Input Convex Lipschitz RNN: A Fast and Robust Approach for Engineering Tasks

Zihao Wang, P S Pravin, Zhe Wu

Abstract—**Computational efficiency and non-adversarial robustness are critical factors in real-world engineering applications. Yet, conventional neural networks often fall short in addressing both simultaneously, or even separately. Drawing insights from natural physical systems and existing literature, it is known that an input convex architecture enhances computational efficiency, while a Lipschitz-constrained architecture bolsters non-adversarial robustness. By leveraging the strengths of convexity and Lipschitz continuity, we develop a novel network architecture, termed Input Convex Lipschitz Recurrent Neural Networks. This model is explicitly designed for fast and robust optimization-based tasks and outperforms existing recurrent units across a spectrum of engineering tasks in terms of computational efficiency and non-adversarial robustness, including real-world solar irradiance prediction for Solar PV system planning at LHT Holdings in Singapore and real-time Model Predictive Control optimization for a nonlinear chemical reactor.**

Index Terms—**Deep Learning, Lipschitz Constrained Neural Networks, Input Convex Neural Networks, Computational Efficiency, Non-Adversarial Robustness, Model Predictive Control, Solar Irradiance Forecasting**

I. Introduction

Two crucial criteria for the resolution of real-world engineering challenges are computational efficiency and non-adversarial robustness. This big-data era has empowered the capabilities of neural networks and transformed them into an incredibly powerful tool, offering comprehensive solutions to numerous engineering problems such as image classification and process modeling. This work draws inspiration from nature-inspired design to develop a fast and robust solution to model nonlinear systems in various engineering disciplines.

Model Predictive Control (MPC) represents an advanced control technique applied to manage a process while complying with a set of constraints, akin to an optimization challenge. Traditional MPC based on first-principles models encounters limitations, particularly in scenarios with intricate dynamics, where deriving these models proves infeasible. Consequently, researchers proposed neural network-based MPC (NN-MPC) as a viable alternative to model system dynamics [1]–[8]. However, MPC built on conventional neural networks often falls short in meeting the aforementioned critical criteria. Nonetheless, efforts have been made to mitigate these limitations. For example, the computational efficiency of NN-MPC is enhanced by implementing an input convex architecture in neural networks [9]. Additionally, non-adversarial robustness of NN-MPC is improved by developing Lipschitz-constrained neural networks (LNNs) [10].

In real-world applications, computational efficiency plays a pivotal role due to the imperative need for real-time decision-making, especially in critical engineering processes such as chemical process operations [9]. We define computational efficiency as the speed at which computational tasks are completed within a given execution timeframe. An effective approach to achieve this efficiency is through the transformation from non-convex to convex structures. Convexity is an ubiquitous characteristic observed in various physical systems. In energy potentials, the potential energy function of certain systems, such as simple harmonic oscillators, can exhibit a convex behavior. For example, the potential energy function of a mass on a spring is a classic example of a convex function in physics [11], [12]. In the context of magnetic fields, the region around a single magnetic pole, where another magnetic pole encounters an attractive force, is considered convex [13], [14]. In the theory of general relativity, the curvature of space-time around massive objects like stars or black holes can be described as convex in certain regions. This curvature influences the paths of objects and light rays near these massive bodies [15].

Building upon nature-inspired design and extending its application to neural networks, Input Convex Neural Networks (ICNNs) is a class of neural network architectures intentionally crafted to maintain convexity in their output with respect to the input. Leveraging these models could be particularly advantageous in various engineering problems, notably in applications such as MPC, due to their inherent benefits especially in optimization. This concept was first introduced by [16] as Feedforward Neural Networks (FNN), and subsequently extended to Recurrent Neural Networks (RNN) by [17], and Long Short-Term Memory (LSTM) by [9].

Additionally, in real-world engineering applications, non-adversarial robustness plays a critical role due to the prevalent noise present in most real-world data, which significantly hampers neural network performance. Our definition of robustness aligns closely with the principles outlined by [10]. Given the inherent noise in real-world process data, our goal is to improve neural networks by effectively learning from noisy training data for comprehensive end-to-end applications. Leveraging LNNs offers a potential solution to the robustness challenge. This concept was first introduced by [18] as FNN, and subsequently extended to RNN by [19], and Convolutional Neural Networks (CNN) by [20].

Motivated by the above considerations, in this work, we

Zihao Wang and Zhe Wu are with the Department of Chemical & Biomolecular Engineering, National University of Singapore, Singapore. E-mail: wuzhe@nus.edu.sg

P S Pravin is with the Department of Electronics & Instrumentation Engineering, National Institute of Technology Silchar, India.

Source code is available at https://github.com/killingbear999/ICLRNN introduce an Input Convex Lipschitz RNN (ICLRNN) that amalgamates the strengths of ICNNs and LNNs. This approach aims to achieve the concurrent resolution of computational efficiency and non-adversarial robustness, which remains a challenge in real-world engineering applications. The core motivation is to develop a fast and robust neural network tailored for optimization frameworks, as shown in Fig. 1. In this study, we validate the efficacy of ICLRNN across various engineering problems that include time series forecasting and neural network-based MPC. Specifically, for time-series forecasting, we engage in planning for Solar Photovoltaic (PV) systems with real-time data from LHT Holdings plants in Singapore, which can be further leveraged for energy dispatch optimization. Furthermore, we use a nonlinear chemical reactor example to show the benefits of ICLRNN in the context of MPC.

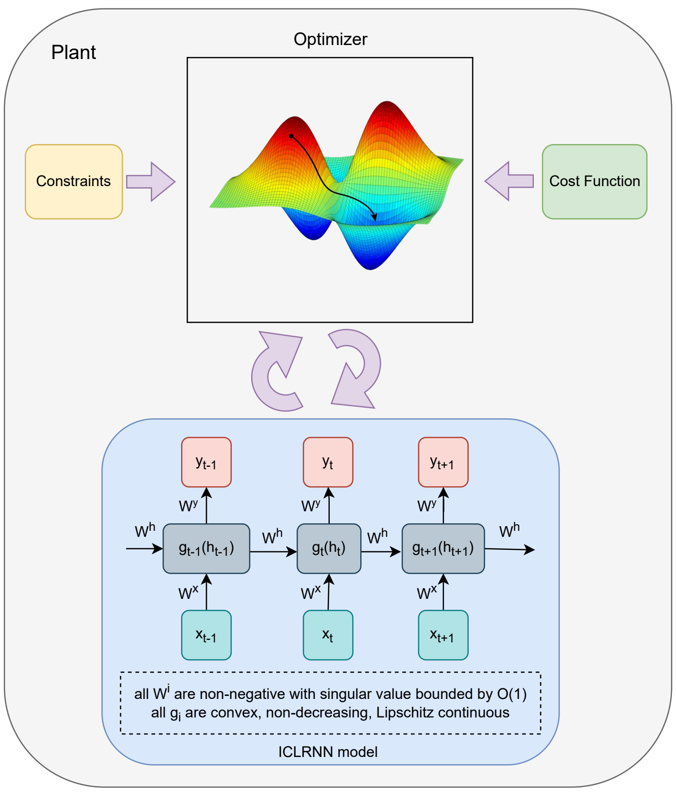

Figure 1: System architecture.

In summary, Section II offers an overview of modern high-performing recurrent units, ICNNs and LNNs. Section III presents the proposed ICLRNN, substantiating its theoretical attributes as input convex and Lipschitz continuous. Section IV illustrates that ICLRNN surpasses state-of-the-art recurrent units in various engineering tasks, including real-time solar irradiance forecasting on Solar PV systems, and optimization and control of chemical processes using MPC. Section V delves into the potential limitations of ICLRNN uncovered through experiments conducted on the Long Range Arena benchmark [21].

II. Background

A. Notations

Let $L(f)$ represents the Lipschitz constant of a function f. "$\circ$" denotes composition. $\|\cdot\|$, $\|\cdot\|_2$, and $\|\cdot\|_F$ denote spectral norm, Euclidean norm, and Frobenius norm of a matrix respectively. $\sigma(\cdot)$ denotes the singular value of a matrix and $r(\cdot)$ denotes the rank of a matrix. For neural networks, x denotes the input, g denotes the activation function, W denotes the weights, h denotes the hidden state, and y denotes the output.

B. High-Performing RNN

In recent years, several high-performing recurrent units have emerged, including UniCORNN [22], rank-coding based RNNs [23], and Linear Recurrent Units [24]. However, these variants primarily emphasize enhancing accuracy, especially for very long time-dependent sequences modelling, which often at the expense of model simplicity and computational efficiency. However, in many engineering applications that we are currently working on with industry partners, accuracy alone does not dictate the suitability of a model. In fact, from an engineering point of view, simplicity and robustness are also important in addition to accuracy, as they ensure the model can be easily adapted and incorporated into the existing industrial operational systems, and can be trusted in an uncertain environment with noise and disturbances. Therefore, our goal is to obtain a model that strikes a balance between these factors while maintaining a satisfactory level of accuracy. This motivation underpins our investigation into the fusion of ICNNs and LNNs.

C. Input Convex RNN

The initiative behind ICNNs is to leverage the power of neural networks, specifically designed to maintain convexity within their decision boundaries [16]. ICNNs aim to combine the strengths of neural networks in modeling complex data with the advantages of convex optimization, which ensures the convergence to a global optimal solution. They are particularly attractive for control and optimization problems, where achieving globally optimal solutions is essential. By maintaining convexity, ICNNs help ensure that the optimization problems associated with neural networks (e.g., neural network-based optimization) remain tractable, thus addressing some of the challenges of non-convexity and the potential for suboptimal local solutions in traditional neural networks.

A foundational work, known as Input Convex RNN (ICRNN), serves as one of the baselines in this paper. The output of ICRNN follows [17]:

$$h_t = g_1(U\mathbf{x}_t + Wh_{t-1} + D_2\mathbf{x}_{t-1})$$
$$y_t = g_2(Vh_t + D_1h_{t-1} + D_3\mathbf{x}_t)$$

The output y_t is convex with respect to the input $\mathbf{x}$ if non-decreasing and convex activation functions are used for g, and $[D_3, D_2, D_1, V, W, U]$ are chosen to be non-negative weights, where $\mathbf{x}$ denotes $\left[\begin{smallmatrix} x \\ -x \end{smallmatrix}\right]$.

D. Lipschitz RNN

By adhering to the definition of Lipschitz continuity for a function f, where an L-Lipschitz continuous f ensures that any minor perturbation to the input results in an output change of at most L times the magnitude of that perturbation. Therefore, constraining neural networks to maintain Lipschitz continuity significantly fortifies their resilience against input perturbations.

A pivotal work, namely the Lipschitz RNN (LRNN), stands as one of the baselines in this paper. The output of LRNN follows [19]:

$$\begin{aligned}\dot{h} &= A_{\beta_A,\gamma_A} h + tanh(W_{\beta_W,\gamma_W} h + Ux + b)\\ y &= Dh\end{aligned}$$

where $\beta_A, \beta_W \in [0,1]$, $\gamma_A, \gamma_W > 0$ are tunable parameters, M_A, M_W, D, U are trainable weights, $\dot{h}$ is the time derivative of h (i.e., h can be updated by the explicit (forward) Euler scheme or a two-stage explicit Runge-Kutta scheme), and $A_{\beta_A,\gamma_A}, W_{\beta_W,\gamma_W}$ is computed as follows:

$$\begin{aligned}A_{\beta_A,\gamma_A} =& (1-\beta_A)(M_A + M_A^T) + \beta_A(M_A - M_A^T)\\ &- \gamma_A I\\ W_{\beta_W,\gamma_W} =& (1-\beta_W)(M_W + M_W^T) + \beta_W(M_W - M_W^T)\\ &- \gamma_W I\end{aligned}$$

III. Input Convex Lipschitz Recurrent Neural Networks

In this section, we develop a novel network architecture, termed ICLRNN, that possesses both Lipschitz continuity and input convexity properties. Specifically, the output of ICLRNN is defined as follows:

$$h_t = g_1(W_t^x x_t + W_t^h h_{t-1} + b_t^h) \tag{1a}$$

$$y_t = g_2(W_t^y h_t + b_t^y) \tag{1b}$$

where all W_i are constrained to be non-negative with singular values small and bounded by $\mathcal{O}(1)$, and all g_i are constrained to be convex, non-decreasing, and Lipschitz continuous.

Similar to LRNN and ICRNN, UniCORNN and Linear Recurrent Units involve high Floating Point Operations (FLOPs). This aspect motivated us to retain the standard RNN architecture while imposing input convexity and Lipschitz constraints on the weights, since many engineering applications can achieve a satisfactory modeling accuracy even using a relatively simple NN structure. However, how to improve the robustness and computational efficiency of these NN models remains an important question (i.e., at this stage, we consider simple RNN as the starting point).

The output of a neural network inherits input convexity and Lipschitz continuity if and only if every hidden state possesses these properties. Unlike [17] and [19], our ICLRNN adheres to this foundational principle by imposing constraints on the weights and activation functions instead of introducing supplementary variables to the original RNN architecture. This approach minimizes the demand of computational resources. We first introduce a proposition to demonstrate that imposing an input convex constraint subsequent to a Lipschitz constraint does not compromise the latter.

The proofs of the following Propositions and Theorems can be found in Appendix A.

Proposition 1. *Let A be an $m \times n$ matrix with its largest singular value at most 1. If all negative elements in matrix A are replaced by 0 to obtain a non-negative matrix B, then the largest singular value of matrix B remains small and bounded by $\sqrt{r(A)}$, and further bounded by $\sqrt{min(m,n)}$.*

For ICLRNN, we first enforce the spectral constraints to ensure that the largest singular values of the weight matrix is at most one [18]. The spectral constraint is implemented in two steps, which is similar to [20]. Firstly, we use spectral normalization as proposed by [25], where the spectral norm is reduced to at most 1 by iteratively evaluating the largest singular value with the power iteration algorithm proposed by [26]. Secondly, we apply the Björck algorithm proposed by [27] to increase other singular values to at most one.

Next, we enforce the non-negative constraint by performing weight clipping to set all negative values in the weights to 0. The theoretical upper bound after the integration of the input convex constraint with the Lipschitz constraint is $\sqrt{r(W)}$ for one single time step as shown in Proposition 1. It is essential to note that the upper bound in Eq. (6) is not precisely tight [28], and thus the upper bound developed in Proposition 1 is not tight. Through experimentation on random large matrices and the experiments in Section IV, we have observed that the Lipschitz constant is normally bounded within the order of $\mathcal{O}(1)$.

Moreover, ICLRNN uses ReLU as the activation function for hidden states instead of the GroupSort function proposed in [18]. This decision aims to maintain convexity within the model architecture. Although GroupSort exhibits gradient norm preservation and higher expressive power [10], its non-convex nature contrasts with our primary objectives. Next, we will proceed to prove the convexity and Lipschitz continuity of ICLRNN.

A. Convexity and Lipschitz Continuity of ICLRNN

Proposition 2. *The Lipschitz constant of an ICLRNN is $\mathcal{O}(1)$ if and only if all weights W_i and all activation functions g_i have a Lipschitz constant bounded by $\mathcal{O}(1)$.*

Proposition 3. *The output y of an ICLRNN is input convex if and only if all W_i are constrained to be non-negative and all g_i are constrained to be convex and non-decreasing.*

By combining Proposition 2 and Proposition 3, the following theorem can be readily derived without requiring additional proof.

Theorem 1. *The output y of an L-layer Input Convex Lipschitz Recurrent Neural Network is a convex, non-decreasing, and Lipschitz continuous function with respect to the input x, where x is in a convex feasible space $\mathcal{X}$, if and only if the following constraints on weights W_i and activation functions g_i are satisfied simultaneously: (1) All W_i are constrained to be non-negative with singular values bounded by $\mathcal{O}(1)$; (2) All g_i are constrained to be convex, non-decreasing, and Lipschitz continuous (e.g., ReLU, Linear, Softmax).*

Next, we proceed by developing a Lyapunov-based MPC (LMPC) framework. Our focus lies in showcasing the transformation of a non-convex NN-based LMPC into a convex optimization problem by utilizing ICNNs.

B. ICLRNN for a Finite-Horizon Convex MPC

1) Class of systems: Specifically, we examine a category of systems that can be expressed through a particular set of ordinary differential equations (ODE), given by the following form:

$$\dot{x} = F(x, u) \tag{2}$$

where $u \in \mathbb{R}^m$ denotes the control action and $x \in \mathbb{R}^n$ denotes the state vector. The function $F : X \times U \to \mathbb{R}^n$ is continuously differentiable, where $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ are connected and compact subsets that enclose an open neighborhood around the origin respectively. In this context, we maintain the assumption throughout this task that $F(0, 0) = 0$, ensuring that the origin $(x, u) = (0, 0)$ serves as an equilibrium point. As it might be impractical to obtain first-principles models for intricate real-world systems, our objective is to construct a neural network to model the nonlinear system specified by Eq. (2), embed it into MPC, and maintain computational efficiency and non-adversarial robustness of MPC simultaneously.

2) LMPC formulation: The optimization problem of the LMPC scheme, which incorporates a neural network model as its predictive element, is expressed as follows [29]:

$$\mathcal{L} = \min_{u \in S(\triangle)} \int_{t_k}^{t_{k+N}} J(\tilde{x}(t), u(t))dt \tag{3a}$$

$$s.t.\ \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{3b}$$

$$u(t) \in U,\ \forall t \in [t_k, t_{k+N}) \tag{3c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{3d}$$

$$V(\tilde{x}(t)) < V(x(t_k)), \forall t \in [t_k, t_{k+N}) \tag{3e}$$

where $S(\triangle)$ denotes the set of piecewise constant functions with period $\triangle$, $\tilde{x}$ denotes the predicted state trajectory, and N denotes the number of sampling periods in the prediction horizon. The objective function $\mathcal{L}$ outlined in Eq. (3a) integrates a cost function J that depends on both the control actions u and the system states x. The system dynamic function $F_{nn}(\tilde{x}(t), u(t))$ expressed in Eq. (3b) is parameterized by an RNN (e.g., the proposed ICLRNN in this work). Eq. (3c) encapsulates the constraint function U, delineating feasible control actions. Eq. (3d) establishes the initial condition $\tilde{x}(t_k)$ in Eq. (3b), referring to the state measurement at $t = t_k$. Lastly, Eq. (3e) represents the Lyapunov-based constraint V, ensuring closed-loop stability within LMPC by mandating a decrease in the value of $V(x)$ over time.

Remark 1 ([9]). *A convex neural network-based LMPC remains convex even when making multi-step ahead predictions, where the prediction horizon is greater than 1. This statement holds true if we incorporate an inherently input convex neural network (e.g., ICLRNN) to the LMPC, as described in Eq. (3b), and if we guarantee that the cost function J in Eq. (3a) is input convex throughout the task.*

IV. Empirical Evaluation

In this section, we assess the performance of the proposed ICLRNN by benchmarking it against other state-of-the-art recurrent units. We aim to exhibit the computational efficiency and non-adversarial robustness of ICLRNN across various engineering challenges. Incorporating real-world data from our industrial partner can illustrate the practical applicability of our proposed method. Unlike abstract benchmark tasks, our real-world scenario aligns more closely with engineering contexts. Demonstrating effectiveness in a practical setting not only enhances the credibility of our approach but also serves commercial interests and addresses stakeholder needs.

Moreover, since linear systems have been well studied with numerous classic data-driven modeling approaches available, making the utilization of neural networks unnecessary in general. However, learning nonlinear dynamics presents a different challenge, and many traditional data-driven modeling techniques fail to obtain an accurate nonlinear dynamic model. Therefore, we choose the Continuous Stirred Tank Reactor (CSTR), which is a widely adopted benchmark example in chemical process modeling, due to its high nonlinearity, especially for states significantly distant from the steady-state.

Drawing from the aforementioned considerations, we meticulously choose two case studies to showcase the computational efficiency and non-adversarial robustness of ICLRNN. These include predicting solar irradiance for Solar PV systems using real-time data from LHT Holdings plants, and implementing an NN-MPC for a chemical reactor. It is worth noting that ablation studies are conducted to ascertain the optimal hyper-parameter values for all subsequent tasks.

Moreover, given that our research encompasses both Lipschitz-constrained neural networks and Input Convex neural networks, we meticulously select the most suitable approach from these categories and ensure consistency by employing a unified set of baseline methods across both tasks. Our primary aim, however, is to showcase the superiority of our proposed method over any alternatives within these two distinct families.

A. Real-World Solar Irradiance Prediction

In this task, we aim to forecast solar irradiance using real-time data sourced from LHT Holdings, a notable wood pallet manufacturing industry based in Singapore (for a detailed manufacturing pipeline of LHT Holdings, refer to Fig. 5 in Appendix B). Before the installation of the Solar photovoltaic (PV) system, LHT Holdings relied solely on the main utility grid to fulfill its energy needs. The Solar PV system was successfully installed by 10 Degree Solar in late 2022, as depicted in Fig. 7 in Appendix B. This transition to solar energy was primarily driven by the aim to minimize manufacturing costs. Three significant uncertainties were identified within the industry, namely solar irradiance affecting the Solar PV system's efficiency, the dynamic real-time pricing of energy sourced from the main utility grid, and the fluctuating energy demand stemming from unexpected customer orders.

At present, the Solar PV system serves as the primary energy source for the industrial facility, while the main utility grid acts as a secondary energy source to supplement any deficiencies in solar energy production. Any surplus solar energy generated beyond the current requirements is efficiently stored in batteries for future utilization (i.e., see Fig. 6 in Appendix B). Hence, in order to optimize real-time energy planning and mitigate manufacturing costs associated with energy procurement from the main utility grid, a high level of precision and accuracy in forecasting solar irradiance becomes imperative. Moreover, a simple and robust trained NN with relatively high accuracy can be seamlessly integrated into applications like energy dispatch optimization. To achieve these goals, the Solar Energy Research Institute of Singapore (SERIS) and Singapore Institute of Manufacturing Technology (SIMTech) have played pivotal roles by installing various sensors specifically designed for the Solar PV system. These sensors include irradiance sensors, humidity sensors, wind sensors, and module temperature sensors. Drawing on parameters akin to those employed by [30], we utilize minute-based average global solar irradiance, ambient humidity, module temperature, wind speed, and wind direction to predict solar irradiance in the subsequent minute.

In this time-series forecasting task, all models undergo training and evaluation using identical settings. These settings encompass the mean squared error as the loss function, the Adam Optimizer, the ReLU activation function except for LRNN. For the training and validation phases, we use data spanning from January 14, 2023, to December 16, 2023. Subsequently, data from December 17, 2023, to January 1, 2024, is exclusively reserved for testing and evaluating the models' predictive performance.

Furthermore, from an engineering perspective, the training and implementation of neural network models are often constrained by limited computational resources, which are typically not as ample as those in computer science. Consequently, engineering applications necessitate relatively simple models capable of effectively handling challenges such as data noise and convexity. To ensure the selection of the most suitable model, it is essential to conduct comparisons under consistent settings (i.e., the same model complexity). This approach ensures fairness, as we evaluate not only accuracy but also other critical factors such as computational efficiency, which can be quantified by FLOPs. The comprehensive discussion of results is accessible in Appendix B.

1) Computational Efficiency Analysis: We leverage FLOPs as a measure to assess the computational efficiency of models. As depicted in Table I, it is evident that among the models compared, ICLRNN demonstrates the most efficient computational performance as a result of its lowest FLOPs. Consequently, this superior efficiency positions ICLRNN as the top-performing model within this context when considering computational resources.

2) Non-Adversarial Robustness Analysis: Given the real-world context of this task, introducing artificial noise for evaluating non-adversarial robustness becomes unnecessary as the inherent noise within real-world data suffices. Notably, our observations reveal that ICRNN encounters challenges with the exploding gradient problem when the hypothesis space size is large (i.e., the number of hidden neurons per layer is more than 128), thereby hindering its ability to effectively model complex system dynamics. While RNN and LSTM exhibit a limited capacity to capture system dynamics, they struggle notably when faced with sudden changes in solar irradiance, leading to less accurate predictions.

Table I: Model complexity with layer size of (256, 256) for solar irradiance prediction.

Model	Trainable Parameters	FLOPs
RNN	199,682	399,362
LSTM	797,186	1,596,418
ICRNN	596,482	1,195,010
LRNN	330,754	2,498,562
ICLRNN (ours)	199,682	399,362

In Fig. 2, we present results from the best state-of-the-art model, LRNN, alongside our model's outcomes specifically for December 28, 2023. LRNN displays marked improvements compared to RNN and LSTM, yet it falls short in accurately predicting solar irradiance during sudden fluctuations. Additionally, during periods of minimal solar irradiance at the start and end of the day, LRNN exhibits slight fluctuations. Highlighted within Fig. 2, our ICLRNN outperforms LRNN, particularly in accurately predicting sudden changes in solar irradiance. This highlights the superior predictive capability of ICLRNN, showcasing its effectiveness in capturing and predicting nuanced fluctuations within the solar irradiance data. Notably, these observations persist across all trials and are not specific solely to December 28, 2023.

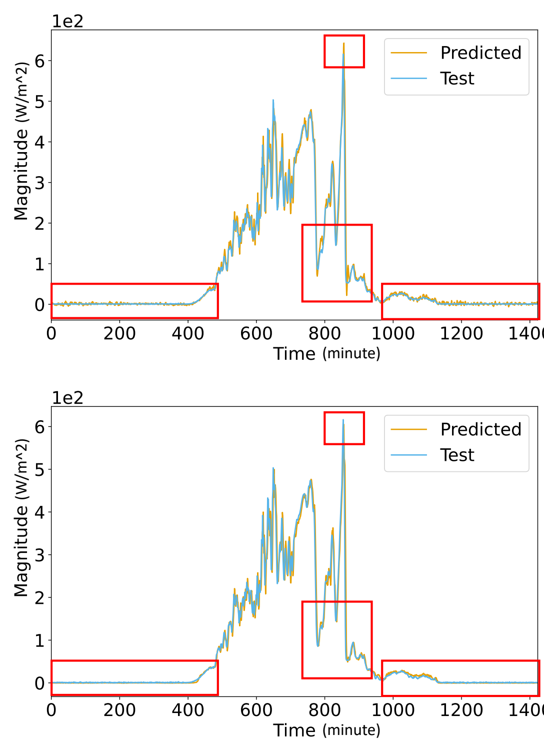

Figure 2: LRNN (top) and ICLRNN (bottom) performances on 2023-12-28 for solar irradiance prediction.

B. MPC for Continuous Stirred Tank Reactor

MPC is a real-time optimization-based control scheme, where the first element of the optimal input trajectory will be executed in the system over the next sampling period, and the MPC optimization problem will be resolved at the subsequent sampling period iteratively until convergence.

Note that in general, solving NN-MPC is time-consuming, since it is a non-convex optimization problem due to the nonlinearity and nonconvexity of neural network models. However, if we integrate an ICNN into a convex MPC problem, the MPC problem will remain convex. In the next subsection, we will demonstrate the improvement of the computational efficiency of MPC using the proposed ICLRNN in Section III.

1) Application to a Chemical Process: We implement LMPC to a nonisothermal and well-mixed continuous stirred tank reactor (CSTR) featuring an irreversible second-order exothermic reaction. This reaction facilitates the conversion of reactant A into product B. The CSTR incorporates a heating jacket responsible for either supplying or extracting heat at a rate Q. The dynamic model of the CSTR is characterized by the ensuing material and energy balance equations [29], [31]:

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \tag{4a}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) + \frac{-\triangle H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{4b}$$

where C_A denotes the concentration of reactant A, F denotes the volumetric flow rate, V denotes the volume of the reactant, C_{A0} denotes the inlet concentration of reactant A, k_0 denotes the pre-exponential constant, E denotes the activation energy, R denotes the ideal gas constant, T denotes the temperature, T_0 denotes the inlet temperature, $\triangle H$ denotes the enthalpy of reaction, ρ_L denotes the constant density of the reactant, C_p denotes the heat capacity, and Q denotes the heat input rate. The detailed constant values and system setup are discussed in [29], [31], and are omitted here.

Moreover, $\triangle Q = Q - Q_s$ and $\triangle C_{A0} = C_{A0} - C_{A0_s}$ denote the manipulated inputs within this system, representing the alterations in the heat input rate and the inlet concentration of reactant A respectively. $x^T = [C_A - C_{As}, T - T_s]$ denotes the system states, where $T - T_s$ and $C_A - C_{As}$ represent the deviations in the temperature of the reactor and the concentration of A from their respective steady-state values. $u^T = [\triangle C_{A0}, \triangle Q]$ denotes the control actions. The primary objective of the controller involves operating the CSTR at (C_{As}, T_s), which is the unstable equilibrium point, by manipulating $\triangle Q$ and $\triangle C_{A0}$ respectively. This manipulation is executed via the LMPC strategy outlined in Eq. (3) by ultimately stabilizing the state at its target steady-state.

We conduct open-loop simulations for the CSTR of Eq. (4). These simulations aim to collect data that mimics the real-world data for training purposes (i.e., as it is difficult to obtain real-world chemical plant data). All models undergo training and evaluation using uniform configurations, incorporating the Categorical Cross-entropy Loss, the Adam Optimizer, and the ReLU activation function except for LRNN. The LMPC reaches the convergence state when $|x_1| = |C_A - C_{As}| < 0.1\ kmol/m^3$ and $|x_2| = |T - T_s| < 3\ K$ simultaneously. The initial conditions are chosen within the stability region, which is defined as $1060x_1^2 + 44x_1x_2 + 0.52x_2^2 - 372 = 0$ (i.e., an ellipse in state space). For a comprehensive evaluation of model performance across various hypothesis space sizes and initial conditions, detailed outcomes and discussions are presented in Appendix C.

Table II: Non-NN Methods Performance.

MODEL	MSE
LINEAR STATE-SPACE MODEL	1.17×10^{-3}
NARX	6.37×10^{-4}
STANDARD RNN	9.77×10^{-5}

Table III: Model complexity with layer size of (256, 256) for CSTR.

MODEL	TRAINABLE PARAMETERS	FLOPS
RNN	198,658	406,548
LSTM	793,090	1,597,460
ICRNN	596,482	1,204,233
LRNN	329,730	2,505,748
ICLRNN (OURS)	198,658	406,548

2) Non-NN methods Performance: In addition to NN-based methods, we investigate various data-driven models to provide a more comprehensive analysis, including non-NN methods such as the linear state-space model and the nonlinear autoregressive network with exogenous inputs (NARX), which have been widely used for modeling nonlinear dynamic systems for decades. As depicted in Table II, both the linear state-space model and the NARX model fail to outperform even a standard RNN.

3) Model Computational Efficiency: To compare the computational efficiency in modeling the system dynamics in Eq. (4) across models, we quantify their FLOPs. Table III illustrates that ICLRNN has the lowest FLOPs, thereby outperforming or matching state-of-the-art methods. Similarly to the results shown in previous tasks, despite LRNN having a relatively modest count of trainable parameters, its high FLOPs indicate substantial computational demands, which is about 6 times more than ICLRNN. Conversely, ICRNN encounters challenges with exploding gradients, particularly when the network complexity exceeds a certain threshold.

4) Non-Adversarial Robustness Analysis: To scrutinize the non-adversarial robustness of ICLRNN in modeling the system dynamics in Eq. (4), we assess the model's sensitivity with respect to test accuracy when exposed to a series of perturbed inputs influenced by Gaussian noise. As illustrated in Fig. 3, the findings, averaging across 5 trials, reveal that ICLRNN consistently outperforms or matches the performance of state-of-the-art models across varying levels of noise.

5) Computational Efficiency of ICLRNN-based LMPC: Incorporating an ICNN into the aforementioned LMPC in Eq. (3) transforms it into a convex optimization problem.

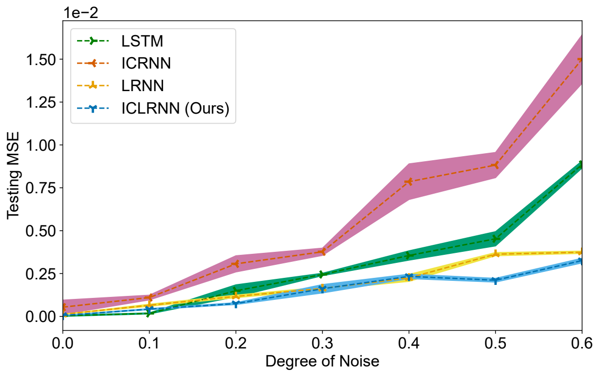

Figure 3: Model performance w.r.t different degree of Gaussian noise for CSTR.

Subsequently, PyIpopt, as a Python version of IPOPT [32], is used to address the LMPC problem. The integration time step is set at $h_c = 10^{-4}\ hr$, while the sampling period remains $\triangle = 0.005\ hr$. The design of the Lyapunov function, denoted as $V(x) = x^T Px$, involves configuring the positive definite matrix P as $\left[\begin{smallmatrix} 1060 & 22 \\ 22 & 0.52 \end{smallmatrix}\right]$, which ensures the convexity of the LMPC.

We execute LMPC for a fixed timeframe for various models and repeat with various initial conditions to evaluate the performance of different NN-based LMPC. By examining Fig. 4, it is evident that the ICLRNN-based LMPC converges, whereas other NN-based LMPC models do not. The fastest convergence speed highlights the superior computational efficiency of the ICLRNN-based LMPC.

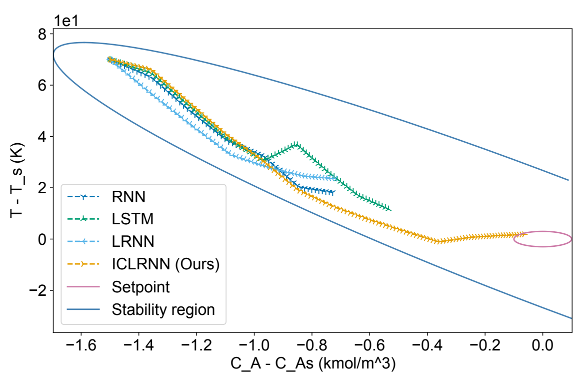

Figure 4: LMPC converging path in a fixed timeframe with an initial condition at $[-1.5\ kmol/m^3, 70\ K]$.

V. Limitations and Future Works

During our experiments, we observed that ICRNN faces challenges related to the exploding gradient problem, prompting us to discuss this issue and its relevance to the exploding and vanishing gradients phenomenon. Fortunately, as a beneficial side-effect of addressing our aforementioned objectives, our proposed method exhibits reduced susceptibility to the exploding gradient problem to a certain extent, attributable to its incorporation of Lipschitz constraints.

Subsequently, we further investigated the potential of ICLRNN by delving into the Long Range Arena benchmark [21], encompassing tasks involving long sequence modeling such as byte-level text sentiment classification and pixel sequence image classification.

Both ICRNN and our ICLRNN exhibit limitations in tasks requiring the modeling of very long time-dependent sequences, such as the byte-level text classification task, where both achieve an accuracy of approximately 50% in this binary sentiment classification task. This limitation stems from the challenge of the exploding gradient problem and lack of representative power due to non-negative weights, which both methods encounter when attempting to model extremely lengthy sequences, and we provide a theoretical explanation as follows.

While the Lipschitz constraint may alleviate the exploding gradient problem by bounding the weights, integrating input convexity introduces a new challenge by raising this upper bound. Applying the Lipschitz constraint first to reduce the spectral norm to 1, followed by imposing the input convex constraint, yields a neural network no longer 1-Lipschitz constrained but rather $n\sqrt{r(W)}$-constrained, where n represents the number of time-steps. Consequently, in modeling very long time-dependent sequences, the Input Convex structure exacerbates the exploding gradient problem, rendering it unmitigated even with Layer Normalization. For readers interested in tackling the challenges of the exploding and vanishing gradient problem within RNNs, we recommend exploring several noteworthy and comprehensive works [33]–[35].

We acknowledge this limitation as inherent to our proposed method, thereby constraining its applicability. However, despite this drawback, we opt to retain the Input Convex structure, as it demonstrates efficacy in many engineering applications where the length of time-series data is not as long as those in Long Range Arena benchmark. For example, in the first case study where we develop a model for real-world solar irradiance prediction, the prediction horizon is relatively short (e.g., an hour or a few hours ahead) since this model is used for real-time energy planning. In the second example where we develop a model for real-time optimization of CSTR, again, the model is to predict one sampling period only, since the controller is typically implemented in a feedback manner with the true measurements of states received at every time step. Therefore, there are many applications in engineering disciplines that focus on short-term prediction, where the proposed ICLRNN will play an important role due to its superiority in computational efficiency and robustness compared to traditional RNNs.

Furthermore, because of the input convex architecture, all ICNNs excel in regression tasks rather than classification tasks, including ICLRNN. This is because, to maintain input convexity, all activation functions must be convex, prohibiting the use of certain activation functions such as Sigmoid, which is standard for classification tasks. However, given our focus in this work on developing a fast and robust neural network tailored for optimization tasks, which are regression-based, we can disregard this limitation.

VI. Conclusion

In conclusion, this work introduces an Input Convex Lipschitz RNN to improve computational efficiency and non-adversarial robustness of neural networks in various engineering applications such as real-world solar irradiance prediction, and process modeling and optimization. Theoretical analyses of its input convexity and Lipschitz continuity properties are provided. Furthermore, from our experiments, we observed that both LRNN and ICRNN require significant computational resources. LRNN hinders the performance of optimizer. Additionally, ICRNN encounters challenges related to exploding gradient problems when operating within large hypothesis space sizes and proved sensitive to input perturbations. Despite potential limitations, the ICLRNN model was demonstrated to surpass state-of-the-art recurrent units across various engineering tasks, ranging from simulated studies to real-world applications, in terms of computational efficiency and non-adversarial robustness.

References

[1] X. Bao, Z. Sun, and N. Sharma, "A Recurrent Neural Network Based MPC for a Hybrid Neuroprosthesis System," in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 4715–4720, IEEE, 2017.

[2] A. Afram, F. Janabi-Sharifi, A. S. Fung, and K. Raahemifar, "Artificial Neural Network (ANN) Based Model Predictive Control (MPC) and Optimization of HVAC Systems: A State of the Art Review and Case Study of a Residential HVAC System," Energy and Buildings, vol. 141, pp. 96–113, 2017.

[3] N. Lanzetti, Y. Z. Lian, A. Cortinovis, L. Dominguez, M. Mercangöz, and C. Jones, "Recurrent Neural Network Based MPC for Process Industries," in 2019 18th European Control Conference (ECC), pp. 1005–1010, IEEE, 2019.

[4] M. J. Ellis and V. Chinde, "An Encoder-Decoder LSTM-Based EMPC Framework Applied to a Building HVAC System," Chemical Engineering Research and Design, vol. 160, pp. 508–520, 2020.

[5] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and Fast Tracking on a Robot Manipulator: Robust MPC and Neural Network Control," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3050–3057, 2020.

[6] Y. Zheng, X. Wang, and Z. Wu, "Machine Learning Modeling and Predictive Control of the Batch Crystallization Process," Industrial & Engineering Chemistry Research, vol. 61, no. 16, pp. 5578–5592, 2022.

[7] Y. Zheng, T. Zhao, X. Wang, and Z. Wu, "Online Learning-Based Predictive Control of Crystallization Processes under Batch-to-Batch Parametric Drift," AIChE Journal, vol. 68, no. 11, p. e17815, 2022.

[8] N. Sitapure and J. S.-I. Kwon, "Neural Network-Based Model Predictive Control for Thin-film Chemical Deposition of Quantum Dots using Data from a Multiscale Simulation," Chemical Engineering Research and Design, vol. 183, pp. 595–607, 2022.

[9] Z. Wang and Z. Wu, "Input Convex LSTM: A Convex Approach for Fast Lyapunov-Based Model Predictive Control," arXiv preprint arXiv:2311.07202, 2023.

[10] W. G. Y. Tan and Z. Wu, "Robust Machine Learning Modeling for Predictive Control using Lipschitz-Constrained Neural Networks," Computers & Chemical Engineering, vol. 180, p. 108466, 2024.

[11] H. Goldstein, C. Poole, and J. Safko, "Classical Mechanics," 2002.

[12] J. R. Taylor and J. R. Taylor, Classical Mechanics, vol. 1. Springer, 2005.

[13] E. M. Purcell, "Electricity and Magnetism," Berkeley University, vol. 2, 1963.

[14] D. J. Griffiths, "Introduction to Electrodynamics," 2005.

[15] J. B. Hartle, "Gravity: an Introduction to Einstein's General Relativity," 2003.

[16] B. Amos, L. Xu, and J. Z. Kolter, "Input Convex Neural Networks," in International Conference on Machine Learning, pp. 146–155, PMLR, 2017.

[17] Y. Chen, Y. Shi, and B. Zhang, "Optimal Control via Neural Networks: A Convex Approach," arXiv preprint arXiv:1805.11835, 2018.

[18] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz Function Approximation," in International Conference on Machine Learning, pp. 291–301, PMLR, 2019.

[19] N. B. Erichson, O. Azencot, A. Queiruga, L. Hodgkinson, and M. W. Mahoney, "Lipschitz Recurrent Neural Networks," arXiv preprint arXiv:2006.12070, 2020.

[20] M. Serrurier, F. Mamalet, A. González-Sanz, T. Boissin, J.-M. Loubes, and E. Del Barrio, "Achieving Robustness in Classification using Optimal Transport with Hinge Regularization," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 505–514, 2021.

[21] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long Range Arena: A Benchmark for Efficient Transformers," arXiv preprint arXiv:2011.04006, 2020.

[22] T. K. Rusch and S. Mishra, "Unicornn: A Recurrent Model for Learning very Long Time Dependencies," in International Conference on Machine Learning, pp. 9168–9178, PMLR, 2021.

[23] A. Jeffares, Q. Guo, P. Stenetorp, and T. Moraitis, "Spike-Inspired Rank Coding for Fast and Accurate Recurrent Neural Networks," arXiv preprint arXiv:2110.02865, 2021.

[24] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De, "Resurrecting Recurrent Neural Networks for Long Sequences," in International Conference on Machine Learning, pp. 26670–26698, PMLR, 2023.

[25] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," arXiv preprint arXiv:1802.05957, 2018.

[26] G. H. Golub and H. A. Van der Vorst, "Eigenvalue Computation in the 20th Century," Journal of Computational and Applied Mathematics, vol. 123, no. 1-2, pp. 35–65, 2000.

[27] Å. Björck and C. Bowie, "An Iterative Algorithm for Computing the Best Estimate of an Orthogonal Matrix," SIAM Journal on Numerical Analysis, vol. 8, no. 2, pp. 358–364, 1971.

[28] S. M. Rump, "Verified Bounds for Singular Values, in particular for the Spectral Norm of a Matrix and Its Inverse," BIT Numerical Mathematics, vol. 51, pp. 367–384, 2011.

[29] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine-Learning-Based Predictive Control of Nonlinear Processes. Part i: Theory," AIChE Journal, vol. 65, no. 11, p. e16729, 2019.

[30] E. Zelikman, S. Zhou, J. Irvin, C. Raterink, H. Sheng, A. Avati, J. Kelly, R. Rajagopal, A. Y. Ng, and D. Gagne, "Short-Term Solar Irradiance Forecasting using Calibrated Probabilistic Models," arXiv preprint arXiv:2010.04715, 2020.

[31] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine-Learning-Based Predictive Control of Nonlinear Processes. Part ii: Computational Implementation," AIChE Journal, vol. 65, no. 11, p. e16734, 2019.

[32] A. Wächter and L. T. Biegler, "On the Implementation of an Interior-Point FilterLine-Search Algorithm for Large-Scale Nonlinear Programming," Mathematical programming, vol. 106, pp. 25–57, 2006.

[33] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5457–5466, 2018.

[34] Y. Hu, A. Huber, J. Anumula, and S.-C. Liu, "Overcoming the Vanishing Gradient Problem in Plain Recurrent Networks," arXiv preprint arXiv:1801.06105, 2018.

[35] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. Schön, "Beyond Exploding and Vanishing Gradients: Analysing RNN Training using Attractors and Smoothness," in International conference on artificial intelligence and statistics, pp. 2370–2380, PMLR, 2020.

[36] S. P. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.

[37] K. Eriksson, D. Estep, and C. Johnson, Applied Mathematics: Body and Soul: Volume 1: Derivatives and Geometry in IR3. Springer Science & Business Media, 2013.

[38] B. Gao and L. Pavel, "On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning," arXiv preprint arXiv:1704.00805, 2017.

[39] A. Virmaux and K. Scaman, "Lipschitz Regularity of Deep Neural Networks: Analysis and Efficient Estimation," Advances in Neural Information Processing Systems, vol. 31, 2018.

Appendix A
Proofs

The definitions of convexity and Lipschitz continuity are given as follows:

Definition 1 ([36]). *A function $f : X \to \mathbb{R}$ is convex if and only if the following inequality holds $\forall\ (x_1, x_2) \in X$, with $x_1 \neq x_2$, and $\forall\ \lambda \in (0, 1)$:*

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$

and it is strictly convex if the relation $\leq$ becomes $<$.

Definition 2 ([37]). *A function $f : X \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant L (i.e., L-Lipschitz) if and only if the following inequality holds $\forall\ (x_1, x_2) \in X$:*

$$\|f(x_1) - f(x_2)\|_2 \leq L\|x_1 - x_2\|_2$$

The following lemmas are provided to prove Proposition 2/4 regarding the Lipschitz continuity of ICLRNN.

Lemma 1 ([37]). *Consider a set of functions, e.g., $f : X \to \mathbb{R}$ and $g : X \to \mathbb{R}$ with Lipschitz constants $L(f)$ and $L(g)$, respectively, by taking their sum, i.e., $h = f + g$, the Lipschitz constant $L(h)$ of h satisfies:*

$$L(h) \leq L(f) + L(g)$$

Lemma 2 ([37]). *Consider a set of functions, e.g., $f : X \to \mathbb{R}$ and $g : X \to \mathbb{R}$ with Lipschitz constants $L(f)$ and $L(g)$ respectively, by taking their composition, denoted as $h = f \circ g$, the Lipschitz constant $L(h)$ of the resultant function h satisfies:*

$$L(h) \leq L(f) \times L(g)$$

Lemma 3. *Given a linear function with a weight $W \in \mathbb{R}^{n\times m}$, we have:*

$$L(W) = \|W\| = \sigma_{max}(W)$$

Lemma 4 ([38], [39]). *The most common activation functions such as ReLU, Sigmoid, and Tanh have a Lipschitz constant that equals 1, while Softmax has a Lipschitz constant bounded by 1.*

Proposition 4. *The Lipschitz constant of an ICLRNN is $\mathcal{O}(1)$ if and only if all weights W_i and all activation functions g_i have a Lipschitz constant bounded by $\mathcal{O}(1)$.*

Proof. By unrolling the recurrent operations over time t in Eq. (1), the output of ICLRNN can be described as a function:

$$\begin{aligned} f(x) = &g_t(W_t^y g_{t-1}^h(W_{t-1}^x x_{t-1} \\ &+ W_{t-1}^h ... g_1(W_1^x x_1 + W_1^h h_0))) \end{aligned} \tag{5}$$

The Lipschitz constant of an ICLRNN is upper bounded by the product of sum of the individual Lipschitz constants (i.e., Lemma 1 and Lemma 2):

$$\begin{aligned} L(f) \leq &L(g_t) \times L(W_t^y) \times L(g_{t-1}) \times (L(W_{t-1}^x) \\ &+ L(W_{t-1}^h) \times ...L(g_1) \times (L(W_1^x) + L(W_1^h))) \end{aligned}$$

If we ensure that the Lipschitz constants of all W_i in ICLRNN are $\mathcal{O}(1)$, the ICLRNN for classification tasks with ReLU activation for hidden states and Softmax activation for output layer has a Lipschitz constant of $\mathcal{O}(1)$, while the ICLRNN for regression tasks with ReLU activation for hidden states and Linear activation for output layer has a Lipschitz constant of $\mathcal{O}(1)$ (i.e., Lemma 3 and Lemma 4). ☐

The following lemmas are provided to prove Proposition 3/5 regarding the convexity of ICLRNN.

Lemma 5 ([36]). *Consider a set of convex functions, e.g., $f : X \to \mathbb{R}$ and $g : X \to \mathbb{R}$, their weighted sum, i.e., $\alpha f + \beta g$, remains convex if coefficients α and β are non-negative.*

Lemma 6 ([36]). *Consider a convex function $f : X \to \mathbb{R}$ and a convex monotone non-decreasing function $g : X \to \mathbb{R}$, the composition of f with g, denoted as $g \circ f$, is convex and non-decreasing.*

Proposition 5. *The output y of an ICLRNN is input convex if and only if all W_i are constrained to be non-negative and all g_i are constrained to be convex and non-decreasing.*

Proof. By unrolling the recurrent operations over time t in Eq. (1), the output of ICLRNN can be described as a function of Eq. (5). The output of an ICLRNN $y = f(x)$ is convex with respect to x (i.e., Lemma 5 and Lemma 6). ☐

The subsequent Proposition 6 aims to establish an upper bound on the weights for a single time step, following the integration of the input convex constraint with the Lipschitz constraint.

Proposition 6. *Let A be an $m \times n$ matrix with its largest singular value at most 1. If all negative elements in matrix A are replaced by 0 to obtain a non-negative matrix B, then the largest singular value of matrix B remains small and bounded by $\sqrt{r(A)}$, and further bounded by $\sqrt{min(m,n)}$.*

Proof. We begin the proof with a commonly admitted fact:

$$\|A\| = \sigma_{max}(A) \leq \|A\|_F = (\sum_{ij} |a_{ij}|^2)^{\frac{1}{2}} = (\sum_k \sigma_k^2)^{\frac{1}{2}} \tag{6}$$

where A is an $m \times n$ matrix with its largest singular value at most 1 (i.e., $\|A\| = \sigma_{max}(A) \leq 1$), a_{ij} represents the elements in A, and the number of singular value k is at most $min(m,n)$. Equality holds if and only if A is a rank-one matrix or a zero matrix. Given that the rank of a matrix is precisely the number of non-zero singular values, we extend the inequality in Eq. (6) further:

$$\|A\|_F \leq (r(A)\sigma_{max}(A)^2)^{\frac{1}{2}} = \sqrt{r(A)}\|A\| \tag{7}$$

where the equality holds if and only if all non-zero singular values are equal.

Let B be a non-negative $m \times n$ matrix transformed by replacing all negative elements in A with 0, where b_{ij} represents the elements in B. Based on the transformation, we have the property:

$$|b_{ij}|^2 \leq |a_{ij}|^2$$

Thus, the following inequality holds:

$$\|B\| \leq \|B\|_F = (\sum_{ij} |b_{ij}|^2)^{\frac{1}{2}} \leq (\sum_{ij} |a_{ij}|^2)^{\frac{1}{2}} = \|A\|_F \tag{8}$$

Based on Eq. (7) and (8) and the assumption that $\|A\| \leq 1$, we conclude:

$$\|B\| \leq \|A\|_F \leq \sqrt{r(A)}\|A\| \leq \sqrt{r(A)} \leq \sqrt{min(m,n)}$$

$\square$

Appendix B
Real-World Solar Irradiance Prediction Results

Fig. 5 illustrates the production pipeline adopted by LHT Holdings. Meanwhile, in Fig. 6, a depiction of the different energy sources employed within the production process is presented. Fig. 7 showcases the physical setup of the Solar PV system, encompassing solar panels, inverters, batteries, meters, humidity and wind sensors, module temperature sensors, and irradiance sensors. Data is systematically recorded on a minute-by-minute basis and seamlessly uploaded online into the system.

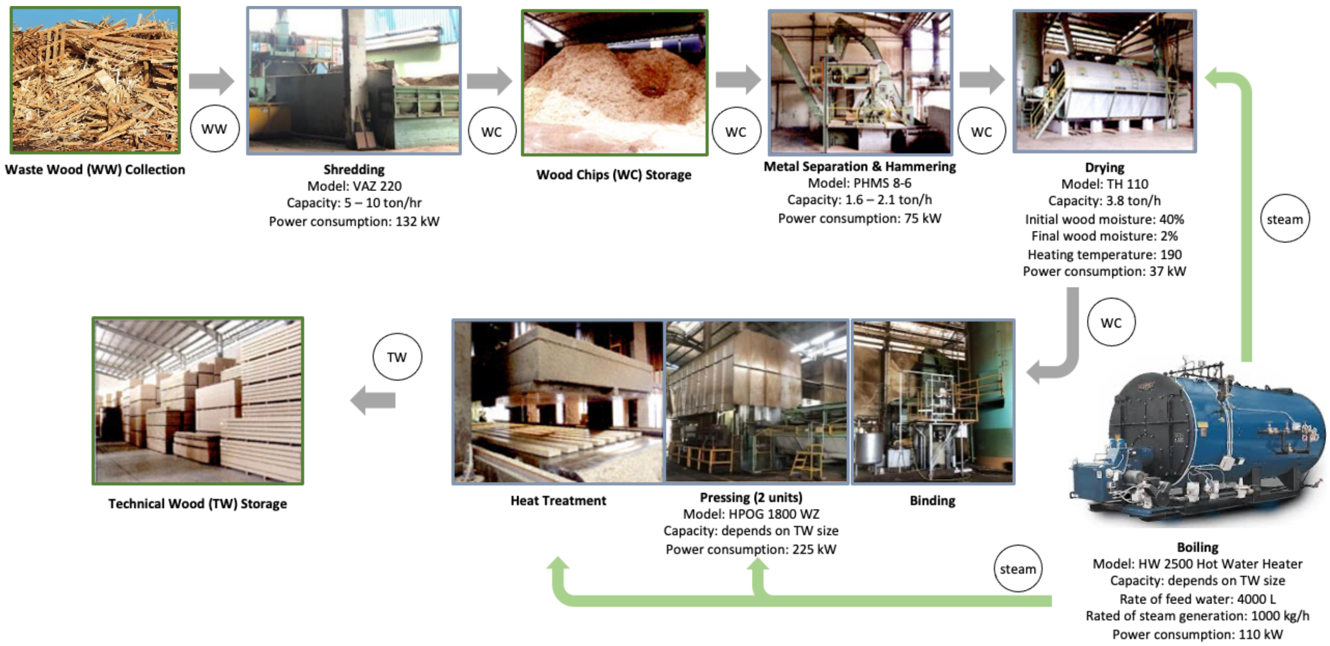

Figure 5: LHT Holdings technical wood production pipeline.

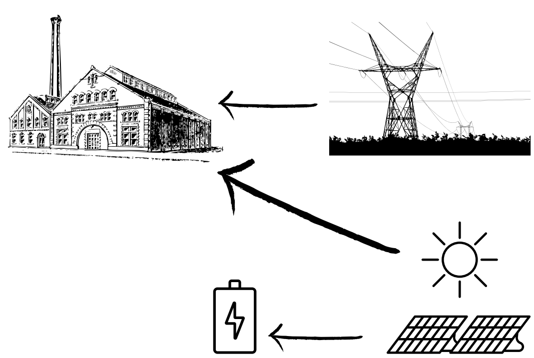

Figure 6: LHT Holdings energy sources for manufacturing.

Fig. 8 shows the model performance on solar irradiance prediction on December 28, 2023. The top two figures display the RNN and LSTM prediction outcomes, respectively. Both models demonstrate poor performance, failing to adjust adequately when faced with sudden changes in solar irradiance, leading to overshooting. Moreover, the middle two figures reveal that the ICRNN encounters issues related to the exploding gradient problem, particularly when the hypothesis space size becomes excessively large. The middle left figure shows the prediction performance of ICRNN with 128 hidden neurons per layer, while the middle right figure shows the prediction performance of ICRNN with 256 hidden neurons per layer. When surpassing 128 hidden neurons per layer, the ICRNN struggles to effectively capture and learn the system's dynamics.

Moving to the bottom two figures, the LRNN and ICLRNN prediction results for the same date, December 28, 2023, are presented. The LRNN, constrained by Lipschitz continuity, exhibits significantly improved performance compared to the RNN and LSTM models. However, it still encounters challenges in accurately predicting solar irradiance during sudden fluctuations, and its excessively high FLOPs signify its computational intensity (i.e., 2,498,562 for LRNN compared to 399,362 for ICLRNN).

Figure 7: LHT Holdings Solar PV system.

Moreover, slight fluctuations in the LRNN predictions occur during periods of minimal solar irradiance at the start and end of the day. Notably, these observations persist across all trials and are not specific solely to December 28, 2023. It is clearly shown in Fig. 2 that our ICLRNN model surpasses state-of-the-art models, demonstrating superior predictive capabilities across various scenarios.

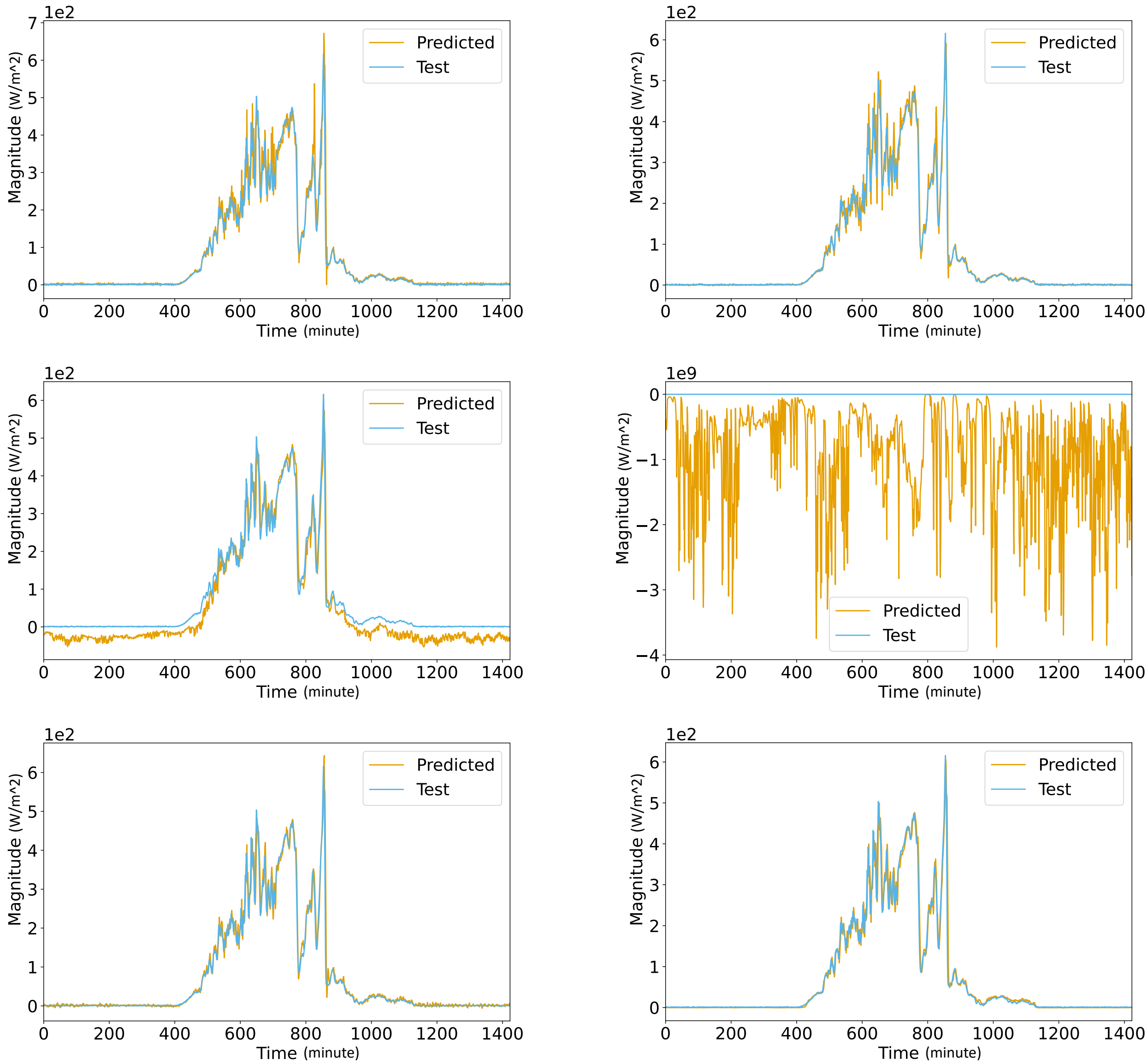

Figure 8: RNN (top left), LSTM (top right), ICRNN (middle left and middle right), LRNN (bottom left), and ICLRNN (bottom right) performance on 2023-12-28 for solar irradiance prediction.

Appendix C
MPC-Based CSTR Results

Fig. 3 showcases the model's testing accuracy with respect to varying degrees of Gaussian noise applied to the input. These findings indicate ICLRNN's non-adversarial robustness across diverse hypothesis space sizes with respect to different Gaussian noise levels. Notably, ICLRNN exhibits superior computational efficiency, as evidenced by its notably lower number of FLOPs compared to other models.

While ICRNN performs well against various degrees of Gaussian noise within smaller hypothesis spaces, it encounters challenges when the hypothesis space expands, succumbing to the issue of the exploding gradient problem, as evidenced in previous task. LRNN demonstrates comparable performance to ICLRNN specifically with a layer size of (512, 512). However, it is computationally demanding, characterized by substantially higher FLOPs (i.e., 9,992,212 for LRNN compared to 1,599,508 for ICLRNN). This computational intensity stands as a significant drawback despite its competitive performance.

We now delve into the control performance assessment of NN-based LMPC. To assess and compare the convergence rates of various NN-based LMPC, we conduct different NN-based LMPC under a fixed timeframe and repeat with different initial conditions. Figs. 9 and 10 provide compelling evidence that the ICLRNN-based LMPC achieves convergence, unlike other NN-based LMPC, within a fixed timeframe. This outcome distinctly indicates that ICLRNN-based LMPC exhibits the fastest convergence speed under identical settings.

In Fig. 9, the left panels show the results corresponding to the initial condition set at $[-1.5\ kmol/m^3, 70\ K]$. Specifically, the top left panel displays the concentration profile, the middle left panel shows the temperature, and the bottom left panel exhibits the converging path. Additionally, the right panels showcase the results from the initial condition at $[1.5\ kmol/m^3, -70\ K]$, featuring the concentration profile in the top right panel, temperature profile in the middle right panel, and the converging path in the bottom right panel. Moving to Fig. 10, the left panels depict the results obtained from the initial condition of $[-1.25\ kmol/m^3, 50\ K]$. Here, the top left panel represents the concentration, the middle left panel illustrates the temperature, and the bottom left panel shows the converging path. Similarly, the right panels exhibit the results of the initial condition at $[1.25\ kmol/m^3, -50\ K]$.

Across various initial conditions, ICLRNN-based LMPC consistently demonstrates the fastest convergence speed, thereby surpassing the state-of-the-art methodologies.

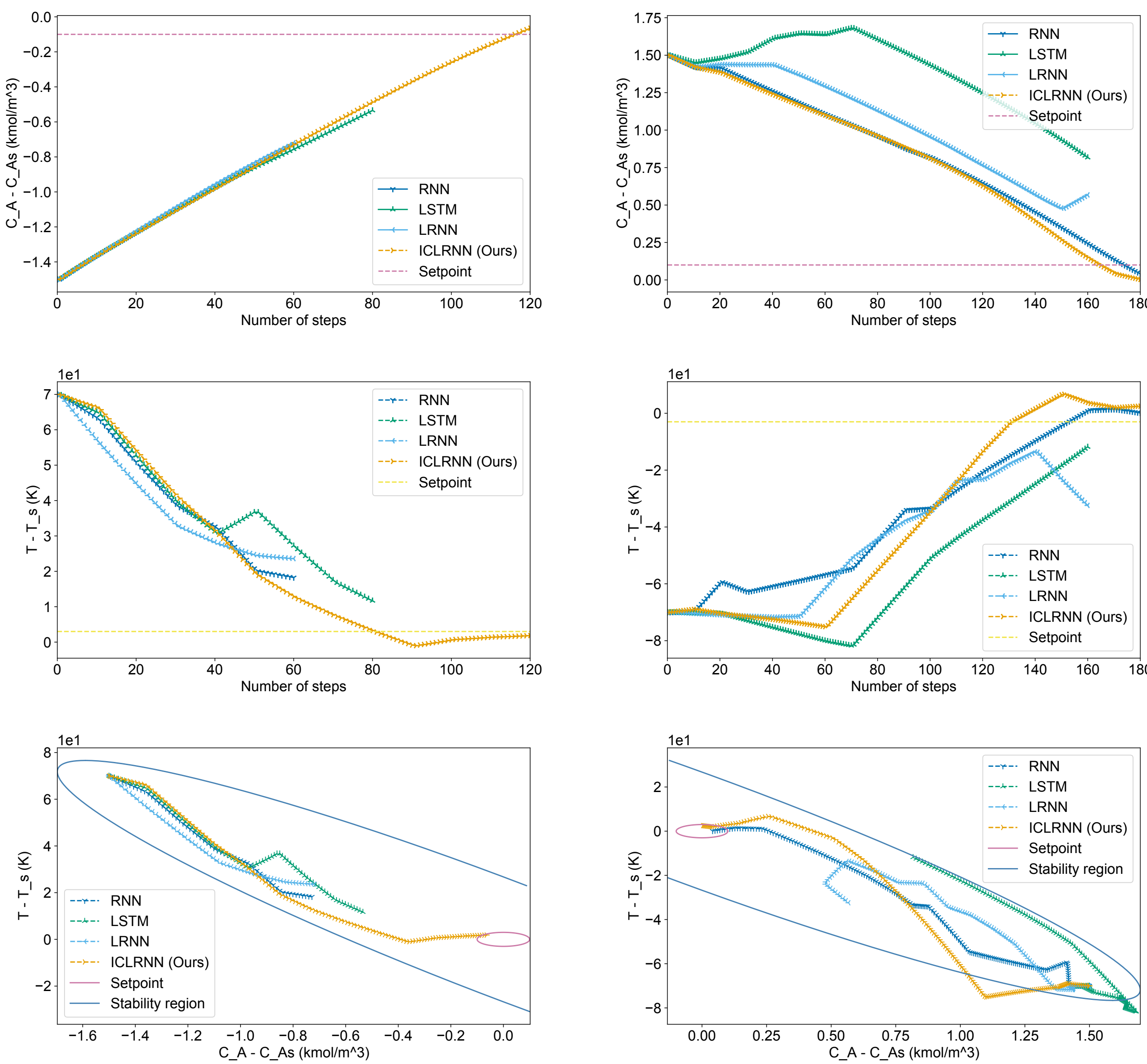

Figure 9: Concentration profile (top left), temperature profile (middle left), converging path (bottom left) in a fixed timeframe with an initial condition at $[-1.5\ kmol/m^3, 70\ K]$ under LMPC, and concentration profile (top right), temperature profile (middle right), converging path (bottom right) in a fixed timeframe with an initial condition at $[1.5\ kmol/m^3, -70\ K]$.

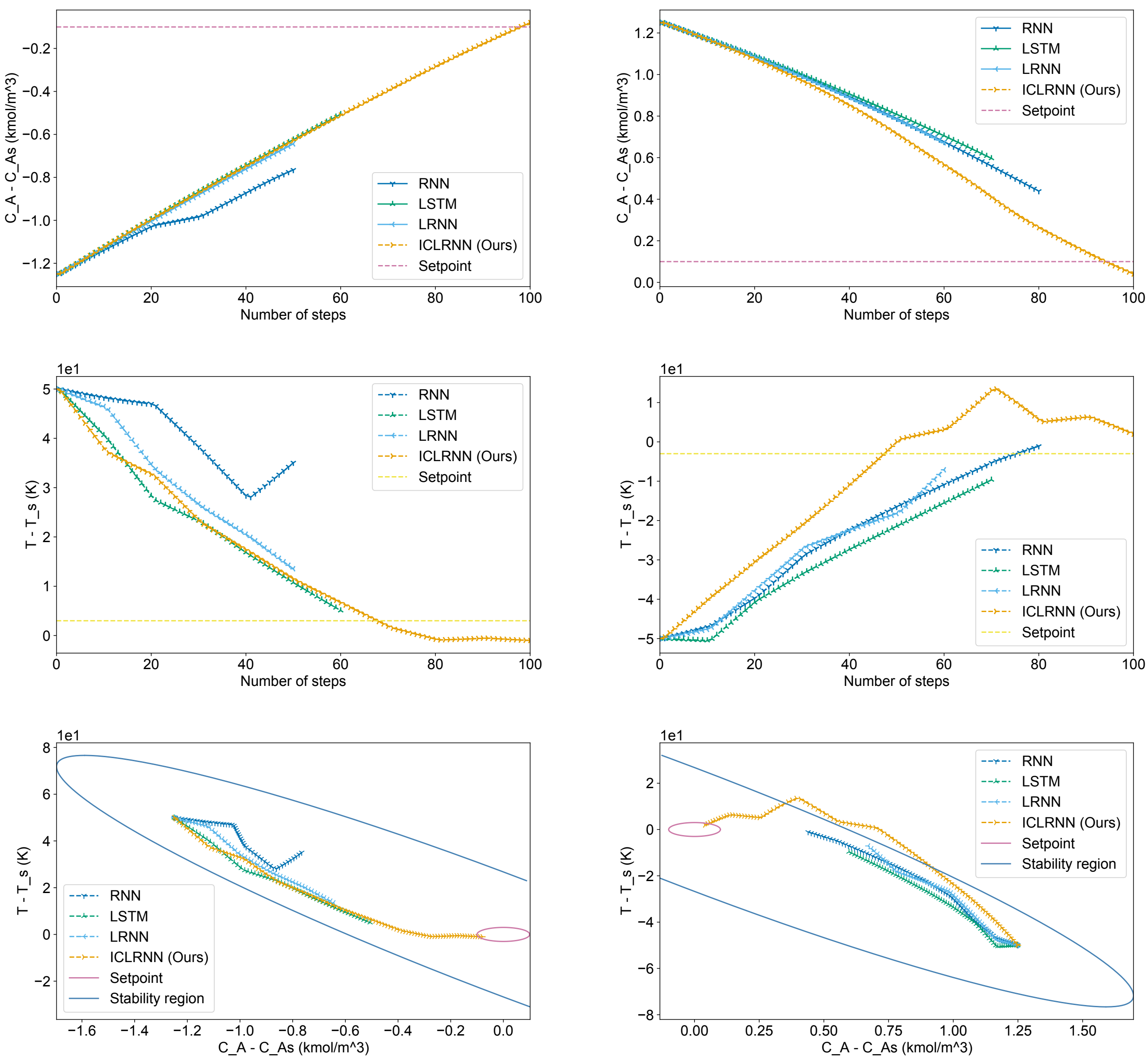

Figure 10: Concentration profile (top left), temperature profile (middle left), converging path (bottom left) in a fixed timeframe with an initial condition at $[-1.25\ kmol/m^3, 50\ K]$, and concentration profile (top right), temperature profile (middle right), converging path (bottom right) in a fixed timeframe with an initial condition at $[1.25\ kmol/m^3, -50\ K]$.