aerospace

MDPI

Article

Sparse Neural Dynamics Modeling for NMPC-Based UAV Trajectory Tracking

Xinyuan Qiu, Changxuan Liu and Jun Li *

School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China
* Correspondence: lijun.uv@zju.edu.cn

Abstract

Accurate and computationally efficient trajectory tracking remains a critical challenge for unmanned aerial vehicles (UAVs), particularly when nonlinear model predictive control (NMPC) is combined with learning-based dynamics models that introduce significant computational burden. This paper proposes a sparse neural dynamics modeling approach by integrating structured pruning and robustness-enhancing fine-tuning techniques to enable efficient nonlinear MPC (NMPC) for UAV trajectory tracking. To this end, a structured neuron-level pruning strategy is introduced, combining L1-norm importance scores with adversarial sensitivity analysis to identify and remove redundant neurons from a neural dynamics model. To preserve smoothness and robustness in closed-loop control, spectral norm constraints and gradient regularization are further incorporated during fine-tuning. The resulting pruned neural dynamics model is embedded into an NMPC framework for online trajectory tracking. Simulation results on a fixed-wing UAV demonstrate that the proposed method reduces the number of trainable parameters by approximately 69% and achieves a 19% reduction in average NMPC solve time, leading to an effective control update frequency of about 39 Hz under the considered simulation settings. Compared with conventional controllers, including TECS and linear MPC, the proposed approach achieves significantly improved trajectory tracking accuracy, as reflected by lower MAE and RMSE across all position axes. These results indicate that structured sparsification of neural dynamics models provides an effective means to enhance both computational efficiency and tracking performance in NMPC-based UAV control.

Keywords: model predictive control; learning-based control; structured pruning; trajectory tracking; UAV

Check for updates

Academic Editor: Gokhan Inalhan

Received: 1 February 2026
Revised: 23 February 2026
Accepted: 27 February 2026
Published: 28 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

1.1. Research Background

Model predictive control (MPC) has gained recognition as an advanced control strategy with strong performance across a wide range of application domains [1], owing to its ability to explicitly handle constraints and optimize control actions over a finite prediction horizon. MPC has been extensively studied in representative complex control systems [2,3], illustrating its general applicability as a constrained optimal control framework. In aerial robotics, MPC has been widely adopted due to its constraint-awareness and predictive optimization capability, enabling precise trajectory tracking under nonlinear dynamics [4].

However, the high dependence on model accuracy and the issue of solution time efficiency remain major challenges for MPC, particularly in uncertain, time-varying, or partially

known environments [5]. To address these limitations, recent advances in machine learning have enabled data-driven modeling approaches that can capture complex and uncertain system dynamics directly from observations. When integrated with MPC, learning-based models provide enhanced flexibility and modeling fidelity, particularly in scenarios where accurate first-principles models are difficult to obtain.

Physics-based models demonstrate excellent generalization capabilities but rely heavily on full-state environmental information, making them challenging to implement for complex dynamic systems [6,7]. Recent advancements have leveraged neural networks (NNs), celebrated for their ability to capture intricate patterns and dynamics [8], to construct control-oriented models by learning system dynamics from observations [9–11]. The combination of data-driven modeling and MPC has demonstrated promising performance in a variety of systems, including aerial robots, robotic arms, and quadrupeds [12–14]. This paradigm not only enables greater flexibility in representing nonlinear and uncertain dynamics, but also reduces reliance on explicit system identification, thereby streamlining the controller design process. These advances have spurred a growing interest in unifying learning-based models with MPC frameworks, enabling adaptive and robust control strategies in dynamic environments.

Despite their remarkable expressive power, NNs are inherently characterized by high nonlinearity and redundant parameters, which pose significant challenges for their efficient and accurate integration into model-based control frameworks, especially in systems with high complexity [15,16]. Model compression has demonstrated great potential in the training of neural networks [17–19], particularly in balancing model size and performance [20,21]. This trade-off between accuracy and compactness can have a significant impact on downstream tasks such as optimization. A growing body of research on network pruning and structured architecture search suggests that learning an over-parameterized model followed by pruning yields better performance than directly learning a compact network [22]. While some works have focused on improving the predictive accuracy and robustness of NN-based dynamic models [23–26], most existing NN-based MPC studies have paid limited attention to the structural optimization of the models themselves. This work investigates an integrated framework that combines structured neural network pruning with control-oriented regularization within an NMPC pipeline. By incorporating both model sparsity and regularization methods, the proposed approach enables efficient and control-aware dynamic modeling for computationally efficient trajectory tracking suitable for NMPC execution.

1.2. Related Work

A prominent line of research in learning-based MPC focuses on learning system dynamics from data, where neural networks serve as flexible approximators for nonlinear systems that are difficult to model analytically. The primary objective of this class of methods is to improve modeling accuracy and flexibility, thereby enabling model-based control in systems with strong nonlinearities or incomplete physical knowledge. Such learned models have been embedded into MPC to enable effective closed-loop control across diverse robotic platforms, ranging from soft robots [27] to aggressive vehicle control near handling limits under varying friction conditions [28]. These studies illustrate the general applicability of neural dynamics models within MPC frameworks across diverse robotic systems.

Another important research direction aims to incorporate learning into MPC while explicitly addressing robustness and safety. These approaches typically focus on handling model uncertainty, external disturbances, or guaranteeing constraint satisfaction through robust or uncertainty-aware control formulations. Gaussian-process-based MPC has been

widely studied in this context, as Gaussian process (GP) models naturally provide uncertainty estimates that can be exploited for robust or chance-constrained MPC designs [29]. Other works combine learning with tube-based MPC, adaptive MPC, or online uncertainty bounds to ensure closed-loop stability [30]. While these methods offer strong theoretical guarantees, they often incur substantial computational cost due to uncertainty propagation, conservative performance, or complex optimization formulations.

Reinforcement learning has also been combined with MPC to enhance control performance. In this paradigm, learning is typically used to approximate value functions, generate warm-starts, or provide high-level guidance for optimization-based controllers. Several studies have shown that RL-assisted MPC can achieve improved long-horizon performance or adapt to complex environments that are difficult to model explicitly [31,32]. However, these approaches often require extensive training data and may suffer from limited interpretability or reduced robustness guarantees compared to classical MPC formulations, which limits their adoption in safety-critical control tasks.

In contrast to the above approaches, this work focuses on improving the computational efficiency and numerical reliability of learning-based NMPC by optimizing the structure of the learned dynamics model itself. While recent studies have proposed diverse learning-assisted MPC frameworks and advanced solver implementations, they typically focus on expanding control formulations or incorporating uncertainty-aware mechanisms. The present work instead concentrates on the structural and numerical characteristics of neural dynamics models when embedded in gradient-based NMPC solvers. Therefore, our investigation emphasizes solver-aware model design and performance-oriented evaluation within a consistent NMPC formulation rather than cross-paradigm benchmarking across heterogeneous learning-based MPC strategies.

Rather than introducing additional uncertainty handling mechanisms or auxiliary learning modules, we retain a standard NMPC formulation and reduce its computational burden through structured neural network pruning and control-oriented regularization. By co-designing model sparsification with NMPC deployment, the proposed approach improves optimization efficiency while preserving tracking accuracy under identical modeling assumptions. This motivates a control-oriented model sparsification strategy that targets both compactness and numerical reliability for gradient-based NMPC solvers. Based on the discussion above, the main contributions of this paper are summarized as follows:

- We propose a control-oriented neural dynamics modeling pipeline for a fixed-wing UAV, which combines structured neuron-level pruning with robustness- and smoothness-promoting fine-tuning to obtain an NMPC-friendly predictor.
- We embed the pruned neural dynamics model into a standard NMPC framework for closed-loop trajectory tracking, where the learned model is used exclusively for multi-step prediction.
- We conduct ablation and comparative simulation studies to quantify the trade-offs between sparsification, solve time, and tracking accuracy using MAE/RMSE metrics.

This paper is structured as follows: Section 2 introduces the mathematical model of the fixed-wing UAV and discusses data-driven dynamics modeling from observations. Section 3 presents the learning-based dynamics model construction using a structured pruning strategy. It includes the iterative pruning–retraining pipeline, adversarial-aware importance scoring, and network regularization techniques that promote robustness and smoothness for controller integration. Section 4 details the design of the model predictive controller for UAV trajectory tracking, including the formulation of the control objectives and constraints. Section 5 provides simulation results and comparative studies to validate the effectiveness and computational benefits of the proposed approach. Finally, Section 6 concludes the paper and discusses potential directions for future research.

1.3. Notation

In this paper, scalars are denoted by lowercase italic letters (e.g., s), vectors by lowercase boldface letters (e.g., $\boldsymbol{v}$), and matrices by uppercase boldface letters (e.g., $\mathbf{M}$). $\mathbb{N}$, $\mathbb{R}$ represent all non-negative integers and real space, respectively. $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$ denote the n-dimensional real space and real matrix space of size $m \times n$, respectively. $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the L_1-norm and Euclidean norm, respectively. $\mathrm{diag}(\cdot)$ creates a diagonal matrix from a vector. $\|\boldsymbol{a}\|_{\boldsymbol{M}}^2 = \boldsymbol{a}^T\boldsymbol{M}\boldsymbol{a}$ denotes the weighted squared norm.

2. UAV Model Structure

This section presents the modeling framework used in this work. A nonlinear fixed-wing UAV model derived from first principles is introduced and used as a simulator to generate offline training data. Based on these data, a neural network is constructed to approximate translational and angular accelerations, while the kinematic relations are preserved in analytical form. Then we describe our methods for learning a dynamics model using environmental observations.

2.1. Fixed-Wing UAV Mathematical Model

A standard six-degree-of-freedom (6-DoF) rigid-body model is adopted to describe the motion of the fixed-wing UAV, incorporating several common simplifying assumptions [33,34]. The coordinate system and the forces and moments acting on the unmanned aerial vehicle are shown in Figure 1.

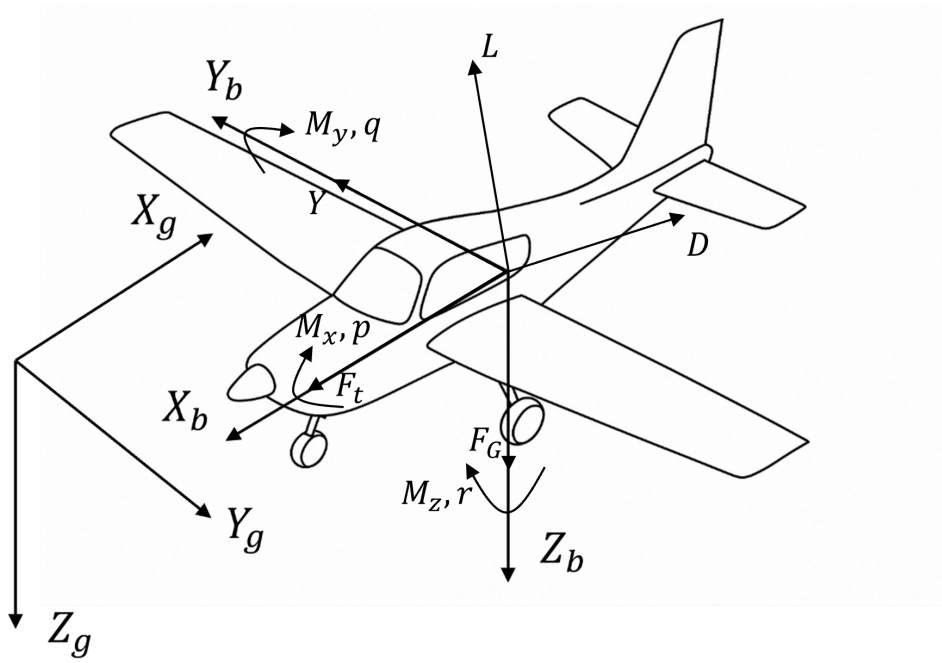

Figure 1. Schematic diagram of the coordinate system of fixed-wing UAV, including the forces and moments.

The UAV is modeled as a rigid body subjected to gravitational, aerodynamic, and thrust forces and moments. The Earth is assumed to be flat and fixed in an inertial frame, with a constant gravitational acceleration. The mass m of the UAV is considered constant, and due to the geometric symmetry in the x-z plane, the products of inertia I_{xy} and I_{yz} are neglected. In addition, the thrust T is assumed to be aligned with the body's longitudinal axis, and rotational effects induced by the propulsion system are ignored.

The dynamics of the fixed-wing UAV model in state-space form are described in twelve coordinates as

$$\boldsymbol{x} = [x, y, z, u, v, w, \phi, \theta, \psi, p, q, r]^T, \tag{1}$$

$$\boldsymbol{u} = [\delta_a, \delta_e, \delta_r, \delta_t]^T, \tag{2}$$

where $\boldsymbol{\xi} = [x, y, z]^T \in \mathbb{R}^3$ denotes position in the North–East–Down inertial frame $\boldsymbol{\Gamma_I}$, $\boldsymbol{V} = [u, v, w]^T \in \mathbb{R}^3$ denotes velocity in the body frame $\boldsymbol{\Gamma_B}$. $\boldsymbol{\Omega} = [\phi, \theta, \psi]^T \in \mathbb{R}^3$ denotes Euler angles in roll, pitch, and yaw axes, respectively, and $\boldsymbol{\omega} = [p, q, r]^T \in \mathbb{R}^3$ denotes angular velocities in the body frame $\boldsymbol{\Gamma_B}$, respectively. $\boldsymbol{u} = [\delta_a, \delta_e, \delta_r, \delta_t] \in \mathbb{R}^4$ represents aileron, elevator, rudder, and throttle percentage, which are the control inputs of the UAV. The general equations of motion and dynamics can be expressed as:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{R}_{\mathcal{BI}} \boldsymbol{V}, \tag{3}$$

$$\dot{\boldsymbol{V}} = \boldsymbol{F}_{\mathcal{B}}/m - \boldsymbol{\omega} \times \boldsymbol{V}, \tag{4}$$

$$\dot{\boldsymbol{\Omega}} = \boldsymbol{R}_{\mathcal{BW}} \boldsymbol{\omega}, \tag{5}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(\boldsymbol{M}_{\mathcal{B}} - \boldsymbol{\omega} \times (\boldsymbol{J}\boldsymbol{\omega})), \tag{6}$$

where m is the mass of the UAV, J is the inertia matrix, $\boldsymbol{R}_{\mathcal{BI}}$ is the rotation matrix mapping vectors from the body frame Γ_B to the inertial frame Γ_I, and $\boldsymbol{R}_{\mathcal{BW}}$ is the attitude rate transformation matrix, their expressions are given as follows, based on the definitions in [35]:

$$\boldsymbol{J} = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{xz} & 0 & I_{zz} \end{bmatrix}, \tag{7}$$

$$\mathbf{R}_{\mathcal{BI}} = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}, \tag{8}$$

$$\mathbf{R}_{\mathcal{BW}} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix}, \tag{9}$$

where $C. = cos(\cdot)$, $S. = sin(\cdot)$, $T. = tan(\cdot)$. For a UAV, $\boldsymbol{F}_{\mathcal{B}}$ represents the total forces acting on the body frame $\boldsymbol{\Gamma_B}$, which can be classified into thrust $\boldsymbol{F}_t$, gravity $\boldsymbol{F}_g$, and aerodynamic forces $\boldsymbol{F}_a$, as shown in Equation (10). The same is true for the moment $\boldsymbol{M}_{\mathcal{B}}$, but it is assumed to act around the center of gravity, thus being independent of gravity, as shown in Equation (11).

$$\boldsymbol{F}_{\mathcal{B}} = \boldsymbol{F}_t + \boldsymbol{F}_g + \boldsymbol{F}_a, \tag{10}$$

$$\boldsymbol{M}_{\mathcal{B}} = \boldsymbol{M}_t + \boldsymbol{M}_a, \tag{11}$$

where the aerodynamic force $\boldsymbol{F}_a = [F_x, F_y, F_z]^T$ and aerodynamic moment $\boldsymbol{M}_a = [M_x, M_y, M_z]^T$ are both expressed in the body-fixed coordinate frame Γ_B. The components $F_x = T + Lsin\alpha - Ycos\alpha sin\beta - Dcos\alpha cos\beta$; $F_y = Ycos\beta - Dsin\beta$; $F_z = -Lcos\alpha - Ysin\alpha sin\beta - Dsin\alpha cos\beta$; L, Y, D are the aerodynamic forces decomposed in the airflow coordinate system, respectively. α is the angle of attack and β is the side slip angle. The aerodynamic forces and moments are given by:

$$\begin{cases} L = QS_{\text{ref}} C_{L,\sigma}, \\ Y = QS_{\text{ref}} C_{Y,\sigma}, \\ D = QS_{\text{ref}} C_{D,\sigma}, \end{cases} \tag{12}$$

$$\begin{cases} M_x = QS_{\text{ref}}L_{\text{ref}}C_l, \\ M_y = QS_{\text{ref}}L_{\text{ref}}C_m, \\ M_z = QS_{\text{ref}}L_{\text{ref}}C_n, \end{cases} \tag{13}$$

where Q is the dynamic pressure, S_{ref} and L_{ref} are the reference area and reference length, and C_L, C_Y, C_D, C_l, C_m and C_n are the aerodynamic force and moment coefficients. These aerodynamic coefficients are nonlinear functions of several flight variables, including the angle of attack α, side slip angle β, body angular rates p, q, r, airspeed, and control inputs elevator δ_e, aileron δ_a, rudder δ_r, and throttle percentage δ_t . Consider the nonlinear control system of the UAV as:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{14}$$

where $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ denotes a mapping given by Equations (3)–(6). While the physics-based model in Equation (14) provides a structured and interpretable representation of UAV dynamics, the exact formulations of the aerodynamic coefficients involved are often difficult to derive analytically, which raises difficulties for the design of efficient controllers. To improve adaptability and capture the system's unmodeled or uncertain behaviors, we learn an approximate dynamics component from data using a neural network while retaining the analytical kinematic relations.

2.2. Offline Training Data Generation

The neural network dynamics model is trained using offline flight data generated in a simulation environment based on a predefined nonlinear fixed-wing UAV dynamics model [36]. The training data are designed to cover the operating conditions encountered in the subsequent NMPC evaluation, while also extending beyond the nominal reference trajectory to enhance coverage within the considered flight envelope. Specifically, flight trajectories are generated by executing NMPC-based trajectory tracking tasks under diverse reference motions, including but not limited to spiral trajectories, with variations in airspeed and attitude. Additive process and measurement noise are introduced during simulation to emulate modeling uncertainties and sensor imperfections.

The raw data are collected as discrete-time sequences with a fixed sampling interval $\Delta t = 0.02$ s, and zero-mean Gaussian noise is added to both states and measurements with standard deviations chosen as 5% of the nominal signal magnitudes. Each sample consists of the system state $\boldsymbol{x}_k$, control input $\boldsymbol{u}_k$, and learning targets $\dot{\boldsymbol{x}}_k$, which are obtained analytically from the predefined dynamics model. Prior to training, all state, input, and target variables are normalized using feature-wise min–max scaling. The normalization is performed independently for each feature dimension using statistics computed from the training dataset only, and the same scaling parameters are reused during validation and closed-loop evaluation. The same scaling parameters are applied during validation and closed-loop NMPC evaluation.

To further characterize the coverage of the training data, the empirical distributions of representative state variables are visualized using kernel density estimation, as shown in Figure 2. These distributions illustrate the range of operating conditions captured in the dataset, rather than implying generalization beyond the considered flight envelope, which is sufficient for the closed-loop NMPC evaluations considered in this work.

Each row separately displays the translational velocities (u, v, w), the attitude angles (ϕ, θ, ψ), and the angular rates (p, q, r). For the translational velocities (u, v, w), the distributions exhibit distinct peaks corresponding to the dominant forward motion and lateral/vertical velocity variations required to track spiral reference trajectories. The comparatively narrow distribution of w indicates that vertical motion remains bounded around nominal climb and descent rates. The roll angle ϕ and pitch angle θ concentrate around

moderate values to maintain maneuverability and stability, while the yaw angle ψ exhibits a broader range due to its unwrapped representation and continuous heading changes along spiral paths. The multi-modal distribution of yaw angle rate r reflects transitions between steady coordinated flight segments and maneuvering phases during spiral tracking.

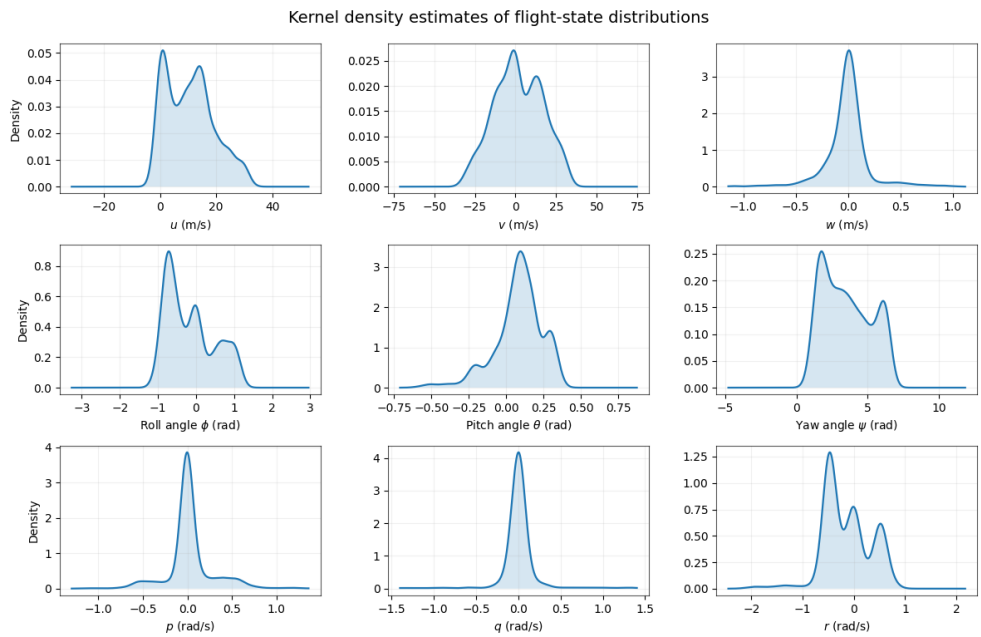

Figure 2. Kernel density estimates (KDEs) of the distributions of key flight states aggregated over all trajectories. The yaw angle ψ is shown in its unwrapped form.

2.3. Learning Neural Dynamics from Observations

In this work, the system dynamics are first described by the physics-based model in Equation (14), which is used as a data-generating process. Based on the resulting state-input-derivative tuples, a neural network is trained to approximate the continuous-time dynamics. Consider the training dataset $\mathcal{D} = \{(\boldsymbol{x}_t^m, \boldsymbol{u}_t^m, \dot{\boldsymbol{x}}_t^m) \mid t = 1, \ldots, T,\ m = 1, \ldots, M\}$ collected via interactions with the environment, where $\boldsymbol{x}_t^m$ and $\boldsymbol{u}_t^m$ denote state and control input obtained at time t in trajectory m. $\dot{\boldsymbol{x}}_t^m$ denotes the state derivative obtained analytically from the predefined dynamics model. The learning objective is to approximate the underlying system dynamics by training a parametric model $\hat{f}_\theta(\boldsymbol{x}, \boldsymbol{u})$:

$$\hat{\dot{\boldsymbol{x}}}_t^m = \hat{f}_\Theta(\boldsymbol{x}_t^m, \boldsymbol{u}_t^m), \tag{15}$$

where the network parameters Θ are trained to minimize the MSE loss between predicted acceleration and observation. The MSE between predicted and observed state derivatives is shown as:

$$\mathcal{L}_{\text{sup}}(\Theta) = \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} \left\| \hat{f}_\Theta(\boldsymbol{x}_t^m, \boldsymbol{u}_t^m) - \dot{\boldsymbol{x}}_t^m \right\|_2^2, \tag{16}$$

In this work, the neural network aims to approximate the dynamics of a fixed-wing UAV by learning to predict both translational and rotational accelerations. Specifically, the output of the neural network consists of the translational accelerations and angular accelerations $\hat{\boldsymbol{a}} = [\dot{u}_{NN}, \dot{v}_{NN}, \dot{w}_{NN}, \dot{p}_{NN}, \dot{q}_{NN}, \dot{r}_{NN}]^T$. The network receives as input the current state and control variables, which include linear and angular velocity $[u, v, w, p, q, r]$, Euler angles $[\phi, \theta, \psi]$, control surface deflections and throttle percentage $[\delta_a, \delta_e, \delta_r, \delta_t]$. The neural network function can be denoted as:

$$\hat{\boldsymbol{a}} = f_\Theta^{NN}(u, v, w, \phi, \theta, \psi, p, q, r, \delta_a, \delta_e, \delta_r, \delta_t), \tag{17}$$

To enable efficient integration into NMPC frameworks, we further compress the learned model by applying structured pruning techniques. The detailed design and implementation of the pruning-based network optimization are presented in the next chapter.

3. Neural Network Modeling

After the initial training phase, neural networks often exhibit significant redundancy, with many neurons and connections contributing little to the overall model performance [37], which leads to a computational burden in the controller design. To enable efficient integration of the learned model into NMPC frameworks, it is essential to develop a lightweight neural network architecture. This chapter presents a data-driven modeling framework based on neural networks. During the training process of the neural networks, a structured pruning strategy is adopted to systematically reduce redundant neurons and their connections, resulting in a compact and computationally efficient model without significant loss of accuracy. In contrast to unstructured pruning methods that often lead to irregular sparsity and inefficient hardware execution, our structured approach targets entire neurons, thus ensuring efficient inference. Figure 3 shows a schematic diagram of the network pruning training.

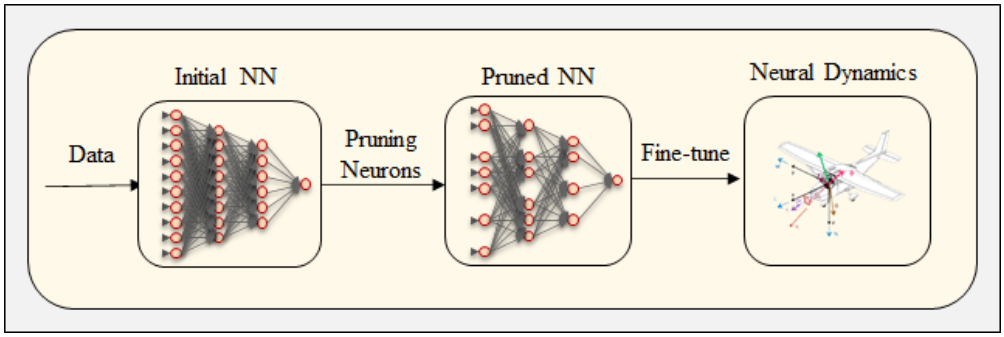

Figure 3. Overview diagram of the proposed neural network modeling framework.

3.1. Structure Pruning

Structured pruning aims to identify, for each layer ℓ, a subset of neurons $\boldsymbol{S}'_\ell = \{s'_{\ell,1}, \ldots, s'_{\ell,n_\ell}\} \subset \boldsymbol{S}_\ell = \{s_{\ell,1}, \ldots, s_{\ell,n_\ell}\}$ such that under a layer-wise pruning ratio r_ℓ, the resulting network achieves minimal performance degradation while maximizing computational acceleration [15].

Our approach follows the widely used pruning-after-training (PAT) paradigm [38], in which a dense fully connected neural network (FCNN) is first pretrained and gradually prunes unimportant neurons while retaining the model's accuracy, yielding a final sparse neural network $f(\boldsymbol{x}, \boldsymbol{W}_F)$. Specifically, a dense FCNN $f(\boldsymbol{x}, \boldsymbol{W}_0)$ is first trained using supervised learning where neurons are ranked based on importance scores that combine magnitude-based and adversarial sensitivity criteria. A structured pruning strategy is applied to remove the least important neurons, resulting in a sparse intermediate network $f(\boldsymbol{x}, \boldsymbol{M}_i)$. To recover the performance loss caused by pruning, we fine-tune the remaining network weights with a reduced number of training epochs [39]. This pruning–fine-tuning cycle is repeated iteratively, progressively increasing the sparsity until a target compression ratio is reached [39,40]. The final pruned model, denoted as $f(\boldsymbol{x}, \boldsymbol{W}_F)$, integrates both efficient structure and retrained parameters optimized for the compressed architecture. The overall structured pruning workflow is illustrated in Figure 4.

To guide the pruning process, we assign an importance score to each neuron based on two complementary criteria: the magnitude of its associated weights and its sensitivity to input perturbations. For the former, we used the L1-norm of weights to measure the magnitude of each neuron's outgoing weights, indicating the contribution to the feature

transformation. For the latter, the adversarial sensitivity is adopted as a complementary criterion to measure each neuron's sensitivity to perturbations in the inputs, which is calculated via gradient backpropagation under adversarial inputs [41]. These scores are designed to identify important structures in the network, thereby determining which redundant parts need to be eliminated.

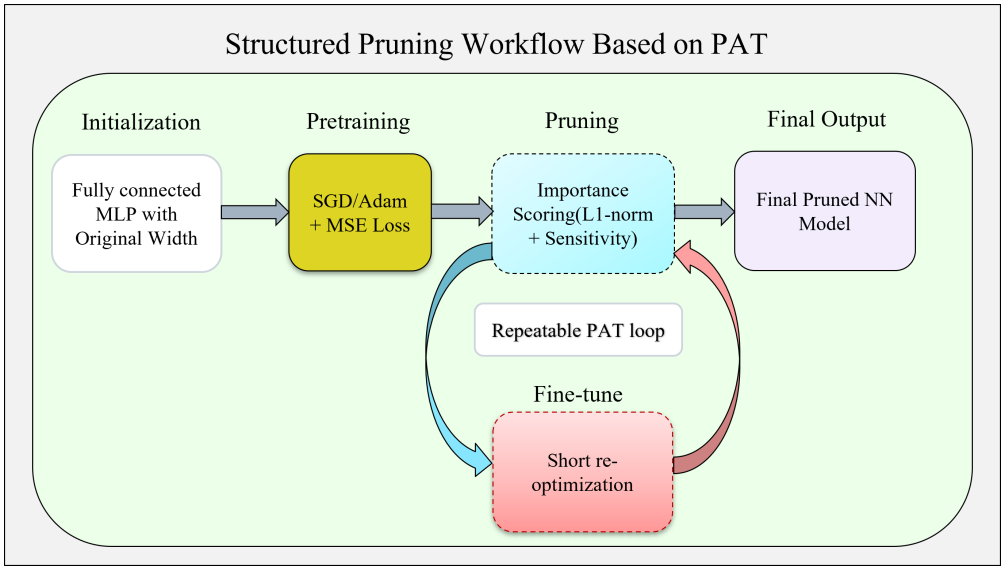

Figure 4. Structured pruning process based on neuron-level L1-norm and adversarial sensitivity scores.

The L1-norm can directly measure the activation intensity of neurons and has been widely used to characterize their structural importance [37,42]. For the i-th neuron in layer l, the L1-norm of its outgoing weights is defined as:

$$a_i^{(l)} = \|\boldsymbol{W}_i^{(l)}\|_1 = \sum_{j=1}^{d} |w_{ij}|, \tag{18}$$

where $\boldsymbol{W}_i \in \mathbb{R}^d$ denotes the weight vector from neuron i to its d downstream units, and w_{ij} is the scalar weight on the j-th connection. Neurons with smaller L1-norm values are typically regarded as contributing less to the forward signal propagation and are therefore candidates for pruning.

To further capture the robustness-related importance of each neuron, we introduce a gradient-based adversarial sensitivity metric. Given an input–target pair $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, an adversarial sample $\boldsymbol{x}_k^{\text{adv}}$ is generated using the Fast Gradient Sign Method (FGSM) [41]:

$$\boldsymbol{x}_k^{\text{adv}} = \boldsymbol{x}_k + \epsilon \cdot \text{sign}\big(\nabla_{\boldsymbol{x}_k} \mathcal{L}_{\text{sup}}(f_\Theta(\boldsymbol{x}_k), \boldsymbol{y}_k)\big), \tag{19}$$

where ϵ denotes the perturbation magnitude, and $\mathcal{L}_{\text{sup}}(\cdot)$ is the supervised loss function. The adversarial sensitivity of neuron i in layer l is then defined as:

$$s_i^{(l)} = \frac{1}{N} \sum_{k=1}^{N} \left\| \nabla_{\boldsymbol{w}_i^{(l)}} \mathcal{L}_{\text{sup}}\left(f_\Theta(\boldsymbol{x}_k^{\text{adv}}), \boldsymbol{y}_k\right) \right\|_1, \tag{20}$$

where $\boldsymbol{w}_i^{(l)}$ denotes the outgoing weights of neuron i.

Since the L1-norm and adversarial sensitivity may exhibit different numerical scales, we apply *layer-wise normalization* to both quantities prior to aggregation. Specifically,

for each layer l, the normalized structural importance and sensitivity scores are computed as:

$$\tilde{a}_i^{(l)} = \frac{a_i^{(l)} - \min_j a_j^{(l)}}{\max_j a_j^{(l)} - \min_j a_j^{(l)}}, \quad \tilde{s}_i^{(l)} = \frac{s_i^{(l)} - \min_j s_j^{(l)}}{\max_j s_j^{(l)} - \min_j s_j^{(l)}}, \tag{21}$$

This normalization ensures that both terms are dimensionless and comparable within each layer. The final importance score is defined as a weighted combination:

$$\text{score}_i^{(l)} = \zeta\, \tilde{a}_i^{(l)} + (1 - \zeta)\, \tilde{s}_i^{(l)}, \tag{22}$$

where $\zeta \in [0, 1]$ controls the trade-off between structural sparsity and robustness.

At each pruning iteration, a fixed fraction r of neurons with the lowest importance scores is removed. Here, r denotes a *per-iteration pruning ratio* that controls the pruning granularity rather than the final sparsity level. After pruning, the remaining network is fine-tuned through short re-optimization cycles. This prune–retrain procedure is repeated until a predefined target sparsity level is reached, which is specified as the ratio between the number of remaining neurons and that of the original network. In this work, the target sparsity level is achieved through a fixed number of pruning iterations with predefined per-iteration pruning ratios, resulting in a predictable overall sparsity.

3.2. Regularization Method

While pruning improves inference efficiency by removing redundant structures, it may compromise the smoothness and robustness of the learned dynamics model—for instance, making the output excessively sensitive to small input variations, which, in turn, can negatively affect the performance of gradient-based optimization methods, such as those used in NMPC. Although fine-tuning can partially recover the performance of the model, it fails to fully compensate for the structural degradation introduced by pruning [20]. To mitigate these issues, we incorporate two robustness-oriented strategies during the fine-tuning phase of training: spectral norm constraints [43] and gradient regularization [44].

Lipschitz continuity is a desirable property for neural networks, as it ensures that the model output maintains stability and smoothness in the presence of input disturbances without significantly affecting performance [45]. Spectral norm regularization is a widely used technique to enforce Lipschitz continuity, which bounds the output variation with respect to input perturbations and ensures output stability [43]. For a feedforward neural network composed of linear layers and Lipschitz-continuous activation functions, the overall Lipschitz constant of the network is upper-bounded by the product of the spectral norms of each layer's weight matrix [46]. The Lipschitz constant of a linear layer is upper-bounded by the spectral norm of the weight matrix $\boldsymbol{W}$, defined as the largest singular value:

$$\|\boldsymbol{W}\|_2 = \sigma_{\max}(\boldsymbol{W}) = \max_{x \neq 0} \frac{||\boldsymbol{W}\boldsymbol{x}||_2}{||\boldsymbol{x}||_2}, \tag{23}$$

To ensure that each layer adheres to a desired Lipschitz bound $\gamma > 0$, we normalize the weights during training by applying spectral normalization:

$$\overline{\boldsymbol{W}} = \frac{\boldsymbol{W}}{\max(1, \sigma_{\max}(\boldsymbol{W})/\gamma)}, \tag{24}$$

where $\overline{\boldsymbol{W}}$ is the normalized weight matrix. This rescaling prevents the layer-wise Lipschitz constants from exceeding γ, thereby constraining the overall sensitivity of the network and improving robustness to input noise.

While spectral normalization constrains the global Lipschitz continuity of the network by bounding the spectral norm of each layer's weight matrix, it does not directly suppress

large local gradients in regions of high sensitivity. To enhance the local smoothness of the learned dynamics model, we additionally adopt a gradient regularization strategy, which penalizes the magnitude of the input gradient of the supervised loss function. This encourages the model to exhibit smoother input–output mappings and reduces sensitivity to small input variations [47]. The gradient regularization term is defined as:

$$\mathcal{L}_{\text{grad}}(\Theta) = \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} \left\| \nabla_{x_t^m} \mathcal{L}_{\text{sup}}(\Theta) \right\|_2^2, \tag{25}$$

where $\nabla_{x_i} \mathcal{L}_{\text{sup}}(\Theta)$ denotes the gradient of the supervised loss with respect to the input state x_t^m, and $\mathcal{L}_{\text{sup}}$ is the MSE loss defined in Equation (16). The total loss used for fine-tuning is then augmented with the regularization term:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sup}} + \lambda_{\text{grad}} \cdot \mathcal{L}_{\text{grad}}, \tag{26}$$

where λ_{grad} is a positive scalar hyperparameter that balances the influence of gradient regularization.

These regularization strategies, including spectral norm constraints and gradient regularization, collectively enhance the generalization capability and robustness of the pruned dynamics model. This not only compensates for potential performance degradation caused by pruning, but also improves stability in closed-loop control scenarios—providing a solid foundation for efficient NMPC implementation. The overall training and pruning pipeline is summarized in Algorithm 1.

Algorithm 1 Structured Pruning with Fine-Tuning

Require: Training data $\mathcal{D}$, initial weights $\boldsymbol{W}_0$, layer-wise pruning ratios $\{r^{(l)}\}_{l=1}^{L}$, sensitivity weighting factor ζ, perturbation magnitude ϵ, Lipschitz bound γ, gradient regularization weight λ_{grad}, maximum pruning iterations I

Ensure: Final pruned model $f(\boldsymbol{x}; \boldsymbol{W}_F)$

```
// Step 1: Pretraining
Train dense network f(x; W_0) on D with MSE loss to obtain W_0
Initialize iter ← 1
while iteration iter ≤ I do
    for each layer l = 1, 2, ..., L do
        Compute L1-norm ||W_i^(l)||_1 based on (18) for each neuron i in layer l
        Generate adversarial input based on (19)
        Compute adversarial sensitivity based on (20)
        Compute the combined importance score based on (22)
        Prune bottom r^(l)% of neurons in layer l with lowest scores
    end for
    // Step 2: Fine-tuning: Retrain the remaining network on D with fewer epochs using:
    Spectral normalization: ||W^(l)||_2 ≤ γ based on (24)
    Compute gradient regularization λ_grad based on (25)
    Augment gradient regularization based on (26)
    Update iter ← iter + 1
end while
return Final pruned model f(x; W_F)
```

Table 1 summarizes the neural network architecture, training configuration, and pruning-related hyperparameters used in all experiments. Unless otherwise stated, these settings are kept fixed throughout the paper. They were chosen following standard practices for neural network regression and structured pruning, and were verified in preliminary trials to yield stable training and reliable closed-loop NMPC behavior.

Smooth activation functions (tanh) are adopted to ensure continuous differentiability of the learned dynamics model, which is essential for gradient-based NMPC solvers. In particular, tanh avoids the dead zones associated with ReLU-type activations, which can hinder solver convergence in practice. The dense network is pretrained for 500 epochs, and each pruned model is subsequently fine-tuned for 250 epochs. For neuron ranking, we set $\zeta = 0.5$ to balance the normalized weight-magnitude score and the adversarial sensitivity score in Equation (22). The pruning ratios are selected in a layer-wise and non-uniform manner. Specifically, we use $\{r^{(l)}\} = \{0.15, 0.25, 0.20\}$ for the three hidden layers. This schedule follows common heuristics in structured pruning: the first hidden layer is closest to the input and tends to play a feature-extraction role, so it is pruned more conservatively; the middle layer is typically more redundant and can tolerate a larger pruning ratio; the last hidden layer is relatively shallow and close to the output, and is therefore pruned moderately to avoid amplifying errors in the predicted accelerations. Our objective is not to fine-tune individual hyperparameters exhaustively, but to demonstrate that the proposed pruning and fine-tuning pipeline consistently yields compact and smooth dynamics models suitable for NMPC integration.

Table 1. Neural network training and pruning hyperparameters used throughout all experiments.

Parameter	Value
Hidden layers	128–64–64
Activation	tanh
Optimizer	Adam
Learning rate	1×10^{-3}
Batch size	64
Training epochs	500
Fine-tuning epochs	250
ζ	0.5
$r^{(l)}$	0.15, 0.25, 0.20
ϵ	10^{-2}
γ	3.0
λ_{grad}	0.05
Pruning iterations I	3

4. Nonlinear Model Predictive Controller Design

4.1. NMPC Formulation

This chapter presents the design of an NMPC framework for the proposed UAV system, aiming to achieve high-precision trajectory tracking. The controller explicitly handles physical constraints on both states and control inputs, and is implemented in a receding-horizon manner using the learned dynamics model as the internal predictor. Let $\boldsymbol{x} \in \mathbb{R}^{n_x}$ from Equation (1) denote the system state, and $\boldsymbol{u} \in \mathbb{R}^{n_u}$ from Equation (2) denote the control input. The goal is to steer the system along a time-varying reference trajectory $\{\boldsymbol{x}_{\text{ref},k}, \boldsymbol{u}_{\text{ref},k}\}_{k=0}^{H}$ over a prediction horizon of length H. Figure 5 illustrates the proposed NMPC framework with pruned neural dynamics.

Following a standard NMPC formulation, we define the cost function to penalize the deviation from the reference trajectory:

$$J(\boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_{i=k}^{k+H-1} \left(\left\| \boldsymbol{x}_{i|k} - \boldsymbol{x}_{\text{ref},i|k} \right\|_{\boldsymbol{Q}}^2 + \left\| \Delta \boldsymbol{u}_{i|k} \right\|_{\boldsymbol{R}}^2 \right) + \left\| \boldsymbol{x}_{k+H|k} - \boldsymbol{x}_{\text{ref},k+H|k} \right\|_{\boldsymbol{S}}^2, \tag{27}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}$ are positive definite weighting matrices to penalize the state tracking error and control input increments, and $\boldsymbol{S} \in \mathbb{R}^{n_x \times n_x}$ is introduced to encourage stabilizing behavior near the reference trajectory and to improve closed-loop

performance. Such a terminal cost is commonly adopted in NMPC formulations to promote practical stability under standard assumptions. The vectors $\boldsymbol{x}_{i|k}$ and $\boldsymbol{u}_{i|k}$ are the predicted states and control inputs over an H-step prediction at time step i. This formulation using control increments $\Delta\boldsymbol{u}_{i|k}$ is motivated by the desire to suppress control input chattering and improve numerical stability.

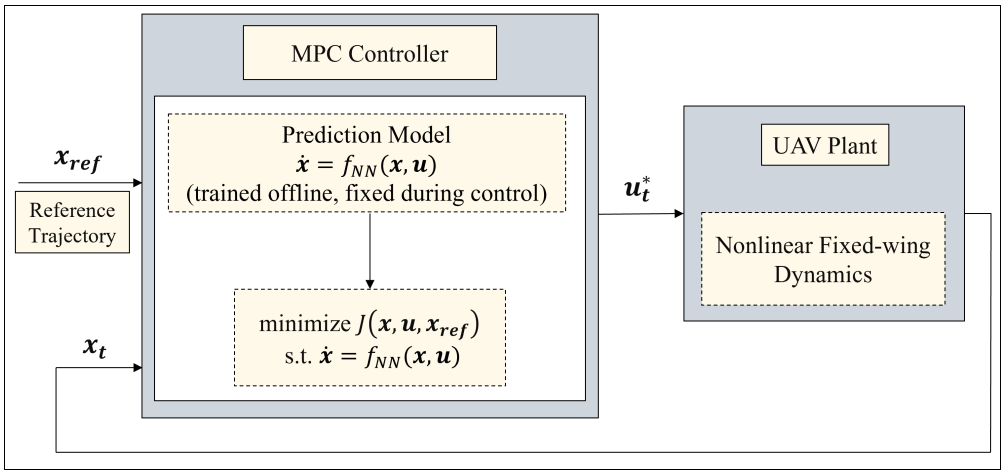

Figure 5. NMPC framework with a pruned neural dynamics model. The pruned neural network is embedded within the NMPC and used exclusively as an internal prediction model for multi-step state propagation. Reference trajectories are provided externally, and the neural dynamics model is trained offline and kept fixed during control execution.

The NMPC optimization problem at each time step k can then be formulated as:

$$\begin{aligned} \min_{\Delta\boldsymbol{u}_{k|k},\ldots,\Delta\boldsymbol{u}_{k+H-1|k}} \quad & J(\boldsymbol{x}_k, \boldsymbol{u}_k), \\ \text{s.t.} \quad & \Delta\boldsymbol{u}_{i|k} = \boldsymbol{u}_{i|k} - \boldsymbol{u}_{i-1|k}, \\ & \dot{\boldsymbol{x}}_{i|k} = f_{\mathrm{NN}}(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}), \quad i = k, \ldots, k+H-1, \\ & \boldsymbol{x}_{i|k} \in \mathcal{X}, \quad \boldsymbol{u}_{i|k} \in \mathcal{U}, \quad \forall i = k, \ldots, k+H-1, \\ & \boldsymbol{x}_{k|k} = \boldsymbol{x}_k, \end{aligned} \tag{28}$$

where $\mathcal{X}$ and $\mathcal{U}$ denote admissible state and control sets, $f_{\mathrm{NN}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ characterizes the state transition, which represents the sparse neural network dynamics model obtained via structured pruning in Section 3.

To construct a complete state-space model compatible with the NMPC framework, we integrate the neural network acceleration predictions from (3)–(6). This yields a hybrid model that combines analytically known kinematic equations with learned dynamics for acceleration prediction. The resulting continuous-time state transition model used in NMPC is given by:

$$\begin{cases} \dot{\boldsymbol{\xi}} = \boldsymbol{R}_{\mathcal{BI}}\boldsymbol{V}, \\ \dot{\boldsymbol{V}} = NN_{\boldsymbol{V}}(\boldsymbol{x}, \boldsymbol{u}), \\ \dot{\boldsymbol{\Omega}} = \boldsymbol{R}_{\mathcal{BW}}\boldsymbol{\omega}, \\ \dot{\boldsymbol{\omega}} = NN_{\boldsymbol{\omega}}(\boldsymbol{x}, \boldsymbol{u}), \end{cases} \tag{29}$$

where $NN_{\boldsymbol{V}}$ and $NN_{\boldsymbol{\omega}}$ denote the body-frame translational and angular accelerations predicted by the pruned neural network. This hybrid formulation ensures that essential geometric and kinematic constraints are explicitly preserved, while the unknown and possibly nonlinear force/torque dynamics are captured through learning. As a result, the constructed model is well-suited for multi-step state prediction in NMPC, balancing physical interpretability with model flexibility.

4.2. Stability Discussion

The proposed controller follows a standard NMPC paradigm, where a finite-horizon optimal control problem is solved repeatedly in a receding-horizon manner using a fixed prediction model. Rather than claiming new Lyapunov-based theoretical guarantees, the closed-loop stability properties of the resulting NMPC can be discussed under classical NMPC assumptions.

Specifically, when the prediction model provides a sufficiently accurate approximation of the system dynamics and the stage cost is positive definite with respect to the tracking error and control input increments, the finite-horizon cost function can be interpreted as a Lyapunov-like function for the closed-loop system. The inclusion of a terminal cost further promotes stabilizing behavior near the reference trajectory and helps mitigate horizon truncation effects. Under these commonly adopted conditions, standard NMPC theory provides qualitative insight that practical closed-loop stability may be expected under commonly adopted assumptions.

In the proposed framework, the neural network dynamics model is trained offline and subsequently pruned to enhance numerical reliability and smoothness, which helps reduce sensitivity to modeling errors and supports stable closed-loop behavior in practice. It is important to note that the neural dynamics model is not updated online during control execution, and the controller relies solely on state feedback from the UAV plant. A rigorous theoretical analysis of closed-loop robustness margins, recursive feasibility, and stability guarantees under bounded disturbances is beyond the scope of the present work and will be investigated in future research.

5. Simulation Results

In this section, we evaluate the effectiveness of the proposed MPC controller with a sparse neural dynamics model through a set of simulation studies. Specifically, we conduct two types of simulation-based evaluation: (1) an ablation study to investigate the impact of pruning and regularization strategies on the control performance, and (2) a comparative study against a conventional total energy control system (TECS) controller [48] and a local linearized MPC (LMPC) controller [49] to demonstrate the superiority of our neural-network-based model predictive control (NNMPC) framework. The simulation is carried out on a fixed-wing UAV model introduced in [50]. The reference trajectory is a three-dimensional spiral path, defined as:

$$\begin{cases} x_r(t) = Rcos(\omega_x t), \\ y_r(t) = Rsin(\omega_y t), \\ z_r(t) = v_z t, \end{cases} \tag{30}$$

where $R = 30\,\mathrm{m}$, $\omega_x = \omega_y = \pi/5\,\mathrm{rad/s}$, $v_z = 1\,\mathrm{m/s}$, and $t \in [0, 10]$. The NMPC is configured with a prediction horizon of $H = 10$ steps and a control interval of $\Delta t = 0.05\,\mathrm{s}$. The cost function is formulated to minimize the trajectory tracking error and control effort, with weight matrices defined as $\boldsymbol{Q} = \mathrm{diag}(10, 10, 10, 10, 10, 10, 1, 1, 10, 1, 1, 1)$ and $\boldsymbol{R} = \mathrm{diag}(1, 1, 1, 1)$. The state and control inputs are constrained within the physical limits of the UAV dynamics. The state constraints are $-50 < u < 50$, $-50 < v < 50$, $-50 < w < 50$, $-\pi/3 < \phi < \pi/3$, $-\pi/2 < \theta < \pi/2$, $-\pi/2 < p < \pi/2$, $-\pi/2 < q < \pi/2$, $-\pi/2 < r < \pi/2$, and the control input constraints are $-\pi/6 < \delta_a < \pi/6$, $-\pi/6 < \delta_e < \pi/6$, $-\pi/6 < \delta_r < \pi/6$, $0 < \delta_t < 100$.

All simulations were performed on a desktop computer equipped with an Intel Core i5-13600K CPU (13th Gen, 3.50 GHz) and 32 GB of RAM. All reported solve times are measured under single-threaded execution. The training of neural networks was executed in PyTorch

2.1.0 version with NVIDIA GeForce RTX 4060, and the NMPC was implemented using CasADi [51] 3.6.5 version and solved by the nonlinear programming solver IPOPT [52].

5.1. Ablation Study

To analyze the effectiveness of each component in the proposed pruned sparse neural dynamics model for NMPC, we conduct an ablation study involving three configurations. The first configuration, referred to as *Unpruned*, uses the original neural dynamics model without any pruning or regularization. The second configuration, *Pruned Only*, applies structured pruning based on a combined score of the L1-norm and input sensitivity, but does not include any regularization terms. The final configuration, *Pruned with Regularization*, represents our complete method, incorporating both pruning and additional regularization techniques, including spectral norm constraints and gradient regularization. To further investigate the effect of progressive pruning on the tracking performance, we additionally evaluate the intermediate models obtained after the first and second pruning stages, before reaching the final pruned configuration. These intermediate models, denoted as *Pruned (1st Stage)* and *Pruned (2nd Stage)*, retain more neurons than the final *Pruned Only* and *Pruned + Reg* configurations, allowing us to observe the trade-off between model compactness and tracking accuracy as pruning progresses.

The baseline neural dynamics model is constructed as a fully connected neural network with three hidden layers, consisting of 128, 64, and 64 neurons, respectively. The model is trained on a dataset containing 4800 state-control pairs sampled from various UAV trajectories. During the iterative pruning and retraining process, we perform three pruning stages followed by fine-tuning. The pruning ratios for the three hidden layers are set to 0.15, 0.25, and 0.20, respectively. The neuron importance score is computed as a weighted combination of the L1-norm and adversarial sensitivity, where the balancing coefficient is set to $\zeta = 0.5$ throughout all pruning iterations. The adversarial sensitivity is evaluated under a perturbation amplitude of $\epsilon = 0.01$. The spectral norm constraint is enforced with a Lipschitz bound $\gamma = 3$, and the gradient regularization coefficient λ_{grad} is scheduled using cosine annealing, with a maximum value of 0.05.

Figure 6 presents the 2D position tracking results on the x, y, and z axes for the three configurations, *Unpruned*, *Pruned Only*, *Pruned + Reg*, along with the reference trajectory. Figure 7 further visualizes the overall trajectory tracking performance in 3D space. Table 2 shows that the proposed pruning and regularization strategies significantly reduce the model size (45%) and computation time while improving tracking accuracy. The MAE and RMSE values are computed based on the per-axis position tracking error with respect to the reference trajectory. Figure 8 shows the control inputs and attitude response of the UAV under the proposed controller configuration, *Pruned + Reg*. The *Unpruned* model exhibits noticeable tracking bias and suffers from low computational efficiency, demonstrating that despite its full parameter capacity, the dense model remains poorly suited for NMPC applications. Notably, the first and second pruning stages (*Pruned (first Stage)* and *Pruned (second Stage)*) have no significant effect on the trajectory tracking, indicating that only a moderate reduction in model size has little impact on tracking performance and solution efficiency. The *Pruned Only* model improves tracking accuracy moderately, benefiting from the removal of redundant neurons, but still suffers from slight deviation. In contrast, the *Pruned with Regularization* configuration achieves the best overall performance, with a 19% reduction in solve time and improvements in tracking precision across all axes compared to the unpruned model. These results validate the effectiveness of both the structured pruning scheme and the stability-oriented regularization techniques in enhancing the NMPC framework's efficiency and accuracy.

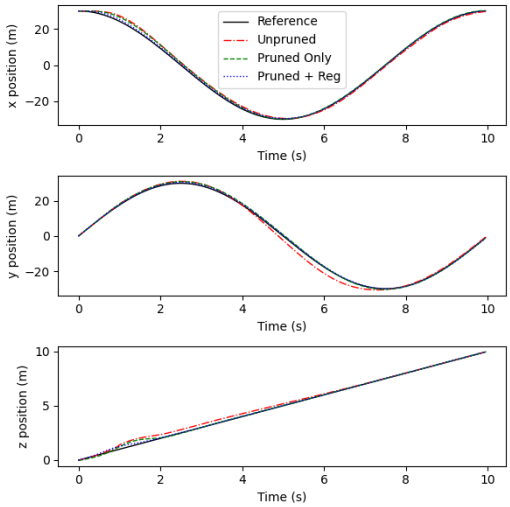

Figure 6. Tracking comparison on x, y, and z axes for different model configurations.

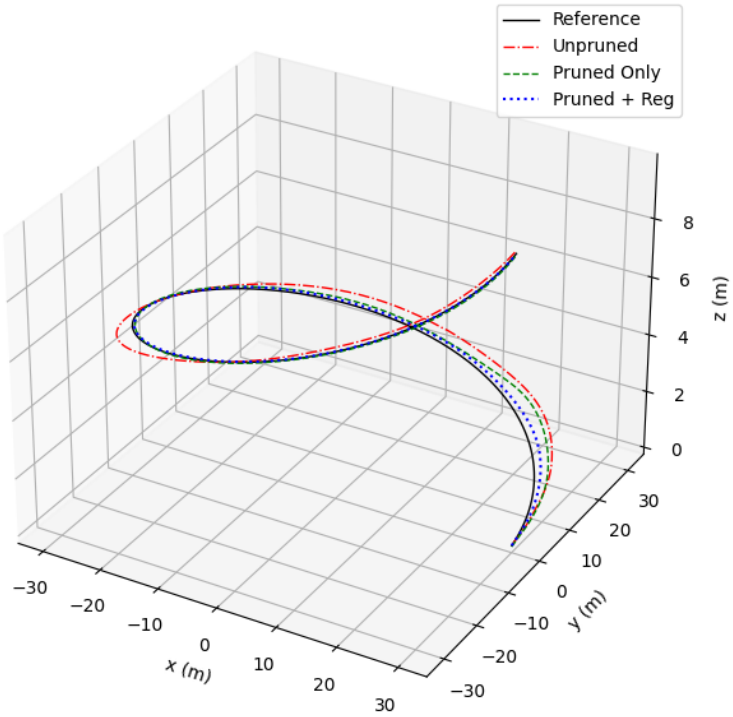

Figure 7. Three-dimensional trajectory tracking performance under different neural dynamics model configurations.

Table 2. Tracking error metrics (MAE/RMSE per axis), average NMPC solve time per update, and model size under different configurations.

Model Variant	Tracking Error [x/y/z] (m)	Avg. Time (ms)	Remaining Neurons
Unpruned	MAE: 1.140/1.159/0.104 RMSE: 1.387/1.574/0.225	31.6	128, 64, 64
Pruned 1st	MAE: 1.113/0.617/0.086 RMSE: 1.329/1.085/0.195	31.7	109, 48, 52
Pruned 2nd	MAE: 0.929/0.531/0.077 RMSE: 1.162/0.873/0.132	30.9	93, 36, 42
Pruned Only	MAE: 0.737/0.498/0.053 RMSE: 0.917/0.636/0.095	29.9	80, 27, 34
Pruned + Reg	MAE: 0.361/0.328/0.037 RMSE: 0.462/0.373/0.070	25.7	80, 27, 34

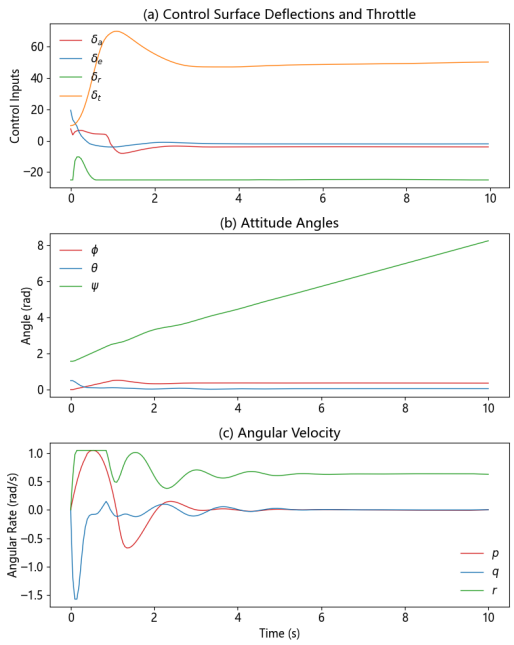

Figure 8. Control inputs and attitude response of the UAV under the proposed controller: (**a**) control surface deflections and throttle command, (**b**) attitude angles (roll, pitch, yaw), and (**c**) body angular rates. In subplot (**a**), δ_a, δ_e, and δ_r denote aileron, elevator, and rudder deflections in degrees (°), while δ_t represents throttle command in percentage (%).

5.2. Comparative Study

To further validate the effectiveness of the proposed NNMPC, we conduct a comparative study against two baselines: the TECS controller and the LMPC controller. The selected baselines (TECS and linear MPC) are chosen due to their practical relevance in fixed-wing flight control and their compatibility with the considered NMPC formulation. A broader comparison with alternative learning-based MPC frameworks would involve fundamentally different modeling assumptions and is beyond the scope of this study. For LMPC, the UAV model is linearized around the helical flight condition, and a quadratic program is solved at each step. The objective is to evaluate the tracking accuracy of different controllers on the same 3D spiral reference trajectory under identical UAV dynamics and initial conditions. To ensure fairness, all controllers are tested with the same simulation setup, including initial states, control frequency, prediction horizon, and simulation duration. The reference trajectory is the 3D spiral path defined in Equation (30), and the nonlinear fixed-wing UAV dynamics remain unchanged across all simulations.

Figure 9 illustrates the time-series tracking performance of the three controllers along the x, y, and z axes. Table 3 summarizes the quantitative tracking results in terms of per-axis MAE and RMSE, together with the average computation time per control update under identical simulation settings. The computation time for TECS is not reported, since it is a conventional feedback controller without online optimization. The TECS controller, serving as a representative baseline of classical energy-based flight control, is able to follow the reference trajectory but exhibits noticeable phase lag and bias, particularly in the lateral channels. The LMPC controller benefits from the computational efficiency of local linearization; however, the mismatch between the linearized model and the underlying nonlinear UAV dynamics results in degraded tracking accuracy, particularly along curved segments of the trajectory. In contrast, the proposed NNMPC achieves closer agreement with the reference trajectory across all axes, consistently demonstrating improved tracking precision and robustness. Figure 10 further illustrates the 3D tracking behavior of the controllers. Both the TECS and LMPC controllers exhibit visible offsets and accumulated

drift along the spiral trajectory, whereas the NNMPC-controlled UAV closely follows the reference trajectory with minimal deviation. These results underscore the advantage of integrating a sparse neural dynamics model within the NMPC framework, which enables a more accurate representation of nonlinear UAV dynamics, leading to significantly improved trajectory tracking performance compared to conventional baseline controllers.

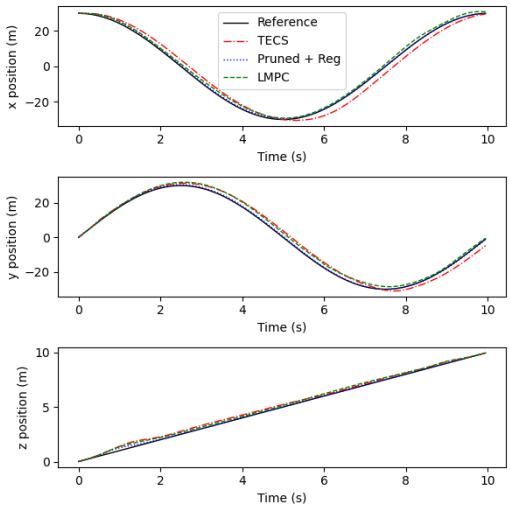

Figure 9. Tracking comparison along x, y, and z axes for NNMPC, LMPC, and TECS controllers.

Table 3. Tracking error metrics (MAE/RMSE per axis) and average computation time per update for different controllers under identical simulation settings.

Controller	MAE [x/y/z] (m)	RMSE [x/y/z] (m)	Avg. Time (ms)
TECS	2.654/2.100/0.195	2.947/2.424/0.226	–
LMPC	1.214/1.681/0.173	1.295/1.888/0.190	14.7
NNMPC (Ours)	0.361/0.328/0.037	0.462/0.373/0.070	25.7

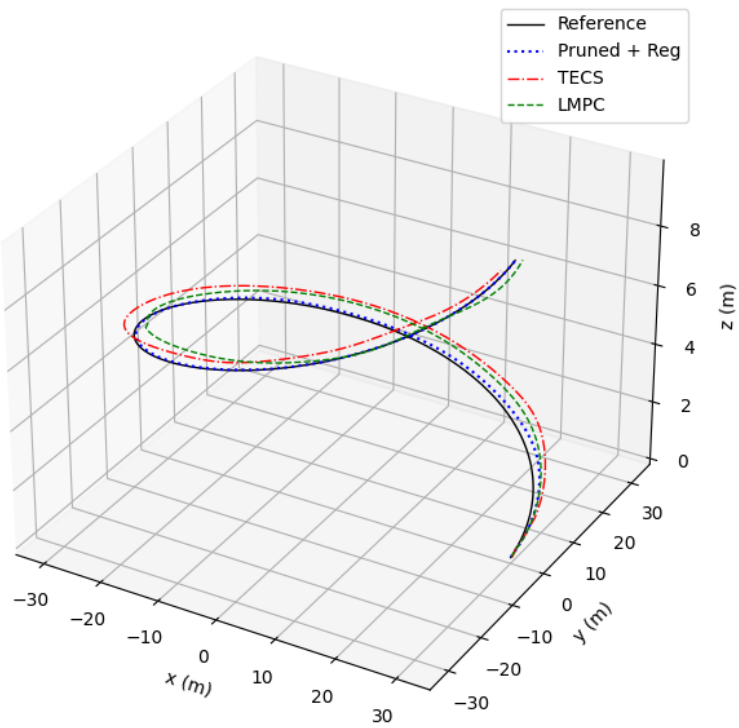

Figure 10. Three-dimensional trajectory tracking performance for NNMPC, LMPC, and TECS controllers.

As can be seen, the proposed NNMPC framework achieves consistently higher tracking accuracy across all dimensions, with a notable improvement in vertical tracking. In addition, the measured solver runtimes under the simulation settings indicate that the proposed controller can be executed at moderate update rates, where the dominant computational cost comes from the NMPC optimization rather than neural network inference. Overall, these results demonstrate the advantage of the proposed method in terms of tracking precision and closed-loop smoothness within the scope of simulation-based evaluation.

6. Conclusions

In this work, we presented a sparse neural dynamics-enhanced NMPC framework for fixed-wing UAV trajectory tracking. A structured pruning strategy was designed to iteratively remove redundant neurons based on a combined importance score of L1-norm and input sensitivity, followed by fine-tuning. To ensure stability and smooth optimization, spectral norm constraints and gradient regularization were further incorporated into the training. The resulting pruned neural dynamics model achieves both efficiency and accuracy, enabling effective integration into an NMPC. Extensive simulations validated the proposed approach. Ablation study highlighted the necessity of pruning and regularization strategies, demonstrating that the full pruned-regularized model achieved the lowest tracking error while maintaining reduced model complexity. A comparative study further showed that the NNMPC outperformed conventional baselines, including TECS and LMPC, by providing more accurate trajectory tracking on a 3D spiral path. These results confirm that sparse neural dynamics modeling is an effective enabler for improving NMPC performance in UAV applications.

Despite these promising results, the present study is limited to a simulation study. Practical challenges about onboard real-time implementation, robustness under disturbances, and adaptive parameter tuning remain open issues. Future work will aim to extend this framework to more complex scenarios such as obstacle-aware trajectory tracking, and to explore integration with robust control or reinforcement learning techniques for improved adaptability.

Author Contributions: Conceptualization, X.Q. and J.L.; methodology, X.Q.; software, X.Q.; validation, X.Q. and C.L.; formal analysis, X.Q.; investigation, X.Q.; resources, X.Q. and J.L.; data curation, X.Q.; writing—original draft preparation, X.Q.; writing—review and editing, X.Q. and J.L.; visualization, X.Q.; supervision, J.L.; project administration, J.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kerrigan, E.C. Predictive Control for Linear and Hybrid Systems [Bookshelf]. *IEEE Control Syst. Mag.* **2018**, *38*, 94–96. [CrossRef]
2. Wang, Z.; Tan, W.G.Y.; Rangaiah, G.P.; Wu, Z. Machine learning aided model predictive control with multi-objective optimization and multi-criteria decision making. *Comput. Chem. Eng.* **2023**, *179*, 108414. [CrossRef]
3. Ubbink, J.; Viljoen, R.; Aertbeliën, E.; Decré, W.; De Schutter, J. From Instantaneous to Predictive Control: A More Intuitive and Tunable MPC Formulation for Robot Manipulators. *IEEE Robot. Autom. Lett.* **2025**, *10*, 748–755. [CrossRef]
4. Mammarella, M.; Capello, E. A Robust MPC-based autopilot for mini UAVs. In *Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 1227–1235. [CrossRef]
5. Piga, D.; Forgione, M.; Formentin, S.; Bemporad, A. Performance-Oriented Model Learning for Data-Driven MPC Design. *IEEE Control Syst. Lett.* **2019**, *3*, 577–582. [CrossRef]

6. Zhou, J.; Hou, Y.; Mason, M.T. Pushing revisited: Differential flatness, trajectory planning, and stabilization. *Int. J. Robot. Res.* **2019**, *38*, 1477–1489. [CrossRef]
7. Hewing, L.; Wabersich, K.P.; Menner, M.; Zeilinger, M.N. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annu. Rev. Control. Robot. Auton. Syst.* **2020**, *3*, 269–296. [CrossRef]
8. Saviolo, A.; Frey, J.; Rathod, A.; Diehl, M.; Loianno, G. Active Learning of Discrete-Time Dynamics for Uncertainty-Aware Model Predictive Control. *IEEE Trans. Robot.* **2024**, *40*, 1273–1291. [CrossRef]
9. Crocetti, F.; Mao, J.; Saviolo, A.; Costante, G.; Loianno, G. GaPT: Gaussian Process Toolkit for Online Regression with Application to Learning Quadrotor Dynamics. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 11308–11314. [CrossRef]
10. Saviolo, A.; Li, G.; Loianno, G. Physics-Inspired Temporal Learning of Quadrotor Dynamics for Accurate Model Predictive Trajectory Tracking. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10256–10263. [CrossRef]
11. Mohajerin, N.; Waslander, S.L. Multistep Prediction of Dynamic Systems with Recurrent Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3370–3383. [CrossRef]
12. Bao, X.; Sun, Z.; Sharma, N. A recurrent neural network based MPC for a hybrid neuroprosthesis system. In *Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, Australia, 12–15 December 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 4715–4720. [CrossRef]
13. Kang, E.; Qiao, H.; Gao, J.; Yang, W. Neural network-based model predictive tracking control of an uncertain robotic manipulator with input constraints. *ISA Trans.* **2021**, *109*, 89–101. [CrossRef]
14. Desaraju, V.R.; Michael, N. Leveraging experience for computationally efficient adaptive nonlinear model predictive control. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 5314–5320. [CrossRef]
15. Cheng, H.; Zhang, M.; Shi, J.Q. A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 10558–10578. [CrossRef] [PubMed]
16. Salzmann, T.; Kaufmann, E.; Arrizabalaga, J.; Pavone, M.; Scaramuzza, D.; Ryll, M. Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms. *IEEE Robot. Autom. Lett.* **2023**, *8*, 2397–2404. [CrossRef]
17. Blalock, D.; Ortiz, J.J.G.; Frankle, J.; Guttag, J. What is the State of Neural Network Pruning? *arXiv* **2020**, arXiv:2003.03033. [CrossRef]
18. Elsken, T.; Metzen, J.H.; Hutter, F. Neural Architecture Search: A Survey. *arXiv* **2019**, arXiv:1808.05377. [CrossRef]
19. Zhou, A.; Ma, Y.; Zhu, J.; Liu, J.; Zhang, Z.; Yuan, K.; Sun, W.; Li, H. Learning N:M Fine-grained Structured Sparse Neural Networks From Scratch. *arXiv* **2021**, arXiv:2102.04010.
20. Wang, H.; Qin, C.; Zhang, Y.; Fu, Y. Neural Pruning via Growing Regularization. *arXiv* **2020**, arXiv:2012.09243.
21. Liu, Z.; Zhou, G.; He, J.; Marcucci, T.; Li, F.F.; Wu, J.; Li, Y. Model-Based Control with Sparse Neural Dynamics. In *Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 10–16 December 2023*; Curran Associates, Inc.: New York, NY, USA, 2023; Volume 36, pp. 6280–6296.
22. Zoph, B.; Le, Q.V. Neural Architecture Search with Reinforcement Learning. *arXiv* **2016**, arXiv:1611.01578.
23. Dong, Y.; Wu, N.; Qi, J.; Chen, X.; Hua, C. Predictive Course Control and Guidance of Autonomous Unmanned Sailboat Based on Efficient Sampled Gaussian Process. *J. Mar. Sci. Eng.* **2021**, *9*, 1420. [CrossRef]
24. Jiang, B.; Li, B.; Zhou, W.; Lo, L.Y.; Chen, C.K.; Wen, C.Y. Neural Network Based Model Predictive Control for a Quadrotor UAV. *Aerospace* **2022**, *9*, 460. [CrossRef]
25. Arena, P.; Patanè, L.; Taffara, S. A Data-Driven Model Predictive Control for Quadruped Robot Steering on Slippery Surfaces. *Robotics* **2023**, *12*, 67. [CrossRef]
26. Zhou, Y.; Quan, Y.; Wang, Y.; Jin, X. Nonlinear Predictive Control Based on ELM Neural Network and Dung Beetle Optimization Algorithm. In *Proceedings of the 2023 5th International Conference on Control and Robotics (ICCR), Tokyo, Japan, 23–25 November 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 78–84. [CrossRef]
27. Gillespie, M.T.; Best, C.M.; Townsend, E.C.; Wingate, D.; Killpack, M.D. Learning nonlinear dynamic models of soft robots for model predictive control with neural networks. In *Proceedings of the 2018 IEEE International Conference on Soft Robotics (RoboSoft), Livorno, Italy, 24–28 April 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 39–45. [CrossRef]
28. Spielberg, N.A.; Brown, M.; Gerdes, J.C. Neural Network Model Predictive Motion Control Applied to Automated Driving with Unknown Friction. *IEEE Trans. Control Syst. Technol.* **2022**, *30*, 1934–1945. [CrossRef]
29. Wang, J.; Zhang, Y. A Tutorial on Gaussian Process Learning-based Model Predictive Control. *arXiv* **2024**,arXiv:2404.03689. [CrossRef]
30. Zhang, X.; Liu, J.; Xu, X.; Yu, S.; Chen, H. Robust Learning-Based Predictive Control for Discrete-Time Nonlinear Systems with Unknown Dynamics and State Constraints. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 7314–7327. [CrossRef]

31. Nagabandi, A.; Kahn, G.; Fearing, R.S.; Levine, S. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 7559–7566. [CrossRef]
32. Peng, B.; Duan, J.; Chen, J.; Li, S.E.; Xie, G.; Zhang, C.; Guan, Y.; Mu, Y.; Sun, E. Model-Based Chance-Constrained Reinforcement Learning via Separated Proportional-Integral Lagrangian. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 466–478. [CrossRef] [PubMed]
33. Carnduff, S. Aircraft System Identification: Theory and Practice V. Klein and E.A. Morelli American Institute of Aeronautics and Astronautics, 1801 Alexander Bell Drive, Suite 500, Reston, VA 20191-4344, USA. 2006. 484pp. Illustrated. 84.95 (AIAA members), 119.95 (non-members). ISBN 1-56347-832-3. *Aeronaut. J.* **2007**, *111*, 602–603.
34. Howard, R.M. Dynamics of Flight: Stability and Control; Third Edition. *J. Guid. Control. Dyn.* **1997**, *20*, 839–840. [CrossRef]
35. Stevens, B.L.; Lewis, F.L.; Johnson, E.N. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*, 3rd ed.; Wiley: Hoboken, NJ, USA, 2016.
36. Hale, L.E.; Patil, M.; Roy, C.J. Aerodynamic Parameter Identification and Uncertainty Quantification for Small Unmanned Aircraft. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Kissimmee, FL, USA, 5–9 January 2015*; American Institute of Aeronautics and Astronautics, Inc.: Reston, VA, USA, 2015. [CrossRef]
37. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both Weights and Connections for Efficient Neural Network. In *Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015*; Curran Associates, Inc.: New York, NY, USA, 2015; Volume 28.
38. Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; Darrell, T. Rethinking the Value of Network Pruning. *arXiv* **2018**, arXiv:1810.05270.
39. Liu, L.; Zhang, S.; Kuang, Z.; Zhou, A.; Xue, J.H.; Wang, X.; Chen, Y.; Yang, W.; Liao, Q.; Zhang, W. Group Fisher Pruning for Practical Network Compression. In *Proceedings of the 38th International Conference on Machine Learning. PMLR, Virtual, 18–24 July 2021*; PMLR: Cambridge, MA, USA, 2021; Volume 139, pp. 7021–7032.
40. Renda, A.; Frankle, J.; Carbin, M. Comparing Rewinding and Fine-tuning in Neural Network Pruning. *arXiv* **2020**, arXiv:2003.02389. [CrossRef]
41. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2015**, arXiv:1412.6572. [CrossRef]
42. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning Filters for Efficient ConvNets. *arXiv* **2016**, arXiv:1608.08710.
43. Yoshida, Y.; Miyato, T. Spectral Norm Regularization for Improving the Generalizability of Deep Learning. *arXiv* **2017**, arXiv:1705.10941. [CrossRef]
44. Ross, A.S.; Doshi-Velez, F. Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients. *arXiv* **2017**, arXiv:1711.09404. [CrossRef]
45. Song, X.; Duan, J.; Wang, W.; Li, S.E.; Chen, C.; Cheng, B.; Zhang, B.; Wei, J.; Wang, X.S. LipsNet: A smooth and robust neural network with adaptive Lipschitz constant for high accuracy optimal control. In *Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023*; JMLR.org; ICML'23; PMLR: Cambridge, MA, USA, 2023.
46. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. *arXiv* **2018**, arXiv:1802.05957. [CrossRef]
47. Shi, Y.; Tang, A.; Niu, L.; Zhou, R. Sparse optimization guided pruning for neural networks. *Neurocomputing* **2024**, *574*, 127280. [CrossRef]
48. Lambregts, A.A. TECS Generalized Airplane Control System Design—An Update. In *Proceedings of the Advances in Aerospace Guidance, Navigation and Control*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 503–534.
49. Kamel, M.; Burri, M.; Siegwart, R. Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles. *IFAC-PapersOnLine* **2017**, *50*, 3463–3469. [CrossRef]
50. Cotting, M.C.; Wolek, A.; Murtha, J.; Woolsey, C. Developmental Flight Testing of the SPAARO UAV. In *Proceedings of the 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, Orlando, FL, USA, 4–7 January 2010*; American Institute of Aeronautics and Astronautics, Inc.: Reston, VA, USA, 2010. [CrossRef]
51. Andersson, J.A.E.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M. CasADi: A software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **2019**, *11*, 1–36. [CrossRef]
52. Wächter, A.; Biegler, L.T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **2006**, *106*, 25–57. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.