Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Learning-Driven Autonomy in Open-Architecture Autopilots

Lakshman Mahto

lakshman@bitmesra.ac.in

Birla Institute of Technology, Mesra

Research Article

Keywords: autonomous systems, autopilot, multimodal perception, Conditional Variational Autoencoder, GRU-attention, model predictive control, control barrier functions, triple modular redundancy, UAV autonomy, safety-critical learning.

Posted Date: June 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9807532/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Learning-Driven Autonomy in Open-Architecture Autopilots

Lakshman Mahto*
Quantitative Economics and Data Science
Birla Institute of Technology Mesra, Ranchi-835215, India

Abstract

This paper proposes a learning-driven autonomy architecture for safety-critical open-architecture autopilots. The framework combines multimodal perception, uncertainty-aware probabilistic forecasting, risk-aware model predictive control, control barrier function-based safety filtering, and triple-redundant command supervision. The core representation is a triad-conditioned autonomy tensor,

$$\xi_t = c_t \| \psi_t^{\text{ctrl}} \| \pi_t \in \mathbb{R}^{104},$$

where $c_t \in \mathbb{R}^{64}$ is a learned multisensor belief vector, $\psi_t^{\text{ctrl}} \in \mathbb{R}^{32}$ encodes recent control-regime information, and $\pi_t \in \mathbb{R}^8$ represents candidate future-plan intent. This tensor conditions a deep Conditional Variational Autoencoder (CVAE), which predicts a distribution over future vehicle states and risk variables.

The resulting forecast distribution is used by a risk-aware Model Predictive Control (MPC) layer to generate a nominal command, which is subsequently projected into a safe-control set by a Control Barrier Function (CBF) safety filter. The system is implemented as a heterogeneous triple-redundant architecture consisting of a primary real-time controller, a learning-based forecasting companion, and an independent safety/voting channel. Simulations on a UAV navigation dataset demonstrate multimodal representation learning, uncertainty-aware forecasting, comparison with deterministic baselines, and safety-filtered closed-loop actuation under obstacle and battery constraints. The proposed design separates prediction, optimization, safety enforcement, and execution, enabling learning-based autonomy while preserving an independent deterministic safety layer.

Keywords: autonomous systems; autopilot; multimodal perception; Conditional Variational Autoencoder; GRU-attention; model predictive control; control barrier functions; triple modular redundancy; UAV autonomy; safety-critical learning.

1 Introduction

Autonomous aerial vehicles are increasingly expected to operate in environments that are dynamic, uncertain, partially observed, and safety-critical. Modern open-architecture autopilots provide reliable low-level stabilization, attitude control, navigation, mission execution, and fail-safe logic. However, higher levels of autonomy require capabilities that extend beyond conventional feedback control, including multimodal perception, predictive reasoning, risk-aware planning, uncertainty quantification, and formal safety filtering. These requirements become especially important when the vehicle operates with noisy sensors, intermittent measurements, limited onboard computation, obstacle uncertainty, battery constraints, and possible distribution shifts between training and deployment.

Recent progress in learning-based perception and sequence modeling has enabled autonomous systems to extract compact representations from heterogeneous sensor data. Nevertheless, direct end-to-end neural control remains difficult to justify in safety-critical autopilot systems

*Corresponding author. Email: lm.optlearning@gmail.com.

because learned policies may be hard to interpret, difficult to certify, and sensitive to out-of-distribution inputs. A practical autonomy architecture should therefore separate prediction, optimization, safety enforcement, and low-level execution. In such a design, learning-based modules provide probabilistic forecasts or candidate plans, while model-based optimization and safety filters ensure that the final actuator command remains admissible. This separation is consistent with the broader safety-critical control literature, where model predictive control (MPC) is used for constrained optimization and control barrier functions (CBFs) are used to enforce forward invariance of safety sets [9, 7, 1, 2].

This paper proposes a layered learning-driven autonomy architecture for open-architecture autopilot systems. The proposed stack consists of five interacting components:

1. a multimodal perception layer that converts heterogeneous sensor streams into a compact belief representation;
2. a probabilistic forecasting layer that predicts future vehicle states and risk under uncertainty;
3. a risk-aware MPC layer that converts forecast distributions into nominal control proposals;
4. a CBF-based safety filter that projects nominal commands into an admissible safe-control set;
5. a classical inner-loop controller that executes the final safety-filtered command.

In addition, an independent safety/voting channel supervises the command stream and may reduce, reject, or override learning-based proposals when uncertainty or risk becomes excessive. Thus, the learning module is used for foresight and decision support rather than direct actuator authority.

A central contribution of this work is the introduction of a triad-conditioned autonomy tensor,

$$\xi_t = c_t \| \psi_t^{\mathrm{ctrl}} \| \pi_t \in \mathbb{R}^{104}, \tag{1}$$

where $c_t \in \mathbb{R}^{64}$ denotes a learned multisensor belief vector, $\psi_t^{\mathrm{ctrl}} \in \mathbb{R}^{32}$ denotes a recent control-regime embedding, and $\pi_t \in \mathbb{R}^8$ denotes a compressed representation of candidate future-plan intent. Unlike conventional forecasting models that rely primarily on the current state estimate, the proposed representation conditions prediction on three complementary sources of information: the current perceptual belief, the recent actuation history, and the future plan being evaluated. Consequently, the forecasting model becomes state-aware, control-aware, and plan-aware.

The probabilistic forecasting module is implemented as a deep Conditional Variational Autoencoder (CVAE). Variational Autoencoders provide a principled latent-variable framework for learning generative models through variational inference and the reparameterization trick [5]. Conditional VAEs extend this idea by modeling structured output distributions conditioned on observed inputs, making them suitable for one-to-many forecasting problems [10]. In the present autopilot setting, identical current conditions may lead to different future trajectories due to wind disturbances, obstacle motion, sensor uncertainty, actuator variation, or battery dynamics. The latent variable z represents such hidden scenario factors, while the decoder generates possible future trajectories conditioned on the triad tensor.

To construct the belief vector c_t, the proposed model first encodes each sensor modality using neural modality encoders. The encoded windows are summarized using Random Fourier Feature Kernel Mean Embeddings (RFF-KME), which provide a computationally efficient approximation of distributional kernel embeddings [8]. A GRU-attention network then aggregates the fused sequence of modality summaries into the learned belief vector. The GRU provides temporal memory for sequential sensor histories [4], while the attention mechanism assigns higher weights to context windows that are more relevant to future prediction [3].

The resulting forecast distribution is passed to a risk-aware MPC layer. The MPC layer evaluates candidate control actions using forecast means, uncertainty estimates, and scenario samples. The nominal MPC command is not directly applied to the vehicle. Instead, it is combined with commands from a primary controller and an independent safety controller through a heterogeneous triple-voting mechanism. The voted command is then filtered by a CBF safety layer. The CBF layer enforces hard safety constraints, such as obstacle-clearance and battery-margin constraints, by projecting the voted command into the safe-control set. This construction ensures that learning-based predictions can influence planning while the final command remains subject to deterministic safety supervision.

The proposed architecture is evaluated through simulations on a UAV navigation dataset. The experiments demonstrate multimodal representation learning, uncertainty-aware CVAE forecasting, comparison against deterministic baselines, visualization of prediction intervals, GRU-attention diagnostics, belief-space embeddings, triad block-norm analysis, and forecast-driven MPC/CBF/voting control. The results show that the deep CVAE provides probabilistic forecasts and modest improvement over a deterministic Ridge baseline, while the safety layer reduces aggressive commands when predicted obstacle or battery margins become unsafe.

The main contributions of this paper are summarized as follows:

1. We propose a triad-conditioned learning-driven autonomy architecture that integrates perception, forecasting, planning, safety filtering, voting supervision, and classical actuation.
2. We introduce a compact autonomy tensor that combines a multisensor belief vector, a control-regime embedding, and a future-plan embedding.
3. We develop a deep CVAE forecasting model based on RFF-KME modality summaries and GRU-attention context encoding.
4. We integrate the probabilistic forecast distribution into an MPC–CBF–voting pipeline for safety-filtered command generation.
5. We provide a theorem-style safety statement for feasible CBF filtering and explain the practical meaning of the CBF feasibility assumption.
6. We present a reproducible simulation workflow with forecasting metrics, uncertainty plots, attention visualizations, belief embeddings, triad diagnostics, CBF margins, voting weights, and 3D trajectory visualization.

2 Related Work

This section reviews the main research areas that motivate the proposed learning-driven autopilot architecture: open autopilot and companion-computer frameworks, conditional generative forecasting, kernel-based multimodal representation learning, recurrent attention mechanisms, model predictive control, control barrier functions, and fault-tolerant redundancy.

2.1 Open Autopilot and Companion-Computer Architectures

Open-architecture autopilot platforms, such as PX4 and ArduPilot, provide practical real-time flight-control infrastructure for unmanned aerial vehicles and other robotic systems. These platforms typically include state estimation, attitude stabilization, waypoint navigation, mission execution, actuator mixing, telemetry handling, and failsafe logic. Their modular software and communication interfaces make them suitable foundations for integrating higher-level autonomy modules on companion computers.

In such architectures, low-level flight-critical functions are usually executed on a real-time flight controller, while perception, mapping, planning, and learning-based inference can be

executed on an onboard companion computer. Communication between these layers is commonly supported by protocols and middleware such as MAVLink, ROS 2, CAN, DroneCAN, and Cyphal. This separation is important for safety-critical autonomy because it allows computationally intensive learning modules to assist decision-making without directly replacing deterministic stabilization and failsafe mechanisms.

The architecture proposed in this paper follows this separation principle. The real-time autopilot layer executes low-level control and actuator commands, the learning companion performs multimodal perception and probabilistic forecasting, and an independent safety/voting layer supervises command validity. This organization enables learning-based autonomy to be integrated into open autopilot systems while preserving a deterministic safety pathway.

2.2 Conditional Generative Forecasting

Probabilistic forecasting is essential in autonomous systems because identical current observations may lead to multiple plausible future outcomes. In UAV autonomy, such multimodality may arise from wind disturbances, obstacle motion, GNSS-denied drift, actuator variability, battery dynamics, sensor noise, and partially observed environments. Deterministic forecasting models often collapse these possibilities into a single mean prediction, which may be insufficient for risk-aware planning.

Variational Autoencoders (VAEs) introduced a principled latent-variable framework for learning generative models using variational inference and the reparameterization trick [5]. Conditional Variational Autoencoders (CVAEs) extend this framework by conditioning the generative process on observed context variables, making them suitable for structured output prediction and one-to-many forecasting problems [10]. In the proposed autopilot architecture, the CVAE models a conditional distribution over future vehicle states and risk variables rather than producing only a point estimate.

The latent variable z in the CVAE represents hidden scenario factors that are not fully determined by the observed state. In the autopilot setting, these latent factors may correspond to unobserved wind changes, obstacle behavior, LiDAR uncertainty, actuator response variation, or battery-discharge variability. By sampling from the conditional prior $p_\theta(z \mid \xi_t)$, the system can generate multiple plausible future trajectories and pass uncertainty estimates to the downstream MPC and CBF layers.

2.3 Kernel Embeddings and Recurrent Attention for Multimodal Perception

Multimodal autonomous systems must combine heterogeneous data streams such as navigation states, inertial measurements, obstacle-distance readings, power variables, environmental conditions, and actuator feedback. A direct concatenation of raw measurements may be sensitive to noise, missing values, and different sampling rates. Therefore, compact and distribution-aware modality representations are useful for robust forecasting and control.

Random Fourier Features provide a scalable approximation of shift-invariant kernels and make kernel-based representations computationally tractable for large datasets [8]. Kernel mean embeddings summarize distributions in a reproducing kernel Hilbert space, and their random-feature approximations can be used to summarize recent sensor windows efficiently. In the proposed model, each modality is first encoded by a neural network and then summarized by an RFF-based kernel mean embedding. This produces a compact representation of recent modality behavior rather than relying on a single instantaneous sample.

Temporal structure is modeled using a gated recurrent unit (GRU), which provides a computationally efficient recurrent architecture for sequence modeling [4]. Attention mechanisms further allow the model to assign different weights to different time windows, emphasizing recent or historically important context segments [3]. The proposed RFF-KME + GRU-attention encoder combines these components to produce the learned belief vector $c_t \in \mathbb{R}^{64}$, which represents

the current multimodal context used by the CVAE forecaster.

2.4 Model Predictive Control and Safety-Critical Planning

Model Predictive Control (MPC) is widely used in autonomous systems because it explicitly handles constraints while optimizing a finite-horizon control objective. MPC repeatedly solves an optimization problem over a prediction horizon, applies the first control action, and updates the solution as new information becomes available. This receding-horizon structure makes MPC suitable for trajectory tracking, energy-aware control, obstacle-aware planning, and actuator-constrained decision-making [9, 7].

In learning-driven autonomy, MPC can serve as an interface between probabilistic forecasts and executable commands. Rather than relying on a deterministic model alone, the MPC layer can evaluate candidate actions using samples or moments from the CVAE forecast distribution. This enables the controller to penalize expected tracking error, energy consumption, predicted risk, actuator effort, and possible constraint violations. In the proposed architecture, the MPC layer generates a nominal command, but this command is not applied directly. It is subsequently supervised by a voting mechanism and filtered by a CBF safety layer.

2.5 Control Barrier Functions for Formal Safety Filtering

Control Barrier Functions provide a formal method for enforcing safety constraints by ensuring forward invariance of a prescribed safe set. Given a safe set defined by $h(x) \geq 0$, a CBF condition restricts the control input so that the system state cannot leave the safe set under feasible control authority. CBF-based quadratic programs have become a common approach for safety-critical control because they minimally modify a nominal command while enforcing barrier constraints [1, 2].

In the proposed autopilot architecture, the CBF layer acts as the final deterministic safety filter before actuation. The nominal command generated by the MPC and voting layers is projected into an admissible safe-control set. Safety margins are also inflated using CVAE forecast uncertainty. For example, if the predicted LiDAR distance is uncertain, the effective obstacle-clearance requirement is increased. Similarly, if the predicted battery level is uncertain, the effective battery threshold is increased. This uncertainty-aware margin inflation makes the controller more conservative when the learned forecast is less certain.

Importantly, the CVAE itself does not guarantee safety. The CVAE provides probabilistic foresight, while the CBF layer enforces hard safety constraints. This separation is central to the proposed architecture: learning improves anticipation, but formal safety filtering determines whether the final command is admissible.

2.6 Fault Tolerance, Voting, and Triple Redundancy

Fault tolerance is a critical consideration in safety-critical autopilot systems. Triple Modular Redundancy (TMR) is a classical strategy in which multiple channels produce outputs that are compared or voted to improve reliability [6]. In traditional TMR, redundant modules may be identical or functionally equivalent. In learning-driven autonomy, however, heterogeneous redundancy can be more appropriate because different channels provide different types of assurance.

The proposed architecture adopts a heterogeneous triple-redundant structure. Channel A is the primary real-time control channel, Channel B is the independent safety and voting channel, and Channel C is the AI/MPC companion channel. The voting mechanism combines commands from these channels using median voting and confidence-weighted averaging. When forecast risk or uncertainty increases, the weight assigned to the AI/MPC channel is reduced. This

prevents learning-based proposals from dominating the command stream under uncertain or unsafe conditions.

This design differs from a purely learning-based control stack because the learning module is deliberately prevented from bypassing independent safety supervision. The resulting architecture combines the adaptability of learning-based forecasting with the robustness of classical control, formal safety filtering, and fault-tolerant voting.

2.7 Positioning of the Present Work

The proposed framework differs from prior work by integrating multimodal representation learning, conditional generative forecasting, MPC optimization, CBF safety filtering, and heterogeneous voting into a single autopilot-oriented architecture. The triad-conditioned tensor

$$\xi_t = c_t \| \psi_t^{\mathrm{ctrl}} \| \pi_t$$

explicitly combines perception, recent control behavior, and future plan intent. This makes the forecasting model not only state-aware, but also control-aware and plan-aware. The architecture therefore provides a structured pathway for using deep probabilistic models in safety-critical autopilot systems without granting the learning module direct actuator authority.

3 Problem Formulation and Assumptions

This section formalizes the learning-driven autopilot problem considered in this work. The objective is to construct an autonomy stack that uses heterogeneous sensor observations and recent control information to forecast uncertain future states, generate nominal control proposals, and apply only commands that satisfy safety constraints. The formulation separates the prediction problem from the safety-filtering problem: the learning module estimates a distribution over plausible futures, whereas the MPC–CBF–voting stack determines whether a proposed command is admissible for execution.

3.1 Stochastic System Model

Consider an autonomous platform evolving in discrete time according to the stochastic dynamics

$$x_{t+1} = f(x_t, u_t, d_t) + \epsilon_t, \tag{2}$$

where $x_t \in \mathcal{X} \subset \mathbb{R}^{n_x}$ denotes the system state, $u_t \in \mathcal{U} \subset \mathbb{R}^{n_u}$ denotes the control input, d_t represents unobserved exogenous disturbances, and ϵ_t denotes process noise. In the UAV setting, the state may include local position, velocity, altitude, attitude-related variables, obstacle-distance information, battery level, actuator load, and risk indicators. The exact state dimension is platform-dependent; therefore, the proposed formulation treats x_t as a general autonomy state rather than as a fixed rigid-body state vector.

At each time step, the platform receives heterogeneous observations from multiple sensing and telemetry sources:

$$o_t = \left[y_t^{\mathrm{nav}}, y_t^{\mathrm{imu}}, y_t^{\mathrm{env}}, y_t^{\mathrm{power}}, y_t^{\mathrm{act}} \right], \tag{3}$$

where y_t^{nav} represents navigation or localization measurements, y_t^{imu} represents inertial measurements, y_t^{env} represents environmental or obstacle-related measurements, y_t^{power} represents battery and energy-state measurements, and y_t^{act} represents actuator or command-feedback measurements.

Because real autonomous systems may experience packet loss, sensor dropout, saturation, or delayed measurements, the observation vector is augmented with a binary availability mask

$$m_{t,i} = \begin{cases} 1, & \text{if observation channel } i \text{ is available at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The perception encoder receives the masked observation

$$\tilde{o}_t = [o_t \odot m_t,\ m_t]\,, \tag{5}$$

where $\odot$ denotes elementwise multiplication. This representation allows the learning model to distinguish a physically meaningful zero-valued measurement from a missing measurement that has been filled numerically.

3.2 Triad-Conditioned Autonomy Representation

The proposed architecture uses a compact triad-conditioned autonomy representation,

$$\xi_t = c_t \| \psi_t^{\text{ctrl}} \| \pi_t, \tag{6}$$

where $\|$ denotes vector concatenation. The three components have distinct roles:

- $c_t \in \mathbb{R}^{64}$ is a learned multisensor belief vector summarizing the recent multimodal observation history.
- $\psi_t^{\text{ctrl}} \in \mathbb{R}^{32}$ is a control-regime embedding summarizing recent actuation, command history, switching behavior, or control effort.
- $\pi_t \in \mathbb{R}^{8}$ is a compact future-plan embedding summarizing the candidate plan or intent evaluated by the planner.

Thus, in the implementation considered in this work,

$$\xi_t \in \mathbb{R}^{104}. \tag{7}$$

The purpose of the triad representation is to condition forecasting not only on the current perceived state, but also on recent control behavior and future plan intent. This is important because two identical state estimates may lead to different future trajectories if the vehicle has recently been controlled aggressively, if actuator limits are nearly active, or if the candidate future plan demands a sharp maneuver.

The forecasting problem is to learn the conditional distribution

$$p_\theta \left(X_{t+1:t+H} \mid \xi_t \right), \tag{8}$$

where

$$X_{t+1:t+H} = [x_{t+1}, x_{t+2}, \ldots, x_{t+H}] \tag{9}$$

denotes the future trajectory over a finite prediction horizon H. In the experiments, the predicted variables include altitude, speed, LiDAR distance, battery level, and an overall risk score; however, the formulation is general and can incorporate additional state, perception, or actuator variables.

3.3 Control and Safety Objective

The learning module provides a forecast distribution rather than a direct actuation command. A nominal control proposal is generated by a risk-aware planner or MPC layer using the predictive distribution in (8). The resulting command is denoted by u_{nom}. Before execution, the command must be validated by the voting and safety-filtering layers.

Let the safe set be defined as

$$\mathcal{S} = \{x \in \mathcal{X} : h_i(x) \geq 0,\ i = 1, \ldots, N_h\}\,, \tag{10}$$

where each $h_i : \mathcal{X} \to \mathbb{R}$ is a continuously differentiable safety function. Typical safety functions may represent obstacle clearance, battery margin, geofence margin, attitude envelope limits, actuator bounds, or other mission-specific constraints.

The safety-filtering problem is to compute an admissible command

$$u_{\text{safe}} = \Pi_{\mathcal{U}_{\text{safe}}}(u_{\text{nom}}), \tag{11}$$

where $\Pi_{\mathcal{U}_{\text{safe}}}$ denotes projection onto the set of controls that satisfy the relevant CBF constraints and actuator bounds. Thus, the final command sent to the classical inner-loop controller is u_{safe}, not the raw learning or MPC proposal.

3.4 Assumptions

The proposed formulation relies on the following assumptions.

Assumption 1 (Temporal Alignment of Multimodal Streams)**.** *All sensor and telemetry streams are synchronized, interpolated, or held to a common temporal grid before being passed to the perception encoder. Remaining timing uncertainty is treated as part of the observation noise.*

Assumption 2 (Explicit Missing-Data Representation)**.** *Missing or unavailable sensor measurements are represented through binary masks. The perception encoder receives both the masked observation values and the mask itself, as in* (5)*.*

Assumption 3 (Separation of Learning and Actuation)**.** *The learning companion may estimate future trajectories, uncertainty, and risk, and may propose nominal actions through the MPC layer. However, it cannot directly bypass the independent safety/voting controller or directly command the actuators.*

Assumption 4 (Availability of Safety-Relevant State Estimates)**.** *The safety layer has access to state estimates and forecast-derived quantities sufficient to evaluate the safety functions $h_i(x)$, including obstacle-clearance and battery-margin constraints in the simulated implementation.*

Assumption 5 (Feasibility of Hard CBF Constraints)**.** *For every state $x \in \mathcal{S}$ encountered during normal operation, there exists at least one control input $u \in \mathcal{U}$ satisfying all hard CBF inequalities and actuator bounds. Equivalently, the CBF-admissible control set is nonempty:*

$$K_{\text{CBF}}(x) = \{u \in \mathcal{U} : L_f h_i(x) + L_g h_i(x) u + \alpha_i h_i(x) \geq 0,\ i = 1, \ldots, N_h\} \neq \emptyset. \tag{12}$$

Remark 1 (Interpretation of CBF Feasibility)**.** *The CBF feasibility assumption is a practical control-authority condition. It may be violated if the vehicle is already too close to an obstacle, if the actuator limits are insufficient for avoidance, if safety constraints conflict, or if uncertainty-inflated margins are overly conservative. In implementation, feasibility is supported by conservative operating envelopes, speed reduction, backup controllers, emergency landing modes, and slack variables for lower-priority soft constraints.*

Assumption 6 (Classical Inner-Loop Tracking)**.** *The low-level autopilot can track the safety-filtered command u_{safe} within bounded error over the time scale considered by the outer-loop planner. This assumption is standard when a high-level planner interfaces with a faster inner-loop attitude or velocity controller.*

4 Preliminaries

This section summarizes the mathematical components used in the proposed learning-driven autopilot architecture. The perception module first compresses heterogeneous sensor windows

using modality-specific encoders and Random Fourier Feature Kernel Mean Embeddings (RFF-KME). The resulting sequence of fused modality summaries is processed by a GRU-attention network to obtain the belief vector c_t. Finally, a Conditional Variational Autoencoder (CVAE) uses the triad-conditioned representation to model a distribution over future states and risk variables.

4.1 Random Fourier Feature Kernel Mean Embedding

Let m denote a sensor modality, such as navigation, inertial, environmental, power, or actuator telemetry. For a raw modality window, the modality encoder first maps each observation to a latent feature vector

$$z_\tau^m = f_m(y_\tau^m), \qquad z_\tau^m \in \mathbb{R}^{d_m}, \tag{13}$$

where $f_m(\cdot)$ is a learned modality-specific encoder and y_τ^m is the raw measurement from modality m at time τ.

To obtain a compact distributional summary of the recent modality window, the encoded samples are mapped through Random Fourier Features. For a shift-invariant kernel, the random feature map is defined as

$$\phi(z) = \sqrt{\frac{2}{D}} \cos(Wz + b), \tag{14}$$

where D is the number of random features, $W \in \mathbb{R}^{d_m \times D}$ is sampled from the spectral distribution associated with the kernel, and $b \in \mathbb{R}^D$ is sampled uniformly from $[0, 2\pi]$. In the implemented model, W and b are fixed after initialization.

The empirical RFF-based kernel mean embedding of modality m over a raw window of length W_{raw} is

$$\mu_t^m = \frac{1}{W_{\text{raw}}} \sum_{\tau=t-W_{\text{raw}}+1}^{t} \phi(z_\tau^m). \tag{15}$$

This embedding summarizes the recent distribution of modality-specific latent features rather than relying on a single instantaneous measurement. Therefore, μ_t^m captures short-term variations, noise patterns, and local temporal statistics of each modality.

For M modalities, the fused multimodal representation at a context time is obtained by concatenating the modality embeddings:

$$f_t = \mu_t^1 \| \mu_t^2 \| \cdots \| \mu_t^M. \tag{16}$$

In the UAV implementation, the modalities correspond to navigation/perception, IMU, and environment/power/risk channels. With $D = 32$ random features per modality and $M = 3$, the fused representation has dimension 96.

4.2 GRU-Attention Belief Encoding

The fused modality summaries form a temporal sequence

$$f_{t-L+1:t} = [f_{t-L+1}, f_{t-L+2}, \ldots, f_t], \tag{17}$$

where $L = L_{\text{seq}}$ is the number of context windows. This sequence is processed by a gated recurrent unit (GRU) to capture temporal dependencies:

$$h_{1:L} = \text{GRU}(f_{1:L}), \qquad h_\tau \in \mathbb{R}^{d_h}. \tag{18}$$

The GRU hidden states encode recent trends in the multimodal sensor history, such as changes in speed, altitude, obstacle distance, wind, battery level, and inertial agitation.

Instead of using only the final hidden state, an attention mechanism is used to compute an adaptive weighted summary of the sequence. The attention score for each context window is

$$e_\tau = w_a^\top h_\tau, \tag{19}$$

and the normalized attention weight is

$$\alpha_\tau = \frac{\exp(e_\tau)}{\sum_{j=1}^{L} \exp(e_j)}. \tag{20}$$

The context summary is then

$$\bar{h}_t = \sum_{\tau=1}^{L} \alpha_\tau h_\tau. \tag{21}$$

Finally, the belief vector is obtained by a learned projection:

$$c_t = W_c \bar{h}_t + b_c, \qquad c_t \in \mathbb{R}^{64}. \tag{22}$$

The attention mechanism allows the encoder to emphasize context windows that are more informative for future prediction. For example, a sudden reduction in LiDAR distance or an increase in inertial activity may receive higher attention than older, less relevant context windows. Thus, c_t serves as a compact learned representation of the current multimodal belief state.

4.3 Conditional Variational Autoencoder

The forecasting module is based on a Conditional Variational Autoencoder. Given the triad-conditioned representation ξ_t, the objective is to model the conditional distribution of the future trajectory:

$$p_\theta \left(X_{t+1:t+H} \mid \xi_t \right). \tag{23}$$

The CVAE introduces a latent variable $z \in \mathbb{R}^{d_z}$ to represent hidden scenario factors that are not fully determined by the observed context. These factors may include unmeasured wind disturbances, obstacle-motion uncertainty, actuator variability, sensor noise, or battery-discharge variability.

The conditional generative model factorizes as

$$p_\theta \left(X_{t+1:t+H} \mid \xi_t \right) = \int p_\theta \left(X_{t+1:t+H} \mid z, \xi_t \right) p_\theta \left(z \mid \xi_t \right) dz. \tag{24}$$

The conditional prior is modeled as a diagonal Gaussian:

$$p_\theta \left(z \mid \xi_t \right) = \mathcal{N} \left(\mu_p(\xi_t), \operatorname{diag} \left(\sigma_p^2(\xi_t) \right) \right). \tag{25}$$

During training, the true future trajectory is available, and a recognition network approximates the posterior distribution:

$$q_\phi \left(z \mid X_{t+1:t+H}, \xi_t \right) = \mathcal{N} \left(\mu_q(X_{t+1:t+H}, \xi_t), \operatorname{diag} \left(\sigma_q^2(X_{t+1:t+H}, \xi_t) \right) \right). \tag{26}$$

To permit gradient-based optimization through the stochastic latent variable, the latent sample is generated using the reparameterization trick:

$$z = \mu_q + \sigma_q \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{27}$$

The decoder then predicts a distribution over future states:

$$p_\theta \left(X_{t+1:t+H} \mid z, \xi_t \right). \tag{28}$$

In the implemented model, the decoder outputs both a mean trajectory and a log-variance trajectory. This allows the forecaster to represent both a nominal future and uncertainty around that future.

At deployment, the recognition network is not used because the future trajectory is unknown. Instead, latent scenarios are sampled from the conditional prior:

$$z^{(s)} \sim p_\theta(z \mid \xi_t), \qquad s = 1, \ldots, S. \tag{29}$$

Each sampled latent variable generates one possible future trajectory. The resulting ensemble is used to estimate predictive means, variances, prediction intervals, and safety-relevant quantities for MPC and CBF filtering.

5 Methods

This section presents the proposed learning-driven autopilot architecture and its main computational components. The method is organized around a safety-oriented separation of responsibilities: the learning module estimates future state distributions, the MPC layer converts these forecasts into nominal control proposals, the voting layer combines heterogeneous command channels, and the CBF layer enforces safety before actuation. This design prevents the learning module from directly commanding the vehicle while still allowing probabilistic forecasts to improve planning and decision-making.

5.1 Triple-Redundant Autopilot Architecture

The proposed autonomy stack is implemented as a heterogeneous triple-redundant architecture. The three channels perform complementary functions rather than simply duplicating the same controller:

- **Channel A: Primary real-time control.** This channel represents the conventional autopilot layer. It is responsible for state estimation, stabilization, attitude or velocity control, actuator mixing, and execution of low-level control commands.
- **Channel B: Independent safety and voting supervision.** This channel monitors command validity, evaluates hard safety limits, performs watchdog checks, applies simplified safety logic when necessary, and supervises command arbitration. It can reduce, reject, or override commands when safety margins are violated.
- **Channel C: Learning and planning companion.** This channel performs multimodal perception, belief-vector construction, CVAE-based probabilistic forecasting, and MPC-based nominal command generation. It improves anticipatory behavior but does not have direct actuator authority.

The final command sent to the low-level controller is obtained only after command voting and CBF safety filtering. This organization allows learning-based forecasting to influence planning while preserving a deterministic safety pathway.

5.2 Multimodal Belief and Triad Construction

At each time t, heterogeneous sensor streams are encoded into a compact belief vector c_t. The perception encoder first computes modality-specific embeddings, summarizes recent windows using RFF-KME features, and then aggregates the fused sequence using GRU-attention. The resulting belief vector

$$c_t \in \mathbb{R}^{64} \tag{30}$$

summarizes the current multimodal context.

Recent control behavior is encoded as

$$\psi_t^{\text{ctrl}} \in \mathbb{R}^{32}, \tag{31}$$

which represents actuation regime, command history, control effort, switching behavior, or saturation tendency. Candidate future-plan information is compressed into

$$\pi_t \in \mathbb{R}^8. \tag{32}$$

The three components are concatenated to form the triad-conditioned autonomy tensor

$$\xi_t = c_t \| \psi_t^{\text{ctrl}} \| \pi_t \in \mathbb{R}^{104}. \tag{33}$$

This representation conditions the forecasting model on perception, recent control memory, and future plan intent.

5.3 Deep CVAE Forecasting Model

The forecasting module is a deep Conditional Variational Autoencoder. Given the triad representation ξ_t, the model estimates a distribution over future trajectories:

$$p_\theta \left(X_{t+1:t+H} \mid \xi_t \right), \tag{34}$$

where H is the forecast horizon. The CVAE introduces a latent variable z to represent hidden future scenario factors such as unmeasured wind variation, obstacle-distance uncertainty, actuator response variability, or battery-discharge uncertainty.

The conditional prior is

$$p_\theta \left(z \mid \xi_t \right) = \mathcal{N} \left(\mu_p(\xi_t), \text{diag}(\sigma_p^2(\xi_t)) \right). \tag{35}$$

During training, the true future trajectory is available, and the recognition model approximates the posterior:

$$q_\phi \left(z \mid X_{t+1:t+H}, \xi_t \right) = \mathcal{N} \left(\mu_q(X_{t+1:t+H}, \xi_t), \text{diag}(\sigma_q^2(X_{t+1:t+H}, \xi_t)) \right). \tag{36}$$

The latent sample is obtained by the reparameterization trick:

$$z = \mu_q + \sigma_q \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{37}$$

The decoder predicts both a future mean trajectory and a log-variance trajectory:

$$(\mu_X, \log \sigma_X^2) = D_\theta \left(z, \xi_t \right). \tag{38}$$

5.4 CVAE Training Objective

The CVAE is trained by minimizing a conditional variational objective consisting of a reconstruction term, a latent regularization term, and a smoothness penalty. Since the decoder outputs a Gaussian distribution for each future state component, the reconstruction term is the Gaussian negative log-likelihood:

$$\mathcal{L}_{\text{rec}} = \frac{1}{2} \sum_{h=1}^{H} \sum_{i} \left[\frac{(x_{t+h,i} - \mu_{h,i})^2}{\sigma_{h,i}^2} + \log \sigma_{h,i}^2 \right]. \tag{39}$$

This term rewards accurate mean predictions while discouraging the decoder from inflating uncertainty arbitrarily.

The KL divergence aligns the training-time posterior with the deployment-time prior:

$$\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}\left(q_\phi(z \mid X, \xi_t) \| p_\theta(z \mid \xi_t)\right). \tag{40}$$

This alignment is necessary because the recognition model has access to the future during training, whereas only the prior is available during deployment.

To discourage nonphysical oscillations in the predicted trajectory, a second-order smoothness penalty is added:

$$\mathcal{L}_{\mathrm{smooth}} = \sum_{h=2}^{H-1} \left\| \mu_{h+1} - 2\mu_h + \mu_{h-1} \right\|^2. \tag{41}$$

The total training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{rec}} + \beta \mathcal{L}_{\mathrm{KL}} + \lambda \mathcal{L}_{\mathrm{smooth}}, \tag{42}$$

where β controls latent regularization and λ controls trajectory smoothness.

At runtime, the recognition network is not used. Instead, latent scenarios are sampled from the conditional prior:

$$z^{(s)} \sim p_\theta(z \mid \xi_t), \qquad s = 1, \ldots, S. \tag{43}$$

The decoder generates an ensemble of future trajectories, from which predictive means, variances, and risk probabilities are estimated.

5.5 MPC Integration

The MPC layer uses the CVAE forecast distribution to generate a nominal control proposal. Given S forecast scenarios,

$$X^{(s)}_{t+1:t+H} \sim p_\theta\left(X \mid \xi_t\right), \qquad s = 1, \ldots, S, \tag{44}$$

the MPC evaluates a candidate control sequence

$$U = [u_t, u_{t+1}, \ldots, u_{t+H-1}]. \tag{45}$$

A generic risk-aware MPC objective is

$$J_{\mathrm{MPC}}(U) = \frac{1}{S} \sum_{s=1}^{S} \sum_{h=1}^{H} J\left(x^{(s)}_{t+h}, u_{t+h}\right), \tag{46}$$

where $J(\cdot)$ may include tracking error, energy cost, predicted risk, control effort, actuator saturation penalties, and smoothness penalties.

The nominal MPC proposal is obtained as

$$U^\star = \arg\min_U J_{\mathrm{MPC}}(U), \tag{47}$$

subject to input constraints, state constraints, and chance constraints approximated using the CVAE scenario samples. Only the first control element is passed to the voting controller:

$$u_C = u_t^\star. \tag{48}$$

Here, u_C denotes the Channel C AI/MPC proposal.

5.6 Confidence-Weighted Voting Controller

Before safety filtering, the system combines three independent command proposals:

- u_A: command from the primary real-time controller;
- u_B: command from the independent safety channel;
- u_C: command from the AI/MPC companion.

The voting controller computes

$$u_{\text{vote}} = \frac{1}{2}\,\text{median}(u_A, u_B, u_C) + \frac{1}{2}\left(w_A u_A + w_B u_B + w_C u_C\right), \tag{49}$$

where the weights satisfy

$$w_A + w_B + w_C = 1, \qquad w_A, w_B, w_C \geq 0. \tag{50}$$

The AI/MPC weight w_C is reduced when the predicted risk, forecast uncertainty, or safety-filter intervention increases. This prevents uncertain learning-based proposals from dominating the command stream.

5.7 CBF Safety Filter

The CBF layer receives the voted command u_{vote} and computes the closest admissible safe command:

$$u_{\text{safe}} = \arg\min_{u \in \mathcal{U}} \|u - u_{\text{vote}}\|^2, \tag{51}$$

subject to the CBF constraints

$$L_f h_i(x) + L_g h_i(x) u + \alpha_i h_i(x) \geq 0, \qquad i = 1, \ldots, N_h. \tag{52}$$

Thus, the CBF filter minimally modifies the voted command while enforcing safety.

In the simulation, obstacle safety is represented by

$$h_{\text{obs}} = \hat{d}_{\text{LiDAR}} - d_{\min}^{\text{eff}}, \tag{53}$$

where the effective clearance margin is inflated by forecast uncertainty and predicted risk:

$$d_{\min}^{\text{eff}} = 12 + 1.25\,(2 + \sigma_{\text{LiDAR}} + 6\hat{r}). \tag{54}$$

Here, $\hat{d}_{\text{LiDAR}}$ is the predicted LiDAR distance, σ_{LiDAR} is the LiDAR forecast uncertainty, and $\hat{r}$ is the predicted risk.

Battery safety is represented by

$$h_{\text{battery}} = \hat{B} - (25 + 0.1\sigma_B), \tag{55}$$

where $\hat{B}$ is the predicted battery level and σ_B is the predicted battery uncertainty.

The final command sent to the classical inner-loop controller is

$$u_{\text{act}} = u_{\text{safe}}. \tag{56}$$

This ensures that learning-based forecasts can influence decision-making, but only safety-filtered commands are executed.

6 Safety Guarantee and CBF Feasibility

This section formalizes the safety role of the CBF layer in the proposed learning-driven autopilot architecture. The CVAE and MPC components provide probabilistic foresight and nominal control proposals, but the final executed command is determined only after voting and CBF-based safety filtering. The safety guarantee therefore applies to the CBF-filtered command, not to the raw learning-based or MPC-generated proposal.

6.1 CBF-Admissible Control Set

Let the safe set be defined as

$$\mathcal{S} = \{x \in \mathcal{X} : h_i(x) \geq 0,\ i = 1, \ldots, N_h\}, \tag{57}$$

where each $h_i : \mathcal{X} \to \mathbb{R}$ is a continuously differentiable barrier function. Each inequality $h_i(x) \geq 0$ represents a safety requirement, such as obstacle clearance, battery margin, geofence margin, attitude-envelope protection, or actuator feasibility.

For a control-affine system

$$\dot{x} = f(x) + g(x)u, \tag{58}$$

the CBF condition associated with h_i is

$$L_f h_i(x) + L_g h_i(x) u + \alpha_i h_i(x) \geq 0, \qquad \alpha_i > 0, \tag{59}$$

where $L_f h_i(x)$ and $L_g h_i(x)$ are Lie derivatives of h_i along the drift and control vector fields, respectively.

The set of admissible controls satisfying all CBF constraints and actuator bounds is

$$K_{\mathrm{CBF}}(x) = \{u \in \mathcal{U} : L_f h_i(x) + L_g h_i(x) u + \alpha_i h_i(x) \geq 0,\ i = 1, \ldots, N_h\}. \tag{60}$$

The CBF safety filter computes the final command as the closest admissible command to the voted nominal command:

$$u_{\mathrm{safe}} = \arg \min_{u \in K_{\mathrm{CBF}}(x)} \|u - u_{\mathrm{vote}}\|^2. \tag{61}$$

6.2 Feasibility Assumption

The CBF feasibility assumption requires that the admissible set $K_{\mathrm{CBF}}(x)$ is nonempty for all states encountered inside the safe set:

$$K_{\mathrm{CBF}}(x) \neq \emptyset, \qquad \forall x \in \mathcal{S}. \tag{62}$$

This condition is not merely a numerical assumption about the optimization solver. It is a physical and operational assumption about the vehicle, actuator authority, environment geometry, and the conservatism of the imposed safety margins.

Feasibility may fail in several realistic cases. For example, the vehicle may already be too close to an obstacle for any admissible control to prevent violation of the clearance constraint. Actuator limits may prevent the system from generating the braking, turning, or climbing authority required by the CBF inequalities. Multiple hard safety constraints may also conflict, such as simultaneous obstacle avoidance, attitude-envelope protection, and low-battery return constraints. Finally, uncertainty-inflated margins may become so conservative that no feasible control remains, especially under severe perception uncertainty.

Remark 2 (Practical management of CBF feasibility)**.** *In implementation, feasibility is supported by conservative operating envelopes, speed limits, emergency braking or landing modes, command saturation management, constraint prioritization, robust safety margins, and slack variables for lower-priority soft constraints. Hard safety constraints, such as collision avoidance and geofence containment, should be assigned higher priority than comfort, tracking accuracy, or energy efficiency.*

6.3 Forward-Invariance Guarantee

Theorem 1 (Forward invariance under feasible CBF filtering)**.** *Assume that f and g are locally Lipschitz, each barrier function h_i is continuously differentiable, the CBF-admissible control set $K_{\mathrm{CBF}}(x)$ is nonempty for all $x \in \mathcal{S}$, and the applied control satisfies*

$$u_{\mathrm{safe}} \in K_{\mathrm{CBF}}(x).$$

If the initial state satisfies $x(0) \in \mathcal{S}$, then the closed-loop trajectory remains in the safe set for all future time:

$$x(t) \in \mathcal{S}, \qquad \forall t \geq 0. \tag{63}$$

Proof. For each barrier function h_i, the CBF filter enforces

$$\dot{h}_i(x, u_{\mathrm{safe}}) + \alpha_i h_i(x) \geq 0. \tag{64}$$

Equivalently,

$$\dot{h}_i(x) \geq -\alpha_i h_i(x). \tag{65}$$

By the comparison lemma, if $h_i(x(0)) \geq 0$, then

$$h_i(x(t)) \geq h_i(x(0)) e^{-\alpha_i t} \geq 0, \qquad \forall t \geq 0. \tag{66}$$

Thus, each individual set

$$\mathcal{S}_i = \{x : h_i(x) \geq 0\} \tag{67}$$

is forward invariant. Since the full safe set is the intersection

$$\mathcal{S} = \bigcap_{i=1}^{N_h} \mathcal{S}_i, \tag{68}$$

the safe set $\mathcal{S}$ is also forward invariant. Hence, $x(t) \in \mathcal{S}$ for all $t \geq 0$. □

6.4 Uncertainty-Aware Safety Margins

The proposed architecture uses the CVAE forecast not only to compute nominal future states, but also to estimate uncertainty. This uncertainty is incorporated into the CBF layer through inflated safety margins. For obstacle avoidance, the effective minimum clearance is defined as

$$d_{\min}^{\mathrm{eff}} = d_{\min} + k_d \sigma_d + k_r \hat{r}, \tag{69}$$

where $d_{\min}$ is the nominal clearance requirement, σ_d is the predicted obstacle-distance uncertainty, $\hat{r}$ is the predicted risk score, and $k_d, k_r > 0$ are design gains. The obstacle barrier is then

$$h_{\mathrm{obs}} = \hat{d}_{\mathrm{obs}} - d_{\min}^{\mathrm{eff}}. \tag{70}$$

Similarly, for battery safety, the effective minimum battery threshold is

$$B_{\min}^{\mathrm{eff}} = B_{\min} + k_B \sigma_B, \tag{71}$$

and the battery barrier is

$$h_{\mathrm{battery}} = \hat{B} - B_{\min}^{\mathrm{eff}}. \tag{72}$$

Corollary 1 (Uncertainty-monotone conservatism)**.** *If $k_d > 0$, $k_r > 0$, and $k_B > 0$, then increasing forecast uncertainty or predicted risk cannot enlarge the admissible set of aggressive controls. In particular, increases in σ_d, $\hat{r}$, or σ_B reduce the corresponding safety margins and tighten the CBF constraints.*

Proof. From (70),

$$h_{\text{obs}} = \hat{d}_{\text{obs}} - d_{\min} - k_d\sigma_d - k_r\hat{r}. \tag{73}$$

Therefore,

$$\frac{\partial h_{\text{obs}}}{\partial \sigma_d} = -k_d < 0, \qquad \frac{\partial h_{\text{obs}}}{\partial \hat{r}} = -k_r < 0. \tag{74}$$

Thus, increasing obstacle-distance uncertainty or predicted risk decreases the obstacle barrier value and tightens the corresponding CBF constraint. Similarly,

$$h_{\text{battery}} = \hat{B} - B_{\min} - k_B\sigma_B, \tag{75}$$

so

$$\frac{\partial h_{\text{battery}}}{\partial \sigma_B} = -k_B < 0. \tag{76}$$

Hence, increasing battery uncertainty tightens the battery safety constraint. Therefore, increasing uncertainty or predicted risk cannot enlarge the admissible aggressive-control set. □

6.5 Scope of the Safety Guarantee

The forward-invariance guarantee applies to the CBF-filtered command under the stated feasibility and modeling assumptions. It does not imply that the CVAE forecast itself is safe, nor does it guarantee safety if the system begins outside the safe set, if actuator authority is insufficient, or if the CBF problem becomes infeasible. The role of the CVAE is to provide probabilistic foresight; the role of the CBF layer is to enforce safety constraints on the command that is ultimately executed. This distinction is central to the proposed architecture.

7 Computational Complexity and Generalization

This section analyzes the computational cost of the proposed learning-driven autopilot stack and discusses the conditions under which the learned forecasting model is expected to generalize. The analysis separates the cost of multimodal perception, temporal context encoding, CVAE scenario generation, and safety-filtered control. This separation is useful because these components may run at different rates in a practical companion-computer autopilot implementation.

7.1 Computational Complexity

Let B denote the batch size, M the number of sensor modalities, L the number of fused context windows, W the raw window length, D the number of Random Fourier Features per modality, d_m the input dimension of modality m, d_z^m the latent dimension of the modality encoder for modality m, h_m the hidden width of the modality encoder, h the GRU hidden dimension, H the prediction horizon, and S the number of CVAE latent scenario samples.

For each modality, the feed-forward encoder maps raw observations to modality-specific latent features. If a single hidden layer of width h_m is used, the leading-order cost for modality m is approximately

$$\mathcal{O}\left(BLWd_m h_m\right). \tag{77}$$

For M modalities, the total modality-encoder cost is therefore

$$\mathcal{O}\left(BLW\sum_{m=1}^{M} d_m h_m\right). \tag{78}$$

After modality encoding, the RFF-KME layer maps each latent feature into a D-dimensional random-feature space and averages over the raw window. The leading-order cost of this operation is

$$\mathcal{O}\left(BLW\sum_{m=1}^{M} d_z^m D\right). \tag{79}$$

Since D is typically moderate, this stage is lightweight compared with large convolutional or transformer-based perception modules.

The fused sequence of modality summaries is then processed by a GRU-attention encoder. For a GRU hidden dimension h, the dominant recurrent cost is approximately

$$\mathcal{O}\left(BLh^2\right), \tag{80}$$

while the scalar attention mechanism adds only

$$\mathcal{O}\left(BLh\right). \tag{81}$$

Thus, the attention overhead is small relative to the recurrent update.

The CVAE prior, recognition network, and decoder are lightweight multilayer or recurrent networks. During training, both the prior and recognition networks are evaluated, and the decoder is evaluated once per training sample. During deployment, the recognition network is not used; latent variables are sampled from the conditional prior. If S scenario samples are generated at runtime, the dominant decoder cost scales as

$$\mathcal{O}\left(SBHh^2\right). \tag{82}$$

Therefore, the runtime cost grows linearly with the number of scenario samples S and the forecast horizon H.

The MPC and CBF layers add additional costs. The MPC cost depends on the number of candidate control sequences, the optimization method, and the number of forecast scenarios used in the objective or chance constraints. The simplified CBF safety filter used in the simulations is a low-dimensional projection or clipping operation. In a full CBF-QP implementation, the cost depends on the number of control inputs and the number of active barrier constraints, but it is typically small relative to high-dimensional perception and forecasting when the control dimension is low.

Overall, the main runtime cost is dominated by

$$\mathcal{O}\left(BLW\sum_{m=1}^{M} d_m h_m + BLW\sum_{m=1}^{M} d_z^m D + BLh^2 + SBHh^2\right). \tag{83}$$

This scaling is suitable for companion-computer inference when L, W, D, H, and S are kept moderate.

7.2 Memory Requirements

The memory requirement is dominated by storing the batch of multimodal input windows, recurrent hidden states, latent variables, and scenario forecasts. The input window memory scales as

$$\mathcal{O}\left(BLW\sum_{m=1}^{M} d_m\right). \tag{84}$$

The fused RFF-KME sequence requires

$$\mathcal{O}(BLMD) \tag{85}$$

memory, and the GRU hidden sequence requires

$$\mathcal{O}(BLh). \tag{86}$$

At deployment, storing all scenario forecasts requires

$$\mathcal{O}(SBHd_x), \tag{87}$$

where d_x is the forecast output dimension. If memory is constrained, the scenario mean, variance, and violation probabilities can be accumulated online without storing all scenario trajectories.

7.3 Real-Time Deployment Considerations

In a practical autopilot stack, different modules can operate at different frequencies. The inner-loop attitude controller typically runs at the highest rate, the safety monitor and CBF filter run at an intermediate rate, and the learning-based forecasting module may run at a lower companion-computer rate. This rate separation is realistic because the CVAE forecast does not directly stabilize the vehicle; rather, it provides anticipatory information for planning and safety-margin adaptation.

The number of latent samples S provides a tunable trade-off between uncertainty quality and runtime cost. A small number of samples may be sufficient for fast onboard decision-making, while a larger number may be used for offline evaluation, calibration, or high-risk maneuvers. In deployment, adaptive sampling can be used: more samples are generated when forecast uncertainty or predicted risk is high, and fewer samples are used during nominal low-risk operation.

7.4 Generalization of the Forecasting Model

The generalization performance of the learned CVAE depends strongly on the diversity and quality of the training data. The model is expected to generalize better when the dataset contains diverse mission phases, wind conditions, obstacle geometries, battery states, speed regimes, and actuator behaviors. Generalization also improves when rare safety-critical events are sufficiently represented, including near-obstacle encounters, low-battery events, actuator saturation, GNSS-denied operation, and CBF intervention cases.

Explicit missing-data masks improve robustness to sensor dropout because the model can distinguish missing measurements from valid zero-valued measurements. Similarly, calibrated uncertainty labels or reliable empirical residuals improve probabilistic calibration. If actuator feedback and controller-mode labels are available, the control-regime embedding ψ_t^{ctrl} can capture meaningful actuation patterns rather than relying on proxy features.

It is important to distinguish predictive generalization from safety assurance. A learned forecasting model may degrade under distribution shift, unseen mission profiles, or sensor faults. For this reason, safety in the proposed architecture does not rely solely on the learned model. The CBF and voting layers remain active during deployment and provide deterministic supervision of the final command. Thus, the learned model improves foresight, but the safety layer determines command admissibility.

7.5 Limitations and Practical Implications

The proposed architecture is computationally tractable for moderate sequence lengths and forecast horizons, but several limitations should be noted. First, increasing the number of scenario samples improves uncertainty estimation but increases latency. Second, the learned latent space may not represent rare events well if those events are underrepresented in the training data.

Third, overly conservative uncertainty-inflated margins may reduce feasibility or cause excessive CBF intervention. Finally, the theoretical CBF guarantee assumes feasibility of the barrier constraints and sufficient low-level tracking authority.

These limitations suggest that practical deployment should combine model compression, adaptive sampling, uncertainty calibration, scenario-balanced data collection, and hardware-in-the-loop validation. In addition, the CBF layer should include fallback strategies such as emergency braking, loitering, return-to-home, or landing when the safe-control set becomes infeasible.

8 Experimental Setup

This section describes the dataset, preprocessing pipeline, model configuration, training protocol, baseline model, evaluation metrics, and simulation outputs used to assess the proposed learning-driven autopilot architecture. The experiments are designed to evaluate three aspects of the framework: (i) multimodal representation learning, (ii) probabilistic multistep forecasting, and (iii) forecast-driven safety-filtered control.

8.1 Dataset and Modalities

The experiments use a UAV navigation dataset containing 5,000 time-indexed samples. Each sample includes navigation variables, inertial measurements, environmental or obstacle-related information, battery state, and an obstacle-event indicator. The raw variables are grouped into three modality classes:

- **Navigation/perception modality:** local position, altitude, speed, and LiDAR distance.
- **IMU modality:** linear acceleration, angular-rate measurements, acceleration norm, and gyroscope norm.
- **Environment, power, and risk modality:** wind speed, battery level, clearance risk, battery risk, wind risk, speed risk, attitude-agitation score, overall risk, and obstacle-event indicator.

The raw latitude and longitude measurements are converted into local planar coordinates to provide a consistent local trajectory representation. Additional derived variables are computed to summarize motion intensity and operational risk. These derived features include acceleration norm, gyroscope norm, clearance risk, battery risk, wind risk, speed risk, attitude agitation, and overall risk. The overall risk score is used only as a risk-related forecasting target and diagnostic variable; it does not replace the CBF safety margins.

8.2 Preprocessing and Chronological Split

The dataset is sorted chronologically before training and evaluation. To avoid temporal leakage, normalization statistics are computed only from the training portion of the dataset and then applied to both training and test samples. Each modality is standardized independently using its training-set mean and standard deviation.

The data are divided using a chronological split, with the first 80% of rows used for training and the remaining 20% reserved for testing. This split reflects a realistic forecasting setting in which the model is trained on past flight data and evaluated on later unseen samples.

8.3 Rolling Window Construction

The model receives a sequence of recent multimodal context windows rather than a single instantaneous measurement. Let W_{raw} denote the number of raw samples in each local sensor

window, L_{seq} denote the number of fused context windows, and H denote the prediction horizon. In the experiments,

$$W_{\text{raw}} = 6, \qquad L_{\text{seq}} = 8, \qquad H = 6.$$

For each current time index t, the model constructs L_{seq} consecutive windows ending at times

$$t - L_{\text{seq}} + 1, \ldots, t.$$

Each such window contains W_{raw} raw samples. Thus, the input to the model contains short-term distributional information inside each window and temporal evolution across successive windows. The future target is the multistep trajectory

$$X_{t+1:t+H}.$$

The preprocessing produced 3,987 training examples and 995 test examples. Each example contains three modality-specific input tensors and one multistep forecasting target.

8.4 Forecast Targets

The deep CVAE is trained to predict five future variables over the horizon $H = 6$:

$$X_{t+1:t+H} = [\text{altitude}, \text{speed}, \text{LiDAR distance}, \text{battery level}, \text{overall risk}]_{t+1:t+H}. \tag{88}$$

These variables are selected because they are directly relevant to trajectory tracking, obstacle avoidance, energy awareness, and safety-margin computation. In particular, LiDAR-distance and battery-level forecasts are used by the CBF-inspired safety filter, while overall-risk forecasts influence the AI/MPC command and the voting weights.

8.5 Model Configuration

The proposed model uses modality-specific neural encoders followed by Random Fourier Feature Kernel Mean Embeddings (RFF-KME). Each modality is summarized using $D = 32$ random Fourier features. Since three modalities are used, the fused context representation has dimension 96 at each sequence step. A GRU-attention network then processes the fused sequence and produces a learned belief vector

$$c_t \in \mathbb{R}^{64}.$$

The CVAE latent dimension is set to

$$d_z = 16.$$

The decoder predicts both the mean and log-variance of the future trajectory. This allows the model to represent forecast uncertainty through both decoder variance and latent scenario sampling.

The learned belief vector is combined with a simulated control-regime embedding and a future-plan embedding to form the triad-conditioned autonomy representation. For the reported test-set diagnostics, the exported triad representation contains:

$$c_t \in \mathbb{R}^{64}, \qquad \kappa_t^{\text{ctrl}} \in \mathbb{R}^{27}, \qquad \psi_t^{\text{ctrl}} \in \mathbb{R}^{32}, \qquad p_t \in \mathbb{R}^{23}, \qquad \pi_t \in \mathbb{R}^{8}.$$

8.6 Training Protocol

The deep CVAE is trained for 10 epochs using the Adam optimizer with learning rate 10^{-3}. The training loss combines Gaussian negative log-likelihood, KL divergence, and a trajectory smoothness penalty:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \beta \mathcal{L}_{\text{KL}} + \lambda \mathcal{L}_{\text{smooth}}.$$

The KL weight is set to

$$\beta = 0.05,$$

and the smoothness weight is set to

$$\lambda = 0.002.$$

Gradient clipping is used to improve numerical stability during training. At test time, the recognition network is not used; instead, future trajectories are generated by sampling latent variables from the learned conditional prior.

8.7 Baseline Model

A deterministic Ridge regression baseline is used for comparison. The baseline receives flattened versions of the same standardized multimodal context windows and predicts the full multistep future trajectory as a deterministic output. This baseline is included to evaluate whether the deep CVAE provides improvement over a simple regularized linear forecasting model. Unlike the CVAE, the Ridge model does not produce latent scenarios or predictive uncertainty intervals.

8.8 Evaluation Metrics

Forecasting performance is evaluated using mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination R^2. Probabilistic forecasting quality is evaluated using empirical prediction-interval coverage. Specifically, the 80% predictive interval is computed from the CVAE scenario distribution and decoder uncertainty, and coverage is estimated as the fraction of test targets lying inside the interval.

The control simulation is evaluated using CBF intervention magnitude, obstacle-margin behavior, battery-margin behavior, voting-weight evolution, and closed-loop speed response. These diagnostics assess how forecast uncertainty influences the safety filter and command arbitration.

8.9 Simulation and Diagnostic Outputs

The simulation pipeline produces several diagnostic plots:

- a 3D UAV trajectory colored by overall risk;
- CVAE training-history curves;
- CVAE versus Ridge RMSE comparison;
- representative multistep forecast plots;
- one-step uncertainty plots for altitude, speed, LiDAR distance, battery level, and overall risk;
- GRU-attention weight visualization;
- PCA projection of the learned belief vector c_t;
- triad block-norm diagnostics;
- MPC/CBF/voting control simulation plots;

- CBF safety-margin and voting-weight diagnostics.

Together, these outputs evaluate forecasting accuracy, uncertainty quality, representation structure, and safety-filtered actuation behavior.

8.10 Reproducibility

All experiments use a fixed random seed for NumPy and PyTorch initialization. The chronological split, normalization procedure, window construction parameters, CVAE hyperparameters, and control-simulation parameters are reported to support reproducibility. The experiment is intended as a simulation-based validation of the proposed architecture rather than as a certified flight-test evaluation.

9 Results

This section reports the forecasting, representation-learning, and safety-filtered control results obtained from the proposed learning-driven autopilot pipeline. The evaluation focuses on four aspects: (i) multistep forecasting accuracy, (ii) probabilistic uncertainty estimates, (iii) learned representation diagnostics, and (iv) forecast-driven MPC–CBF–voting control behavior.

9.1 Forecasting Accuracy

The proposed deep CVAE was evaluated against a deterministic flattened-window Ridge regression baseline. Both models used the same chronological train–test split and the same multimodal context windows. The Ridge model predicts a deterministic future trajectory, whereas the CVAE predicts both a mean trajectory and an uncertainty estimate through latent scenario sampling and decoder variance.

Table 1 reports target-wise RMSE and empirical 80% interval coverage for the CVAE. The CVAE achieved a mean RMSE of 38.29 across the five forecast targets, compared with 38.92 for the Ridge baseline. The improvement is modest but consistent for most state variables. More importantly, the CVAE provides predictive intervals, which are not available from the deterministic baseline and are required by the proposed uncertainty-aware safety filter.

Table 1: Forecasting performance of the deep CVAE and Ridge baseline. The CVAE additionally reports empirical coverage of the nominal 80% prediction interval.

Target	CVAE RMSE	Ridge RMSE	CVAE 80% Coverage
Altitude	132.317	134.406	0.727
Speed	7.277	7.501	0.724
LiDAR distance	28.068	28.588	0.770
Battery level	23.696	23.993	0.733
Overall risk	0.109	0.110	0.844
Mean	38.293	38.920	0.760

The results suggest that the deep CVAE captures some nonlinear and multimodal structure in the sensor history while maintaining calibrated uncertainty intervals for several targets. The coverage values are close to, but generally below, the nominal 80% level for altitude, speed, and battery level, indicating mild undercoverage. The overall-risk forecast exhibits higher empirical coverage, suggesting that the uncertainty band for the aggregate risk variable is comparatively conservative.

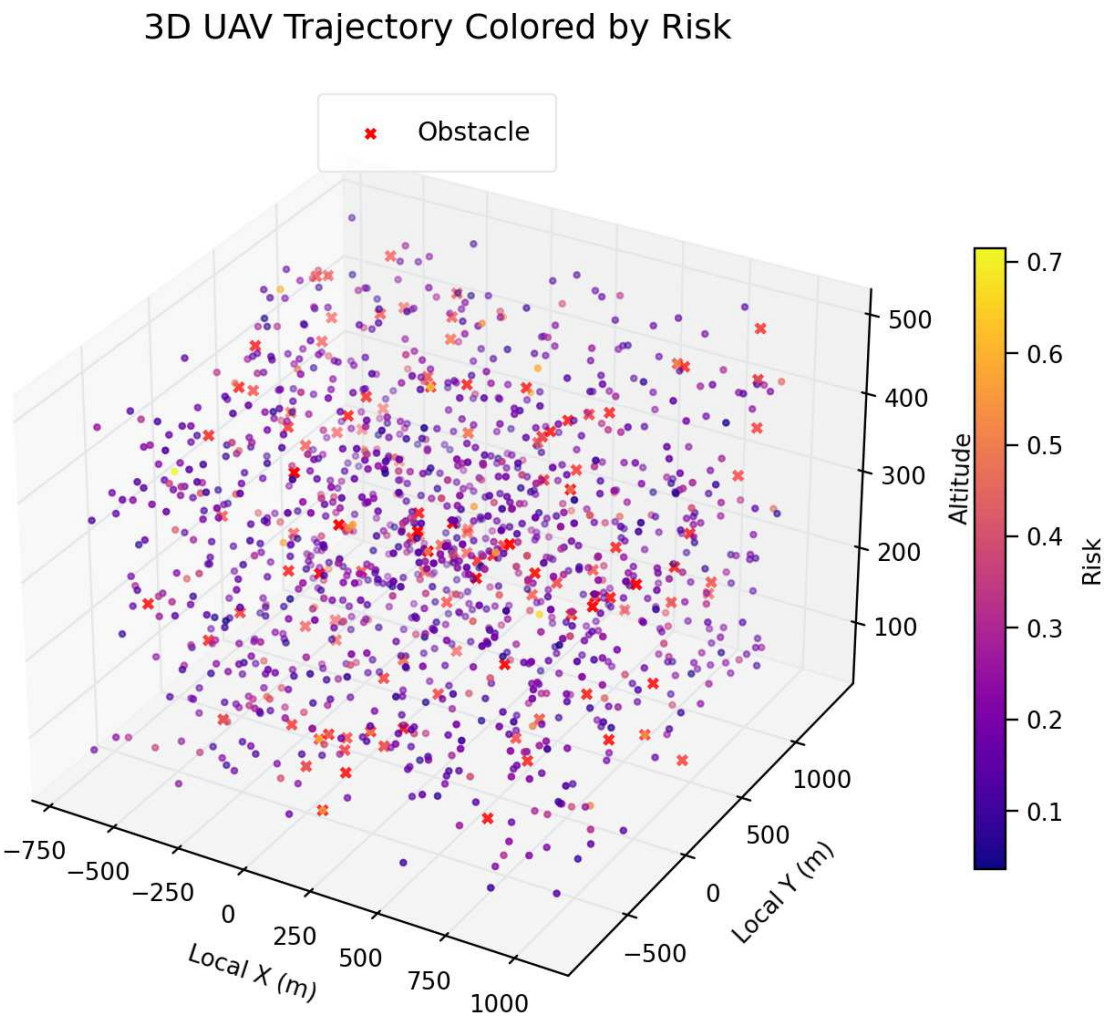


Figure 1: Three-dimensional UAV trajectory in a local coordinate frame. The horizontal axes represent local displacement in meters, the vertical axis represents altitude, and the color map indicates the computed overall risk score. Red markers indicate obstacle-detection events.

9.2 Trajectory-Level Risk Visualization

Figure 1 visualizes the UAV trajectory in a local coordinate frame and colors each point by the computed overall risk score. The figure provides a mission-level view of where higher-risk states and obstacle events occur along the trajectory. The spatial distribution of risk supports the need for continuous risk-aware forecasting rather than isolated event detection. Since risk depends on perception, vehicle state, battery condition, and recent dynamics, this plot also motivates the use of a multimodal belief representation.

9.3 CVAE Training Behavior

Figure 2 shows the training history of the deep CVAE. The total objective combines reconstruction negative log-likelihood, KL regularization, and trajectory smoothness. A stable decrease in the reconstruction component indicates that the decoder learns to predict future trajectories from the belief representation and latent variable. The KL term regularizes the latent space by aligning the recognition posterior with the conditional prior used during deployment.

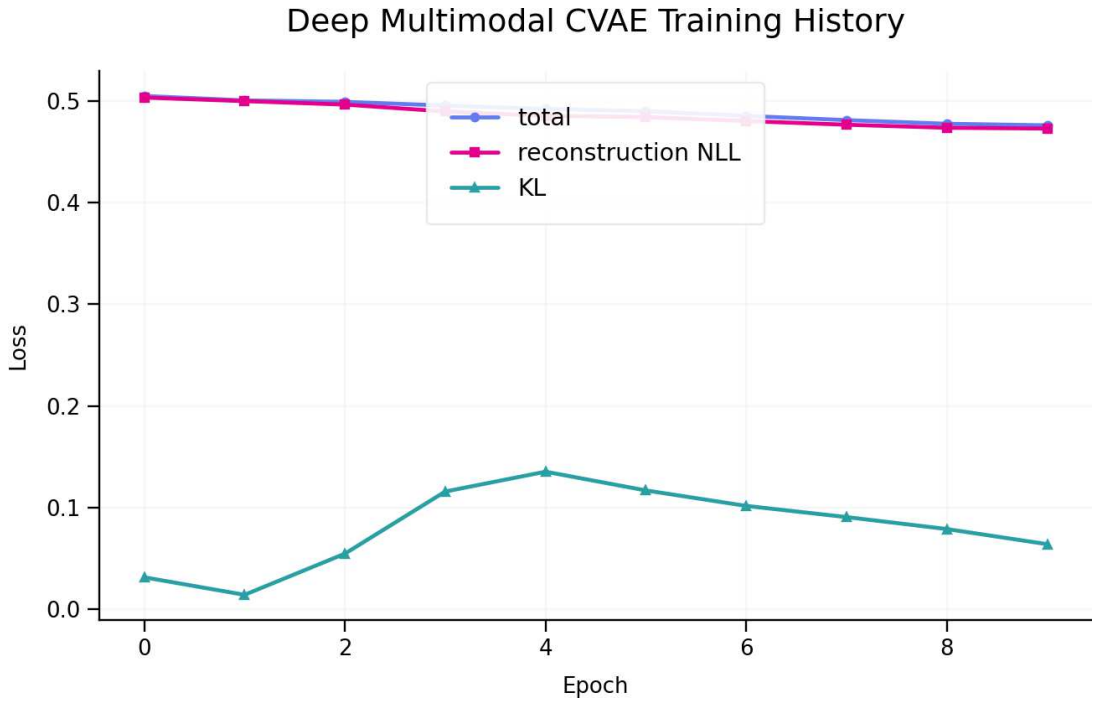


Figure 2: Training history of the deep multimodal CVAE. The total loss combines Gaussian reconstruction negative log-likelihood, KL divergence, and smoothness regularization.

9.4 Baseline Comparison

Figure 3 compares target-wise RMSE between the CVAE and the Ridge baseline. The CVAE achieves slightly lower RMSE for most targets. The largest absolute errors occur for altitude because the altitude variable has a much larger numerical scale than the normalized risk score. The comparison should therefore be interpreted together with uncertainty coverage and downstream control behavior, not only point-estimation error.

9.5 Representative Multistep Forecast

Figure 4 shows a representative six-step forecast for altitude, speed, and LiDAR distance. The CVAE predictive mean provides a smoothed estimate of the future trajectory, while the prediction interval represents uncertainty induced by latent scenario sampling and decoder variance. For safety-critical use, this interval is as important as the mean forecast because the CBF layer uses uncertainty to enlarge safety margins when future obstacle distance or battery state is uncertain.

9.6 One-Step Forecast Uncertainty

Figure 5 is placed at the beginning of this subsection because it introduces the one-step uncertainty analysis. Figures 5–9 report the one-step predictive mean and uncertainty intervals for altitude, speed, LiDAR distance, battery level, and overall risk, respectively.

Figures 5–9 show that the CVAE produces predictive distributions rather than only point forecasts. The LiDAR-distance and battery-level uncertainty estimates are particularly important because they are used directly by the CBF safety filter. When LiDAR-distance uncertainty increases, the effective obstacle-clearance requirement is increased. Similarly, battery uncertainty increases the effective battery threshold and reduces the admissibility of aggressive commands.

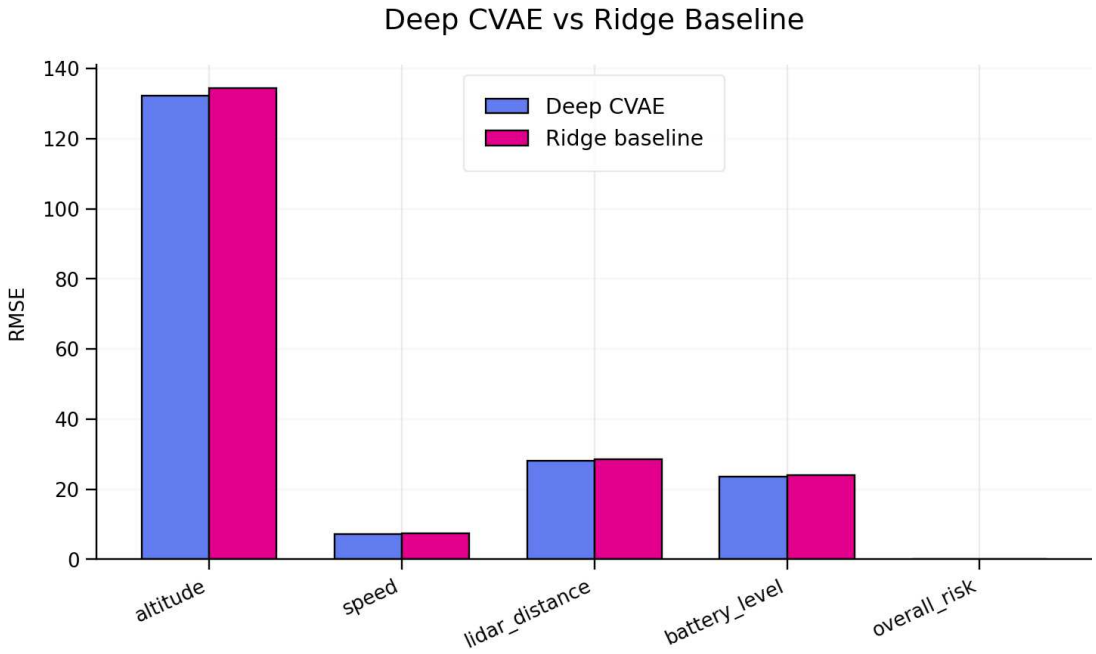


Figure 3: Forecasting RMSE comparison between the proposed deep CVAE and a flattened-window Ridge regression baseline.

9.7 Attention and Belief-Space Diagnostics

Figure 10 shows the attention weights assigned to recent multimodal context windows. The attention mechanism allows the encoder to emphasize windows that are more informative for forecasting. Figure 11 projects the learned belief vectors into two principal components and colors them by future risk. The embedding provides a qualitative diagnostic of whether the learned context representation organizes operating regimes according to risk-relevant structure. In the present dataset, the risk values are not perfectly separated, suggesting overlap among operating regimes and motivating richer future datasets with more explicit obstacle and actuator labels.

9.8 Triad Representation Diagnostics

Figure 12 shows the relative magnitudes of the three components of the triad representation. The belief vector norm is comparatively smooth, indicating that the GRU-attention encoder compresses multimodal context into a stable representation. The control-regime and plan-embedding norms vary more strongly, reflecting changes in simulated actuation intensity, safety margins, and plan structure. These diagnostics help identify whether the autonomy tensor is dominated by perception, recent control history, or future-plan information during different operating regimes.

9.9 Forecast-Driven MPC, CBF, and Voting Control

Figure 13 shows the forecast-driven control simulation. The AI/MPC channel proposes commands using CVAE forecasts, while the safety channel imposes a conservative command envelope. The CBF layer reduces the voted command when the predicted obstacle or battery margins become unsafe. The CBF intervention curve quantifies how much the safety filter modifies the voted command before actuation.

Figure 14 provides additional safety diagnostics. The obstacle margin h_{obs} indicates whether the predicted obstacle-clearance constraint is active. The battery margin h_{battery} indicates

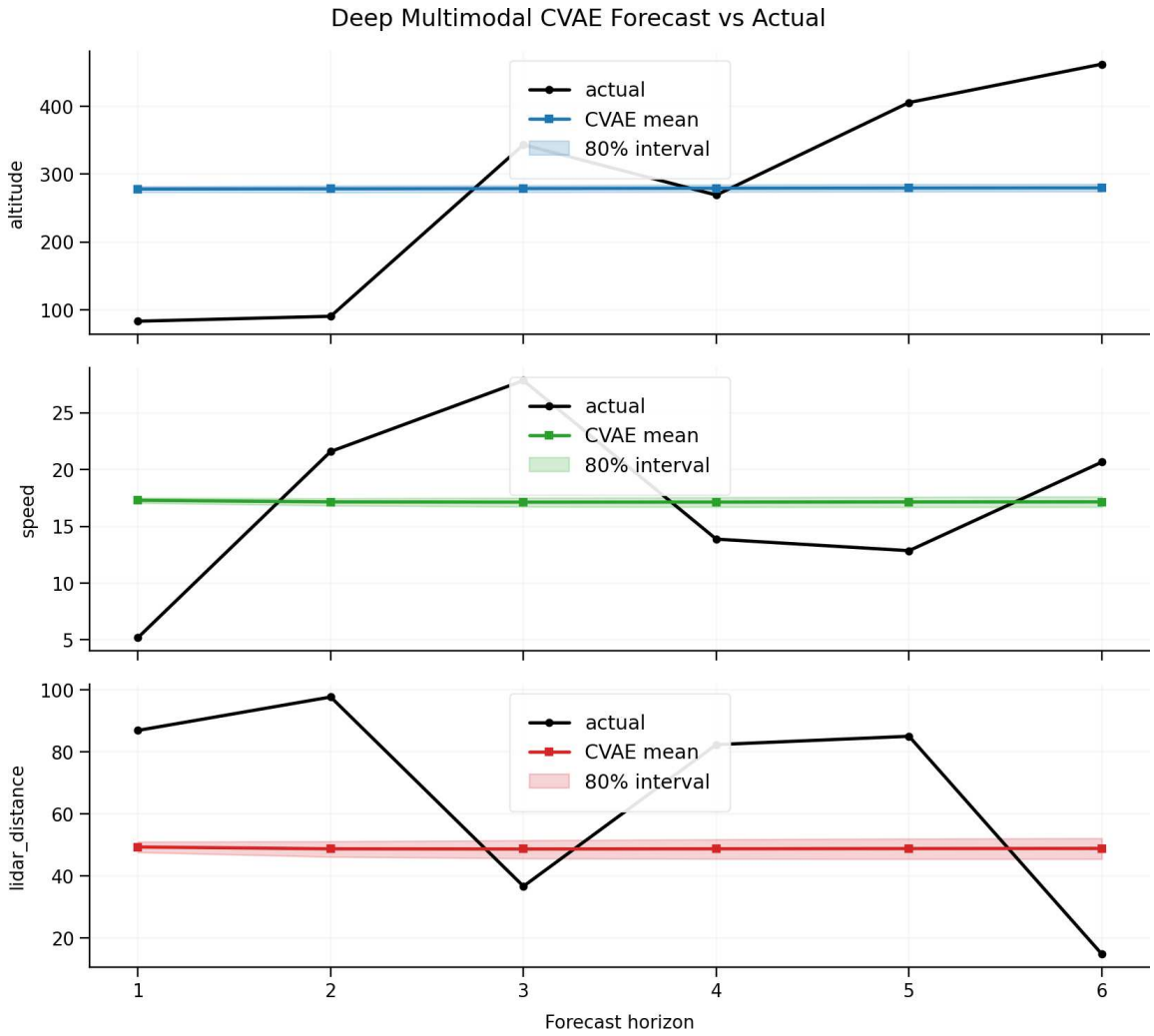


Figure 4: Representative multistep forecast from the deep CVAE. The plot shows actual and predicted trajectories for altitude, speed, and LiDAR distance over the forecast horizon.

whether energy safety is limiting. The voting weights show how authority is distributed among the primary controller, safety controller, and AI/MPC companion. When predicted risk or uncertainty increases, the AI/MPC weight is reduced and the safety channel becomes more influential.

Overall, the results demonstrate the intended separation of responsibilities in the proposed architecture: the CVAE provides probabilistic foresight, MPC converts forecasts into nominal commands, voting supervises heterogeneous command proposals, and the CBF layer enforces safety before actuation.

10 Discussion

The results demonstrate that deep probabilistic forecasting can be integrated into an autopilot architecture without bypassing deterministic safety mechanisms. The CVAE provides uncertainty intervals and slightly improves forecasting relative to the Ridge baseline. The CBF layer uses forecast uncertainty to inflate safety margins and reduce unsafe commands. The voting controller further reduces reliance on the AI/MPC channel when predicted risk or uncertainty is high.

The modest performance gain also indicates that dataset quality is central. Richer datasets

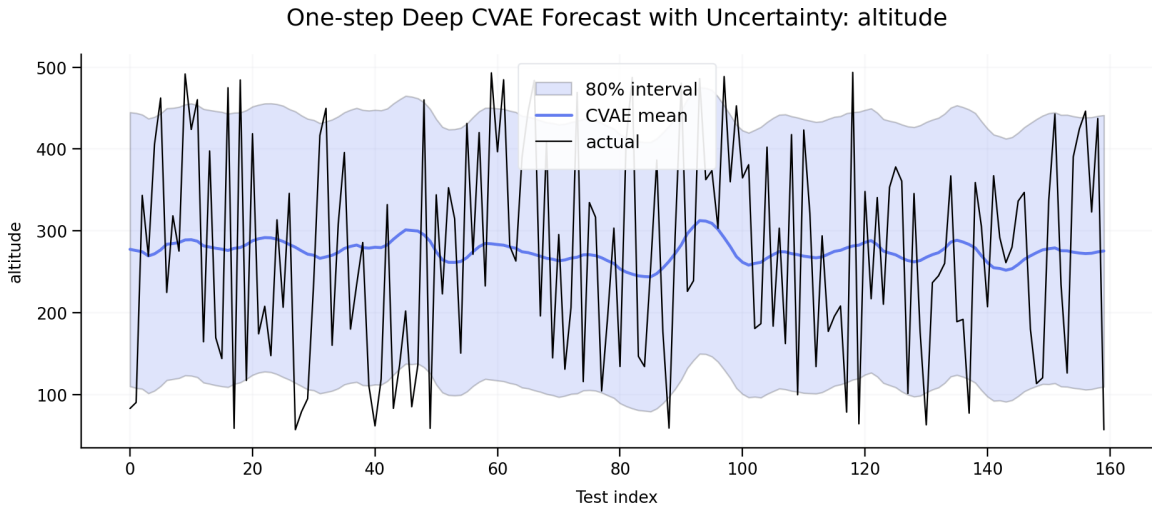


Figure 5: One-step altitude forecasting with uncertainty. The plot shows the observed altitude, the CVAE predictive mean, and the 80% uncertainty interval.

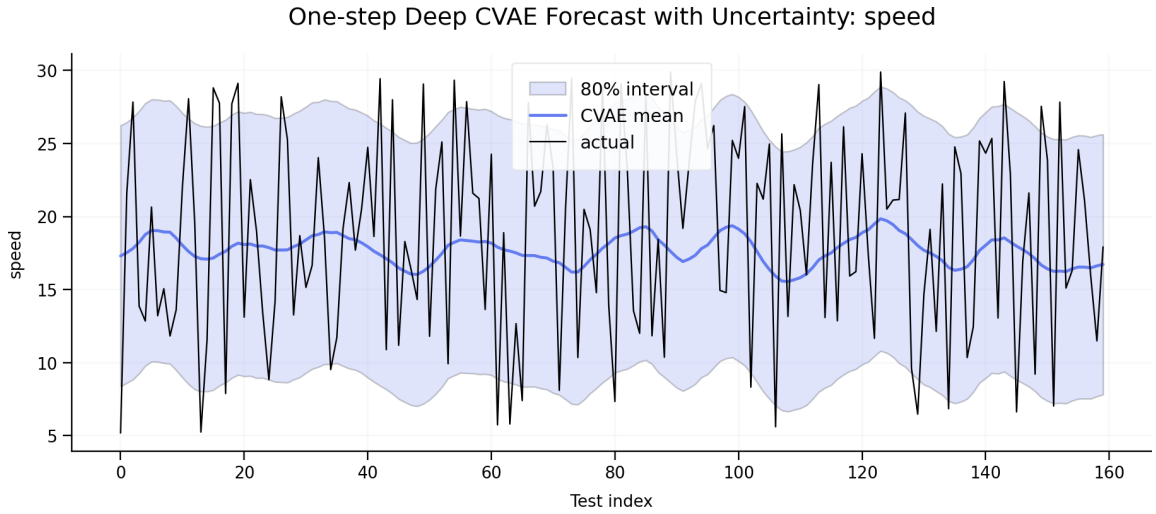


Figure 6: One-step speed forecasting with uncertainty. The plot shows the observed speed, the CVAE predictive mean, and the 80% uncertainty interval.

should include ESC telemetry, motor commands, actuator saturation, GNSS covariance, VIO confidence, obstacle geometry, controller modes, MPC feasibility labels, and CBF intervention labels. Such data would allow the triad representation to encode physically meaningful perception-control-plan interactions.

11 Conclusion

This paper presented a learning-driven autopilot architecture integrating multimodal perception, triad-conditioned CVAE forecasting, risk-aware MPC, CBF safety filtering, and triple-redundant voting. The CVAE provides probabilistic foresight, MPC proposes feasible controls, CBF enforces safety, and the classical controller executes safe actuation. The architecture is suitable for UAV autonomy, GNSS-denied navigation, field robotics, and autonomous irrigation systems.

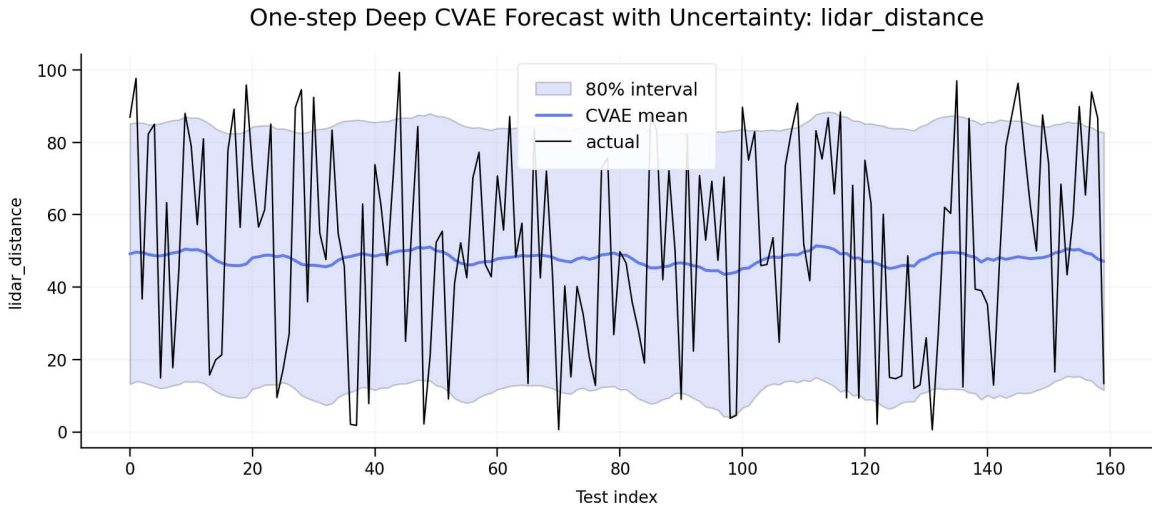


Figure 7: One-step LiDAR-distance forecasting with uncertainty. The uncertainty band is used by the CBF layer to inflate the effective obstacle-clearance margin.

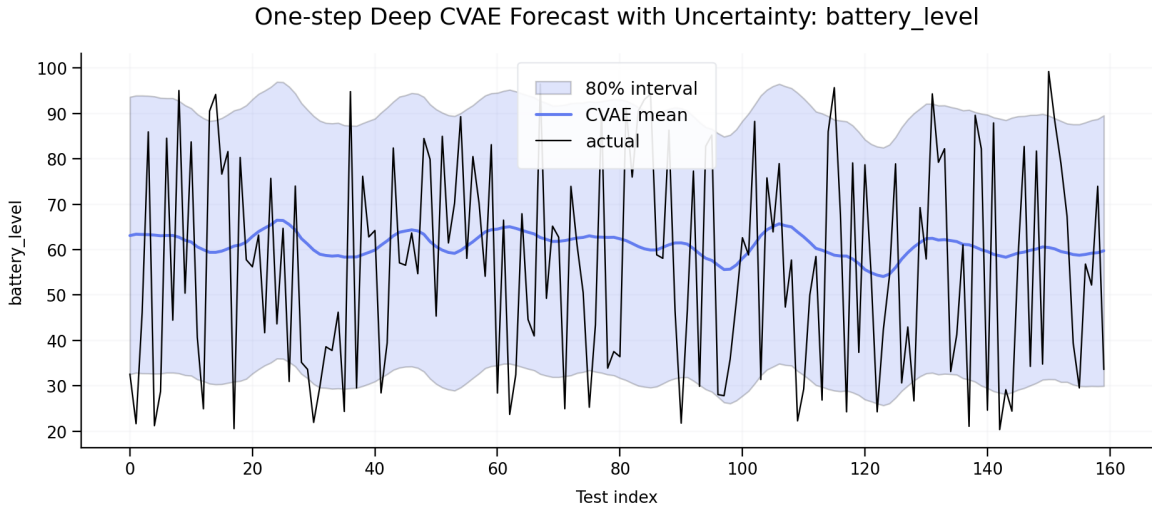


Figure 8: One-step battery-level forecasting with uncertainty. The uncertainty band is used to form an uncertainty-aware battery safety margin.

Funding

This work was supported by SEED MONEY SCHEME under Grant $DRIE/SMS/DRIE - 10954/2025 - 26/2847$.

Author Contributions

Lakshman Mahto: Conceptualization, methodology, mathematical formulation, simulations, writing, validation.

References

[1] Ames, A. D., Xu, X., Grizzle, J. W., & Tabuada, P. (2017). Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control, 62*(8), 3861–3876. https://doi.org/10.1109/TAC.2016.2638961

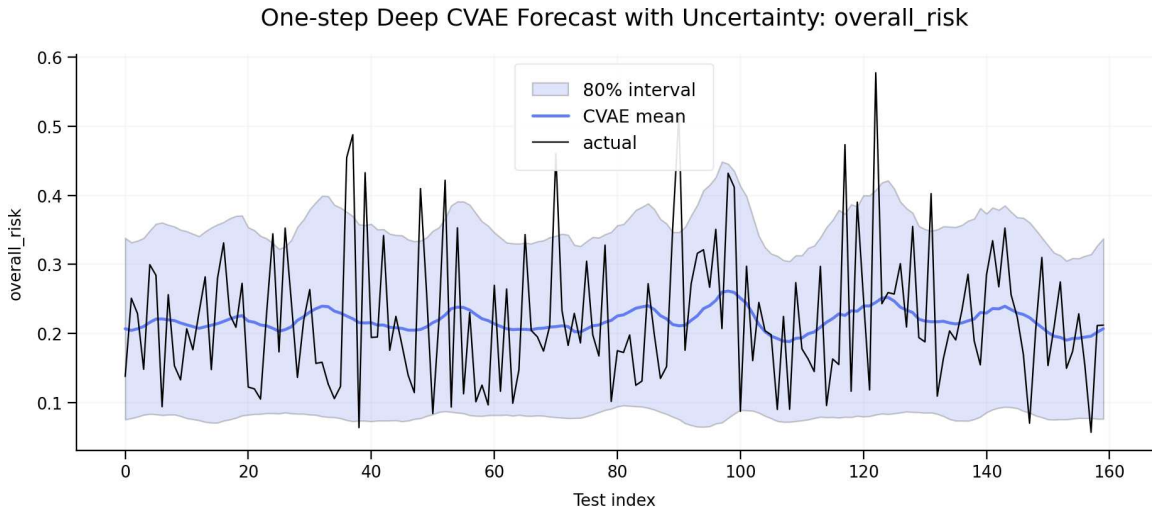


Figure 9: One-step overall-risk forecasting with uncertainty. The predicted risk influences the AI/MPC command and the voting weight assigned to the learning channel.

[2] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., & Tabuada, P. (2019). Control barrier functions: Theory and applications. In *Proceedings of the European Control Conference* (pp. 3420–3431). https://doi.org/10.23919/ECC.2019.8796030

[3] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

[4] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1724–1734). https://doi.org/10.3115/v1/D14-1179

[5] Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

[6] Lyons, R. E., & Vanderkulk, W. (1962). The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development, 6*(2), 200–209. https://doi.org/10.1147/rd.62.0200

[7] Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica, 36*(6), 789–814. https://doi.org/10.1016/S0005-1098(99)00214-9

[8] Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*.

[9] Rawlings, J. B., Mayne, D. Q., & Diehl, M. (2017). *Model predictive control: Theory, computation, and design* (2nd ed.). Nob Hill Publishing.

[10] Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*.

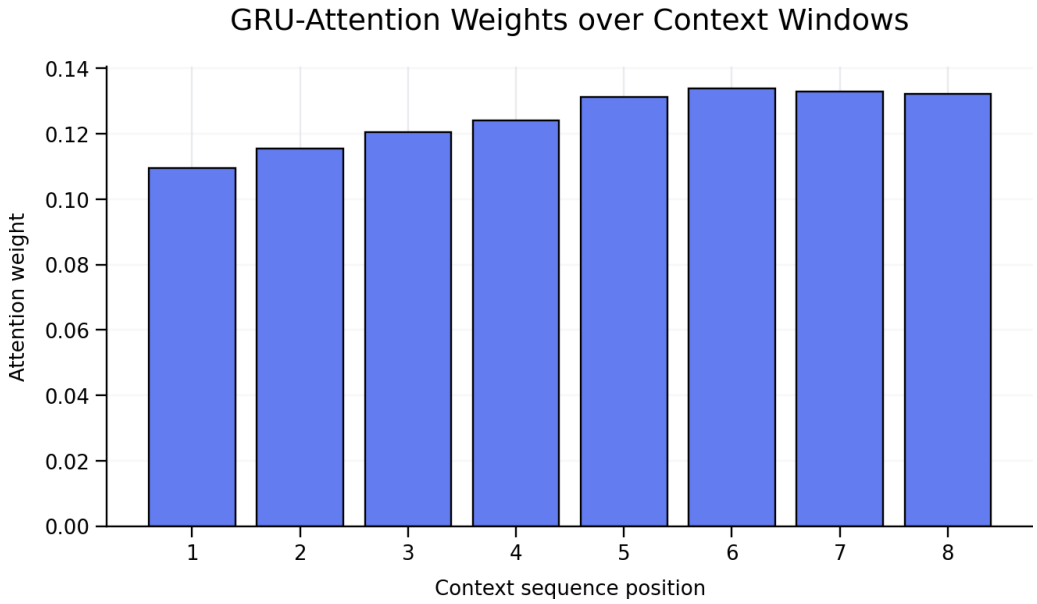


Figure 10: GRU-attention weights over recent multimodal context windows. Larger weights indicate context windows that contribute more strongly to the learned belief vector.

A Pseudocode

Algorithm 1 Multimodal Window Construction

Require: Sensor table $\mathcal{D}$, raw window length W_{raw}, context length L_{seq}, horizon H

Compute local coordinates and derived risk variables.
Standardize each modality using training rows only.
for $t = W_{\text{raw}} + L_{\text{seq}} - 2$ to $N - H$ **do**
  for $s = t - L_{\text{seq}} + 1$ to t **do**
    Extract navigation, IMU, and environment windows.
  end for
  Set future target $X = Y_{t+1:t+H}$.
end for
return Multimodal context-target pairs.

Algorithm 2 Deep CVAE Training

Require: Context windows and future trajectory X

Compute RFF-KME modality summaries.
Compute belief vector c_t with GRU-attention.
Compute prior $(\mu_p, \log \sigma_p^2) = P_\theta(c_t)$.
Compute posterior $(\mu_q, \log \sigma_q^2) = Q_\phi(c_t, X)$.
Sample $z = \mu_q + \sigma_q \odot \epsilon$.
Decode $(\mu_X, \log \sigma_X^2) = D_\theta(c_t, z)$.
Minimize $\mathcal{L}_{\text{rec}} + \beta \mathcal{L}_{\text{KL}} + \lambda \mathcal{L}_{\text{smooth}}$.

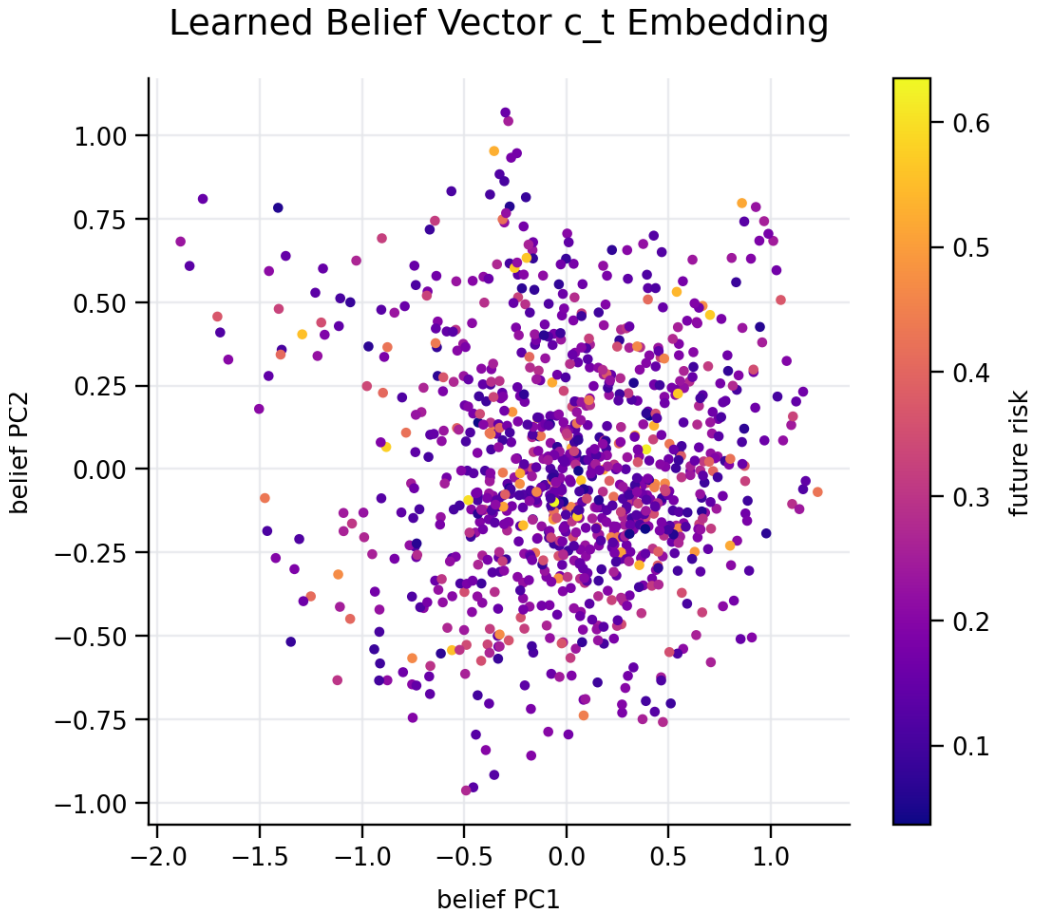


Figure 11: Two-dimensional PCA projection of the learned belief vector $c_t \in \mathbb{R}^{64}$, colored by future overall risk.

Algorithm 3 Forecast-Driven MPC/CBF/Voting Control

Require: Belief c_t, CVAE forecast, commands u_A, u_B, u_C

1: Estimate forecast mean and uncertainty from CVAE samples.
2: Compute voting weights w_A, w_B, w_C.
3: Compute u_{vote} using median and weighted voting.
4: Compute h_{obs} and h_{battery}.
5: Reduce $u_{\max}^{\text{safe}}$ if margins are unsafe.
6: Output $u_{\text{safe}} = \text{clip}(u_{\text{vote}}, 0, u_{\max}^{\text{safe}})$.

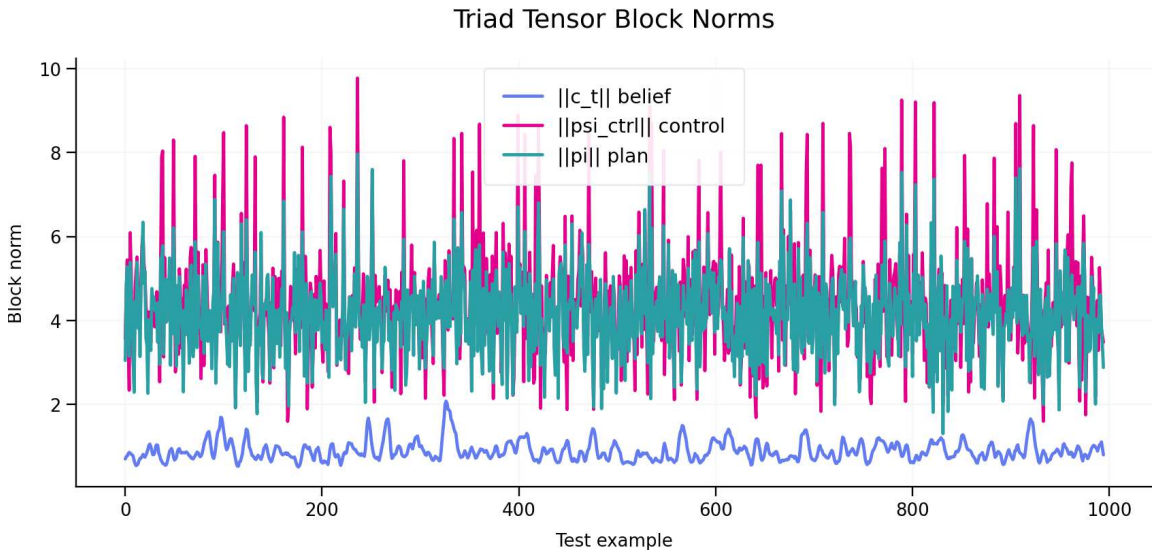


Figure 12: Evolution of triad tensor block norms over the test segment. The three curves correspond to the belief norm $\|c_t\|$, the control-regime norm $\|\psi_t^{\text{ctrl}}\|$, and the plan-embedding norm $\|\pi_t\|$.

B Hyperparameters

Table 2: Data and model hyperparameters.

Parameter	Value	Description
W_{raw}	6	Raw sensor window length
L_{seq}	8	Context sequence length
H	6	Forecast horizon
RFF dimension	32	Per modality
GRU hidden dimension	64	Context encoder
Belief dimension	64	Dimension of c_t
Latent dimension	16	CVAE latent variable z
Epochs	10	Training epochs
Learning rate	10^{-3}	Adam optimizer
KL weight β	0.05	Latent regularization
Smoothness weight λ	0.002	Forecast smoothness

Table 3: Control and safety hyperparameters.

Parameter	Value	Description
Target speed	18.0	Nominal speed reference
Target altitude	300.0	Nominal altitude reference
Base obstacle margin	12.0	Nominal clearance term
LiDAR uncertainty gain	1.25	Safety margin inflation
Risk gain	6.0	Risk margin multiplier
Battery threshold	25.0	Minimum battery level
Battery uncertainty gain	0.1	Battery margin inflation
AI weight range	$[0.05, 0.35]$	Channel C voting weight

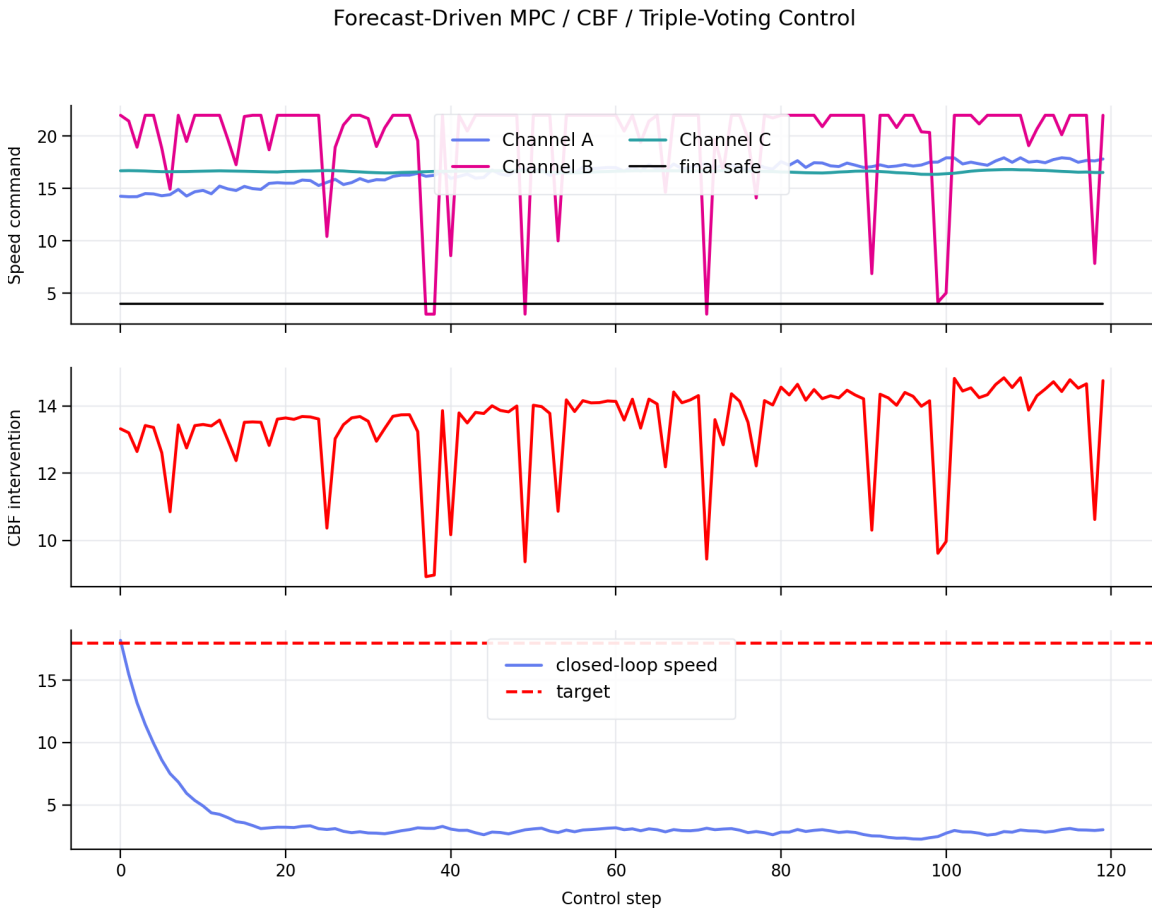


Figure 13: Forecast-driven MPC, CBF, and triple-voting control simulation. The figure compares the primary command, safety command, AI/MPC proposal, final CBF-filtered command, CBF intervention, and resulting closed-loop speed response.

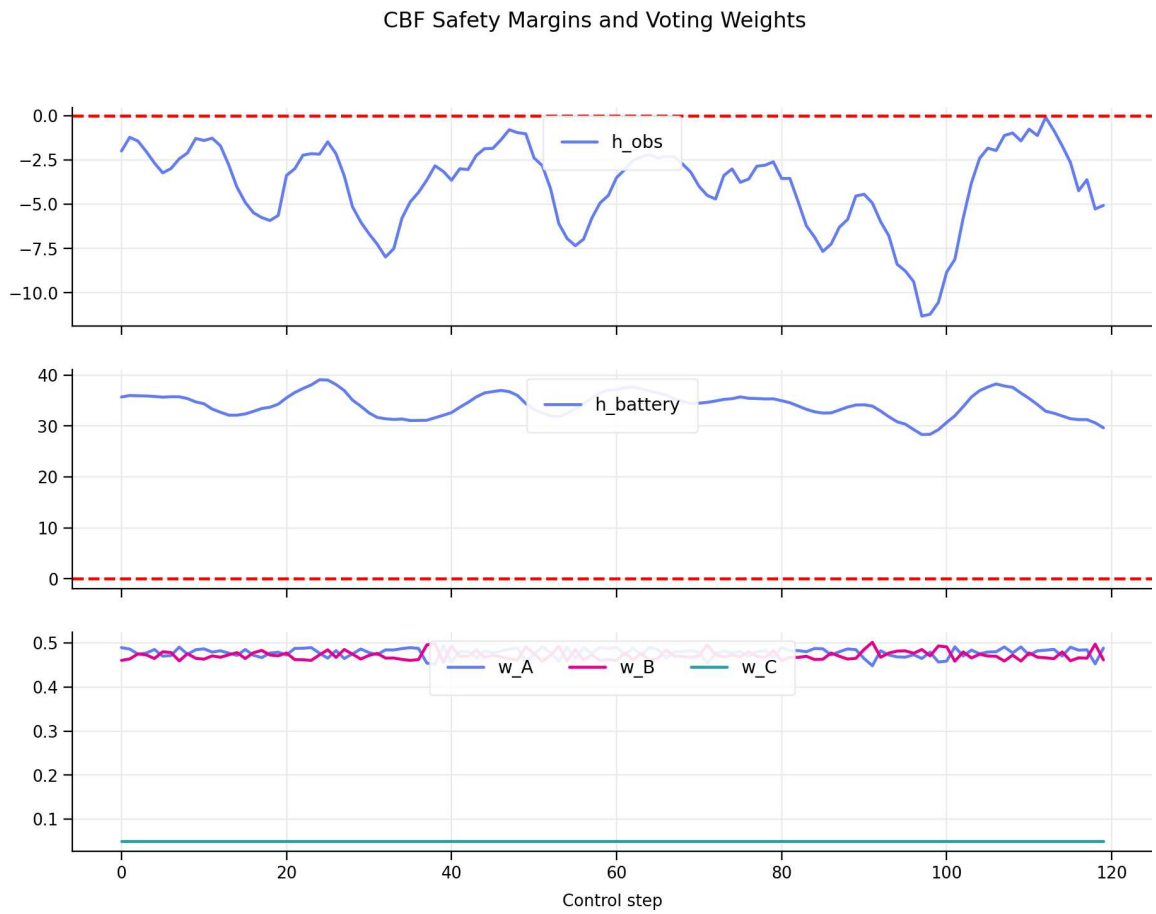


Figure 14: CBF safety margins and confidence-weighted voting diagnostics. The figure shows obstacle margin, battery margin, and voting weights for the primary, safety, and AI/MPC channels.