IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.DOI

Autonomous Aircraft Tactical Pop-Up Attack Using Imitation and Generative Learning

JOAO P. A. DANTAS[1], MARCOS R. O. A. MAXIMO[1], TAKASHI YONEYAMA[1]
[1]Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, 12288-900, Brazil

Corresponding author: Joao P. A. Dantas (e-mail: jpdantas@ita.br).

The Article Processing Charge for the publication of this research was funded by the Coordination for the Improvement of Higher Education Personnel (CAPES) - Brazil (ROR identifier: 00x0ma614). For open access purposes, the authors have assigned the Creative Commons CC BY license to any accepted version of the article. The work of Takashi Yoneyama and Marcos R. O. A. Maximo was supported in part by the National Research Council of Brazil (CNPq) through grants 304134/2-18-0 and 307525/2022-8, respectively.

ABSTRACT This study presents a methodology for developing models that replicate the complex pop-up attack maneuver in air combat operations, using flight data from a Brazilian Air Force pilot in a 6-degree-of-freedom flight simulator. By applying imitation learning techniques and comparing three algorithms – Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) – the research trains models to predict aircraft control inputs through sequences of state-action pairs. The performances of these models were evaluated in terms of Root Mean Squared Error (RMSE), coefficient of determination (R^2), training time, and inference time. To further enhance the training dataset with the aim of improving the robustness of the models, a Variational Autoencoder (VAE) was employed to generate synthetic data. These findings demonstrate the potential for deploying such models in fully autonomous aircraft, enhancing autonomous combat systems' reliability and operational effectiveness in real-world scenarios.

INDEX TERMS Air combat operations, generative learning, imitation learning, pop-up attack.

I. INTRODUCTION

The integration of autonomous systems into modern warfare is rapidly increasing, offering enhanced capabilities and operational efficiencies by taking over roles that traditionally required human involvement [1]. These systems are deployed across various military applications, from surveillance and reconnaissance to direct combat engagement, where they provide significant strategic benefits [2]. However, a key challenge in their development remains to enable them to execute complex tasks, such as air combat maneuvers, with the proficiency of human pilots [3].

A pop-up attack, for instance, is a maneuver where a fighter aircraft quickly ascends from a low altitude to engage a ground target, followed by a rapid descent to avoid counterattacks. This maneuver is a critical part of air combat operations and involves elements of surprise and precise execution under extreme conditions, making it particularly challenging to replicate autonomously [4].

To address this challenge, imitation learning, specifically Behavior Cloning (BC), provides a promising approach for training autonomous systems. BC allows agents to learn by mimicking expert demonstrations, capturing human decision-making and execution skills from collected flight data [5]. This method is particularly effective for structured tasks where expert strategies are well-established, such as air combat maneuvers. In this work, BC is used to enable autonomous agents to replicate complex maneuvers like the pop-up attack with high fidelity [6].

While reinforcement learning could also be applied to this problem, it introduces challenges such as the need for carefully designed reward functions and long training times for convergence [7]. Although reinforcement learning can discover novel strategies through trial and error, its learning process is often unstable and computationally expensive. Nonetheless, RL has already been applied in air combat scenarios, demonstrating its potential for maneuver optimization and tactical decision-making [8], [9]. In contrast, BC follows a supervised learning approach, making training more stable and data-efficient [10]. Moreover, when expert demonstrations are available, imitation learning enables

models to directly replicate successful strategies rather than searching for optimal actions independently. This advantage is particularly relevant in air combat, where decision-making is guided by established tactics and prior knowledge. Given these factors, BC was chosen as the primary method for training autonomous agents in this work, ensuring they can effectively learn from expert pilots and execute maneuvers with precision and reliability.

In addition to imitation learning, this work incorporates generative learning techniques [11] to generate synthetic flight data based on flight data from human pilots collected from a simulation environment. The synthetic data aims to improve the model's performance, enabling better generalization and enhancing the ability of autonomous systems to perform complex air combat maneuvers. This approach helps mitigate issues related to data limitations, which are common when working with complex military operations where collecting large amounts of real data may be impractical or costly [12].

The main contribution of this study is the development of models for autonomous pop-up attack maneuvers using imitation learning, specifically BC. By training these models on flight data collected from human pilots in a simulation environment, the study aims to replicate human decision-making in air combat operations. Additionally, synthetic data generation using generative learning is introduced to expand the dataset, enhancing the robustness of the models. This work enhances our understanding of autonomous systems' ability to perform reliably in dynamic and unpredictable combat scenarios, emulating the proficiency of human pilots.

The remainder of this work begins with Section II, which provides an operational background on the pop-up attack maneuver, a key technique in air-to-ground combat. Following this, in Section III, we present an overview of related work, surveying key contributions in the fields of imitation and generative learning applied to autonomous systems, with a particular focus on the air domain. In Section IV, the study details the data collection process, emphasizing the role of a flight simulator model to capture the maneuver's complexities. This section also describes the development of multiple BC models using different techniques and explores generative learning methods to synthesize additional flight data. In Section V, we present the results and provide a performance analysis of the models across various configurations. Finally, Section VI concludes the work by discussing the research outcomes and suggesting future directions.

II. OPERATIONAL BACKGROUND

The pop-up attack maneuver is a key technique in air-to-ground combat, designed to enable a fighter aircraft to approach and engage a target while minimizing its exposure to enemy defenses [13]. This maneuver is initiated from a preplanned Pop-Up Point (PUP), which is strategically selected to optimize the aircraft's approach path and timing for the attack sequence [4]. As depicted in Figure 1, the offset pop-up maneuver typically involves an approach angle ranging from $15°$ to $90°$ relative to the final attack heading. This angular approach allows the pilot to visually acquire the target early and maintain visual contact until the weapon is released.

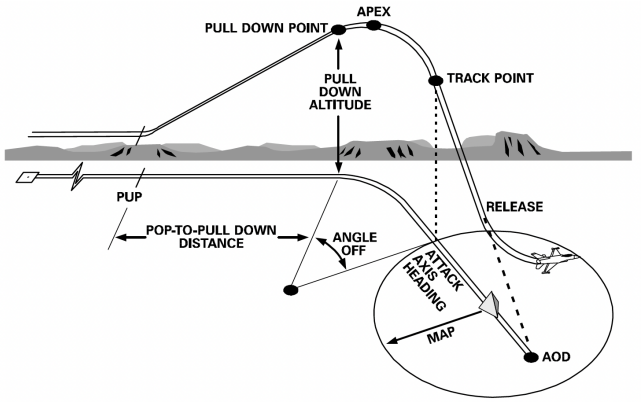

FIGURE 1. Flight profile for offset pop-up delivery. Source: [14]

The maneuver begins as the aircraft reaches the PUP, with the pilot initiating a climb at a minimum speed of 450 knots calibrated speed. At this point, the pilot selects the appropriate power setting and executes a $3-4\,G$ wings-level pull-up to achieve the planned climb angle. Simultaneously, a chaff/flare countermeasure program is activated, which releases metallic strips (chaff) and infrared flares to confuse radar-guided and heat-seeking missiles, respectively, protecting the aircraft from potential surface-to-air threats. During the climb, the pilot must precisely maintain the planned climb angle while carefully monitoring the altitude gain [13], [14].

As the aircraft approaches the predetermined pull-down altitude, the pilot executes a smooth roll towards the target, followed by a $3-5\,G$ pull-down maneuver to intercept the planned dive angle. During this phase, the pilot passes through the track point, where the aircraft's trajectory is adjusted to ensure alignment with the target [14]. Achieving this dive angle while aligning with the aim-off point (AOD), which is the ground distance from the target where the nose of the aircraft is pointed during tracking, is essential for maintaining the preplanned delivery parameters. Minor deviations in the attack heading are generally acceptable and can be corrected during the final phase of the maneuver [13].

Throughout the maneuver, the pilot must make real-time adjustments for any deviations in the pop-up point or unanticipated wind conditions encountered during the climb. The apex, the highest altitude reached in the pop-up delivery profile, is typically reached approximately halfway through the pull-down maneuver, providing the necessary altitude and positioning to execute the attack effectively. Additionally, the distance from the final maneuver point to the target (MAP) must be carefully managed as it combines bomb range and horizontal distance covered while tracking [13].

Given the precision required at each stage, the pop-up attack maneuver is a highly complex and demanding technique.

Successfully executing this maneuver presents significant challenges, especially for autonomous systems, which must replicate the decision-making and real-time adjustments that human pilots perform under dynamic conditions.

III. RELATED WORK

Research on air combat operations has long recognized the complexity and dynamic nature of these military engagements. These operations require rapid decision-making, precise execution, and the ability to adapt to evolving threats in real-time [15]. Historically, human pilots have been the base of air combat success, leveraging their skills and decision-making abilities to outmaneuver adversaries. However, with advancements in autonomous systems, there is a growing focus on replicating and enhancing these capabilities through automation [16], [17]. Integrating autonomous technologies into air combat offers potential benefits such as increased mission effectiveness, reduced cognitive load on operators, and improved survivability, though it also presents challenges in developing systems that can emulate human decision-making in critical environments [18].

In this context, recent research has increasingly focused on developing autonomous systems capable of executing complex air combat maneuvers. These systems are designed to handle a broad range of missions, including both close-range air-to-air engagements (dogfighting) and beyond-visual-range (BVR) combat, where detecting, tracking, and engaging adversaries occur at long distances [19]. Autonomous systems capable of excelling in air-to-air [20], [21] and ground-to-air combat [22], as well as in air-to-ground tactical operations, as presented in this study, represent a significant advancement in modern military aviation [23].

Imitation learning has emerged as a highly effective method for training autonomous systems by enabling them to mimic the decision-making processes of experienced pilots. For instance, deep feature representation has been used to map flight observations to continuous control actions in autonomous helicopters, showcasing the feasibility of transferring expert-level skills to autonomous systems [24]. Similarly, intelligent autopilot systems that learn piloting skills through imitation have successfully replicated both low- and high-level flight skills, including complex maneuvers [25]. Further work has explored the cloning of fighter pilot strategies through imitation learning, allowing autonomous systems to replicate intricate combat tactics, proving effective in replicating decision-making patterns under combat conditions [26]. Additionally, the DAgger algorithm has been applied to train neural network-based autopilots in unmanned aerial systems (UAS), showing significant improvements in generalization across diverse flight maneuvers and proving effective in maintaining flight stability and adaptability during complex aerial tasks [27].

Recent research has extended imitation learning to air combat scenarios. For instance, exploratory studies have highlighted its potential for modeling fighter pilot behavior, enabling autonomous agents to replicate combat tactics and decision-making strategies under dynamic and uncertain environments [28]. This includes the development of methods to capture the complex behaviors of pilots and translate them into effective autonomous system strategies. Complementary work has focused on data-driven behavioral modeling for military applications, emphasizing the integration of imitation learning techniques to replicate the nuanced decision-making processes of expert operators in defense scenarios [29]. These studies demonstrate the critical role of imitation learning in enhancing the operational capabilities of autonomous systems, both in civilian and military aviation contexts, by improving their ability to adapt to complex and high-stakes situations.

Generative learning has been explored as a complementary approach to imitation learning, particularly for data augmentation in training models. For example, generative models have been employed to create motion control policies, enabling autonomous systems to generalize to new scenarios with synthesized data [30]. Additionally, Generative Adversarial Networks (GANs) have been applied to iteratively improve data efficiency in reinforcement learning, enhancing model performance [31]. Another approach uses a GAN-based training model to generate high-quality synthetic data for lightweight convolutional neural networks, addressing the shortage of training data and improving classifier accuracy [32]. Generative models have also been utilized to improve the robustness of object detection systems in low-visibility environments, such as in [33], where synthetic data augmentation effectively counters natural perturbations like low light and blur, improving model robustness in real-world scenarios.

Synthetic data has also been explored in military decision support systems, focusing on its fidelity and applicability in real-world scenarios. For instance, [34] evaluates the effectiveness of synthetic data in replicating operational conditions and enhancing decision-making processes in complex environments. This work highlights the importance of assessing synthetic data quality to ensure its reliability in augmenting training datasets and supporting robust system performance in military applications. Also, [35] introduced resampling techniques to generate synthetic data to address imbalanced data, which improved model reliability for supervised learning in autonomous aircraft systems in the context of aerial combat.

Previous work has examined the pop-up attack maneuver as an optimization-based approach to tactical mission planning, emphasizing weapon delivery precision, ballistic trajectory control, and detection avoidance, thus serving as a critical foundation for applications in both human-piloted and autonomous aircraft systems [4]. Additionally, intent inference models have been applied in air defense contexts, using flight profile analysis to predict potential weapon delivery points in pop-up scenarios, thereby enabling proactive threat assessment and enhancing situational awareness [14].

IV. METHODOLOGY

This section outlines the methodological framework employed in this work, focusing on developing both imitation and generative learning models. The following subsections detail the characteristics of the flight data and the approach used to develop and evaluate multiple imitation learning models. Additionally, this section introduces a generative learning technique to produce synthetic data, aiming to expand the dataset and enhance the robustness of the models.

A. FLIGHT DATA

The dataset for this research comprises 30 flight recordings of pop-up attack maneuvers executed by a Brazilian Air Force (FAB) fighter pilot. The flight data was collected using AEROGRAF [36], a 6DOF (Six Degrees of Freedom) flight simulator model based on the F-16 Fighting Falcon aircraft, which was developed by the FAB. AEROGRAF served as the predecessor to the Aerospace Simulation Environment (ASA), also developed by the FAB [37], [38].

All flights in the dataset start from exactly the same initial point and heading, located 5.9 nautical miles from the target and with an altitude difference of about 146 meters, thus providing a standardized initial condition for each maneuver. These distances, altitudes, and headings were determined in collaboration with subject matter experts to represent a common operational scenario for this maneuver.

To ensure uniformity, each flight recording was trimmed to match the shortest sequence length across all samples. This process standardizes the data, enabling consistent inputs for model training and reducing variability in training data, which supports more reliable model performance. Figure 2 illustrates the flight patterns within this dataset, showing the trajectories for each of the 30 flights utilized in this study.

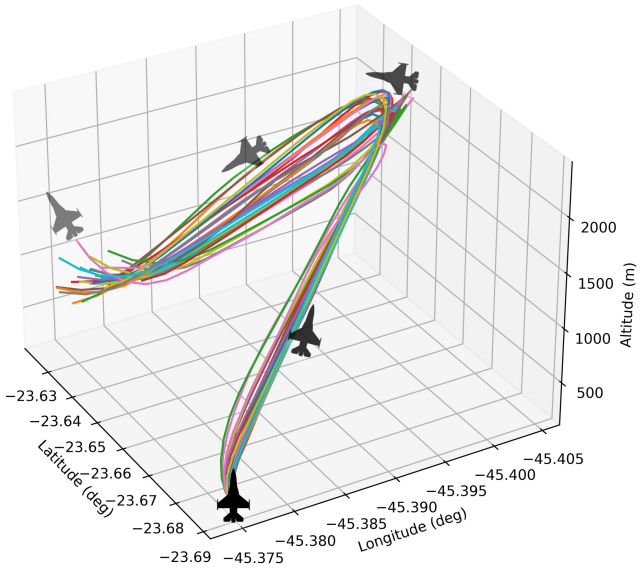

FIGURE 2. Adjusted flight data for the 30 flights of the pop-up attack maneuver executed by the human pilot.

While this dataset provides valuable insights for training and evaluating autonomous agents, transitioning from simulated environments to real-world deployment presents several challenges. A key issue is the potential dynamic mismatch between the simulator and a real aircraft. Unless a high-fidelity, certified simulator is used, discrepancies in system dynamics, response times, and environmental factors could affect the transferability of trained models to real-world scenarios. Given these challenges, the deployment of autonomous agents should begin with extensive validation in simulation, including tests with variations in model parameters to assess robustness before considering real-world implementation. In this context, the flight data used in this study serves as an initial exploration for developing autonomous agents capable of performing air combat maneuvers, providing a first study for future advancements.

B. IMITATION LEARNING MODELS

In this subsection, we will develop imitation learning models designed to replicate the actions of a human pilot during a pop-up attack maneuver. The models are trained to predict control inputs based on the aircraft's state, effectively learning to perform the maneuver autonomously. We will detail the process of constructing the imitation learning models, beginning with the preparation of state and action vectors from the recorded flight data, followed by presenting the architectures of the different models. Finally, we will clarify the model training process, how flight trajectory predictions are made, and the methods used for evaluation and analysis.

1) State and Action Vectors

To develop the imitation learning models, the flight data was segmented into sequences of state-action pairs. The state vectors, defined relative to the aircraft's body frame, included key variables: altitude, pitch, roll, and yaw angles, along with radial angle, distance to the target, and altitude difference between the aircraft and the target. The action vectors consisted of the control inputs commanded by the pilot: pitch, roll, and throttle. To ensure unbiased training, all data was normalized using the mean and standard deviation computed across the entire dataset.

All imitation learning models were trained in two configurations: a baseline version, which excluded derived variables such as linear velocities, angular velocities, and accelerations, and a full version, which included all available state variables. This approach was designed to emphasize the importance of these derived variables and to demonstrate their impact on model performance by comparing simpler models that lack temporal dependency handling with those capable of capturing it. The variables used in the state and action vectors are detailed in Table 1, which outlines the units and descriptions of each variable.

2) Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) model, a type of Feed-forward Neural Network, was designed to efficiently learn

TABLE 1. State and action variables used in the imitation learning model.

Variable	Units	Description	Type
ALT (m)	Meters	Altitude in meters	State
Phi (deg)	Degrees	Pitch angle (positive for nose-up)	State
Theta (deg)	Degrees	Roll angle (positive for left roll)	State
Psi (deg)	Degrees	Yaw angle	State
Vx (m/s)	Meters/second	Velocity in the pitch direction	State
Vy (m/s)	Meters/second	Velocity in the roll direction	State
Vz (m/s)	Meters/second	Velocity in the yaw direction	State
P (deg/s)	Degrees/second	Pitch angular velocity	State
Q (deg/s)	Degrees/second	Roll angular velocity	State
R (deg/s)	Degrees/second	Yaw angular velocity	State
Nx (m/s^2)	Meters/second^2	Lateral acceleration	State
Ny (m/s^2)	Meters/second^2	Longitudinal acceleration	State
Nz (m/s^2)	Meters/second^2	Vertical acceleration	State
Radial (deg)	Degrees	Angular position of the aircraft relative to the target	State
Distance (m)	Meters	Horizontal distance between the aircraft and the target (ground range)	State
DeltaAlt:Anv-Tgt (m)	Meters	Altitude difference between aircraft and target	State
JX	–	Positive for nose-up pitch	Action
JY	–	Positive for left roll	Action
Throttle	–	Throttle position	Action

the relationship between flight states and actions by utilizing fully connected layers that apply non-linear transformations to capture complex patterns between state variables and action predictions [39]. We propose a model based on this MLP architecture, implemented using TensorFlow [40], whose final structure is illustrated in Figure 3, depicting the layers of the MLP-based imitation learning model. The architecture includes the following layers:

- **Dense Layer 1:** A fully connected layer with 128 units, ReLU activation, serving as the first internal layer to capture high-dimensional relationships in the input data.
- **Dense Layer 2:** A fully connected layer with 64 units, ReLU activation, providing further non-linear transformations for improved model accuracy.
- **Dense Layer 3:** A fully connected layer with 32 units, ReLU activation, refining the learned representations.
- **Output Layer:** The final output layer has 3 units corresponding to the predicted actions (JX, JY, and Throttle), without any activation function, to output direct action values.

3) Long Short-Term Memory Network

The Long Short-Term Memory (LSTM) network, an advanced type of Recurrent Neural Network (RNN), was designed to capture complex temporal dependencies in sequential data. LSTMs are particularly effective in learning long-term dependencies, making them ideal for time-series data

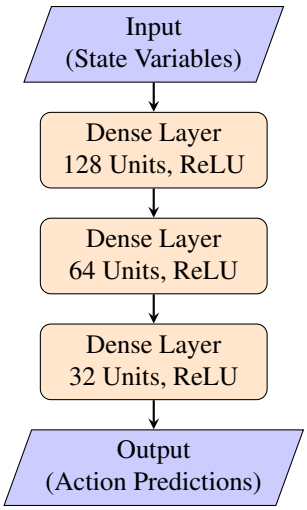

FIGURE 3. Architecture of the MLP-based imitation learning model.

where each output relies on prior inputs [41]. We propose a model based on an LSTM architecture, also implemented using TensorFlow, with its final structure illustrated in Figure 4 and consisting of the following layers:

- **LSTM Layer:** The first layer of the network is an LSTM (Long Short-Term Memory) layer with 128 units, which captures temporal dependencies across the entire sequence. It processes each step of the input data while maintaining a sequential nature, allowing the model to learn from past steps and make predictions based on long-term dependencies in the data.
- **TimeDistributed Dense Layer 1:** The second layer is a TimeDistributed Dense layer with 64 units and ReLU activation. The TimeDistributed wrapper applies the Dense layer to each time step independently, allowing the network to learn complex relationships between the input state and the predicted action at each time step in the sequence.
- **TimeDistributed Dense Layer 2:** Another TimeDistributed Dense layer with 32 units and ReLU activation further refines the predictions. Like the previous layer, it is applied independently to each time step, adding non-linear transformations that enhance the network's ability to handle complex sequential patterns.
- **Output Layer:** The final layer is a TimeDistributed Dense layer with 3 units (corresponding to the actions: JX, JY, and Throttle) and no activation function. This layer produces the directly predicted values for each action at each time step, allowing the network to output continuous values, which is typical for regression tasks where real-valued outputs are predicted.

4) Gated Recurrent Unit

The Gated Recurrent Unit (GRU) model provides a simplified alternative within the RNN family. Compared to Long Short-Term Memory (LSTM) networks, GRUs utilize

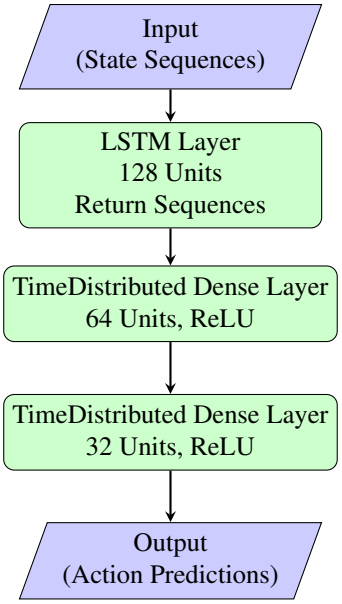

FIGURE 4. Architecture of the LSTM-based imitation learning model.

a simpler structure with fewer parameters by merging the forget and input gates into a single update gate and removing the cell state [42]. This efficient design allows GRUs to achieve performance similar to LSTMs while lowering computational costs, making them effective for tasks like trajectory prediction. Despite its simplicity, the GRU can maintain essential temporal relationships in the data, making it a practical choice for this work. We propose a model based on this GRU architecture, with the final design illustrated in Figure 5, which depicts the layers of the GRU-based imitation learning model. The architecture includes the following layers:

- **GRU Layer:** The first layer consists of a GRU with 128 units, configured to return sequences. Like the LSTM, the GRU captures temporal dependencies across the entire sequence, processing each time step while preserving the sequence's sequential nature. The GRU layer is similar to the LSTM but is more computationally efficient, as it uses fewer parameters while still learning long-term dependencies in the data.
- **TimeDistributed Dense Layer 1:** A TimeDistributed Dense layer with 64 units and ReLU activation is applied to each time step independently. This allows the model to learn complex relationships between the input states and the corresponding action predictions at each time step in the sequence.
- **TimeDistributed Dense Layer 2:** A second TimeDistributed Dense layer with 32 units and ReLU activation further refines the predictions by introducing additional non-linear transformations. Just like the first TimeDistributed layer, this layer is applied independently to each time step, enhancing the model's capacity to capture intricate patterns in sequential data.
- **Output Layer:** The final TimeDistributed Dense layer has 3 units (corresponding to the actions: JX, JY, and Throttle), producing the directly predicted values for each action at each time step in the sequence. The absence of an activation function ensures that the network can output continuous values, typical of regression tasks.

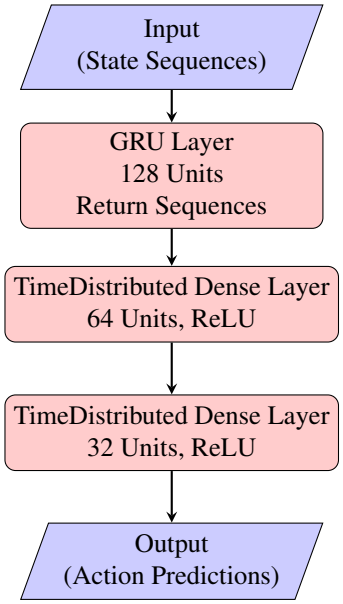

FIGURE 5. Architecture of the GRU-based imitation learning model.

5) Model Training

The training process involved 5-fold cross-validation to ensure robust evaluation of the models. The dataset was split, with 30% reserved for testing and the remaining 70% used for training and validation. During cross-validation, the models were trained and evaluated across each fold, using the Adam optimizer [43] with a learning rate of 10^{-5} and the Mean Squared Error (MSE) loss function, which is suitable for continuous regression problems. A batch size of 32 was employed to maintain computational efficiency while balancing memory usage and training stability.

Within each fold, early stopping was implemented with a patience of 20 epochs, monitoring the validation loss to prevent overfitting. This allowed the training process to halt if no improvement was detected in validation loss over 20 consecutive epochs, thus reducing the risk of overfitting to a particular fold. Hyperparameter tuning was carried out using random search [44], covering a range of configurations for activation functions, the number of units per layer, the depth of the model (number of layers), learning rate, and batch size. This randomized search process allowed for the exploration of diverse model architectures, optimizing the choice of hyperparameters to enhance the models' performance while maintaining generalization capabilities. In this study, only the best-performing architectures for each model, as determined

through tuning, are presented, focusing on configurations that maximize performance and stability.

Once cross-validation was complete, the final models were trained on the entire dataset, leveraging all available data to enhance model performance. In this phase, early stopping was not employed since no dedicated validation group was available. Instead, the number of epochs for this training was determined based on the average number of epochs achieved during cross-validation, providing an empirically informed stopping criterion to ensure that the models were sufficiently trained without excessive epochs. By incorporating all data, the models gained the advantage of a more comprehensive training set, which could contribute to improved generalization on unseen samples.

To address the high variability observed in the results with the addition of synthetic data, the final model was trained five times using different seeds. This approach aimed to enhance the robustness of the results by averaging the performance across multiple runs, thereby mitigating the effects of random initialization and stochastic processes during training. This methodology ensured more stable and reliable results, particularly given the variability introduced by the synthetic data.

The entire training process was conducted on a system with 20 cores of the Intel Xeon Gold 6230R CPU, running at 2.10 GHz, and 40 GB of RAM, providing sufficient computational resources to handle the training workload efficiently.

6) Prediction of Flight Trajectories

Different methods were employed to predict flight trajectories based on the model type. For the MLP model, a single-step prediction approach was used, where each state vector was treated independently. The MLP model predicted the corresponding action for each time step without considering a sequence of states. This was implemented using a function that normalizes the input state sequence and uses the trained model to predict actions across the entire sequence. The actual and predicted actions were then collected for comparison, with the results plotted to show the mean and standard deviation of the trajectories.

In contrast, the LSTM and GRU models, designed to capture temporal dependencies, used a sliding window approach (Figure 6). In this method, the input data was segmented into overlapping state sequences of fixed length, allowing the models to learn patterns across time effectively. The sequence length was set to 5, chosen based on experimentation with values of 3, 5, 10, 15, and 20. This choice proved effective, as it balances capturing temporal dependencies efficiently and aligns well with the maneuver length of 39 frames. Each window contained a sequence of consecutive time steps, with a fixed overlap (stride = 1), ensuring that the model maintained continuity across predictions [45], [46]. These sequences were provided to the model to generate action predictions, where each window produced a prediction for the last time step, and these predictions were accumulated sequentially to reconstruct the entire trajectory. Any remaining time steps not covered by the sliding windows were filled by repeating the first predicted action, ensuring consistency with the original sequence length. The function for the LSTM and GRU models included normalization of each window, model prediction, and denormalization of the output actions. The complete process for this sliding window approach is outlined in Algorithm 1.

Algorithm 1 Sliding window approach for predicting full flight trajectories (LSTM and GRU)

Input: Full sequence of state vectors
Output: Predicted full trajectory
Initialize an empty list for storing predicted actions
Divide the full sequence into overlapping windows of fixed length L (where $L = 5$) with a stride of 1
for each window **do**
    Reshape the window to match the input shape expected by the model
    Feed the window into the model to predict the action for the last time step
    Append the predicted action to the list of predicted actions
end for
if any remaining time steps were not covered by the windows **then**
    Fill the first uncovered time steps with the first predicted action, repeating it
end if
Return the complete predicted trajectory

7) Evaluation and Analysis

The predicted trajectories were compared with the actual recorded maneuvers to assess the model's performance in replicating realistic flight actions. The evaluation metrics, including Root Mean Squared Error (RMSE) and the Coefficient of Determination (R^2), were calculated on the denormalized output values to ensure that predictive accuracy reflects the original scales of the actions. Additionally, a comparative analysis of the mean and standard deviation of the predicted actions was conducted against the actual actions, providing insights into the precision and consistency of the model's predictions. This analysis highlighted any areas where variability diverged from the real maneuvers, allowing for a deeper understanding of the model's stability across different time steps and scenarios and identifying specific areas for further refinement.

C. GENERATIVE LEARNING MODEL

This subsection details the generative learning model employed in this work. A Variational Autoencoder (VAE), implemented using TensorFlow, is used to generate synthetic flight data that replicates the dynamics of pop-up attack maneuvers. The VAE architecture allows the model to learn a low-dimensional latent representation of the high-dimensional flight data, which is then sampled to generate synthetic sequences. These sequences enhance the training dataset and improve model robustness.

1) Data Preprocessing

To prepare the flight data for VAE training, all available recordings of the pop-up attack maneuver were processed and normalized. The data was loaded from text files and stripped of irrelevant columns. Next, the data was normalized by subtracting the mean and dividing by the standard devia-

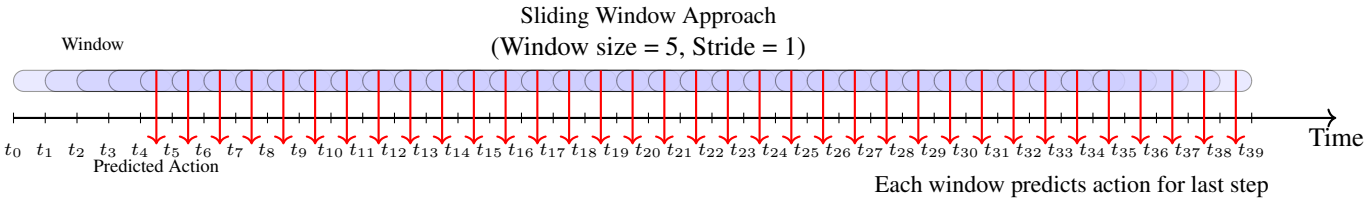

FIGURE 6. Sliding window approach with overlapping sequences to predict flight trajectory across 39 timesteps. Only the first window and predicted action are labeled to illustrate the pattern followed by all subsequent windows.

tion computed across all flight records, ensuring a consistent scale across variables.

2) Architecture

The VAE model consists of three main components: the encoder, the latent space sampling, and the decoder. The encoder compresses the input flight data into a latent representation, the latent space sampling introduces variability into the model, and the decoder reconstructs the data from the latent space.

- **Encoder:** The encoder takes an input sequence of specified length and dimensionality, passing it through a bidirectional LSTM layer with 256 units and L2 regularization to prevent overfitting [47]. This is followed by a Dense layer with 128 units and ReLU activation. A dropout layer with a rate of 0.4 is applied after the Dense layer to further enhance regularization [48]. The encoder outputs two vectors representing the mean and variance of the latent space distribution.
- **Latent Space Sampling:** To introduce variability in the synthetic data, a sampling function generates latent vectors by combining the mean and variance vectors with Gaussian noise. The latent dimension was set to 50, enabling the VAE to capture the underlying structure of the flight data while maintaining computational efficiency.
- **Decoder:** The decoder reconstructs the input data from the sampled latent vector. Since the latent vector represents a single compressed representation of the input sequence, it is first passed through a `RepeatVector` layer to match the sequence length, enabling the decoder to process each timestep independently. The `RepeatVector` effectively "repeats" the latent vector across the number of timesteps, preparing it for the next layers. Following this, a Dense layer with 128 units and ReLU activation is applied, with a dropout layer (rate 0.4) to provide additional regularization. A TimeDistributed LSTM layer with 256 units processes the repeated latent vectors over time, followed by an output layer with linear activation that produces the final reconstructed sequence.

3) Loss Function

The VAE's loss function consists of two terms:

- **Reconstruction Loss:** This term, based on the mean squared error, measures the difference between the original input sequence and its reconstruction, summed over all timesteps. It ensures that the generated sequences closely resemble the original data.
- **KL Divergence Loss:** The Kullback-Leibler (KL) divergence term regularizes the latent space, encouraging it to follow a standard Gaussian distribution [49]. This term is weighted by a factor β to balance between generating realistic data and maintaining a smooth latent space.

4) Training Procedure

The VAE was trained using an Adam optimizer with a learning rate scheduler that decays exponentially over training epochs, starting with an initial learning rate of 1×10^{-3}, decay steps of $10,000$, and a decay rate of 0.9. This approach facilitates efficient convergence by reducing the learning rate as training progresses. Early stopping with the patience of 20 epochs was employed to stop training once the validation loss plateaued, ensuring efficient learning without overfitting. The training was conducted with a large maximum number of epochs (1,000,000) to ensure convergence, though early stopping typically halted training much earlier. The training was conducted with a batch size of 32 and a validation split of 20%.

5) Synthetic Data Generation

After training, the decoder component of the VAE was extracted and used to generate synthetic flight sequences by sampling from the latent space. Synthetic samples were generated in quantities of 7, 15, 30, 45, 60, and 150 to assess the impact of varying amounts of synthetic data on model performance, with generation reaching up to 5 times the size of the original dataset (30). These synthetic sequences were combined with real flight data to enrich the training dataset, ultimately enhancing the model's robustness in learning from a diverse set of trajectories.

6) Fine-Tuning of Synthetic Data

To enhance the realism of the synthetic flight data, a fine-tuning process was applied to the generated sequences. This process involved using a centered moving average to smooth the generated trajectories, reducing noise while preserving

essential maneuver characteristics. Through experimentation, a moving average window of 10 frames was found to provide a suitable balance, ensuring that the synthetic data aligns more closely with the patterns observed in real flight data.

Additionally, subject matter experts (SMEs) from FAB reviewed and qualitatively validated the generated data through visual inspection, ensuring consistency with how a real pilot would execute the maneuver. Their expertise helped confirm that the synthetic sequences followed expected flight dynamics, reinforcing the reliability of the augmented dataset.

D. EVALUATION OF SYNTHETIC DATA IMPACT ON MODEL PERFORMANCE

To evaluate the impact of synthetic data on model performance, synthetic samples generated by the Variational Autoencoder (VAE) were added to the training and validation datasets. In each configuration, we maintained a consistent test set across all experiments to ensure that any observed performance differences were attributable exclusively to the additional synthetic data.

For each configuration (baseline and full), we trained models (MLP, LSTM, and GRU) on these augmented training sets and evaluated them using metrics such as R^2, RMSE, training time, and inference time. It is important to note that the reported inference time corresponds to predicting the entire test dataset, which remained the same across all models for consistency. This process enabled a thorough analysis of how incorporating synthetic data affects model training dynamics and performance potential, particularly in the context of replicating complex maneuvers. By gradually increasing the synthetic data volume, we aimed to assess its role in improving model robustness and generalization.

V. RESULTS AND DISCUSSION

In this section, we present the outcomes of our experiments and analyze the impact of the proposed methods. In the first subsection, V-A, we evaluate the performance of imitation learning techniques in replicating specific maneuvers based on actual flight data. The second subsection, V-B, examines the role of synthetic data in enhancing model performance, particularly in scenarios with limited real-world data. Together, these analyses provide insights into the effectiveness of combining imitation learning with synthetic data generation for robust trajectory prediction and maneuver execution.

A. IMITATION LEARNING FOR MANEUVER EXECUTION

The performance of the imitation learning models was evaluated on the test set, with models trained in both baseline (without derived variables) and full (with all state variables) configurations. Table 2 summarizes the performance metrics for each model configuration, presenting the final R^2 and RMSE values, as well as training and inference times.

The results indicate that including all state variables (full configuration) consistently improves model performance across all architectures, as reflected in higher R^2 and lower RMSE compared to the baseline. This effect is particularly beneficial for models that do not explicitly capture temporal dependencies, as they cannot infer hidden states over time. In contrast, recurrent architectures such as LSTM and GRU can leverage sequential patterns to partially compensate for missing state information.

For models designed to handle sequential data, such as LSTM and GRU, the full configuration also led to substantial improvements, reinforcing the importance of leveraging temporal dependencies for enhanced predictive accuracy. These models exhibited significant gains over the baseline, showing that access to complete state information contributes to better trajectory estimation.

Regarding computational efficiency, the GRU model had the shortest training time in the baseline configuration, while MLP maintained the fastest inference time across both configurations. The inclusion of additional state variables slightly increased computational demand across all models, but the improvements in predictive performance justify the trade-off.

Among all tested models, GRU in the full configuration achieved the best balance of accuracy and computational efficiency, making it a strong candidate for real-time maneuver prediction tasks. However, in real-world scenarios, accessing all state variables may not always be feasible due to sensor limitations, communication delays, or operational constraints. These factors highlight the relevance of the baseline configuration, where synthetic data played a key role in compensating for missing information, demonstrating its potential to enhance model robustness under real-world constraints.

Figure 7 provides a detailed comparison between the actual and predicted actions for JX, JY, and Throttle during the pop-up attack maneuver, specifically illustrating the results from the best-performing model, the GRU in the full configuration. At the beginning of the maneuver, there is a slight delay of 5 steps due to the window size, as the full sequence is divided into overlapping windows of fixed length $L = 5$. Solid blue lines represent the mean of the actions taken by the pilot, while the dashed red lines illustrate the model's predictions. The shaded regions around the mean of the pilot's actions indicate the standard deviation, calculated from the dataset of recorded maneuvers, to visually represent variability in the pilot's actions. Similarly, the shaded regions around the predicted actions show the standard deviation of the model's predictions across multiple test sequences. This highlights the model's consistency in capturing the maneuver patterns, as well as its robustness in generating realistic action sequences across different test instances.

Examining the plots in Figure 7, we observe the following key insights regarding the model's ability to replicate the pop-up maneuver:

- **JX:** The GRU-based model closely mirrors the actual pitch actions, maintaining alignment with the general trend throughout the maneuver. However, during segments with higher variability, the model slightly under-

TABLE 2. Performance metrics for imitation learning models (baseline and full configurations). Values are presented as mean ± standard deviation, calculated over multiple executions with five different seeds. The best values in each column are highlighted in **blue**.

Model	Configuration	R^2	RMSE	Training Time (s)	Inference Time (ms)
MLP	Baseline	0.323 ± 0.009	8.850 ± 0.015	76.11 ± 1.12	267.52 ± 19.19
	Full	0.726 ± 0.070	1.977 ± 0.054	140.84 ± 0.40	$\mathbf{266.92 \pm 12.25}$
LSTM	Baseline	0.672 ± 0.003	8.708 ± 0.059	54.08 ± 17.33	1432.22 ± 122.63
	Full	0.970 ± 0.001	1.743 ± 0.088	74.12 ± 3.66	1283.86 ± 119.91
GRU	Baseline	0.670 ± 0.003	8.698 ± 0.055	$\mathbf{35.85 \pm 4.52}$	1716.36 ± 304.40
	Full	$\mathbf{0.971 \pm 0.002}$	$\mathbf{1.738 \pm 0.064}$	74.63 ± 2.47	1206.61 ± 65.94

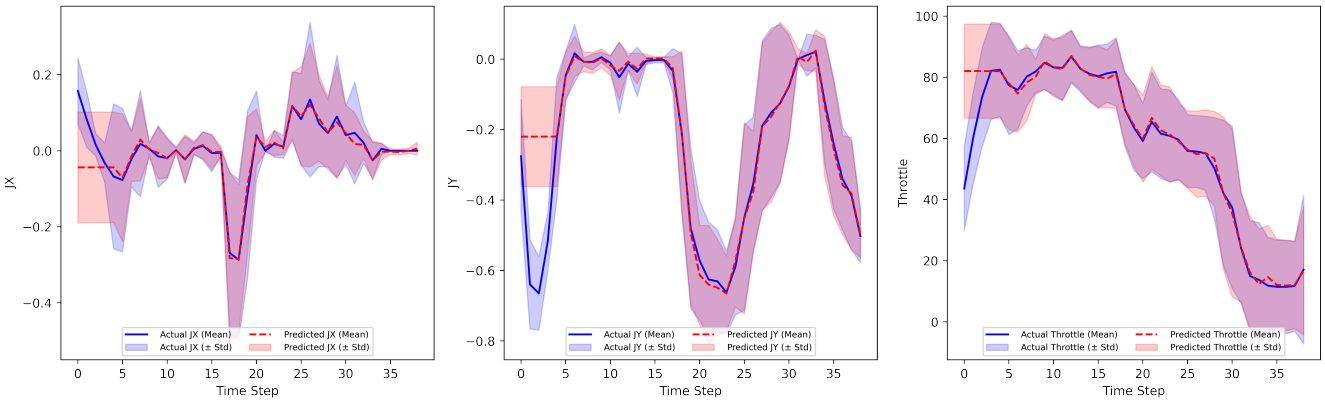

FIGURE 7. Trajectory comparison – actual vs predicted actions with mean and standard deviation for pitch (JX), roll (JY), and Throttle during the pop-up attack maneuver.

estimates the amplitude. This underestimation is still within the standard deviation range, indicating an acceptable degree of accuracy.

- **JY:** The roll predictions are remarkably accurate, especially in capturing sharp transitions. The model's predictions align closely with the actual roll actions, deviating only slightly at certain peaks. These minor deviations fall within the shaded variability range, highlighting the model's effectiveness in tracking rapid changes in roll.
- **Throttle:** The model demonstrates a high level of accuracy in predicting throttle actions, especially during prolonged phases of consistent throttle application. Minor discrepancies appear during abrupt throttle transitions; however, these remain well within the standard deviation range, showcasing the model's robustness in handling throttle adjustments.

Examining the plots in Figure 7, we observe that the visual alignment between the predicted and actual actions across all three control inputs reinforces the quantitative performance metrics, such as R^2 and RMSE. This confirms the GRU model's ability to capture the intricate dynamics of the pop-up maneuver. The shaded regions around the actual actions, which represent pilot variability, demonstrate that the model's predictions consistently fall within the expected range. This further supports the model's potential applicability in both pilot training and autonomous maneuver systems.

B. SYNTHETIC DATA FOR MODEL IMPROVEMENT

To assess the impact of synthetic data on model performance, we incrementally added synthetic flight samples to the training and validation datasets while keeping the test dataset consistent across all experiments. This approach enabled a systematic evaluation of how synthetic data affects model generalization and prediction accuracy for maneuver trajectories. Since the inference time of the models should not depend on the amount of training data and was already analyzed in previous sections, we only report the training time in this subsection. We present our findings separately for the **baseline configuration** (Section V-B1) and the **full configuration** (Section V-B2) to highlight how the inclusion of synthetic data influences model performance under different levels of available real-world features.

1) Baseline Configuration

Table 3 presents the results for the baseline configuration, bringing the number of synthetic samples, R^2, RMSE, and training time in seconds. Results are shown as mean ± standard deviation over five runs. The best R^2, RMSE, and training time for each model are highlighted in **blue**.

For this configuration, synthetic data had a notable impact on the performance of models, particularly for the MLP architecture. The inclusion of synthetic samples improved the MLP's R^2, with the highest value observed at 60 synthetic samples. However, RMSE slightly increased beyond this point, suggesting a reduced benefit as more synthetic data was added. This trend likely occurs because, while additional synthetic data expands the training set, it may also introduce redundancy or noise, limiting further improvements in predictive accuracy.

In contrast, recurrent architectures such as LSTM and

TABLE 3. Performance metrics for imitation learning models in **baseline configuration** when synthetic samples are added to the training and validation datasets. Values are presented as mean ± standard deviation over five runs. The best R^2, RMSE, and Training Time for each model are highlighted in **blue**.

Model	Number of Synthetic Samples Added	R^2	RMSE	Training Time (s)
MLP	7	0.277 ± 0.069	$\mathbf{8.831 \pm 0.012}$	77.33 ± 0.82
	15	0.312 ± 0.066	8.895 ± 0.019	$\mathbf{74.41 \pm 4.82}$
	30	0.414 ± 0.074	9.141 ± 0.034	242.93 ± 3.55
	45	0.408 ± 0.121	9.158 ± 0.016	261.88 ± 1.48
	60	$\mathbf{0.455 \pm 0.053}$	9.313 ± 0.047	315.89 ± 4.34
	150	0.439 ± 0.043	9.324 ± 0.024	453.48 ± 3.58
LSTM	7	0.680 ± 0.003	8.638 ± 0.043	$\mathbf{43.85 \pm 0.89}$
	15	0.679 ± 0.005	8.644 ± 0.079	54.76 ± 1.54
	30	0.680 ± 0.004	8.663 ± 0.077	77.03 ± 4.85
	45	$\mathbf{0.681 \pm 0.003}$	$\mathbf{8.557 \pm 0.044}$	85.81 ± 7.15
	60	0.680 ± 0.005	8.618 ± 0.112	107.61 ± 3.00
	150	0.681 ± 0.004	8.581 ± 0.134	214.97 ± 27.72
GRU	7	$\mathbf{0.683 \pm 0.002}$	8.660 ± 0.076	$\mathbf{35.59 \pm 0.81}$
	15	0.683 ± 0.005	8.692 ± 0.119	64.91 ± 10.91
	30	0.681 ± 0.004	8.691 ± 0.131	72.95 ± 8.01
	45	0.682 ± 0.005	8.578 ± 0.092	79.99 ± 4.13
	60	0.680 ± 0.007	8.692 ± 0.111	113.06 ± 9.86
	150	0.681 ± 0.006	$\mathbf{8.520 \pm 0.143}$	142.02 ± 20.36

GRU exhibited more stable performance across different numbers of synthetic samples, with only slight variations in R^2 and RMSE. This stability suggests that these models already extract sufficient temporal patterns from the available real data, making additional synthetic samples less impactful. Unlike MLP, which does not explicitly model temporal dependencies and can benefit from increased data diversity, LSTM and GRU inherently capture sequential patterns, reducing their reliance on dataset augmentation.

Training time increased consistently with the number of synthetic samples for all models. However, MLP showed the most significant increase, likely due to its fully connected structure requiring more updates per additional sample. In contrast, GRU remained the most computationally efficient model, benefiting from its simplified gating mechanism compared to LSTM. The key reason for GRU's efficiency can be attributed to its architectural design. Unlike LSTM, which has separate forget, input, and output gates, GRU combines these operations into fewer gates, specifically the update and reset gates. This more compact structure reduces the number of parameters and matrix operations required per timestep, leading to faster training times [39].

2) Full Configuration

Table 4 presents the results for the full configuration. As in the baseline case, the table reports the number of synthetic samples, R^2, RMSE, and training time in seconds, with the best results for each model highlighted in **blue**.

In this configuration, synthetic data was not only less useful but, in many cases, made performance worse for all models. Unlike in the baseline setup, where MLP still benefited from synthetic data, in the full configuration, adding synthetic samples consistently led to lower R^2 values and higher RMSE across MLP, LSTM, and GRU. The best results for all models occurred with the smallest number of synthetic samples (7 for LSTM, GRU, and MLP), and performance dropped as more synthetic data was added.

A possible reason for this is that, in the full configuration, models already had access to a richer set of real-world features that provided enough information for learning. The synthetic data, instead of adding useful diversity, may have introduced inconsistencies or patterns that did not fully match the real data distribution. This could have confused the models, leading to worse generalization.

Training time also increased as more synthetic samples were added, which was expected. However, since the extra data did not improve performance, this additional computational cost brought no real benefit in this configuration.

3) Summary of Findings

The evaluation of synthetic data inclusion across different configurations highlights key insights:

- **MLP improved in the baseline configuration:** MLP showed the most gains with synthetic data, reaching the highest R^2 at 60 synthetic samples. Beyond this, RMSE increased, likely due to overfitting or redundant patterns.
- **LSTM and GRU remained stable in the baseline configuration:** Both models showed minimal variation across different amounts of synthetic data, indicating that they effectively captured temporal dependencies from real data without requiring augmentation.
- **Synthetic data reduced performance in the full configuration:** Unlike in the baseline, synthetic samples consistently lowered R^2 and increased RMSE in the full configuration for all models, likely introducing inconsistencies instead of useful variability.
- **Training time increased:** As expected, adding synthetic data raised training time for all models. MLP saw the highest increase, while GRU remained the most efficient.

Notably, the baseline configuration provides a more realistic approximation of real-world operational constraints, as pilots often rely on a partial state space with a limited subset of available variables for decision-making. In this context, synthetic data proved to be effective, expanding the training set and introducing additional variability that contributed to improved performance. These findings highlight the potential of synthetic data to mitigate data scarcity and enhance model

TABLE 4. Performance metrics for imitation learning models in **full configuration** when synthetic samples are added to the training and validation datasets. Values are presented as mean ± standard deviation over five runs. The best R^2, RMSE, and Training Time for each model are highlighted in **blue**.

Model	Number of Synthetic Samples Added	R^2	RMSE	Training Time (s)
MLP	7	$\mathbf{0.698 \pm 0.085}$	$\mathbf{2.230 \pm 0.050}$	186.20 ± 3.00
	15	0.616 ± 0.094	2.530 ± 0.099	$\mathbf{171.13 \pm 2.38}$
	30	0.598 ± 0.078	2.499 ± 0.089	235.22 ± 0.62
	45	0.527 ± 0.146	2.497 ± 0.091	280.90 ± 9.47
	60	0.472 ± 0.140	2.620 ± 0.057	328.74 ± 97.39
	150	0.397 ± 0.213	2.782 ± 0.146	544.65 ± 15.61
LSTM	7	$\mathbf{0.957 \pm 0.002}$	$\mathbf{2.099 \pm 0.115}$	$\mathbf{71.71 \pm 7.53}$
	15	0.951 ± 0.002	2.271 ± 0.093	80.02 ± 2.21
	30	0.950 ± 0.003	2.362 ± 0.076	77.50 ± 2.10
	45	0.947 ± 0.004	2.472 ± 0.093	107.83 ± 1.35
	60	0.944 ± 0.004	2.475 ± 0.128	119.31 ± 0.90
	150	0.939 ± 0.003	2.883 ± 0.172	144.32 ± 11.47
GRU	7	$\mathbf{0.958 \pm 0.002}$	$\mathbf{2.132 \pm 0.073}$	$\mathbf{51.38 \pm 1.18}$
	15	0.954 ± 0.004	2.244 ± 0.139	61.49 ± 9.20
	30	0.948 ± 0.005	2.478 ± 0.157	94.79 ± 13.24
	45	0.945 ± 0.004	2.537 ± 0.146	104.07 ± 12.96
	60	0.943 ± 0.005	2.672 ± 0.155	138.90 ± 6.39
	150	0.937 ± 0.007	2.805 ± 0.201	164.40 ± 13.79

robustness in environments where real-world features are limited.

VI. CONCLUSION AND FUTURE WORK

This study successfully developed models to replicate the complex pop-up attack maneuver in air combat, utilizing simulated flight data from an FAB fighter pilot. Through imitation learning techniques with MLP, LSTM, and GRU networks, the research demonstrated that these models could effectively predict aircraft control inputs, closely mimicking the execution patterns of experienced pilots. Validated through cross-validation and test group evaluations, the models showed strong potential for integration into autonomous combat systems.

The BC approach achieved satisfactory replication of the pop-up maneuver, especially when using state variables that reflect temporal aspects, such as angular and linear velocities and accelerations. Among the tested models, the GRU network achieved the highest performance, effectively capturing temporal dependencies while maintaining computational efficiency, making it particularly suitable for autonomous maneuver execution.

Additionally, we explored generative learning techniques to produce synthetic data that closely mirrors the collected flight data. This approach has the potential to enhance model performance, particularly in scenarios where the agent's state space is not fully known or when non-temporal-dependent algorithms are used. Initial findings suggest that carefully integrated synthetic data can improve model performance and enhance generalization.

While these results are promising, there is room for improvement. Future work will focus on expanding the dataset by collecting additional flight recordings from pilots with different profiles and capturing a more diverse range of strategies and variations in maneuver execution. Expanding data collection to include a wider variety of maneuvers relevant to air combat operations, such as missile evasion and engagement with enemy aircraft, will further improve model generalization. Additionally, evaluations with real flight data will be considered, enabling model updates and reducing the gap between simulation-based training and real-world applicability.

Despite the challenges associated with reinforcement learning, as discussed at the beginning of this work, it remains a valuable research direction for this domain. Future work could also investigate how reinforcement learning can be used to discover innovative approaches for executing ground attack maneuvers, potentially identifying strategies that differ from traditional expert-driven tactics. Moreover, future studies could explore testing alternative imitation and generative learning approaches, such as adversarial imitation learning (AIL) and transformer-based sequence models, which have shown promise in sequential decision-making tasks [50], [51]. Combining reinforcement learning with imitation learning could also be explored as a way to balance stability with adaptability, allowing autonomous agents to refine their skills while maintaining the benefits of expert guidance [52].

Furthermore, future studies should explore how well the models handle sudden and unforeseen operational states, such as extreme maneuvers or unexpected inputs, ensuring that autonomous agents can adapt beyond the conditions seen during training. Another important direction is addressing the risk of excessive dependence on synthetic data by improving the balance between real and synthetic samples during training, validating the models with real flight data, and refining synthetic data generation to better match real-world conditions.

SOURCE CODE

The source code for this research is publicly available at https://github.com/jpadantas/pop-up_attack_generative. This repository contains the scripts and models used in this work. The flight data, however, is not available due to its classified nature.

REFERENCES

[1] Paul D Scharre. The Opportunity and Challenge of Autonomous Systems. Autonomous Systems: Issues for Defence Policymakers, pages 3–26, 2015.

[2] Thomas R Ryan and Vikram Mittal. Potential for Army Integration of Autonomous Systems by Warfighting Function. Military Review, 99(5):122, 2019.

[3] James S McGrew. Real-time maneuvering decisions for autonomous air combat. PhD, Massachusetts Institute of Technology, 2008.

[4] Nan Wang, Lin Wang, Yanlong Bu, Guozhong Zhang, and Lincheng Shen. Tactical Aircraft Pop-Up Attack Planning Using Collaborative Optimization. In Proceedings of the Intelligent Computing 5th International Conference on Emerging Intelligent Computing Technology and Applications, ICIC'09, pages 361–370, Berlin, Heidelberg, Germany, 2009. Springer, Springer-Verlag.

[5] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. ACM Comput. Surv., 50(2):1–35, apr 2017.

[6] Huan Wang, Xiaofeng Liu, and Xu Zhou. Autonomous UAV Interception via Augmented Adversarial Inverse Reinforcement Learning. In Proceedings of 2021 International Conference on Autonomous Unmanned Systems (ICAUS 2021), pages 2073–2084, Singapore, 2022. Springer.

[7] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA, 2018.

[8] Patrick Ribu Gorton, Andreas Strand, and Karsten Brathen. A survey of air combat behavior modeling using machine learning, 2024.

[9] Joao P. A. Dantas, Marcos R. O. A. Maximo, and Takashi Yoneyama. Autonomous Agent for Beyond Visual Range Air Combat: A Deep Reinforcement Learning Approach. In Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'23), SIGSIM-PADS'23, pages 13–24, New York, NY, USA, June 2023. Association for Computing Machinery.

[10] Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges, 2023.

[11] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11):7327–7347, 2022.

[12] Su-Jeong Park, Soon-Seo Park, Han-Lim Choi, Kyeong-Soo An, and Young-Gon Kim. An Expert Data-Driven Air Combat Maneuver Model Learning Approach. In AIAA Scitech 2021 Forum. American Institute of Aeronautics and Astronautics, 2021. View Video Presentation: https://doi.org/10.2514/6.2021-0526.vid.

[13] United States Air Force. Multi-Command Handbook 11-F16, Volume 5: Air Combat Command. United States Air Force, Washington, D.C., USA, May 1996. Volume 5.

[14] Pek Hui Foo, Gee Wah Ng, Khin Hua Ng, and Rong Yang. Application of Intent Inference for Air Defense and Conformance Monitoring. Journal of Advances in Information Fusion (JAIF), 4(1):3–26, 2009.

[15] Joao P. A. Dantas, Andre N. Costa, Diego Geraldo, Marcos R. O. A. Maximo, and Takashi Yoneyama. Engagement Decision Support for Beyond Visual Range Air Combat. In Proceedings of the 2021 Latin American Robotics Symposium, 2021 Brazilian Symposium on Robotics, and 2021 Workshop on Robotics in Education, pages 96–101, Natal, RN, Brazil, 2021. IEEE.

[16] Xiangming Dou, G. Tang, Aoyu Zheng, Han Wang, and Xiaolong Liang. Research on Autonomous Decision-Making in Manned/Unmanned Coordinated Air Combat. In Proceedings of the 9th International Conference on Control, Automation and Robotics (ICCAR), pages 1–6, Beijing, China, April 2023. IEEE.

[17] Zhigang Wang, Kecheng Li, Lei Wang, and Xiaoxiong Liu. Research on an Air Combat Maneuvering Decision Control Method. In Proceedings of the 42nd Chinese Control Conference (CCC), pages 1–6, Xi'an, China, July 2023. IEEE.

[18] Gregg H. Gunsch, Douglas E. Dyer, Mark J. Gerken, Laurence D. Merkle, and Michael A. Whelan. Autonomous Agents as Air Combat Simulation Adversaries. In Applications of Artificial Intelligence 1993: Knowledge-Based Systems in Aerospace and Industry, pages 50–60, Orlando, FL, USA, March 1993. SPIE.

[19] Joao P. A. Dantas, Marcos R. O. A. Maximo, Andre N. Costa, Diego Geraldo, and Takashi Yoneyama. Machine Learning to Improve Situational Awareness in Beyond Visual Range Air Combat. IEEE Latin America Transactions, 20(8), 2022.

[20] Joao P. A. Dantas, Andre N. Costa, Diego Geraldo, Marcos R. O. A. Maximo, and Takashi Yoneyama. Weapon Engagement Zone Maximum Launch Range Estimation Using a Deep Neural Network. In André Britto and Karina Valdivia Delgado, editors, Intelligent Systems, pages 193–207, Cham, 2021. Springer.

[21] Joao PA Dantas, Andre N Costa, Diego Geraldo, Marcos ROA Maximo, and Takashi Yoneyama. PoKER: a probability of kill estimation rate model for air-to-air missiles using machine learning on stochastic targets. The Journal of Defense Modeling and Simulation, 0(0):15485129241309675, 0.

[22] Joao P. A. Dantas, Diego Geraldo, Felipe L. L. Medeiros, Marcos R. O. A. Maximo, and Takashi Yoneyama. Real-Time Surface-to-Air Missile Engagement Zone Prediction Using Simulation and Machine Learning. In Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), Orlando, FL, USA, November 2023. National Training and Simulation Association (NTSA).

[23] Jodi A Miller, Paul D Minear, Albert F Niessner Jr, Anthony M DeLullo, Brian R Geiger, Lyle N Long, and Joseph F Horn. Intelligent unmanned air vehicle flight systems. Journal of Aerospace Computing, Information, and Communication, 4(5):816–835, 2007.

[24] Shaofeng Chen, Yang Cao, Yu Kang, Pengfei Li, and Bingyu Sun. Deep Feature Representation Based Imitation Learning for Autonomous Helicopter Aerobatics. IEEE Transactions on Artificial Intelligence, 2(5):437–446, 2021.

[25] Haitham Baomar and Peter J. Bentley. An Intelligent Autopilot System that Learns Piloting Skills from Human Pilots by Imitation. In 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pages 1023–1031, Arlington, VA, USA, June 2016. IEEE.

[26] Viktoria Sandström, Linus Luotsinen, and Daniel Oskarsson. Fighter Pilot Behavior Cloning. In Proceedings of the 2022 International Conference on Unmanned Aircraft Systems (ICUAS), pages 686–695, Dubrovnik, Croatia, June 2022. IEEE.

[27] Daksh Shukla, Shawn Keshmiri, and Nicole M. Beckage. Imitation Learning for Neural Network Autopilot in Fixed-Wing Unmanned Aerial Systems. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pages 1508–1517, Athens, Greece, September 2020. IEEE.

[28] Patrick Gorton, Martin Asprusten, and Karsten Brathen. Imitation Learning for Modelling Air Combat Behaviour — An Exploratory Study. Technical Report 22/02423, Norwegian Defence Research Establishment (FFI), Kjeller, Norway, January 2023.

[29] Maarten Schadd, Nico de Reus, Sander Uilkema, and Jeroen Voogd. Data-driven behavioural modelling for military applications. Journal of Defence & Security Technologies, 4(1):12–36, 2022.

[30] Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. ACM Trans. Graph., 41(6), November 2022.

[31] Yang Liu, Yifeng Zeng, Yingke Chen, Jing Tang, and Yinghui Pan. Self-Improving Generative Adversarial Reinforcement Learning. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, page 52–60, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.

[32] Ishfaq Hussain Rather and Sushil Kumar. Generative adversarial network based synthetic data training model for lightweight convolutional neural networks. Multimedia Tools Appl., 83(2):6249–6271, May 2023.

[33] Nilantha Premakumara, Brian Jalaian, Niranjan Suri, and Hooman Samani. Improving Object Detection Robustness against Natural Perturbations through Synthetic Data Augmentation. In Proceedings of the 2023 Asia Conference on Computer Vision, Image Processing and Pattern Recognition, CVIPPR '23, New York, NY, USA, 2023. Association for Computing Machinery.

[34] Louis F. Weyland, Maarten P. D. Schadd, and Henk C. Henderson. Exploring the Fidelity of Synthetic Data for Decision Support Systems in Military Applications. In Proceedings of the 2024 International Conference on Military Communication and Information Systems (ICMCIS), pages 1–8, Koblenz, Germany, April 2024. IEEE.

[35] Joao P. A. Dantas, Andre N. Costa, Felipe L. L. Medeiros, Diego Geraldo, Marcos R. O. A. Maximo, and Takashi Yoneyama. Supervised Machine Learning for Effective Missile Launch Based on Beyond Visual Range Air Combat Simulations. In Proceedings of the Winter Simulation Conference, WSC '22, Singapore, December 2022. IEEE.

[36] J. F. Petersen, M. R. C. Aquino, and R. N. Salles. Plataforma AEROGRAF: um SIG voltado para a Força Aérea. Revista Spectrum, n. 11, Comando-Geral de Operações Aéreas, Setembro 2008.

[37] Joao P. A. Dantas, Andre N. Costa, Vitor C. F. Gomes, Andre R. Kuroswiski, Felipe L. L. Medeiros, and Diego Geraldo. ASA: A Simulation Environment for Evaluating Military Operational Scenarios, July 2022.

[38] Joao P. A. Dantas, Diego Geraldo, Andre N. Costa, Marcos R. O. A. Maximo, and Takashi Yoneyama. ASA-SimaaS: Advancing Digital Transformation through Simulation Services in the Brazilian Air Force. In Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE2023), page 6, São José dos Campos, Brazil, September 2023. Instituto Tecnológico de Aeronáutica (ITA).

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[40] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A System for Large-Scale Machine Learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pages 265–283, Savannah, GA, USA, 2016. USENIX Association.

[41] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. Neural Comput., 9(8):1735–1780, November 1997.

[42] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[43] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In International Conference on Learning Representations (ICLR), San Diego, CA, USA, December 2014. ICLR.

[44] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13(10):281–305, 2012.

[45] Yufeng Yu, Yuelong Zhu, Shijin Li, and Dingsheng Wan. Time Series Outlier Detection Based on Sliding Window Prediction. Mathematical Problems in Engineering, 2014(1):879736, 2014.

[46] H. S. Hota, Richa Handa, and A. K. Shrivas. Time Series Data Prediction Using Sliding Window Based RBF Neural Network. International Journal of Computational Intelligence Research, 13(5):1145–1156, 2017.

[47] Andrew Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In Proceedings of the Twenty-First International Conference on Machine Learning, ICML '04, page 78, New York, NY, USA, 2004. Association for Computing Machinery.

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014.

[49] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114, 2013.

[50] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, page 4572–4580, Red Hook, NY, USA, 2016. Curran Associates Inc.

[51] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. In Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc.

[52] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In 2018 IEEE International Conference on Robotics and Automation (ICRA), page 6292–6299. IEEE Press, 2018.

JOAO P. A. DANTAS received his B.Sc. degree in Mechanical-Aeronautical Engineering from the Aeronautics Institute of Technology (ITA) in Brazil in 2015. During this time, he participated in a year-long exchange program at Stony Brook University (SBU) in the USA. He also obtained his M.Sc. degree from ITA's Graduate Program in Electronic and Computer Engineering in 2019. In 2022, he worked as a visiting researcher in the AirLab at the Robotics Institute of Carnegie Mellon University. He is pursuing his Ph.D. in the same program at ITA and working as a researcher for the Brazilian Air Force at the Institute for Advanced Studies (IEAv). His research interests include artificial intelligence, machine learning, robotics, and simulation.

MARCOS R. O. A. MAXIMO received the B.Sc. degree (Hons.) (summa cum laude) in computer engineering, and the M.Sc. and Ph.D. degrees in electronic and computer engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2012, 2015, and 2017, respectively. He is currently a Professor at ITA, where he is a member of the Autonomous Computational Systems Laboratory (LAB-SCA) and leads the Robotics Competition Team: ITAndroids. He is especially interested in humanoid robotics. Moreover, his research interests include mobile robotics, dynamical systems control, and artificial intelligence.

TAKASHI YONEYAMA received a bachelor's degree in electronic engineering from the Aeronautics Institute of Technology (ITA), Sao Jose dos Campos, Brazil, in 1975, an MD degree in medicine from the Taubate University, Taubate, Brazil, in 1993, and the Ph.D. degree in electrical engineering from the Imperial College London, London, U.K., in 1983. He is a professor of control theory at the Department of Electronics, ITA. He has more than 300 published papers, has written four books, and supervised more than 70 theses. His research is concerned mainly with stochastic optimal control theory. Prof. Yoneyama was the President of the Brazilian Automatics Society from 2004 to 2006.