An Intelligent Autopilot System that Learns Flight Emergency Procedures by Imitating Human Pilots

Haitham Baomar, Peter J. Bentley

Dept. of Computer Science, University College London, Gower Street, London, WClE 6BT, U.K.
Email: {h.baomar, p.bentley} @ cs.ucl.ac.uk

Abstract— We propose an extension to the capabilities of the Intelligent Autopilot System (IAS) from our previous work, to be able to learn handling emergencies by observing and imitating human pilots. The IAS is a potential solution to the current problem of Automatic Flight Control Systems of being unable to handle flight uncertainties, and the need to construct control models manually. A robust Learning by Imitation approach is proposed which uses human pilots to demonstrate the task to be learned in a flight simulator while training datasets are captured from these demonstrations. The datasets are then used by Artificial Neural Networks to generate control models automatically. The control models imitate the skills of the human pilot when handling flight emergencies including engine(s) failure or fire, Rejected Take Off (RTO), and emergency landing, while a flight manager program decides which ANNs to be fired given the current condition. Experiments show that, even after being presented with limited examples, the IAS is able to handle such flight emergencies with high accuracy.

I. INTRODUCTION

Human pilots are trained to handle flight uncertainties or emergency situations such as severe weather conditions or system failure. For example, pilots are exposed to scenarios of forced or emergency landing which is performed by executing standard emergency procedures. Usually, the main phase of an emergency landing is known as gliding which is the reliance on the aerodynamics of the aircraft to glide for a given distance while altitude is lost gradually. This happens when the aircraft has lost thrust due to full engine failure in relatively high altitudes.

In contrast, Automatic Flight Control Systems (AFCS/Autopilot) are highly limited, capable of performing minimal piloting tasks in non-emergency conditions. Autopilots are not capable of handling flight emergencies such as engine failure, fire, performing a Rejected Take Off, or a forced (emergency) landing. The limitations of autopilots require constant monitoring of the system and the flight status by the flight crew to react quickly to any undesired situation or emergencies. The reason for such limitations of conventional AFCS is that it is not feasible to anticipate everything that could go wrong with a flight, and incorporate all of that into the set of rules or control models "hardcoded" in an AFCS.

This work aims to address this problem by expanding the capabilities of the Intelligent Autopilot System (IAS) [1] to be able to learn flight emergency procedures from human pilots by applying the Learning by Imitation concept with Artificial Neural Networks. By using this approach, we aim to extend the capabilities of modern autopilots and enable them to autonomously adapt their piloting to suit multiple scenarios ranging from normal to emergency situations.

This paper is structured as follows: part (II) reviews related literature on fault/failure tolerant systems, and the application of multiple ANNs or Artificial Neural Circuits. Part (III) explains the Intelligent Autopilot System (IAS). Part (IV) describes the experiments, Part (V) describes the results by comparing the behaviour of the human pilot with the behaviour of the Intelligent Autopilot System, and part (VI) provides an analysis of the results. Finally, we provide conclusions and future work.

II. BACKGROUND

A review of the Autopilot problem, Artificial Neural Networks, and Learning by Imitation for Autonomous Flight Control is presented in our previous work [1].

A. *Fault/Failure Tolerant Systems for Flight Control*

Current operational autopilots fall under the domain of Control Theory. Classic and modern autopilots rely on controllers such as the Proportional Integral Derivative (PID) controller, and Finite-State automation [2]. Many recent research efforts focus on enhancing flight controllers by adding fault/failure tolerant capabilities. With respect to flight control systems, a fault is "an unpermitted deviation of at least one characteristic property of the system from the acceptable, usual, standard condition." [3], while failure is "a permanent interruption of a system's ability to perform a required function under specified operating conditions." [3].

To handle faults and failures, recent research efforts have been focusing on designing Fault Detection and Diagnosis (FDD) systems that can either stream information to ground crew members especially in the case of UAVs, or feed fault tolerant systems that are capable of handling system faults. The first type of such systems are known as the Passive Fault Tolerant Controllers which can handle moderate faults such as parameters deviations by using a robust feedback controller. However, if the faults are beyond the capabilities of such controllers, another type of fault tolerant systems becomes a necessity. This type is known as an Active Fault Tolerant control

system which includes a separate FDD system that adds an extended and enhanced level of fault tolerance capabilities [4].

In case of emergency situations, mainly engine failure, engine fire, flight instruments failure, or control surface damage or failure, continuing to fly becomes either impossible or can poses a serious threat to the safety of the flight. In such circumstances, a forced or emergency landing on a suitable surface such as a flat field becomes a must especially if it is not possible to return safely to the runway [5]. In [6], an emergency landing controller is proposed for an Unmanned Aerial Vehicle by segmenting the emergency landing period into four sub-levels known as slipping guiding, straight line down, exponential pulling up, and shallow sliding. Each level uses different control strategies aimed at insuring the safe execution of the complete emergency landing. For example, during the exponential pulling up level, the system maintains a certain pitch without causing the UAV to stall. Using a simulator, the proposed approach showed its ability to handle emergency landing [6].

B. *Multiple ANNs or Artificial Neural Circuits*

The problem of coordinating multiple sensor-motor architectures found in complex robotic systems is challenging. This is due to the simultaneous and dynamic operation of these motors while insuring rapid and adaptive behaviour, and due to the need to properly handle the fusion of data from disparate sources. In nature, animals manage this problem by the large number of neural circuits in the animals' brains. For example, neural circuits which are responsible for motion are connected to the muscles (motor systems), and operate simultaneously and dynamically while handling changes in the environment [7]. This has inspired the field of complex robotics to develop multiple neural-based controllers and integrate them together to tackle larger problems such as long-endurance locomotion under uncertainties. For example, the problem of coordinating multiple sensor-motor architectures is addressed in the context of walking by developing a neural circuit which generates multiple gaits adaptively, and coordinates the process of walking with different behavioural-based processes in a hexapod robot. The results showed the ability of the biology-inspired system to detect and stabilize multiple instability scenarios, and to determine what needs to be controlled at each moment which allows the system to handle changes in the environment [7].

Multiple Artificial Neural Networks were applied to the problem of detecting roads visually. In [8], different inputs are fed into multiple ANNs to handle multiple segments of the image. The proposed approach allows the system to detect and classify multiple factors of the environment ahead which leads to an enhanced performance compared to other computer-vision solutions [8]. In [9], Multiple ANNs were applied to tackle the limitations problem of traffic light control systems that are based on conventional mathematical methods. In simulation, the results showed that the approach of using multiple ANNs to address this problem presented an improvement in performance compared to other methods [9]. Another proposed system inspired by biology; is presented in [10] which is designed to handle the challenging problem of gesture recognition. The system shares similarities with the human visual system by developing multiple spiking ANNs. The outputs of the spiking ANNs are used to generate a fusion of multiple data from different segments of the gesture. The results proved the system's ability to handle dynamic visual recognition with the presence of complex backgrounds [10].

The approach of segmenting or breaking down the problem, and using multiple ANNs to handle multiple segment shows the potential to enhance the properties of ANNs as explained in [11]. A large ANN is split into parallel circuits that resemble the circuits of the human retina. During training, the Backpropagation algorithm runs in each circuit separately. This approach does not only decrease training time, but it also enhances generalization [11].

III. The Intelligent Autopilot System

The proposed Intelligent Autopilot System (IAS) in this paper can be viewed as an apprentice that observes the demonstration of a new task by the experienced teacher, and then performs the same task autonomously. A successful generalization of Learning by Imitation should take into consideration the capturing of low-level models and high-level models, which can be viewed as rapid and dynamic sub-actions that occur in fractions of a second, and actions governing the whole process and how it should be performed strategically. It is important to capture and imitate both levels in order to handle flight uncertainties successfully.

The IAS is made of the following components: a flight simulator, an interface, a database, a flight manager program, and Artificial Neural Networks. The IAS implementation method has three steps: A. Pilot Data Collection, B. Training, and C. Autonomous Control. In each step, different IAS components are used. The following sections describe each step and the components used in turn.

A. *Pilot Data Collection*

Fig. 1 illustrates the IAS components used during the pilot data collection step.

1) *Flight Simulator*

Before the IAS can be trained or can take control, we must collect data from a pilot. This is performed using X-Plane which is an advanced flight simulator that has been used as the simulator of choice in many research papers such as [12] [13] [14].

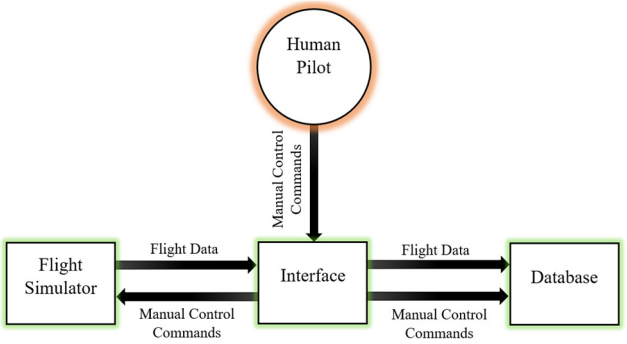

Fig. 1. Block diagram illustrating the IAS components used during the pilot data collection step.

X-Plane is used by multiple organizations and industries such as NASA, Boeing, Cirrus, Cessna, Piper, Precession Flight Controls Incorporated, Japan Airlines, and the American Federal Aviation Administration.[1] X-Plane can communicate with external applications by sending and receiving flight status and control commands data over a network through User Datagram Protocol (UDP) packets. For this work, the simulator is set up to send and receive packets comprising desired data every 0.1 second. In X-Plane, it is possible to simulate a number of flight emergencies for the purpose of training pilots. Emergencies range from severe weather conditions to system failure such as engine failure or fire.

2) The IAS Interface

The IAS Interface is responsible for data flow between the flight simulator and the system in both directions. The Interface contains control command buttons that provide a simplified yet sufficient aircraft control interface which can be used to perform basic tasks of piloting an aircraft such as take-off and landing in the simulator while being able to control other systems such as fuel and fire systems. It also displays flight data received from the simulator.

Data collection is started immediately before demonstration, then; the pilot uses the Interface to perform the piloting task to be learned. The Interface collects flight data from X-Plane over the network using UDP packets, and collects the pilot's actions while performing the task, which are also sent back to the simulator as manual control commands. The Interface organizes the collected flight data received from the simulator (inputs), and the pilot's actions (outputs) into vectors of inputs and outputs, which are sent to the database every 1 second.

3) Database

An SQL Server database stores all data captured from the pilot demonstrator and X-Plane, which are received from the Interface. The database contains tables designed to store: 1. Flight data as inputs, and 2. Pilot's actions as outputs. These tables are then used as training datasets to train the Artificial Neural Networks of the IAS.

B. Training

1) Artificial Neural Networks

After the human pilot data collection step is completed, Artificial Neural Networks are used to generate learning models from the captured datasets through offline training. Fig. 2 illustrates the training step.

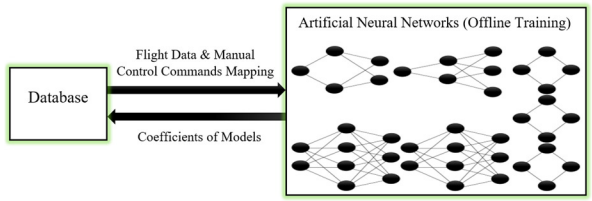

Fig. 2. Block diagram illustrating the IAS components used during training.

Ten feedforward Artificial Neural Networks comprise the core of the IAS. Each ANN is designed and trained to handle specific controls and tasks. The ANNs are: Taxi Speed Gain ANN, Take Off ANN, Rejected Take Off ANN, Aileron ANN, Rudder ANN, Cruise Altitude ANN, Cruise Pitch ANN, Fire Situation ANN, Emergency Landing Pitch ANN, and Emergency Landing Altitude ANN. The inputs and outputs which represent the gathered data and relevant actions, and the topologies of the ten ANNs are illustrated in Fig. 3.

The method for choosing ANN topologies in this work is based on a rule-of-thumb [15] which indicates that problems requiring more than one hidden layer are rarely encountered. This rule follows an approach that tries to avoid under-fitting caused by too few neurons in the hidden layer, or over-fitting caused by too many neurons, by having the number of hidden neurons less than or equal to twice the size of the input layer.

Before training, the datasets are normalized, and retrieved from the database. Then, the datasets are fed to the ANNs. Next, Sigmoid (1) [15] and Hyperbolic Tangent (Tanh) (2) [15] functions are applied for the neuron activation step, where $f(x)$ is the activation function for each neuron, and x is the relevant input value:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$f(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{2}$$

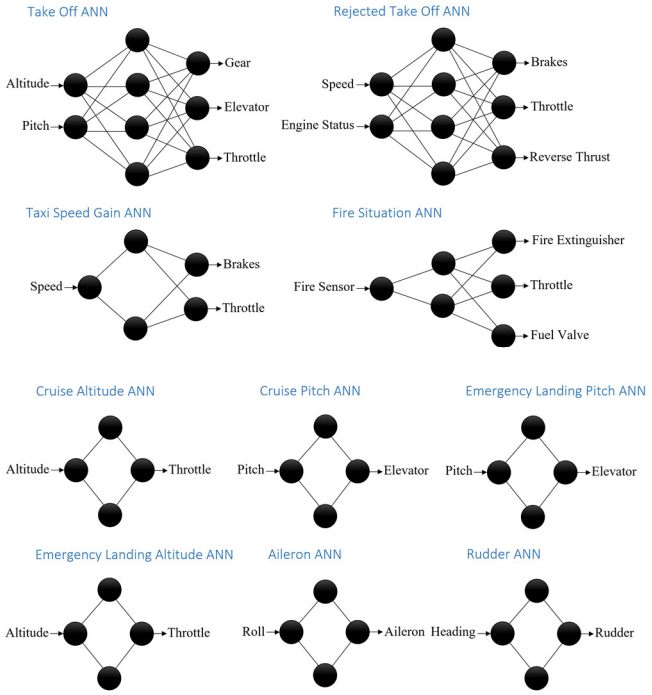

Fig. 3. Inputs, outputs, and the topologies of the ten ANNs representing the core of the Intelligent Autopilot System. Each ANN is designed and trained to handle a specific task.

[1] "X-Plane 10 Global
http://www.x-plane.com

The Sigmoid activation function (1) is used by the Taxi Speed Gain ANN, Take Off ANN, Emergency Landing Altitude ANN, Rejected Take Off ANN, and the Fire Situation ANN, while (2) is used by the rest since their datasets contain negative values.

Next, Backpropagation is applied. Based on the activation function, (3) [16], or (4) [16] are applied to calculate the error signal (δ) where t_n is the desired target value and a_n is the actual activation value:

$$\delta_n = (t_n - a_n)a_n(1 - a_n) \quad (3)$$

$$\delta_n = (t_n - a_n)(1 - a_n)(1 + a_n) \quad (4)$$

Finally, coefficients of models (weights and biases) are updated using (5) [17] where $\delta w_{i,j}$ is the change in the weight between nodes j and k.

$$w_{i,j} = w_{i,j} + \delta w_{i,j} \quad (5)$$

When training is completed, the learning models are generated, and the free parameters or coefficients represented by weights and biases of the models are stored in the database.

C. Autonomous Control

Once trained, the IAS can now be used for autonomous control. Fig. 4 illustrates the components used during the autonomous control step.

1) The IAS Interface

Here, the Interface retrieves the coefficients of the models from the database for each trained ANN, and receives flight data from the flight simulator every 0.1 second. The Interface organizes the coefficients into sets of weights and biases, and organizes data received from the simulator into sets of inputs for each ANN. The relevant coefficients, and flight data input sets are then fed to the Flight Manager and the ANNs of the IAS to produce outputs. The outputs of the ANNs are sent to the Interface which sends them to the flight simulator as autonomous control commands using UDP packets every 0.1 second.

2) The Flight Manager Program

The Flight Manager is a program which resembles a Behaviour Tree [18]. The purpose of the Flight Manager is to manage the ten ANNs of the IAS by deciding which ANNs are to be used simultaneously at each moment. The Flight Manager starts by receiving flight data from the flight simulator through the interface of the IAS, then it detects the flight condition and phase by examining the received flight data, and decides which ANNs are required to be used given the flight condition (normal/emergency/fire situation) and phase (taxi speed gain/take off/cruise/emergency landing). Fig. 5 illustrates the process which the Flight Manager follows.

3) Artificial Neural Networks

The relevant set of flight data inputs received through the Interface is used by the ANNs' input neurons along with the relevant coefficients to predict control commands given the flight status by applying (1) and (2). The values of the output layers are sent to the Interface which sends them to the flight simulator as autonomous control commands. Taxi Speed Gain ANN is used while on the runway just before take off to predict the suitable brakes and throttle command values. Take Off ANN is used after a certain take off speed is achieved to predict gear, elevator, and throttle command values. Rejected Take Off ANN is used to abort take off if necessary by predicting brakes, throttle, and reverse throttle command values. Aileron ANN is used to control the aircraft's roll immediately after take off. Rudder ANN is used to control the aircraft's heading before take off, and yaw when airborne in case one engine fails and creates drag. Cruise Altitude ANN is used to control the aircraft's desired cruising altitude by predicting the throttle command value. Cruise Pitch ANN controls the pitch while cruising by predicting the elevator command value. Fire Situation ANN is used in case of fire by predicting fuel valve and fire extinguishing control commands. Emergency Landing Pitch ANN maintains a certain pitch during emergency landing to lose speed without stalling and to prevent a nose first crash. Emergency Landing Altitude ANN controls the throttle in case of a single engine failure.

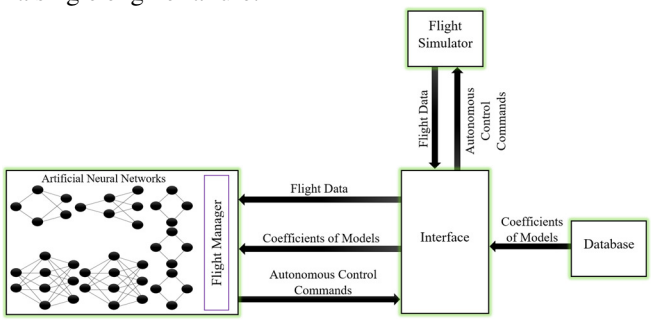

Fig. 4. Block diagram illustrating the IAS components used during autonomous control.

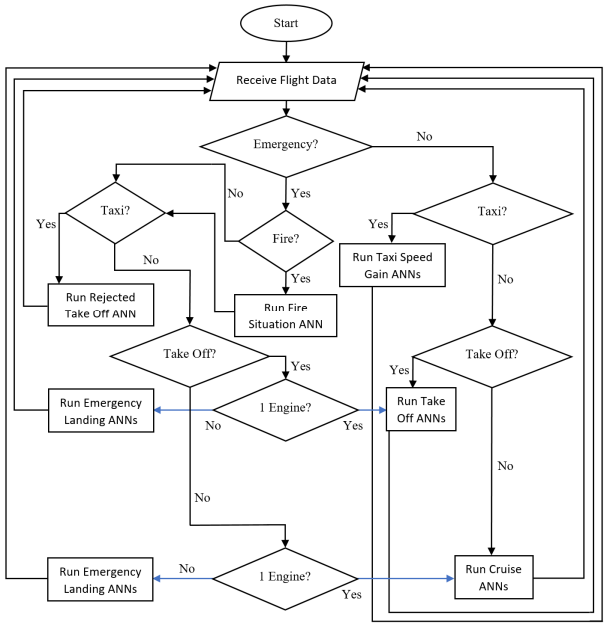

Fig. 5. A Flowchart illustrating the process which the Flight Manager program follows to decided which ANNs are to be used.

IV. EXPERIMENTS

Our previous work [1] provides detailed explanations of the experiments of autonomous taxi speed gain, take off, climb, and applying rudder and aileron to correct heading and roll deviations under normal and severe weather conditions. The new approach in this paper is to segment the training dataset of taxi speed gain, take off, and climb into three different sets that are handled separately by three ANNs (Taxi Speed Gain ANN, Take Off ANN, and Cruise ANN) instead of just one ANN. This work also introduces four new ANNs in order to learn flight emergency procedures for the first time.

In order to assess the effectiveness of the proposed approach in this paper, the Intelligent Autopilot System was tested in four experiments: A. Rejecting take off, B. Emergency landing, C. Maintaining a cruising altitude, and D. Handling single engine failure/fire while airborne. Each experiment is composed of 20 attempts by the IAS to perform autonomously under the given conditions.

The human pilot who provided the demonstrations is the first author. The simulated aircraft used for the experiments is a Boeing 777 as we want to experiment using a more complex model with more than one engine rather than a light single-engine model. The experiments are as follows:

A. *Rejecting Take Off*

The purpose of this experiment is to assess the behaviour of the IAS compared to the behaviour of the human pilot when a Rejected Take Off (RTO) is required.

1) *Data Collection*

In this experiment, the human pilot used the IAS Interface to perform the following in the flight simulator: reject take off when one engine fails or catches fire, and when two engines fail or catch fire (one demonstration for each scenario). The flight simulator was set to simulate the failure or fire conditions for one or two engines immediately after the user presses a hot key on the keyboard. Rejecting take off is performed by going to full reverse thrust and engaging brakes. In case of fire, the human pilot turned off the fuel valve, turned on the fire extinguishing system, and went to full throttle to burn the fuel left in the engine(s). While the pilot performed the demonstration, the Interface collected speed and engine status as inputs, and brakes, throttle, and reverse thrust control data as outputs. The Interface stored the collected data in the database as the training dataset for the Rejected Take Off ANN. The Interface also collected fire sensor readings as input, and fire extinguisher, throttle, and fuel valve control data as outputs. The Interface stored the collected data in the database as the training dataset for the Fire Situation ANN.

2) *Training*

For this experiment, the Rejected Takeoff ANN, and the Fire Situation ANN were trained until low Mean Squared Error (MSE) values were achieved (below 0.001).

3) *Autonomous Control*

After training the ANNs on the relevant training datasets, the aircraft was reset to the runway in the flight simulator to test autonomous RTO multiple times under different scenarios (one and two engine(s) failure and fire), the simulator was set to simulate the desired emergency scenario, and the IAS was engaged. When the flight manager detects the emergency, it stops the Taxi Speed Gain ANN, and runs the Rejected Takeoff ANN and the Fire Situation ANN simultaneously to reject take off and handle fire autonomously. Through the Interface, ANNs receive: 1. Relevant flight data from the flight simulator as inputs, and 2. Coefficients of the relevant models from the database to predict and output command controls that are sent to the flight simulator. This process allows the IAS to autonomously perform the learned task: rejecting take off if necessary. This was repeated 20 times for each scenario to assess performance consistency.

B. *Emergency Landing*

The purpose of this experiment is to assess the behaviour of the IAS compared to the behaviour of the human pilot when a forced or emergency landing is required.

1) *Data Collection*

In this experiment, the human pilot used the IAS Interface to perform the following in the flight simulator: emergency landing when two engines fail or catch fire (one demonstration for each scenario). The flight simulator was set to simulate the failure or fire conditions for two engines immediately after the user presses a hot key on the keyboard. Emergency landing is performed by maintaining a controlled glide using the elevators to insure a gradual loss of speed and altitude without stalling the aircraft, by maintaining a slight positive pitch. If there is any power left in the engines, the throttle is used to aid the gliding phase. In case of fire, the human pilot turned off the fuel valve, and turned on the fire extinguishing system. In this scenario going to full throttle to burn the fuel left in the engines is not possible since both engines do not have sufficient power. While the pilot performed the demonstration, the Interface collected pitch as input, and elevator control data as output. The Interface stored the collected data in the database as the training dataset for the Emergency Landing Pitch ANN. The Interface also collected altitude as input, and throttle control data as output. The Interface stored the collected data in the database as the training dataset for the Emergency Landing Altitude ANN.

2) *Training*

For this experiment, the Emergency Landing Pitch ANN, and the Emergency Landing Altitude ANN were trained until low Mean Squared Error (MSE) values were achieved (below 0.001 for the Emergency Landing Pitch ANN and below 0.2 for the Emergency Landing Altitude ANN).

3) *Control*

After training the ANNs on the relevant training datasets, the aircraft was reset to the runway in the flight simulator to test autonomous emergency landing multiple times under different scenarios (both engines failure or fire), the simulator was set to simulate the desired emergency scenario, and the IAS was engaged. After the IAS took the aircraft airborne, and when the flight manager detects the emergency, it stops the Take Off ANN (during climb), or the cruise ANNs, and runs the Emergency Landing Pitch ANN, and the Emergency Landing Altitude ANN simultaneously to maintain a controlled glide while descending to the ground. Through the Interface, the

ANNs receive: 1. Relevant flight data from the flight simulator as inputs, and 2. Coefficients of the relevant models from the database to predict and output command controls that are sent to the flight simulator. This process allows the IAS to autonomously perform learned task: emergency landing by maintaining a controlled glide. This was repeated 20 times for each scenario to assess performance consistency.

C. Maintaining a Cruising Altitude

The purpose of this experiment is to assess the behaviour of the IAS compared to the behaviour of the human pilot while maintaining a desired cruising altitude.

1) Data Collection

In this experiment, the human pilot used the IAS Interface to maintain a cruising altitude in the flight simulator by increasing and decreasing the throttle, and by using the elevator to maintain a fairly leveled pitch (one demonstration). While the pilot performed the demonstration, the Interface collected altitude as input, and throttle control data as output. The Interface stored the collected data in the database as the training dataset for the Cruise Altitude ANN. The Interface also collected pitch as input, and elevator control data as output. The Interface stored the collected data in the database as the training dataset for the Cruise Pitch ANN.

2) Training

For this experiment, the Cruise Altitude ANN, and the Cruise Pitch ANN were trained until low Mean Squared Error (MSE) values were achieved (below 0.02 and 0.001 respectively).

3) Autonomous Control

After training the ANNs on the relevant training datasets, the aircraft was reset to the runway in the flight simulator to test the ability of maintaining a desired cruise altitude autonomously, and the IAS was engaged. After the IAS took the aircraft airborne, continued to climb, and reached the proximity of the desired altitude, the system's ability to maintain the given altitude was observed. Through the Interface, the ANNs receive: 1. Relevant flight data from the flight simulator as inputs, and 2. Coefficients of the relevant models from the database to predict and output command controls that are sent to the flight simulator. This process allows the IAS to autonomously perform learned task: maintain a desired cruising altitude. This was repeated 20 times for each scenario to assess performance consistency.

D. Handling Single Engine Failure/Fire while Airborne

The purpose of this experiment is to assess the behaviour of the IAS in case of an engine failure or fire while airborne.

1) Data Collection

In this experiment, the human pilot did not provide an explicit demonstration for the single engine failure. Instead, it was intended to test the already trained ANNs, and determine whether their models are able to generalize well in this new scenario where the failed engine creates a drag, and forces the aircraft to descend, and creates a yaw deviation towards the failed engine's side.

2) Training

For this experiment, the previously trained models of the Cruise Altitude ANN, the Cruise Pitch ANN, and the rudder ANN from our previous work [1] were used.

3) Autonomous Control

After setting the simulator to simulate the desired emergency scenario (single engine failure or fire), and after the IAS took the aircraft airborne, when the flight manager detects the emergency, it continues to use the same ANNs (Take Off ANN, or cruise ANNs), and runs the Fire Situation ANN if fire is detected, to fly autonomously using the power left from the engine that operates normally. Through the Interface, the ANNs receive: 1. Relevant flight data from the flight simulator as inputs, and 2. Coefficients of the relevant models from the database to predict and output command controls that are sent to the flight simulator. This was repeated 20 times for each scenario to assess performance consistency.

Throughout all the experiments, the Rudder and Aileron ANNs from our previous work [1] are used normally during the different phases.

V. Results

The following section describes the results of the conducted tests. The 20 attempts by the IAS to handle each scenario autonomously were averaged and compared with the performance of the human pilot when applicable.

A. Rejecting Take Off

Two models were generated with the MSE values as table I shows. Fig. 6 illustrates the behaviour of the IAS when controlling the transition of flight modes under normal conditions, while Fig. 7 illustrates the behaviour of the IAS when engine(s) failure or fire is detected and a Rejected Take Off (RTO) is performed. The results of the 20 experiments showed strong consistency by following the correct procedure in each experiment with a 100% accuracy rate.

B. Emergency Landing

Two models were generated with the MSE values as table I shows. Fig. 8 and 9 illustrate a comparison between the human pilot and the IAS while maintaining a positive pitch during emergency landing, and their altitude (sink rate). The pitch Mean Absolute Deviation (MAD) results (0.024 for the IAS and 0.196 for the human pilot) show less deviation and a steady behaviour of the IAS due to the good model fit as can be seen in Fig. 8. Fig. 10 illustrates the behaviour of the IAS when both engines failure or fire is detected and a forced or emergency landing is performed. The results of the 20 experiments showed strong consistency by following the correct procedure in each experiment with a 100% accuracy rate.

TABLE I
MSE VALUES OF THE MODELS GENERATED FOR THE REJECTED TAKE OFF AND THR EMERGENCY LANDING EXPERIMENTS.

ANN	MSE
Rejected Takeoff ANN	0.000999
Fire Situation ANN	0.000999
Emergency Landing Pitch ANN	0.000997
Emergency Landing Altitude ANN	0.196117

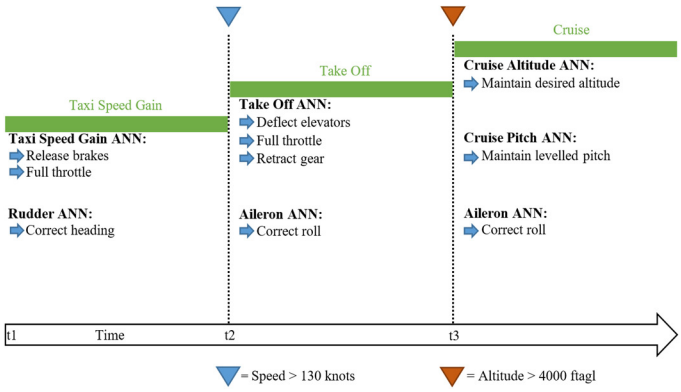

Fig. 6. The behaviour of the IAS when controlling the transition of flight modes under normal conditions. Different ANNs are used in each flight mode.

Fig. 7. (Rejected Take Off experiment) The behaviour of the IAS when engine(s) failure or fire is detected and a Rejected Take Off (RTO) is performed. The Fire Situation ANN is used only when fire is detected.

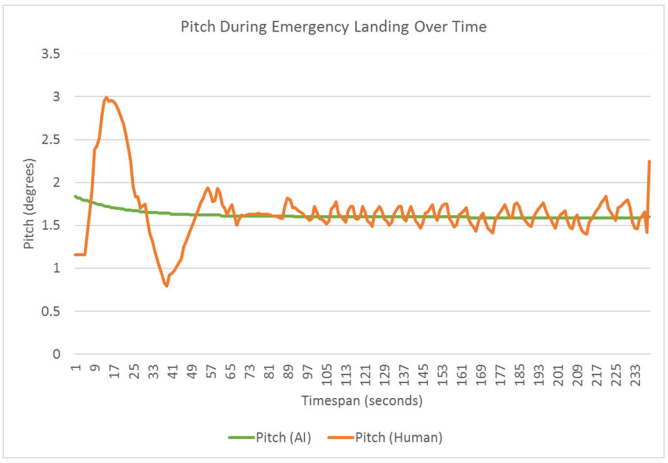

Fig. 8. (Emergency landing experiment) A comparison between the human pilot and the Intelligent Autopilot System's pitch during emergency landing. In this case the human pilot struggled to generate perfect training data so our training approach was designed to prevent overfitting, instead creating a general model (good fit) which provided the desired performance.

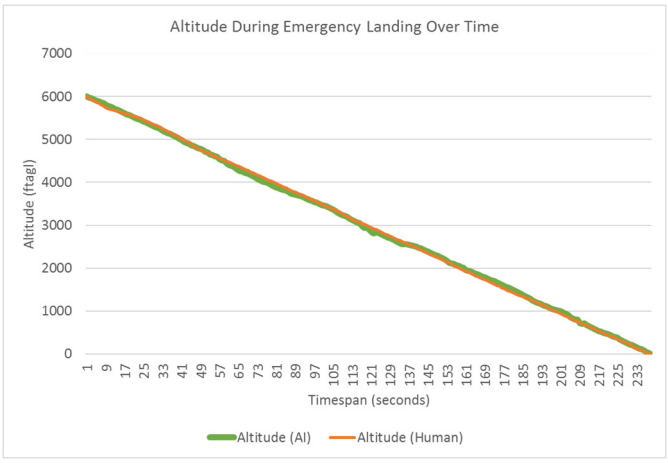

Fig. 9. (Emergency landing experiment) A comparison between the human pilot and the Intelligent Autopilot System's altitude during emergency landing. The results show a significantly close sink rate of about 1500 ftagl per minute.

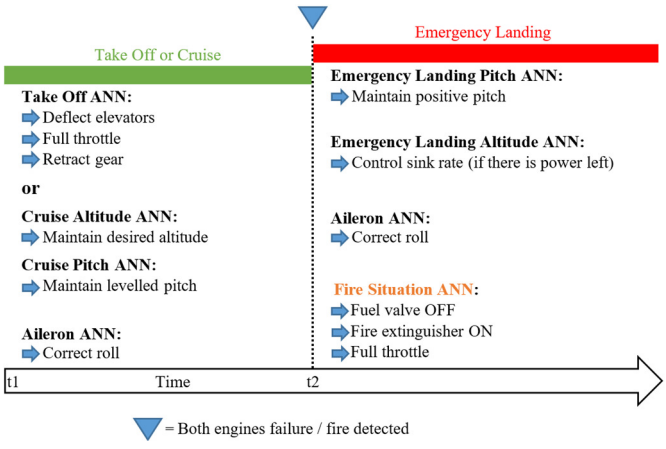

Fig. 10. (Emergency landing experiment) The behaviour of the IAS when both engines failure or fire is detected during either take off or cruise, and an emergency landing is performed. The Fire Situation ANN is used only when fire is detected.

C. *Maintaining a Cruise Altitude*

Two models were generated with the MSE values as table II shows. Fig. 11 and 12 illustrate a comparison between the human pilot and the IAS while maintaining a desired cruising altitude. The altitude Mean Absolute Deviation (MAD) results (85.8 for the IAS and 204.58 for the human pilot) shows less deviation of altitude and a steady behaviour of the IAS due to the good model fit as can be seen in Fig. 11.

TABLE II
MSE VALUES OF THE MODELS GENERATED FOR THE CRUISE EXPERIMENT.

ANN	MSE
Cruise Altitude ANN	0.017574
Cruise Pitch ANN	0.000835

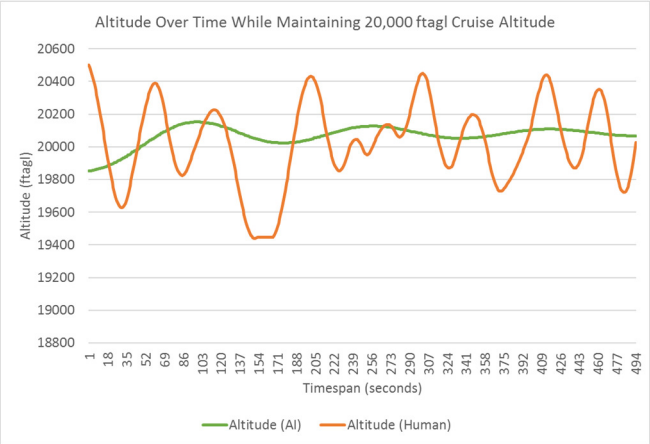

Fig. 11. (Maintaining a cruise altitude experiment) A comparison between the human pilot and the Intelligent Autopilot System's altitude during cruising. While the human pilot demonstrator struggled to maintain a desired cruise altitude of 20,000 ftagl, the IAS performed better due to the good fit of the generated learning model.

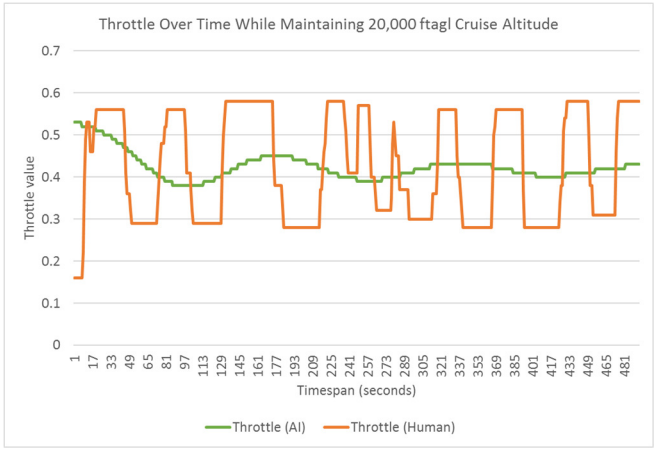

Fig. 12. (Maintaining a cruise altitude experiment) The IAS manipulation of throttle to maintain a desired cruise altitude of 20,000 ftagl compared with the human pilot. The IAS manipulated the throttle smoothly compared to the human pilot due to the good fit of the generated learning model.

D. Handling Single Engine Failure/Fire while Airborne

As mentioned in part (IV) the human pilot did not provide an explicit demonstration for the single engine failure scenario. Instead, it was intended to test the already trained ANNs, and determine whether their models are able to generalize well in this new scenario's experiment. Fig. 13 illustrates the behaviour of the IAS when a single engine fails or catches fire during take off or cruise. The system was intended to carry on flying, apply the rudder ANN from our previous work [1], and run the Fire Situation ANN in case of fire. The results of the 20 experiments showed strong consistency by following the correct procedure in each experiment accurately. Fig. 14 illustrates how the IAS continues to fly while losing altitude gradually compared to the aircraft's autopilot under the same situation.

VI. ANALYSIS

As can be seen in Fig. 7, the rejected take off experiment presented excellent results. The IAS was capable of imitating the human pilot's actions and behaviour with excellent accuracy, and strong consistency by following the correct procedure in each experiment accurately.

Fig. 8 to 10 (the emergency landing experiment) show very desirable results of the ability of the IAS to imitate the human pilot's demonstration of controlling an emergency landing. They show the ability of the IAS to perform the learned sink rate which enabled the aircraft to hit the ground smoothly without being severely wrecked. The flight simulator measures the G force effect on the aircraft's frame, and informs the user in case of an unsurvivable crash. It should be mentioned that selecting a suitable landing surface is not within the scope of this work.

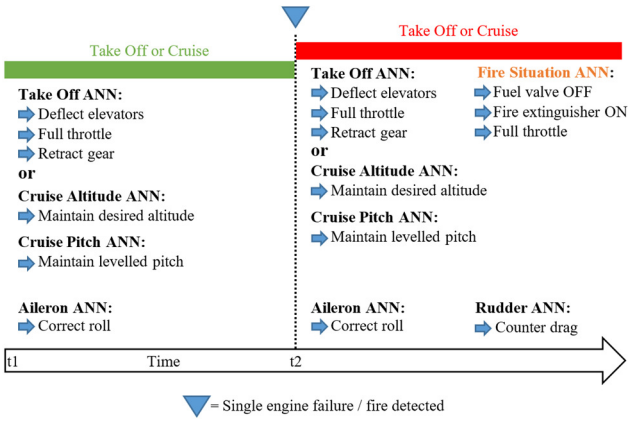

Fig. 13. (Handling single engine failure/fire experiment) The behaviour of the IAS when a single engine failure or fire is detected during either take off or cruise. The Fire Situation ANN is used only when fire is detected. The ANNs used during Take Off or Cruise perform the same tasks as Fig. 6 shows, while the Aileron ANN continues to correct roll.

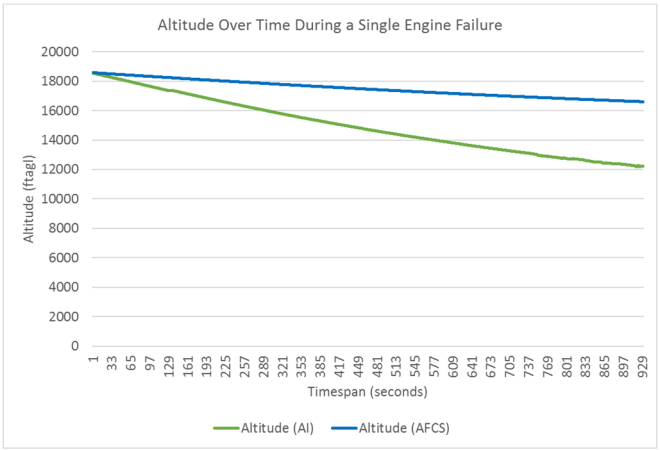

Fig. 14. (Handling single engine failure/fire experiment) Comparing the altitude loss rate of the IAS and the aircraft's AFCS. Since the AFCS is not aware of the single engine failure situation, it compensates by increasing the throttle aggressively, which results in a smaller altitude loss rate, but puts excessive stress on the single operating engine.

Fig. 11 and 12 (maintaining a cruise altitude experiment) show very desirable results of the ability of the IAS to learn how to use throttle and elevator to maintain a given altitude. They illustrate the ability of the IAS to perform better than the human pilot teacher due to the achieved good fit of the learning models. This can also be seen in Fig. 8 (the emergency landing experiment).

As can be seen in Fig. 13 and 14, the single engine failure/fire experiment presented excellent results. The IAS was capable of using the already learned models to continue flying while gradually losing altitude. Although the aircraft's standard autopilot maintained a better altitude in the short term, by aggressively increasing engine thrust it increases the likelihood of engine failure in the remaining engine, with potentially catastrophic results.

The system was able to imitate multiple human pilot's skills and behaviour after being presented with very limited examples. This is due to the approach of segmenting the problem of autonomous piloting while handling uncertainties into small blocks of tasks, and assigning multiple ANNs specially designed and trained for each task, which resulted in the generation of highly accurate models as tables I, and II show.

VII. Conclusion & Future Work

In this work, a robust approach is proposed to "teach" autopilots how to handle uncertainties and emergencies with minimum effort by exploiting Learning by Imitation also known as Learning from Demonstration.

The experiments were strong indicators towards the ability of Supervised Learning with Artificial Neural Networks to capture low-level piloting tasks such as the rapid manipulation of the elevator and throttle to maintain a certain pitch or a given altitude. The experiments showed the ability of the IAS to capture high-level tasks such as coordinating the necessary actions to reject take off and extinguish fire.

Breaking down the piloting tasks, and adding more Artificial Neural Networks enhanced performance and accuracy, and allowed the coverage of a wider spectrum of tasks.

The aviation industry is currently working on solutions which should lead to decreasing the dependence on crew members. The reason behind this is to lower workload, human error, stress, and emergency situations where the captain or the first officer becomes incapable, by developing autopilots capable of handling multiple scenarios without human intervention. We anticipate that future Autopilot systems which make of methods proposed here could improve safety and save lives.

Future effort will focus on giving the IAS the ability to learn how to fly a pre-selected course, and land safely in an airport. The IAS should be capable of avoiding no-fly zones that are either pre-identified, or detected during the flight such as severe weather systems detected by the aircraft's radar.

The Flight Manager program should be redesigned to utilize Artificial Neural Networks to classify the situation (normal or emergency), and predict the suitable flight control law or mode given the situation.

The problem of sensor fault and denial should be investigated to test the feasibility of teaching the IAS how to handle such scenarios.

References

[1] H. Baomar and P. J. Bentley, "An Intelligent Autopilot System that learns piloting skills from human pilots by imitation," *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, USA, 2016, pp. 1023-1031. doi: 10.1109/ICUAS.2016.7502578

[2] Nelson, Robert C. Flight stability and automatic control. Vol. 2. WCB/McGraw Hill, 1998.

[3] G. J. J. Ducard, "Fault-Tolerant Flight Control and Guidance Systems: Practical Methods for Unmanned Aerial Vehicles," Springer, 2009.

[4] Sadeghzadeh, I. & Zhang, Y. (2011). A Review on Fault-Tolerant Control for Unmanned Aerial Vehicles (UAVs). St. Louis, Missouri, USA: The American Institute of Aeronautics and Astronautics.

[5] Rao, Faheem Muhammad et al. "UAV Emergency Landing Site Selection System Using Machine Vision". J MACH INTELL 1.1 (2016): n. pag. Web.

[6] P. Li, X. Chen and C. Li, "Emergency landing control technology for UAV," *Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese*, Yantai, 2014, pp. 2359-2362.

[7] Steingrube, Silke et al. "Self-Organized Adaptation Of A Simple Neural Circuit Enables Complex Robot Behaviour". Nat Phys 6.3 (2010): 224-230. Web.

[8] P. Y. Shinzato, V. Grassi, F. S. Osorio and D. F. Wolf, "Fast visual road recognition and horizon detection using multiple artificial neural networks," *Intelligent Vehicles Symposium (IV), 2012 IEEE*, Alcala de Henares, 2012, pp. 1090-1095.

[9] M. B. W. D. Oliveira and A. D. A. Neto, "Optimization of Traffic Lights Timing Based on Multiple Neural Networks," *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, Herndon, VA, 2013, pp. 825-832.

[10] L. Huang, Q. Wu, Y. Chen, S. Hong and X. Huang, "Gesture Recognition Based on Fusion Features from Multiple Spiking Neural Networks," *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, Gwalior, 2015, pp. 1167-1171.

[11] Kien Tuong Phan, T. H. Maul and Tuong Thuy Vu, "A parallel circuit approach for improving the speed and generalization properties of neural networks," *Natural Computation (ICNC), 2015 11th International Conference on*, Zhangjiajie, 2015, pp. 1-7.

[12] Wei, F., Amaya-Bower, L., Gates, A., Rose, D. and Vasko, T. (2016). The Full-Scale Helicopter Flight Simulator Design and Fabrication at CCSU. 57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference.

[13] Jirgl, M., Boril, J., Jalovecky, R. (2015). The identification possibilities of the measured parameters of an aircraft model and pilot behavior model on the flight simulator. International Conference on Military Technologies (ICMT), vol., no., pp.1-5.

[14] Kaviyarasu, A. and Senthil Kumar, K. (2014). Simulation of Flapping-wing Unmanned Aerial Vehicle using X-plane and Matlab/Simulink. Defence Science Journal, 64(4), pp.327-331.

[15] Heaton, J. (2005). Introduction to neural networks with Java. St. Louis: Heaton Research.

[16] McClelland, J. (2015). Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises (2nd ed.). Stanford.

[17] Tveter, D. (1995). Chapter 2, The Backprop Algorithm.

[18] K. Winter, I. J. Hayes and R. Colvin, "Integrating Requirements: The Behavior Tree Philosophy," 2010 8th IEEE International Conference on Software Engineering and Formal Methods, Pisa, 2010, pp. 41-50.