Deep Learning Approach for Unmanned Aerial Vehicle Landing

Utkarsh R. Moholkar, Dipti D. Patil, Vinod Kumar, Archana Patil

***Abstract:** It is one of the biggest challenges to land an unmanned aerial vehicle (UAV). Landing it by making its own decisions is almost impossible even if progress has been made in developing deep learning algorithms, which are doing a great job in the Artificial Intelligence sector. But these algorithms require a large amount of data to get optimum results. For a Type-I civilization collecting data while landing UAV on another planet is not feasible. But there is one hack all the required data can be collected by creating a simulation that is cost-effective, time-saving, and safe too. This is a small step toward making an Intelligent UAV that can make its own decisions while landing on a surface other than Earth's surface. Therefore, the simulation has been created inside gaming engine from which the required training data can be collected. And by using that training data, deep neural networks are trained. Also deployed those trained models into the simulation and checked their performance.*

***Keywords:** Artificial intelligence, deep learning, Unmanned Aerial Vehicle*

I. INTRODUCTION

Over the past few years, the computation power of CPU and GPU has tremendously increased. A massive amount of labeled data is also available to train deep neural networks. Many machine learning algorithms can accurately process images by converting image data into NumPy arrays. The goal is either classification or regression task. Therefore, in the future, deep learning algorithms will play vital roles in autonomous UAV navigation. This work builds a robust simulator based on Unreal Engine [10]. In this Simulator, the vehicle can be controlled by using the keyboard and Mouse keys and custom API. This API is one of the simplest APIs ever written. This simulator generates training data by manually flying a vehicle in a simulated environment and writing a custom python script using custom API. The training parameters are assumed in 3D space. It means that UAV's location, rotation, velocity, and acceleration are all along (x,y,z) axes, where the z-axis is upward. For guidance and control, a UAV prototype in the simulated planetary gaming environment of Unreal Engine is programmed. To make a vehicle truly autonomous, they drive cars for hours to collect image data using a supervised learning approach. By using this image data, steering angles get predicted using a deep neural network. A similar approach has followed here. The difference is that this works in 3D space. For the proposed work, a simulated planetary gaming environment is used with the help of the Unreal Engine. Video data is recorded while landing smoothly on a plain planetary surface. Trained Linear Regression model, which predicts optimum thrust values that help in a safe landing.

Manuscript received on 16 August 2022 | Revised Manuscript received on 22 August 2022 | Manuscript Accepted on 15 September 2022 | Manuscript published on 30 September 2022.

***Correspondence Author**
Utkarsh R. Moholkar*, Research Scholar, Artificial Intelligence & Robotics, College of Engineering, Pune (Maharashtra), India. E-mail: moholkarur20.mfg@coep.ac.in
Dipti D. Patil, Associate Professor, Department of Information Technology, MKSSS's Cummins College of Engineering for Women, Pune (Maharashtra), India. E-mail: dipti.patil@cumminscollege.in
Vinod Kumar, Director, U.R. Rao Satellite Centre, Indian Space Research Organization, Bengaluru (Karnataka), India. E-mail: vinodkkaushik@gmail.com
Archana Patil, Assistant Professor, Department of Computer Engineering & Information Technology, College of Engineering, Pune (Maharashtra), India. E-mail: abp.comp@coep.ac.in

II. LITERATURE REVIEW

Not much work has been done on the deep learning approach in autonomous UAVs in a simulated gaming environment. But here is some research that is referred to while implementing this research. Dosovitskiy et al. [1] proposed an approach for autonomous training of vehicles known as CARLA (Car Learning to Act), a simulator. It is based on Unreal Engine. They have created Python API to communicate with Unreal Engine's internal functions. They have deployed a server on Unreal Engine, and the Python client can be communicated. The simulation environment can be manipulated for agents, weather, cars, signals, etc., by using Python clients from outside and can even get the live stream of the dashboard camera by using the NumPy arrays and OpenCV. Deep Learning and Reinforcement Learning can be achieved for cars. Shah S et al. [2] proposed Airsim to train Autonomous cars and UAVs. AirSim is also based on Unreal Engine. Using AirSim, datasets can be generated for autonomous cars and autonomous multirotor vehicles. They also have their client-server architecture from python clients. Simulation can control environments, cars, and drones. Using their well-documented Python API, a video dataset can be generated to train deep neural networks. For deep learning algorithms, feature extraction is crucial. Extracting feature points from video frames is achieved in [3]. Moghe et al.[4] proposed an approach of LIDAR Scans to get surface data. To detect safe landing spots, they have used LIDAR scans. And Semantic Segmentation [5] is used to make classes of hazards and secure locations. They have used geometric techniques to generate ground truth and a UNet neural network. Ciabatti et al.[6] proposed simulation in PyBullet.

They have done excellent work in autonomous landing. They are using transfer learning with deep reinforcement learning to solve the landing issue in unknown or barely known environments. For this purpose, they have modeled a physics simulator by using PyBullet Library. They have also used virtual Lander in that simulation. Zhu et. al.[7] proposed optimal fuel trajectories. They have trained deep neural networks offline to generate optimal fuel trajectories and real-time optimal control actions used in the online application stage. They have also used a control switching strategy to improve the accuracy of the lunar landing. D'Ambrosio et. al.[8] proposed craters detection using the Canny Algorithm. This paper focuses on guidance and control, and for craters detection and avoidance, they are using the Canny algorithm. The canny algorithm detects edges in the images or video. They have tested their work in a real-time facility using Cartesian Robot. The cartesian robot is equipped with a 320 x 240 pixels resolution camera. Furfaro et. al.[9] proposed image-based lunar landing approach. They have trained a deep neural network that can predict the optimal fuel thrust magnitude by using a sequence of images taken by an onboard camera. They worked on a verticle landing problem to get an optimum fuel trajectory. They demonstrated a fuel-efficient way that a UAV could be dropped vertically toward the moon's surface. For this case, a Convolutional neural network is trained and tested to predict the thrust during free fall. The DAgger algorithm has achieved the desired accuracy in the following case. They considered the 2D planer Lunar landing problem. They designed an RNN-LSTM-CNN network that can predict both thrust and direction using the sequence of images.

III. DATASET GENERATION

This is one of the crucial phases of this research. A negligible amount of data sets are available related to this kind of work. to predict optimum thrust values for a safe landing requires actual footage while landing so that the neural network can be trained based on that video as shown in figure 1. As deep neural networks require a tremendous amount of data sets to get optimum results, data generation is an important task. Video feed has been recorded while landing UAV by manual control of simulation and experimented with thrust values so that the UAV could be landed safely. Finally, a live video stream is recorded while landing the UAV safely in simulation and used video stream to train a deep neural network.

In this paper, training and testing of the deep neural network have been done, predicting optimum thrust value. Many deep neural networks [11]-[16] are analyzed, and a suitable network is chosen for the thrust prediction task.

IV. TRAINING A DEEP NEURAL NETWORK

In this paper, training and testing of the deep neural network have been done, predicting optimum thrust value. Many deep neural networks [11]-[16] are analyzed, and a suitable network is chosen for the thrust prediction task.

A. Training a Regression Deep Neural Network

Generally, UAV landing is done by using safe trajectories. Here free-fall landing case has been considered. In a free-fall case, to land a UAV safely, it needs to apply the Main Engine Thrust against the direction of gravity. Here the input is a live video stream from the camera sensor. A deep neural network predicts main engine thrust values; therefore, it is a regression problem as thrust values are predicted. Images and corresponding thrust values were captured while landing a UAV in a simulation environment to train this network. After experimentation studies, it is found that the NVIDIA model gave better results than other regression models. NVDIA has done good research in self driving cars their model predicts driving angles by taking images as input in this research we leveraged transfer learning and used NVIDIA model for thrust prediction.

NVIDIA model architecture is implemented, except that the normalization layers have been removed. The normalization has been done outside the model, and a few added dropout layers make the model robust. Mean Squared Error (MSE) is used as a loss function as it fits well for regression model. Rectified Linear Unit (ReLU) is the activation function used. NVIDIA model takes input images in 200x66 pixel resolution, but 120x120 resolution input images are given as an input as the model predicts the main engine thrust and not the driving angles. The NVIDIA model architecture is shown in "Fig. 3".

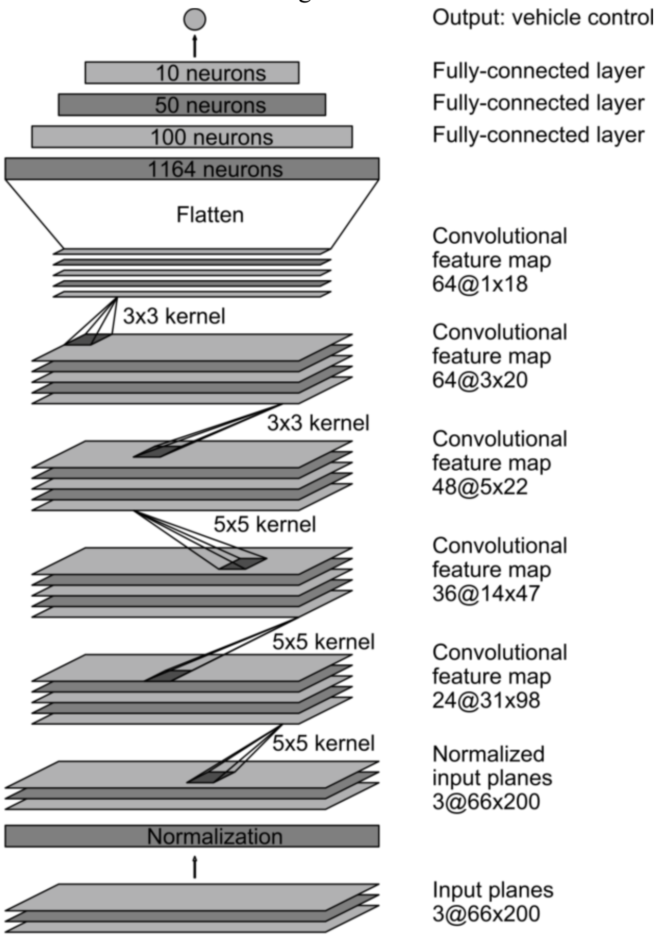
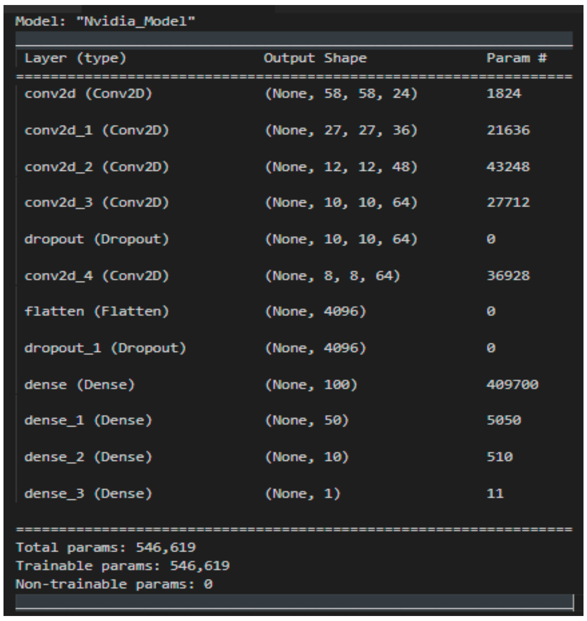


Fig. 1.NVIDIA Model [18] Architecture

"Fig. 4" enlists the details of trainable parameters at each layer of the NVIDIA Model. The first column describes the operation performed at the layer.

The corresponding second column describes the output shape of the images after the operation specified in column one is performed. The third column shows the number of parameters trained for a particular operation in the deep neural network of the NVIDIA model. The convolution operation is specifically used for generating feature maps from the input images. This feature extraction [17] process is followed in a deep neural net. In all, a total 546619 number of parameters are trained for building a regression model. The regression model so built is utilized for predicting main engine thrust values. In this research fine-tuning of the Nvidia model [18] is done as per requirement. While fine-tuning, the optimizer is set as Adam, and the learning rate is 0.001. Finally created the model with 546,619 parameters.



```
Model: "Nvidia_Model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 58, 58, 24)        1824

 conv2d_1 (Conv2D)           (None, 27, 27, 36)        21636

 conv2d_2 (Conv2D)           (None, 12, 12, 48)        43248

 conv2d_3 (Conv2D)           (None, 10, 10, 64)        27712

 dropout (Dropout)           (None, 10, 10, 64)        0

 conv2d_4 (Conv2D)           (None, 8, 8, 64)          36928

 flatten (Flatten)           (None, 4096)              0

 dropout_1 (Dropout)         (None, 4096)              0

 dense (Dense)               (None, 100)               409700

 dense_1 (Dense)             (None, 50)                5050

 dense_2 (Dense)             (None, 10)                510

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 546,619
Trainable params: 546,619
Non-trainable params: 0
```

Fig. 2. Fine Tuned NVIDIA Model

The plot for training and validation sets is shown in figure 5. Training & validation loss shows a declined trend and keeps at low after epoch 1000. There is no significant difference after 5000 epochs. Validation loss is seen fluctuating quite often as compared to the training loss. But the fluctuations are lesser than 0.1 and hence can be neglected. It ascertains that the model is quite stable and performs consistently for training and validation sets.

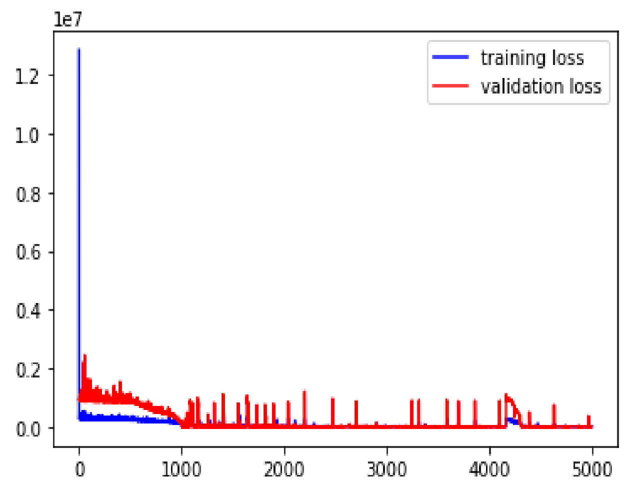


Fig. 3.Training and Validation Loss

The R² metric is also used to check the performance of the proposed system. The R² is the measure of how data fits a particular regression model. The desired value of the metric is generally closer to 1 or 100% for predictive models. It is calculated and found to be 98.71%, which means the model performs pretty well. Figure 6 shows test samples and their thrust predictions. The thrust values are measured in $g\frac{m}{s^2}$. For the first image sample "Fig. 6", the error of 70.54 is observed in the predicted value, while for thesecond image sample"Fig. 6" error is 92.80. This error range is only 1-1.5% of the actual value, which is permissible.

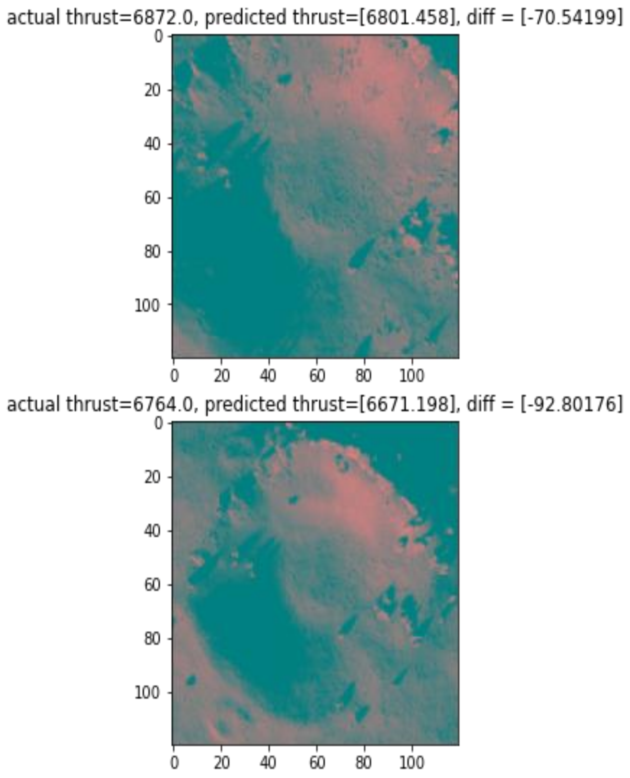

actual thrust=6872.0, predicted thrust=[6801.458], diff = [-70.54199]
0
20
40
60
80
100
actual thrust=6764.0, predicted thrust=[6671.198], diff = [-92.80176]

Fig. 4.Sample Predictions on The Test Data

V. CONCLUSION AND FUTURE SCOPE

In this research, a simulator for a UAV landing is created. A UAV is landed successfully in the simulated environment of the Unreal Engine. For this, the NVIDIA model is trained with the generated dataset, which gives a mean squared error closer to 0, and R² of 98.71%, meaning a good fit for the model. In this research, only a free-fall landing case is considered, and the main engine thrust is predicted. NVIDIA model also can be used in other landing cases, and it may predict Main Engine Thrust and Roll, Pitch, and Yaw engine thrust values too. In future work, we will consider a complete trajectory of the UAV rather than a free-fall landing.

REFERENCES

1. Dosovitskiy, Alexey, Germán Ros, Felipe Codevilla, Antonio M. López and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator." ArXiv abs/1711.03938 (2017): n. pag.

2. Shah S., Debadeepta Dey, Chris Lovett, and Ashish Kapoor. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles." FSR (2017). [CrossRef]
3. J. H. Borse, D. D. Patil, and V. Kumar, "Tracking Keypoints from Consecutive Video Frames Using CNN Features for Space Applications," Teh. Glas., vol. 15, no. 1, pp. 11–17, Mar. 2021, doi: 10.31803/tg20210204161210. [CrossRef]
4. Moghe, Rahul and Renato Zanetti. "A Deep Learning Approach to Hazard Detection for Autonomous Lunar Landing." The Journal of the Astronautical Sciences 67 (2020): 1811-1830. [CrossRef]
5. Janhavi Borse, Dipti Patil, V. K. "Deep Semantic Classification Of Visual Inputs For Hazard-Free Lunar Landing," vol. 3, no. June, pp. 14–18, 2021.
6. Ciabatti, Giulia, Shreyansh Daftry and Roberto Capobianco. "Autonomous Planetary Landing via Deep Reinforcement Learning and Transfer Learning." 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2021): 2031-2038. [CrossRef]
7. Zhu, Lingchao, Jianbo Ma and Shuquan Wang. "Deep Neural Networks Based Real-time Optimal Control for Lunar Landing." IOP Conference Series: Materials Science and Engineering (2019): n. pag. [CrossRef]
8. D'Ambrosio, Andrea, Andrea Carbone, Dario Spiller and Fabio Curti. "PSO-Based Soft Lunar Landing with Hazard Avoidance: Analysis and Experimentation." Aerospace (2021): n. pag. [CrossRef]
9. Furfaro, Roberto, Ilaria Bloise, Marcello Orlandelli, Pierluigi Di, Lizia, Francesco Topputo and Richard Linares. "AAS 18-363 DEEP LEARNING FOR AUTONOMOUS LUNAR LANDING." (2018).
10. Epic Games, 2019. Unreal Engine, Available at: https://www.unrealengine.com.
11. Belagoune, Soufiane, et al. "Deep learning through LSTM classification and regression for transmission line fault detection, diagnosis and location in large-scale multi-machine power systems." Measurement 177 (2021): 109330. [CrossRef]
12. Lee, Hojun, et al. "Deep learning model for real-time prediction of intradialytic hypotension." Clinical Journal of the American Society of Nephrology 16.3 (2021): 396-406. [CrossRef], [PMid], [PMCid]
13. Haq, Anwar Ul, et al. "Forecasting daily stock trend using multi-filter feature selection and deep learning." Expert Systems with Applications 168 (2021): 114444. [CrossRef]
14. Ravani, Khilan, S. Mathavaraj, and Radhakant Padhi. "Site detection for autonomous soft-landing on asteroids using deep learning." Transactions of the Indian National Academy of Engineering 6.2 (2021): 365-375. [CrossRef]
15. Osco, Lucas Prado, et al. "A review on deep learning in UAV remote sensing." International Journal of Applied Earth Observation and Geoinformation 102 (2021): 102456. [CrossRef]
16. Afifi, Ghada, and Yasser Gadallah. "Unmanned Aerial Vehicles 3-D Autonomous Outdoor Localization: A Deep Learning Approach." 2022 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2022. [CrossRef]
17. J. H. Borse and D. D. Patil, "Empirical Analysis of Feature Points Extraction Techniques for Space Applications," Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 9, pp. 81–87, 2021, doi: 10.14569/ijacsa.2021.0120910. [CrossRef]
18. Bojarski, Mariusz & Testa, Davide & Dworakowski, Daniel & Firner, Bernhard & Flepp, Beat & Goyal, Prasoon & Jackel, Larry & Monfort, Mathew & Muller, Urs & Zhang, Jiakai & Zhang, Xin & Zhao, Jake & Zieba, Karol. (2016). End to End Learning for Self-Driving Cars.

AUTHORS PROFILE

Utkarsh R. Moholkar, M.Tech. Research Scholar, Artificial Intelligence & Robotics, College Of Engineering, Pune, India. Utkarsh has worked as a Junior Data Engineer for Vamstar, India. And he is currently working as a Research Assistant at MKSSS's Cummins College of Engineering for Women, a funded research in the area of "Autonomous Space Missions" in collaboration with ISRO.

Dipti D. Patil, is currently working as Associate professor in Information Technology department at MKSSS's Cummins College of Engineering for Women, Pune. She is registered patent attorney of India. She has wide experience of patent drafting and filing. Dr. Dipti consults on national and international patent drafting and filing. Dr. Patil is serving as member, board of studies of Information Technology at Savitribai Phule Pune University, VIIT, Pune, AISSMS IOIT, Pune and St. Vincent Pallotti College of Engineering & Technology, Nagpur, RTM Nagpur University. Dr. Dipti is currently carrying funded research in the area of "Autonomous Space Missions" in collaboration with ISRO (Indian Space Research Organisation), U.R. Rao Satellite Centre, Bangalore.

Vinod Kumar, Director, Promotion Directorate, IN-SPACe, Department of Space, Executive Secretary, Astronautical Society of India. Experienced Deputy Project Director skilled in Matlab, C++, Engineering, Aerospace, and LabVIEW. Strong program and project management professional with a Doctor of Philosophy (PhD) focused in Aerospace Engineering from Indian Institute of Technology, Bombay.

Archana Patil, is Assistant Professor working at College of Engineering Pune, Maharashtra, India. Her teaching experience is in different subjects like Databases, Data Mining, Information Retrieval and research interests include Big Data, Machine Learning and Artificial Intelligence. She is also a life member of Indian Society for Technical Education (ISTE).