Simulation and Evaluation of Deep Learning Autoencoders for Image Compression in Multi-UAV Network Systems

1st Gabryel Silva Ramos
Electronic Engineering Department
CEFET-RJ
Rio de Janeiro, Brazil
gabryelsr@gmail.com

2nd Amaro Azevedo de Lima
Electronic Engineering Department
CEFET-RJ
Rio de Janeiro, Brazil
amaro.lima@cefet-rj.br

3rd Luciana F. Almeida
Electronic Engineering Department
CEFET-RJ
Rio de Janeiro, Brazil
luciana.almeida@cefet-rj.br

4th Jose Lima
Research Centre in Digitalization and Intelligent Robotics (CeDRI)
Instituto Politécnico de Bragança
Bragança, Portugal
jllima@ipb.pt

5th Milena Faria Pinto
Electronic Engineering Department
CEFET-RJ
Rio de Janeiro, Brazil
milena.pinto@cefet-rj.br

2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE) | 979-8-3503-1538-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/LARS/SBR/WRE59448.2023.10332986

***Abstract*—Mobile multi-robot systems are versatile alternatives for improving single-robot capacities in many applications, such as logistics, environmental monitoring, search and rescue, photogrammetry, etc. In this sense, this kind of system must have a reliable communication network between the vehicles, ensuring that information exchanged within the nodes has little losses. This work simulates and evaluates the use of autoencoders for image compression in a multi-UAV simulation with ROS and Gazebo for a generic surveillance application. The autoencoder model was developed with the Keras library, presenting good training and validation results, with training and validation accuracy of 70%, and a Peak Signal Noise Ratio (PSNR) of 40dB. The use of the CPU for the simulated UAVs for processing and sending compressed images through the network is 25% faster. The results showed that this compression methodology is a good choice for improving the system's performance without losing too much information.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The current state of the art in mobile robot applications research has little to debate about the fact that Unmanned Aerial Vehicles (UAVs) are one of the most versatile robot platforms for a wide range of applications [1, 2]. To cite some uses, there is surveillance [3], naval operations assistance [4], search and rescue [5], inspection [6], package delivering [7], environmental monitoring [8], topography [9], among others. The UAV systems also have improved their capacities when combined in groups or swarms, where coordinated vehicles can share the same tasks or individually execute parts of a bigger mission, gaining efficiency or even being able to accomplish operations where a single robot wouldn't be enough [10].

An essential aspect of the UAV multi-vehicle systems is the communication strategy between each part. The vehicles need to be able to send and receive data in a real-time pipeline to know each other's position, velocity, orientation, and flight autonomy and communicate with the mission's Ground Control Stations (GCS) [11]. These network systems may be designed as IoT-based (Internet of Things) architectures or even be nodes of an IoT architecture itself, like fog [12], fog-cloud [13], or edge-fog-cloud networks [14].

Some applications, such as surveillance, may also demand that the vehicles send and receive sensing data, for example, lidar readings, thermal images, or simple video footage frames [15]. Therefore, for designing reliable network communication systems for multi-UAV applications, some factors must be considered, such as communication latency, data throughput, package loss, interference, processing delay in network nodes, and power constraints [16]. Given the significant volume of data flowing through this system, efficient compression strategies are fundamental to reducing data volume without losing too much information.

According to [17], thousands of data compression research papers are published in the most important scientific bases. The author categorizes the techniques in compression methodologies based on data quality, coding schemes, data types, and applications. The present work mainly concerns compression based on application (wireless sensor networks) and data quality (lossy compression). Advances in Machine Learning (ML) and Deep Learning (DL) allowed the development of versatile models for data compression [18].

Once DL is mainly based on training a model to extract features from input data (coding) and then reconstructing the

979-8-3503-1538-7/23/$31.00©2023 IEEE

input at the output based on those reduced features (decoding), it is possible to use those properties to reduce (code) the data generated in network nodes, and then recreate it on its destination (decode). If necessary, the same node may send the compressed data by coding and decoding the original data, with some loss, a methodology described as autoencoder [19]. This paper aims to propose a linear autoencoder architecture based on a DL model and evaluate its training, test, and validation results. The trained autoencoder is integrated with a multi-UAV control simulation which captures video data, evaluating the results of transmitting compressed and uncompressed image data during the simulation workout.

The present work aims to develop a multi-UAV simulation framework with ROS and Gazebo packages and integrate it with a data (image) compression DL model, evaluating its robustness within the robot system. According to [20] state-of-the-art review about multi-UAV systems development, the published works present only simulated results, with physical tests mostly being executed indoors with only dozens of robots, and even fewer tests are executed outdoors.

Main contributions of the present work are *(i)* The generic multi-UAV development platform with reconfigurable separated sensors, controls, and sensors modules built with ROS; *(iii)* The multi-UAV development framework ready for outdoors field tests using up to 10 vehicles and capacity to coordinate missions that demand large distances between robots;*(iii)* The linear trained model for image compression, developed with Keras and already integrated to the control modules in ROS.

II. Proposed Methodology

The proposed work is divided into three main parts, which are the development, training, and validation of the autoencoder model for image compression, the development of the multi-agent simulation with ROS and Gazebo, and the integration of the trained and validated model into the ROS control node, evaluating the real-time results of image compression and the controller behavior operating with and without compressed data.

A. Autoencoder model

Autoencoders are a type of neural network that tries to rebuild the input data at its output but with an initial stage of reducing the input data to essential features and then building the output from this reduced set of features. The first stage of the network is called the encoder, and the second is called the decoder. The encoder outputs are the reduced features that reconstruct the original input data. Autoencoders use unsupervised training to learn how to effectively reduce the information (code) and then represent the input again at its output (decode), with some loss of information (compression). The TensorFlow Keras library was used for the model construction and training.

The proposed autoencoder's input and output layers have 96×96×3 shapes, representing an RGB input image with 96×96 pixels. The encoder structure consists of 4 groups of 4 convolutional 2D layers interspersed and stacked with 4 batch normalization layers, totaling eight layers per group. The convolutional 2D layers create a convolution kernel convoluted with the layer input to produce a tensor of outputs. Those kernels represent the information extracted from the input data. The batch normalization layers apply a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. That's why a batch normalization layer must be between two consecutive convolutions (the input data must also be normalized before compression). Between each group of layers, there is a maximum pooling 2D layer, which downsamples the number of features extracted from each group, followed by a dropout layer to help prevent overfitting during training. The encoded input at the end of the encoder stage has dimension 12×12×32.

The decoder structure comprises four groups of 4 convolutional 2D layers interspersed and stacked with 4 batch normalization layers. Still, instead of the maximum pooling 2D layers followed by the dropout layer, there is an upsampling 2D layer for reconstructing the data to its original format. The decoder output has shape 96×96×3, the same as the input, which also represents an RGB image of 96×96 pixels. In other words, the decoder tries to reproduce the input image data with a reduced set of features after the encoder process. In order to train the model, it was used the STL-10 dataset [21], which is composed of 10,000 RGB generic images of 96×96 pixels, with 6,000 being used for training and 4,000 used for the test. The training was performed in 100 epochs, with a batch size of 10, and Adam optimizer with a learning rate 0.0005. The loss and validation loss criteria were the minimum squared error and the accuracy criteria performed for the evaluation.

In order to validate the model, it was studied the loss and validation loss curves, as well as the accuracy and validation accuracy curves. It also evaluated the peak-to-signal noise ratio (PSNR) variation for 1 to 10 consecutive compressions, presented in Equation 1.

$$PSNR = 10\log_{10}\left(\frac{{MAX_I}^2}{MSE}\right) \tag{1}$$

Where:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(I_{(i,j)} - K_{(i,j)})^2 \tag{2}$$

MAX_I is the maximum possible pixel value of the image, and MSE is the minimum squared error between the pixels (i, j) of images I (original) and K (compressed), of shape m×n. The structural similarity index (SSIM, given in Equation 3) and the multi-scale structural similarity (MS-SSIM, presented in Equation 4) were evaluated as well.

$$SSIM = \frac{[2\mu_x\mu_y + (k_1L)^2][2\sigma_{xy} + (k_2L)^2]}{[{\mu_x}^2 + {\mu_y}^2 + (k_1L)^2][{\sigma_x}^2 + {\sigma_y}^2 + (k_2L)^2]} \tag{3}$$

$$MS - SSIM = \frac{1}{M}\sum_{j=0}^{M} SSIM(x_j, y_j) \tag{4}$$

The SSIM measures the similarity between two images, where μ_x and μ_y are the average values of x and y respectively, ${\sigma_x}^2$ and ${\sigma_y}^2$ are the variances of x and y respectively, σ_{xy} is the covariance between x and y, and finally $(k_1 L)^2$ and $(k_2 L)^2$ are two stabilization constants, where in this analysis $k_1 = 0.01$, $k_2 = 0.03$ and $L = 2^n - 1$ where n is the number of bits per pixel. MS-SSIM is nothing else but the average SSIM for a set of images. All training and evaluation were executed from an Intel NUC 38i3BEK, using Google Collab computational resources to speed up the training.

B. Multi-UAV simulation with ROS and Gazebo

The first step for creating the simulation in ROS to integrate with the autoencoder for image compression was establishing a generic application that uses aerial imaging to accomplish its goals. It was chosen as a mission in an offshore environment where the UAVs will follow programmed paths for environmental surveillance (search for oil spills, for example). The UAV swarm should take off from the oil production facility's helideck and send its camera images to the control station. First, to simulate the environment in which the mission will take place, an oil rig and a shuttle tanker vessel were designed with Blender 3D modeling software, Gazebo packages for designing the sky, clouds, sea, and ODE plugins for wind and sea parameters, the world on which the simulation took place was created. Figure 1 presents the results.

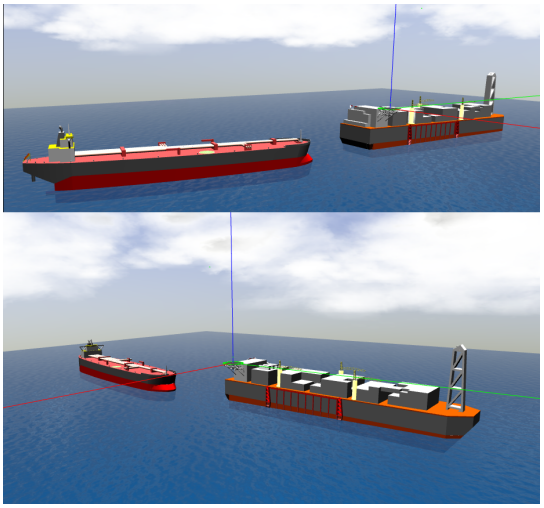

Fig. 1. World created in Gazebo for simulating the image transmission system on an offshore mission.

The main idea adopted for developing the UAVs simulation was to launch separated agents with the PX4 FCU and the developing custom nodes representing offboard controllers. These offboard nodes will communicate to their respective FCU through MAVLINK protocol (in this case, MAVROS) to receive data from sensors, alarms, status, position, velocity, and any other variable that is important for the operational controller of each robot. In real life, this offboard controller may be a companion computer like a Raspberry Pi attached to the UAV, and the communication between the UAV and the companion computer may happen through serial communication via USB or the MAVLINK telemetry radio (one attached to the FCU and one attached to the companion computer). In the offboard controller, packages will be developed for controlling its respective UAV, communicating with the simulated network, and using a different sensor that can't be embedded in the UAV's FCU. In the simulation, this sensor is the camera, which is a Gazebo external plugin streaming the image "seen" by each UAV through a pipeline using the Python GStreamer library and OpenCV, transformed to a camera node that communicates to each offboard node. The active nodes diagram for a 3 UAV group developed under this logic is presented in Figure 2.

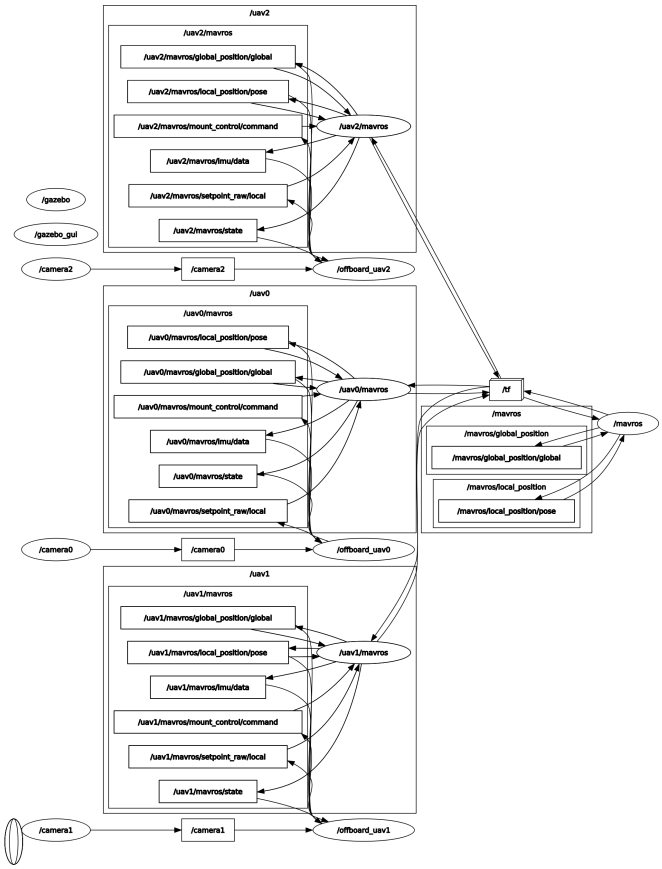

Fig. 2. Active ROS nodes and topics for 3 UAV groups in simulation.

The image compression autoencoder model is also integrated into the offboard nodes, which take the camera's image from the camera nodes and transmit it to the ground station. In real life, this may be addressed with WiFi, ZigBee, or other wireless communication technology associated with the companion computer. The simulated UAV was a Typhoon H480 model, and the simulation was performed with groups of 1 to 10 vehicles. Figure 3 presents the simulated UAVs in the Gazebo environment.

For the image transmission test, the offboard controllers were programmed for a 10 meters takeoff, then described a squared trajectory of 20 m $\times$ 20 m side around the oil rig.

Fig. 3. Swarm of 10 Typhoon H480 in simulation.

The results will be presented in the next session.

III. Results and Discussion

The results will be presented in the following subsections, one detailing training and static evaluation of the developed autoencoder and one containing the results obtained when integrating the autoencoder into the multi-UAV simulation.

A. Autoencoder evaluation

Figure 4 presents the loss and validation loss during training the autoencoder.

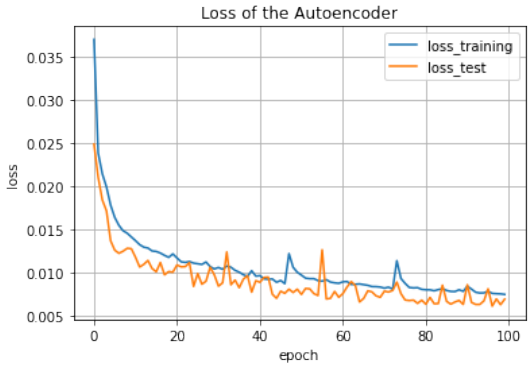

Fig. 4. Loss and Validation Loss in the autoencoder's model training.

The training began with 0.0370 for loss and 0.0249 for validation loss at the end of the first epoch and ended with 0.0075 for loss and 0.0069 for validation loss. Usually, validation loss is greater than training loss, as the network is trained with those data, but the dropout layers in the encoder structure penalize the model variance by randomly freezing neurons in a layer during model training. Since it applies only to the training process, it affects training loss, leading to a validation loss lower than training loss. This is a good indicator that the autoencoder could compress generic images from the dataset, even if it is new (validation images). Figure 5 below presents the training's accuracy and validation accuracy results.

The final accuracy for the training was 0.7053, and for the test, it was 0.6982. This is also similar to the loss and indicates that the model could predict the output data given the input data with almost 70% precision. For an object detector, this

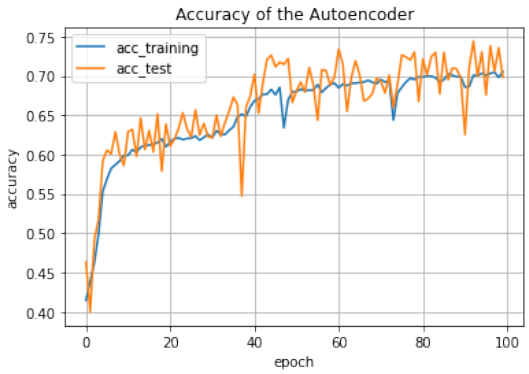

Fig. 5. Accuracy and Accuracy Loss in the autoencoder's model training.

could be considered a low result, but since the network goal is to reduce transmitted bytes in the proposed system, this is a solid result and shows that the trained model can accomplish the task. A compression result example is presented in Figure 6, which shows the input (a random image from the STL-10 dataset) data and the output compressed data.

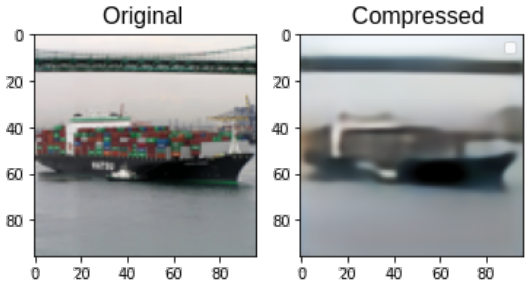

Fig. 6. Compression example.

As the model prediction is fast, improving byte transmission reduction (with the cost of losing data) compresses the image several times before streaming it. The number of compressions for each UAV transmitting images could be defined in the offboard controller nodes, being a collective decision of the group which agent should detail more or less the data they're sending. In order to validate the autoencoder's performance, an image from outside the training and validation dataset was used, being compressed from 1 to 10 times, and the PSNR, SSIM, and MS-SSIM were measured. Figure 7 presents the variation results for each indicator.

The similarity of the compressed image decreases with successive compression as expected, and the signal/noise relation also decays fast. After the seventh compression, the image is practically unrecognizable. Figure 8 presents the result of the image subjected to successive compression with the autoencoder.

With the autoencoder ready to use, the next step was integrating it with the multi-UAV simulation system.

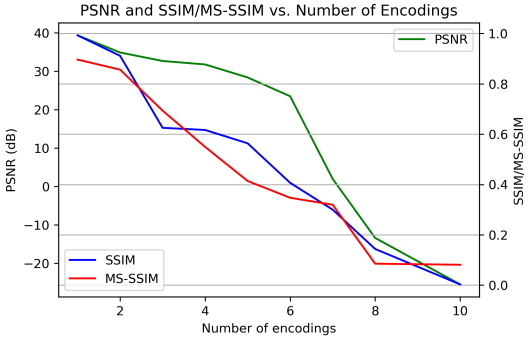

Fig. 7. PSNR, SSIM, and MS-SSIM variation for interlinked compression.

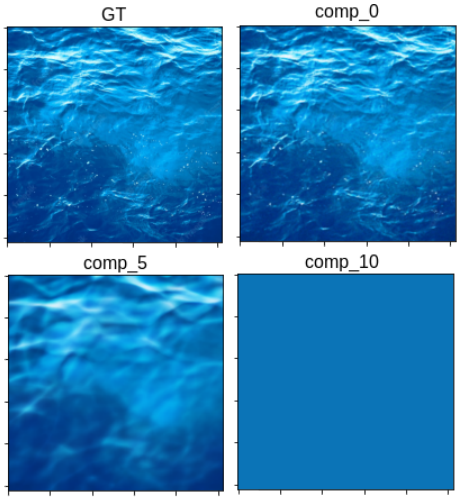

Fig. 8. Interlinked compression example.

B. Image compression evaluation in simulation

Nine simulations were carried out, each one increasing the number of drones from 1 to 10 units. In [22], it is possible to see a video of an example using 2 UAVs. Each time one UAV was added, the simulation ran slower due to computational restrictions. Figure 9 presents the route flight by each UAV in the simulation with ten agents.

All robots could compress and stream their camera's images through the designed network. The images were presented at a 25 fps rate, and the autoencoder decreased this to 15 when performing ten successive compressions. The CPU use of each offboard controller was evaluated with and without the image compression, using only one encoding. The results are presented in Figure 10.

The CPU is in less demand when the images are compressed because the process deals with less data. The streamed data is also much smaller than the original image, reconstructed at the origin. In a real application scenario, the network could process which UAV must stream images with more or less quality, ensuring signal continuity.

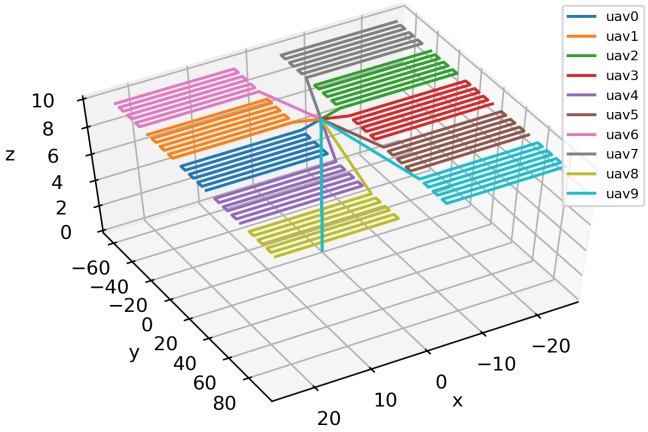

Fig. 9. Route performed by the UAV swarm in simulation.

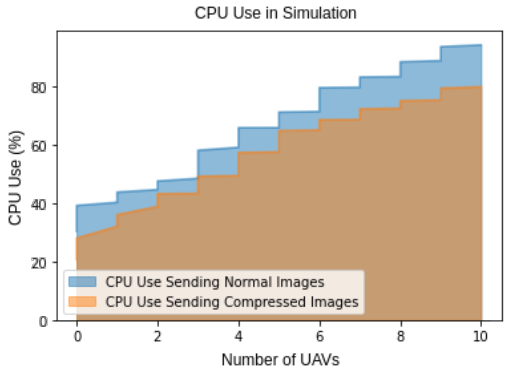

Fig. 10. CPU use during the simulation.

IV. Conclusions and Future Work

The autoencoder was satisfactorily trained and tested, with robust validation results. The proposed strategy to simulate and control the UAV swarm also worked as expected, opening the possibility for future experiments since several intelligent routines, new sensors, and communication pipelines can be integrated with the offboard controllers. Those packages are also ready to be used in real robots using the PX4 FCU. This was also proved by integrating the encoder and decoder models into the controllers for compressing and streaming data, reducing the necessity of CPU usage and dealing with fewer bytes. For future works, this system must be integrated with a network simulator like OMNET++ or NS-3 to verify other constraints regarding the communication of robot swarms and the compression performance with different communications protocols. The swarm framework must also be developed and presented as a generic application for swarm simulation. Finally, field tests must be performed to validate the proposed framework.

Acknowledgment

The authors thank CEFET/RJ, UFF, UFRJ, and the Brazilian research agencies CAPES, CNPq, and FAPERJ. Besides, the

authors are grateful to the Foundation for Science and Technology (FCT, Portugal) for financial support through national funds FCT/MCTES (PIDDAC) to CeDRI (UIDB/05757/2020 and UIDP/05757/2020) and SusTEC (LA/P/0007/2021).

REFERENCES

[1] J. Parikh and A. Basu, *Unmanned Aerial Vehicles: State-of-the-Art, Challenges and Future Scope*, ch. 2, pp. 29–42. John Wiley Sons, Ltd, 2021.

[2] G. S. Ramos, M. F. Pinto, F. O. Coelho, L. M. Honório, and D. B. Haddad, "Hybrid methodology based on computational vision and sensor fusion for assisting autonomous uav on offshore messenger cable transfer operation," *Robotica*, vol. 40, no. 8, pp. 2786–2814, 2022.

[3] M. F. Pinto, A. G. Melo, A. L. Marcato, and C. Urdiales, "Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pp. 1324–1329, IEEE, 2017.

[4] G. S. Ramos, D. Barreto Haddad, A. L. Barros, L. de Melo Honorio, and M. Faria Pinto, "Ekf-based vision-assisted target tracking and approaching for autonomous uav in offshore mooring tasks," *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 3, no. 2, pp. 53–66, 2022.

[5] C. D. Rodin, L. N. de Lima, F. A. de Alcantara Andrade, D. B. Haddad, T. A. Johansen, and R. Storvold, "Object classification in thermal images using convolutional neural networks for search and rescue missions with unmanned aerial systems," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2018.

[6] G. S. Berger, M. Teixeira, A. Cantieri, J. Lima, A. I. Pereira, A. Valente, G. G. de Castro, and M. F. Pinto, "Cooperative heterogeneous robots for autonomous insects trap monitoring system in a precision agriculture scenario," *Agriculture*, vol. 13, no. 2, p. 239, 2023.

[7] L. D. P. Pugliese, F. Guerriero, and G. Macrina, "Using drones for parcels delivery process," *Procedia Manufacturing*, vol. 42, pp. 488–497, 2020. International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019).

[8] D. R. Green, J. J. Hagon, C. Gomez, and B. J. Gregory, "Chapter 21 - using low-cost uavs for environmental monitoring, mapping, and modelling: Examples from the coastal zone," in *Coastal Management* (R. Krishnamurthy, M. Jonathan, S. Srinivasalu, and B. Glaeser, eds.), pp. 465–501, Academic Press, 2019.

[9] S. Manfreda, P. Dvorak, J. Mullerova, S. Herban, P. Vuono, J. J. Arranz Justel, and M. Perks, "Assessing the accuracy of digital surface models derived from optical imagery acquired with unmanned aerial systems," *Drones*, vol. 3, no. 1, 2019.

[10] A. Tahir, J. Boling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles — a survey," *Journal of Industrial Information Integration*, vol. 16, p. 100106, 2019.

[11] G. Asaamoning, P. Mendes, D. Rosário, and E. Cerqueira, "Drone swarms as networked control systems by integration of networking and computing," *Sensors*, vol. 21, no. 8, 2021.

[12] A. Gupta and S. K. Gupta, "Uav aided fog network (uafn): A proposal framework for better qos," in *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, pp. 265–270, 2022.

[13] H. A. Alharbi, B. A. Yosuf, M. Aldossary, J. Almutairi, and J. M. H. Elmirghani, "Energy efficient uav-based service offloading over cloud-fog architectures," *IEEE Access*, vol. 10, pp. 89598–89613, 2022.

[14] Z. Chen, N. Xiao, and D. Han, "A multilevel mobile fog computing offloading model based on uav-assisted and heterogeneous network," *Wireless Communications and Mobile Computing*, vol. 2020, p. 8833722, Jul 2020.

[15] J. Scherer and B. Rinner, "Persistent multi-uav surveillance with energy and communication constraints," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1225–1230, 2016.

[16] M. F. Pinto, A. L. Marcato, A. G. Melo, L. M. Honório, and C. Urdiales, "A framework for analyzing fog-cloud computing cooperation applied to information processing of uavs," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[17] U. Jayasankar, V. Thirumal, and D. Ponnurangam, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 2, pp. 119–140, 2021.

[18] J. Park, H. Park, and Y.-J. Choi, "Data compression and prediction using machine learning for industrial iot," in *2018 International Conference on Information Networking (ICOIN)*, pp. 818–820, 2018.

[19] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 415–419, 2018.

[20] M. Abdelkader, S. Güler, H. Jaleel, and J. S. Shamma, "Aerial swarms: Recent applications and challenges," *Current Robotics Reports*, vol. 2, pp. 309–320, Sep 2021.

[21] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 215–223, PMLR, 11–13 Apr 2011.

[22] G. S. Ramos, "Multi uav environmental surveyllance simulation with ros/gazebo." https://www.youtube.com/watch?v=u_XFkuzfRcs, 2022. Accessed in 04/12/2022.