2024 Brazilian Symposium on Robotics (SBR), and 2024 Workshop on Robotics in Education (WRE) | 979-8-3503-9108-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/SBR/WRE63066.2024.10837937

Recognizing Human Actions: A Deep Learning Model for UAV Piloting

Iohana A. Torres Cabral, Marco Aurelio Wehrmeister, André Eugenio Lazzaretti, Heitor Silverio Lopes
Universidade Tecnológica Federal do Paraná (UTFPR)
Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI),
Av. Sete de Setembro, 3165, 80230-901 Curitiba, Brasil
iohana@alunos.utfpr.edu.br, {wehrmeister, lazzaretti, hslopes}@utfpr.edu.br

***Abstract*—This project aims to explore action recognition through a deep learning model generated by Convolutional Neural Networks, establishing the foundation for human-robot interaction in a scenario where Unmanned Aerial Vehicles (UAV) are controlled exclusively by visual commands. The model analyzes images captured by an onboard camera using and classifies them into nine categories. Each category issues a specific command based on human actions performed by individuals properly equipped with personal protective equipment. The results demonstrate the feasibility of the proposed approach, opening room for improvements aiming its use in more complex scenarios.**

I. Introduction

Even before the Industrial Revolution, literature fantasized about artificial beings coexisting with humans. The term ”robot” itself came from the Czech word ”robota,” originated from a 1920 tail. “R.U.R”, by Karel Čapek depicted ”robota” as a humanoid created for servitude [1]. With the growing evolution of cinema, interest in the theme spread and fueled human creativity regarding the possibilities of automation. As technology advances and robots become increasingly integrated into our society, human-robot interaction will continue to play an important role in a wide range of fields, from healthcare and education to industry and entertainment. With this understanding, the amount of research is growing not only on ways to communicate effectively with machines but also on more practical, intuitive, and accessible control alternatives.

Aiming to stimulate research in the field of human-robot interactions and the development of autonomous and intelligent UAVs, the Flying Robot Trial League [2] of RoboCup Brazil has devised a challenge for the national competition that involves controlling a UAV using only visual commands: The robots must land on two fixed suspended bases and four mobile bases placed in the arena.

The UAV focus of this project is a quadcopter, commonly known as a drone. Figure 1 shows the 4 movements of a quadcopter:

- Yaw is the rotation around the frame’s center of mass.
- Pitch is the horizontal movement forward/backward.
- Roll is the horizontal movement to the right/left.
- Throttle is the vertical movement up/down.

979-8-3503-9108-4/24/$31.00©2024 IEEE

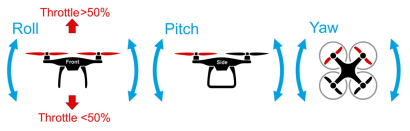


Fig. 1: UAV movements [3]

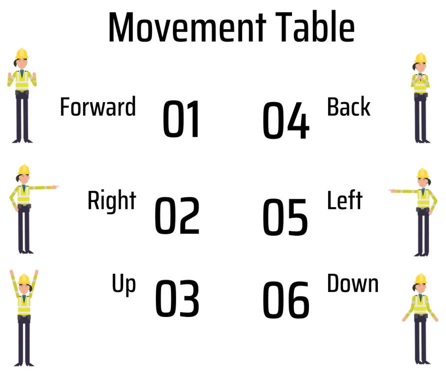


Fig. 2: Table of gestures to be recognized by the model with their respective commands.

We propose an approach to pilot the UAV (i.e., control its movement) by using visual command as depicted in Figure 2. In other words, the operator performs a gesture, and the UAV moves according to the visual command it identifies. The focus is on the following movements: ascend, descend, move right, move left, move forward, and move backward (Yaw will not be considered). We created a computer vision system based on YOLOv8 that detects visual command and provides input to a PID system that controls UAV movement accordingly. The results are encouraging and indicate that the proposed approach is feasible, as the vision system achieved a precision of 95,83% in 9 classification classes (i.e, visual commands).

This paper is organized as follows: Section II analyses the related work. Section III discusses the construction of the dataset and the tools used for its processing; Section IV presents the results and discusses some improvements; Finally, Section V draws some conclusions and discusses limitations and the future work.

II. Related Work

Since the 1960s, object detection in images has been a core focus of computer vision, laying the foundation for many artificial intelligence systems. Over time, various methods have been developed to detect and classify object categories. However, when applied to aerial vehicles, object detection becomes more challenging due to factors like limited aerial datasets and the inherent complexity of human actions, which are difficult to model accurately. These issues compound the challenges already faced by machine learning and deep learning models in achieving precise action recognition.

One notable study in human action recognition using aerial datasets is [4], which achieved an accuracy of 87.80% using a Multilayer Perceptron (MLP) and 87.77% using a Support Vector Machine (SVM). Five other classifiers used in the study showed significantly lower performance. Another approach, highlighted in [5], reached 95% accuracy by classifying keypoints extracted from videos using CNNs, although it required high computational power.

In more recent developments, as detailed in [6], a fast DNN object detector and a custom LDES-ODDA visual tracker were combined in a unified detection-and-tracking system, alternating between detection and tracking tasks. This approach yields a bounding box around the detected human, which is then used for gesture recognition and human state estimation. Following this, a 2D skeleton representing visible body joints in pixel coordinates is extracted using an enhanced multi-branch CNN model, as described in [7].

III. Metodology

A. Dataset

1) Collecting Data: Let us consider that the robot is not in a controlled environment. To ensure that the dataset encompasses a wide variety of examples and relevant use cases, the video collection was done with the following conditions:

- Recordings at 30 frames per second;
- Different camera resolutions;
- Various filming angles;
- Distinct lighting conditions;
- Indoor environments;
- Outdoor environments;
- Controller using PPE;
- Controller without PPE;
- Backgrounds with movement;
- Backgrounds without people.

Since this is an experimental dataset, it is not publicly available due to privacy and ethical regulations concerning image authorization and sensitive data [8]. Initially, videos were recorded with all six positions of interest, along with an additional one that associates the drone with a stop command. This position involves the operator standing with their arms close to their body. Additionally, two more categories were added for observational analysis: ”w_ppe” represents everybody that is not wearing personal protective equipment, and ”x” represents any frame where the operator is performing an action not previously listed. As mentioned, the three added classes (stopped, x, and w_ppe) are for analytical purposes, and the intent is for none of these to send movement commands to the drone.

2) Manual preprocessing: These videos were divided into short clips and labeled with the corresponding actions (1-backward, 2-down, 3-forward, 4-left, 5-right, 6-stopped, 7-up, 8-w_pee, 9-x). Since YOLOv8 [9] (the method used) does not require a model trained on video files for accurate real-time recognition, to reduce computational load and data size, a Python script was programmed using the OpenCV [10] library to extract the last frames from all videos containing actions. Figure 3 shows a flowchart of all processes executed up to the final dataset.

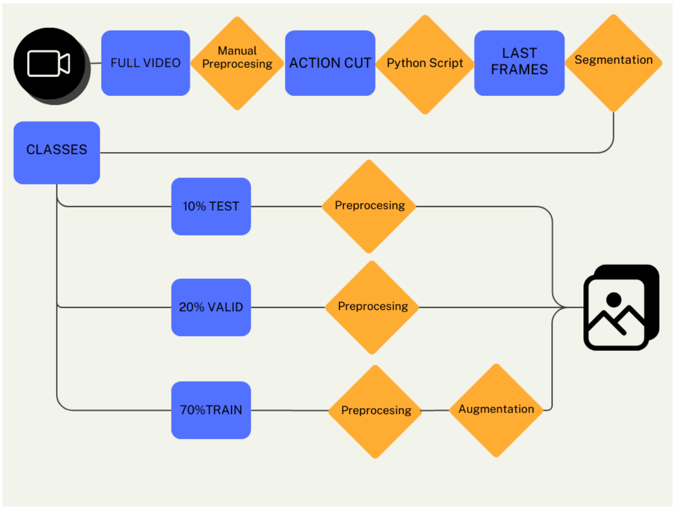


Fig. 3: Dataset Preparation Flowchart

3) Classification and Division: To properly label the images, the Roboflow platform was used. To ensure a balanced set of examples, the nine classes were individually labeled and split into 70% training, 20% validation, and 10% test sets. Only after this all the sets (training, validation, and test) were combined. Figure 4 shows the state of the dataset at the end of this process.

4) Preprocessing: The goal of pre-processing is to reduce training time and improve performance by transforming the dataset images. To ensure consistency, it is necessary for the data to be normalized. In this dataset, the following transformations were applied to all images:

- Conversion to ”.jpg” format to ensure data consistency;
- Resizing to a width of 320 pixels while keeping the height variable to maintain aspect ratio and normalize the dataset. See Figure 5;
- Auto-orientation to ensure that inferences use the image metadata, capturing orientation information at the time of capture;

Here, the term ”real-time” is used in the popular sense to refer to an online or instant-response system.

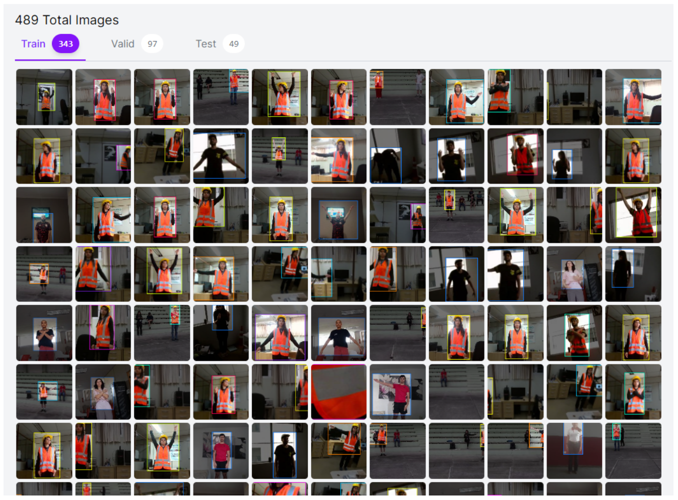


Fig. 4: Sample of the dataset after bounding box labeling

- Auto-contrast applies contrast based on the image histogram to improve normalization and line detection under adverse lighting conditions. See Figure 6.

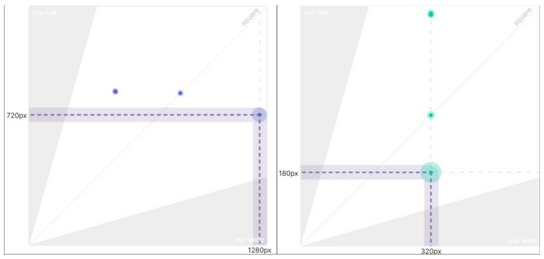


Fig. 5: The left chart represents the average dimensions of the images before normalization (1280x720). The right chart represents the average dimensions of the images after preprocessing (320x180).

5) Data Augmentation: Data Augmentation is a data enhancement technique where transformations or variations are applied to existing images in the training set to create new examples. These techniques can help improve model generalization and reduce overfitting. Below are the strategies adopted to augment the datasets for this project:

- Grayscale. Converts 15% of the training images to grayscale, which reduces the model's emphasis on color as a signal.
- Hue. Randomly alters the colors of an image between -15º and 15º, ensuring that the model does not memorize the colors of objects in the scene. This helps the model recognize objects even if the color of the operator's protective equipment changes.
- Saturation. Randomly adjusts the color saturation between -15% and 15% so that the model performs better even with slight color changes.
- Brightness. Varies the brightness of the image between -15% and +15% to help the model be more resistant to changes in lighting and camera settings.

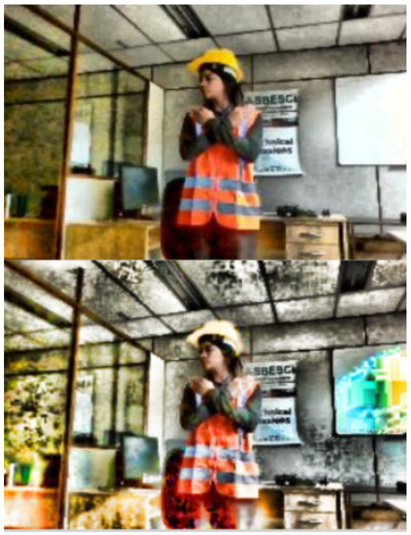


Fig. 6: Two examples used in training. The top image is the original, and just below is the normalized image with auto-contrast using the equalizer adjustment.

- Exposure. Varies the exposure of the image between -15% and +15% to help the model be more resilient to lighting changes and camera settings.
- Blur. Adds random Gaussian blur between 0 and 1.5 pixels to help the model be more resilient to focus changes in the camera.
- Noise. Adds "salt and pepper" noise to 1.49% of the pixels to help the model be more resistant to noise that might decrease image readability.

After all processes, the final dataset consists of 1,175 images, with 1,029 for training, 97 for validation, and 49 for testing (see Table I and Figure 7).

TABLE I: Classes banlance

Class	Sample
stoped	320
w_ppe	311
up	86
x	84
backward	82
right	76
left	75
forward	72
down	69

B. Training

For the final application of this project, reducing computational costs for energy savings and achieving high accuracy are the primary justifications for the choice of model. As stated, this project is based on a method for which all versions have been considered state-of-the-art at the time of their release.

1) YOLOv1: YOLO (You Only Look Once) [12] is a computer vision tool launched in 2015 for object detection. The premise of the method is to use a convolutional neural network to extract features by looking at the image only

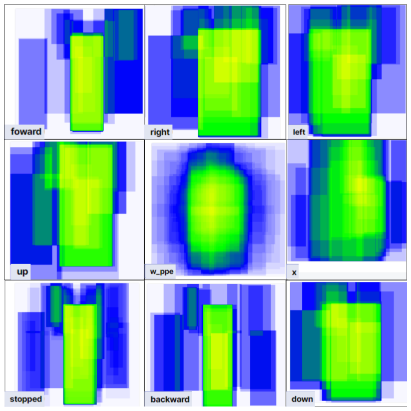


Fig. 7: classes heatmaps

once (single pass). The YOLO architecture consists of a single-path CNN (Convolutional Neural Network) that uses 1×1 convolutions followed by 3×3 convolutions and two final fully connected layers (where the output is interpreted). The activation function applied in all layers is Leaky ReLU [13], except in the final layer, where a linear activation function is used. Starting from SSE (sum of squared errors) due to its optimization convenience, the creators noticed that the model suffered from instability. This was because the loss function equally weighted the localization error with the classification error, and the cell scores approached zero since many of them did not contain any objects. To address this, the loss function was adapted, resulting in:

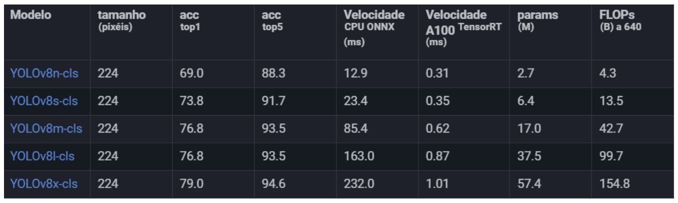
$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \nVdash_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \nVdash_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \nVdash_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \nVdash_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \nVdash_{i}^{\text{obj}} \sum_{c \in \text{ classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{1}$$

2) YOLOv8: As of the start of this project, the latest version of YOLO was v8, so this is the architecture we will be working with. An official paper has not yet been released, but the Ultralytics documentation [14] is quite detailed. Regarding accuracy, Binary Cross-Validation is widely recommended in the literature to reduce multi-class classification errors. In this model, the loss function operates according to the following equation:

$$\begin{aligned}
\mathcal{L} = &\frac{\lambda_{\text{box}}}{N_{\text{pos}}} \sum_{x,y} \nVdash_{c^*_{x,y}} \left[1 - q_{x,y} + \frac{\left\| b_{x,y} - \hat{b}_{x,y} \right\|_2^2}{\rho^2} + \alpha_{x,y} \nu_{x,y} \right] \\
&+ \frac{\lambda_{\text{cls}}}{N_{\text{pos}}} \sum_{x,y} \sum_{c \in classes} y_c \log(\hat{y}_c) + (1 - y_c) \log(1 - \hat{y}_c) \\
&+ \frac{\lambda_{\text{dfl}}}{N_{pos}} \sum_{x,y} \nVdash_{c^*_{x,y}} \left[- \left(q_{(x,y)+1} - q_{x,y} \right) \log \left(\hat{q}_{x,y} \right) \right. \\
&\left. + \left(q_{x,y} - q_{(x,y)-1} \right) \log \left(\hat{q}_{(x,y)+1} \right) \right]
\end{aligned} \tag{2}$$

Each cell determines its best candidate for predicting the object's bounding box. This loss function includes the CIoU loss as the box_loss, the standard binary cross-entropy for multi-label classification as the classification loss (allowing each cell to predict more than one class), and the distribution focal loss. In terms of speed, YOLOv8, unlike its predecessors, uses anchor-free detection to enhance generalization. Anchor-based detection reduces learning speed on custom datasets due to pre-defined anchor boxes, which may negatively impact this project. Anchor-free detection allows the model to predict the center point of an object directly, reducing the number of bounding box predictions, accelerating Non-Maximum Suppression (NMS), a preprocessing step that eliminates inaccurate predictions [15].

Given its functionality, Ultralytics has provided five YOLOv8 models with different initial weights to be adapted according to the developer's needs. Observing Figure 8 about these models, we can infer that the 8n-cls performs worse than the 8x-cls in terms of accuracy but excels in speed due to its optimization. For comparison, both extreme models (8n-cls and 8x-cls) were trained in classification mode.



Modelo	tamanho (pixéis)	acc top1	acc top5	Velocidade CPU ONNX (ms)	Velocidade A100 TensorRT (ms)	params (M)	FLOPs (B) a 640
YOLOv8n-cls	224	69.0	88.3	12.9	0.31	2.7	4.3
YOLOv8s-cls	224	73.8	91.7	23.4	0.35	6.4	13.5
YOLOv8m-cls	224	76.8	93.5	85.4	0.62	17.0	42.7
YOLOv8l-cls	224	76.8	93.5	163.0	0.87	37.5	99.7
YOLOv8x-cls	224	79.0	94.6	232.0	1.01	57.4	154.8

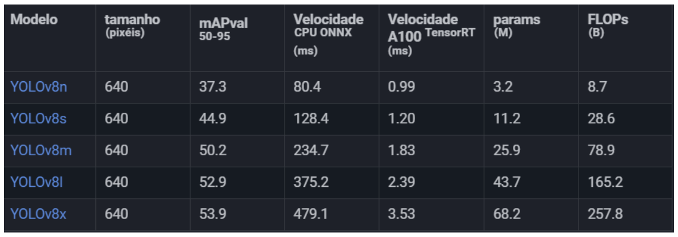
Fig. 8: Pre-trained Classification Models [14]

The two equivalent models (8n and 8x) were also trained for the detection mode. Although this requires a higher computational cost than the classification models: see the "FLOPs" column in Figures 8 and 9; the output of bounding boxes may be useful in the future for camera position control.

IV. Results and Discussions

Inferences from the first two models, one for testing and one for classification, showed confusion between "left" and "right" as well as between "up" and "down." This information is supported by the normalized confusion matrix shown in Figure 10b.

This is because YOLOv8, by default, performs data augmentation before training by rotating images horizontally and



Modelo	tamanho (pixéis)	mAPval 50-95	Velocidade CPU ONNX (ms)	Velocidade A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Fig. 9: Pre-trained detection models. [14]

vertically. To correct these errors, undesirable parameters such as ”fliplr” and ”translate” were disabled in subsequent trainings, along with ”erasing” (which removes part of the image). On the other hand, ”scale” was increased to adapt the model to perceive images at different distances, as classification errors were also observed when the operator was farther from the camera.

Minimizing false detections is a priority in a scenario involving drones and humans. For safety, it is acceptable for the machine to remain static if it does not identify a command, but it is not acceptable for it to move involuntarily, potentially causing a critical accident. For this reason, the metric of interest is ”precision,” aiming to minimize false positives. The ”cls” was adjusted to ”1” to ensure that classification precision has a greater impact than the bounding box location in the loss function. Training parameter adjustments can be seen in Table II.

TABLE II: Arguments for Training the Models

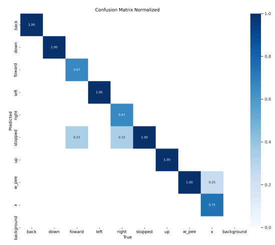
Arg	**Config**
mode	train
epochs	80
patience	15
val	True
translate	0
fliplr	0
scale	1.0
erasing	0
cls	1

The precision of the initial models 8x and 8n-cls were 0.86887 and 0.91667, respectively. Following parameter adjustments, the results are presented in Table III.

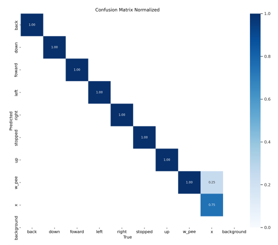
TABLE III: Models Accuracy

Model	**Accuracy**
YOLOv8n-cls	0,9325
YOLOv8x-cls	0,9583
YOLOv8n	0,9027
YOLOv8x	0,9104

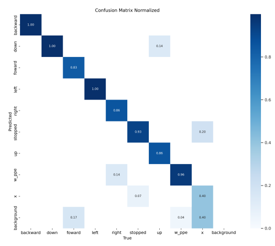
The analysis of the test videos shows quite satisfactory results, with no significant failures occurring among the six main commands. Misclassifications in the test samples mainly involve ”stopped,” ”x,” and ”w_pee.” These classifications are



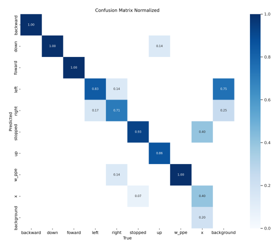
(a) YOLOv8n-cls.



(b) YOLOv8x-cls.



(c) YOLOv8n.



(d) YOLOv8x.

Fig. 10: Confusion Matrices

not considered commands for the machine. By narrowing the model to validate only 6 out of the 9 classifications, a precision of approximately 0.98 is expected.

Confusion matrices for the 4 models are shown in Figure 10. The results for the best classification and regression models are depicted in Figures 11 and 12, respectively.

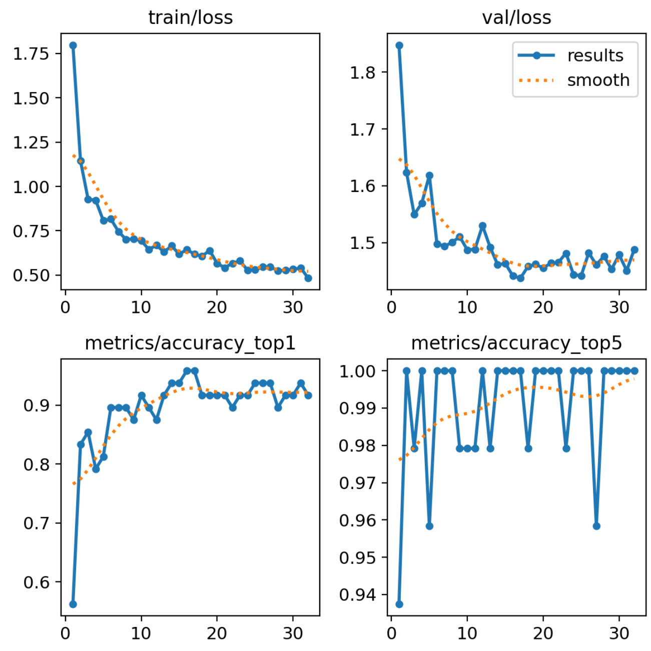


Fig. 11: YOLOv8x-cls metrics.

V. Conclusion

The results of the final models demonstrated satisfactory accuracy, with precision values ranging from 0.9027 to 0.9583.

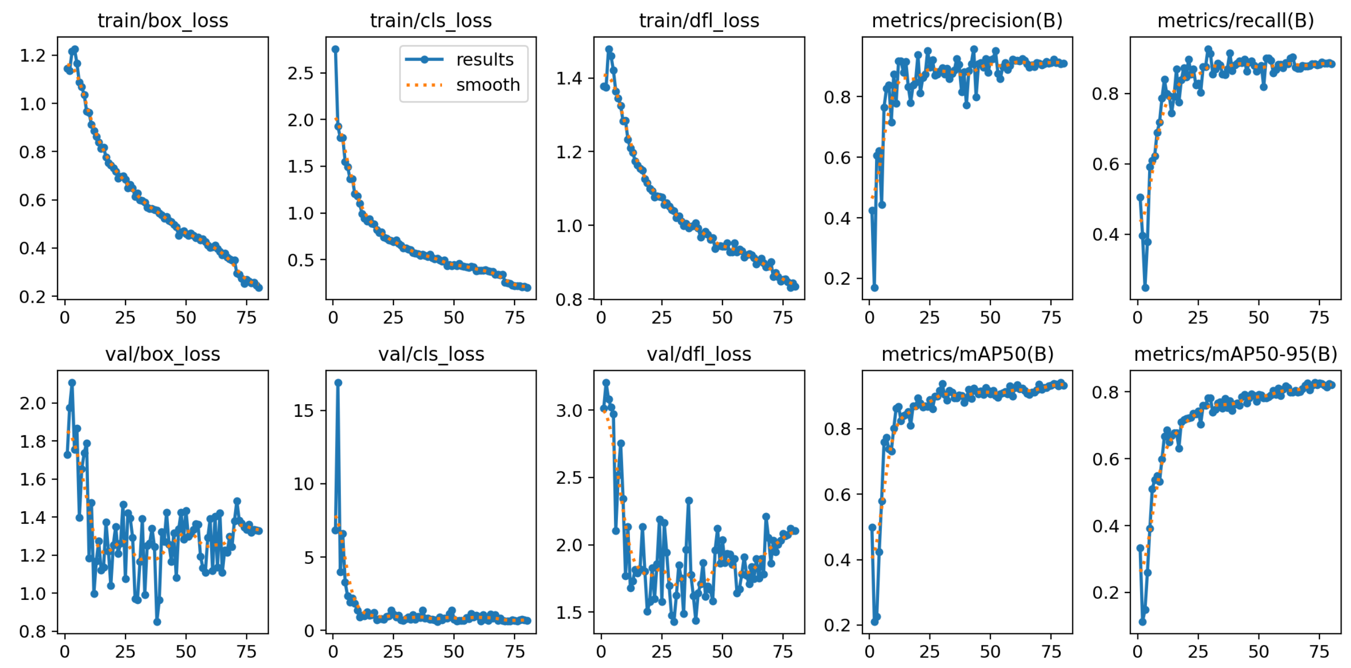


Fig. 12: YOLOv8x metrics.

Analysis of the test videos confirmed the models' effectiveness in classifying the six main commands, with minimal occurrences of significant errors.

It is also important to note that, regardless of the model's precision, a good project practice will be to add a filter as a redundancy measure between the inference result and the input to the machine's control system.

Although the models achieved promising results, there is still room for ongoing improvements, especially in reducing incorrect inferences and optimizing the overall system performance. However, the obtained results represent a significant advancement in the field of human-robot interactions and the practical application of autonomous and intelligent UAVs.

Ultimately, this work contributes not only to the advancement of drone technology controlled by visual commands but also to the growing integration of machines into society and the development of safe and reliable autonomous systems.

Acknowledgment

This study was financed in part by the coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

References

[1] K. Čapek, *RUR*, Standard Ebooks, 2020.

[2] T. Nascimento, "RegrasDesafioDrones_v2.9," 2024. [Online]. Available: https://cbr.robocup.org.br/wp-content/uploads/2024/05/RegrasDesafioDrones_v2.9-1.pdf. [Accessed: Aug. 12, 2024].

[3] "PX4 Autopilot User Guide," PX4, [Online]. Available: https://docs.px4.io/v1.14/en/. [Accessed: Dec. 20, 2023].

[4] S. Kapoor, A. Sharma, A. Verma, V. Dhull, and C. Goyal, "A Comparative Study on Deep Learning and Machine Learning Models for Human Action Recognition in Aerial Videos," *The International Arab Journal of Information Technology*, vol. 20, 2023. DOI: 10.34028/iajit/20/4/2.

[5] U. Azmat, S. S. Alotaibi, M. Abdelhaq, N. Alsufyani, M. Shorfuzzaman, A. Jalal, and J. Park, "Aerial Insights: Deep Learning-based Human Action Recognition in Drone Imagery," *IEEE Access*, 2023.

[6] V. Krátký, G. Silano, M. Vrba, C. Papaioannidis, I. Mademlis, R. Pěnička, and I. Pitas, "Gesture-Controlled Aerial Robot Formation for Human-Swarm Interaction in Safety Monitoring Applications," *arXiv preprint arXiv:2403.15333*, 2024.

[7] C. Papaioannidis, I. Mademlis, and I. Pitas, "Fast CNN-based Single-Person 2D Human Pose Estimation for Autonomous Systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, pp. 1-1, Sep. 2022. DOI: 10.1109/TCSVT.2022.3209160.

[8] Brasil, *Constituição da República Federativa do Brasil de 1988*, Disponível em: http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm. [Accessed: Jan. 20, 2024].

[9] Ultralytics, *Introducing Ultralytics YOLOv8*, 2023. Disponível em: https://www.ultralytics.com/pt/blog/introducing-ultralytics-yolov8. [Accessed: Jan. 20, 2024].

[10] G. Bradski, *The OpenCV Library*, Dr. Dobb's Journal of Software Tools, 2000.

[11] Roboflow, *Roboflow: End-to-End Computer Vision for Every Developer*, 2024. Disponível em: https://roboflow.com/. Acesso em: 03 jan. 2024.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[13] Papers with Code, *Leaky ReLU*, 2024. Available: https://paperswithcode.com/method/leaky-relu. Accessed: 2024-08-14.

[14] Ultralytics, *Ultralytics YOLOv8 Documentation*, 2023. Available: https://docs.ultralytics.com/. [Accessed: Jan. 15, 2024].

[15] R. Dias and F. Pereira de Figueiredo, *Comparação de Modelos YOLOv5 e YOLOv8 para Detecção de Imagens de Áreas Rurais*, 2023. Available: https://doi.org/10.13140/RG.2.2.30587.90400. [Accessed: Jan. 15, 2024].

[16] Roboflow, *What's New in YOLOv8?*, 2023. Available: https://blog.roboflow.com/whats-new-in-yolov8/. Accessed on: Aug. 15, 2024.