Design and Implementation of a Multi-Agent Image Preprocessing Framework for Deep Learning

Yago Salerno Dayub Campos[1,2], Thiago de Medeiros Rocha[1], Gilson Alexandre Ostwald Pedro da Costa[1,2], Guilherme Lucio Abelha Mota[1,2], Vera Maria Benjamim Werneck[1,2], Tassio Ferenzini Martins Sirqueira[1]

[1]Institute of Mathematics and Statistics - Rio de Janeiro State University (UERJ)
Rio de Janeiro – RJ – Brasil

[2]Graduate Program in Computational Sciences and Mathematical Modeling
Rio de Janeiro State University (UERJ) – Rio de Janeiro – RJ – Brasil

{yagosalerno,mrthiago09}@gmail.com

{gilson.costa,guimota,vera,tassio.sirqueira}@ime.uerj.br

Abstract. *Artificial Intelligence techniques, such as neural networks and vision transformers, have become the preferable approaches in computer vision. However, training these algorithms depends on high-quality data, which makes preprocessing, image crop, resize, normalization, training and test sets split among others, a fundamental stage. Preprocessing is often time-consuming and complex task. Thus, this work aims at contributing to optimize this process. The idea is substituting the manual work by a multi-agent system capable of performing various preprocessing algorithms on image datasets. The proposed method is based on a modular and expandable architecture, in which specialized agents execute actions in an isolated and organized manner. The implementation demonstrates the feasibility of the proposed architecture, providing a modular and extensible framework for image preprocessing. The proposed solution facilitates the integration of new preprocessing techniques while preserving a low-coupling architecture. The herein presented experiments have shown that the proposed architecture provides an efficient framework for integrating the desired preprocessing functionalities, facilitating the development of neural networks training pipelines.*

1. Introduction

Deep neural networks have established themselves as essential tools for solving complex problems across various fields. However, training these architectures depends both on the complexity of the algorithms, and on the quality and consistency of the data used. Data volume is a relevant factor despite not being determinant, so the quality of training data must ensure clear and representative information about the problem of interest.

In this scenario, the preprocessing stage acts as a bridge between raw data and useful information. Often, the original data is heterogeneous or noisy. Therefore, the preprocessing stage aims to transform this raw state into a clean, normalized, and standardized structure, optimizing feature extraction by the network layers. Thus, obtaining models with high generalization capacity and accuracy is intrinsically associated to the effectiveness of data training preprocessing.

However, managing these processing steps poses significant challenges. The diverse nature of computer vision problems requires taylored workflows, where the order and choice of techniques (such as resizing and normalization) directly impact computational cost and performance. The need for a system that facilitates these steps arises from the difficulty of orchestrating complex transformations in a scalable and modular way, preventing the data pipeline from becoming a rigid and hard-to-maintain process.

Inspired by recent approaches to dataset automation, such as the one presented by [Sun et al. 2025], which leverages collaboration among agents for the autonomous construction of datasets, this paper proposes developing a multi-agent system designed to manage and execute various image preprocessing tasks in a modular manner. The need for flexibility and autonomy justifies the choice of an agent-based architecture. By decomposing the pipeline into specialized agents, the system allows each component to perform specific transformations in a coordinated manner, facilitating the inclusion of new methods and ensuring that data preprocessing be adaptable to the demands of different neural network architectures. Furthermore, it disregards complex execution routines into a simplified workflow.

This work provides three main contributions to the field of multi-agent systems. First, it introduces a modular architecture comprising autonomous, specialized agents, each one responsible for a specific image preprocessing activity. Second, the architecture promotes loose coupling and extensibility, enabling new preprocessing capabilities to be incorporated through additional agents without modifying existing components. Third, the system employs an agent coordination mechanism through the Interface Agent, which manages task delegation, execution flow, and communication among preprocessing agents, allowing the dynamic composition of preprocessing workflows according to user requirements. The user may select different combinations of preprocessing agents according to the desired workflow.

Section 2 details the main preprocessing techniques and the theoretical foundations that motivate their applications. Section 3 reviews related works, analyzing different methods applied to distinct Computer Vision problems. Section 4 describes the modeling and architecture of the proposed multi-agent system, based on the Prometheus methodology. Section 5 presents the implementation of specific agents and the results obtained with real image sets. Finally, Section 6 summarizes the conclusions and main contributions of this system.

2. Preprocessing

Data preprocessing is an important step in deep learning, consisting of a set of transformations applied to raw data to optimize training and favor neural network convergence. This step aims to convert unstructured data into consistent numerical representations, minimizing noise and highlighting features relevant to the model.

The first step in the process is to extract the information contained in image files (such as Joint Photographic Experts Group (JPEG) or Portable Network Graphics (PNG) and convert it into manipulable mathematical structures. Images are read and decoded into tensors (N-dimensional arrays), where each element represents the intensity of a pixel. For color images, this representation typically follows the format (H, W, C), corresponding to the height, width, and the three color channels (RGB). Converting to a matrix

format, such as NumPy (Python), is essential, because it enables high-performance matrix operations, which are crucial for the large-scale processing required by complex models.

Several convolutional neural network (CNN) architectures require fixed input images dimensions. Therefore, when that is the case, it is necessary to ensure that the entire dataset has uniform dimensions:

i. **Resize:** This involves altering the dimensions of the original image to match the network's input layer. While effective in maintaining the overall content of the image, resizing can introduce distortions if the original aspect ratio is not respected.
ii. **Crop:** This technique allows the extraction of a specific region of interest (ROI) from the image. The use of the crop function is justified by the need to remove irrelevant or noisy information, such as irrelevant information located at the edges, focusing on the central attributes that define the object's class. According to [Soekarta and Ku-Mahamud 2025], strategic crop helps the neural network extract the most relevant features, mitigating the impact of complex backgrounds that could confuse the network.

On the other hand, normalization is the process of projecting pixel intensity values originally integers in the range $[0, 255]$ to a continuous scale, specifically $[0, 1]$ or $[-1, 1]$. This transformation is vital for the numerical stability of the training. Input values on very wide scales can cause abrupt oscillations, hindering convergence. As highlighted by [Ioffe and Szegedy 2015], normalization reduces variance between samples, resulting in faster and more stable training.

To ensure the statistical validity of the results, the preprocessed dataset is divided into distinct groups. The training set is used to adjust the network's internal parameters (weights), while the test set serves to evaluate the model's ability to generalize with unseen data. This separation is the main way to avoid overfitting, ensuring that the network is not merely memorizing data, but learning patterns.

The choice of preprocessing techniques should be guided by the nature of the problem. [Tachibana et al. 2019] demonstrates that customized data processing strategies can significantly reduce ambiguity in diagnostic imaging. The multi-agent system proposed in this work addresses this need through a modular architecture, where each agent (such as the Grayscale Agent or the Normalization Agent) performs a specialized task. This approach allows experimentation with different preprocessing combinations, enabling the removal or addition of steps based on the observed performance in the model's final accuracy.

3. Related Works

This chapter is dedicated to analyzing relevant works that explore the role of preprocessing in image analysis and in the training of deep learning models, with the aim of highlighting how the proper handling of input data can maximize the performance and accuracy of these models. The scope of this section also includes agent based preprocessing initiatives.

In [Krizhevsky et al. 2012], preprocessing is established as a fundamental step to enable the training of deep neural networks. The authors demonstrate that dimensional standardization and the subtraction of the mean intensity of pixels are essential for model

convergence. Furthermore, the study validates the use of data augmentation as a necessary preprocessing strategy to increase network robustness and mitigate overfitting in large datasets.

[DeVries and Taylor 2017] proposes a preprocessing technique called Cutout, which removes regions from the input image during training. This approach acts as an efficient form of regularization, forcing the model to extract features from multiple regions of the image instead of focusing on isolated details. Applying this technique in the data preparation stage increases the network's robustness and improves its generalization ability.

According to [Shorten and Khoshgoftaar 2019], these techniques allow the artificial expansion of the dataset and the teaching of spatial invariance to the network. By transforming the raw data through normalization and data augmentation, preprocessing ensures that the network learns robust patterns instead of memorizing noise from the original dataset, which is unnecessary in scenarios of high computational complexity.

The work of [Tachibana et al. 2019] investigated the usefulness of various preprocessing strategies specifically to mitigate ambiguity in clinical diagnoses based on deep learning. The authors found that applying different image processing methods allows for the independent training of models, facilitating the identification of the strategy that best reduces uncertainties in the analysis of medical images. The study highlights that the careful selection of preprocessing techniques is crucial to overcome interpretation problems in neural networks, adapting the data input to the specific demands of the clinical domain.

[Soekarta and Ku-Mahamud 2025] investigated the impact of various preprocessing steps on diverse sets of images, highlighting how the choice of methods influences the effectiveness of recognition and classification algorithms. Among the proven techniques, the following stand out: normalization, applied to attenuate specific variations in intensity and brightness; the use of filters, such as the Gaussian filter, to reduce noise and improve visual quality; the crop technique, essential for eliminating irrelevant areas and focusing the analysis on relevant appearance features; and data augmentation, used to increase the robustness of the training set, consequently optimizing the accuracy and generalization capacity of the models.

[Kumari et al. 2023] provides a critical analysis of image enhancement techniques, which is essential for optimizing the input data of neural network architectures. By systematically exploring spatial and frequency domain methods, the authors demonstrate that the effectiveness of deep learning models is heavily dependent on the quality of the preprocessing pipeline. Their evaluation of the synergy between different techniques highlights that a well-structured workflow is vital to mitigate noise and enhance relevant features, ultimately ensuring more robust feature extraction and improving the overall performance of neural networks in computer vision tasks.

Recently, [Sun et al. 2025] proposed DatasetAgent, a multi-agent system designed for the independent construction of datasets. The work utilizes an architecture composed of four specialized agents (analysis, processing, labeling, and supervision) that collaborate to transform raw images into structured datasets. The results demonstrate that optimizing the data flow in independent agents reduces human intervention and increases

data consistency. This approach validates the premise of this work that modularizing preprocessing through agents is an efficient strategy for handling data complexity in deep learning (DL).

Table 1. Comparison of related works and the proposed architecture

Work	MAS	Preproc.	Architecture Proposal	DL
[Krizhevsky et al. 2012]	No	Yes	No	Yes
[DeVries and Taylor 2017]	No	Yes	No	Yes
[Shorten and Khoshgoftaar 2019]	No	Yes	No	Yes
[Tachibana et al. 2019]	No	Yes	No	Yes
[Soekarta and Ku-Mahamud 2025]	No	Yes	No	Yes
[Kumari et al. 2023]	No	Yes	No	Yes
[Sun et al. 2025]	Yes	Limited	Yes	Partial
Proposed Architecture	**Yes**	**Yes**	**Yes**	**Yes**

Table 1 summarizes the main characteristics of the reviewed studies. Most existing works focus on evaluating preprocessing techniques and their impact on deep learning performance. While these studies demonstrate the importance of preprocessing, they do not provide an extensible architecture for orchestrating preprocessing tasks.

Among the analyzed works, DatasetAgent [Sun et al. 2025] is the closest to the present proposal, as it adopts a multi-agent approach. However, its primary objective is the autonomous construction and labeling of datasets. In contrast, the architecture proposed in this work focuses specifically on image preprocessing workflows and employs the Prometheus methodology to define agent responsibilities, interactions, and system extensibility. This distinction highlights the contribution of the proposed approach as a modular framework for organizing and executing preprocessing tasks through specialized agents.

4. Method

To model the system, the Prometheus methodology [Padgham and Winikoff 2004] was adopted. The main advantage of the Prometheus methodology is its practical approach to organizing ideas and quickly and concisely creating a prototype. The Prometheus structure is based on three macro-phases: System Specification, Architectural Design and Detailed Design.

4.1. System Specification

The System Specification phase serves as the initial step in the Prometheus methodology, defining what the multi-agent system must accomplish and establishing its operational context through four core viewpoints: System Goals, which decompose objectives into structured sub-goals; Use Case Scenarios, detailing interactions between agents, humans, and external entities; Functionalities, grouping goals and tasks into core capabilities; and Actions and Percepts, specifying environmental inputs and system outputs.

In the proposed system, the goal is to preprocess a set of images provided by the user. The system should be able to normalize the data, perform crop, resize the images, divide the set into training and test sets, and apply the Gaussian filter, according to the diagram in the Figure 1.

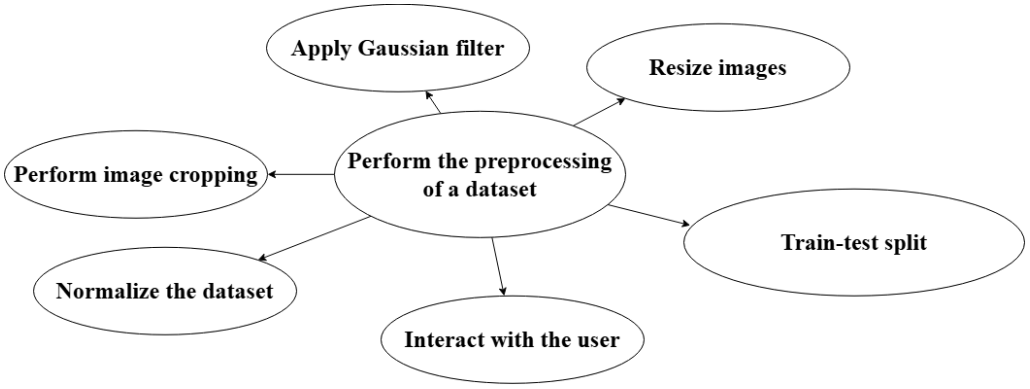

Figure 1. Goals Diagram

4.2. Architectural Design and Detailed Design

The architectural design phase aims to assign roles and goals to each agent, as well as how communication will be carried out, specifying communication protocols, the perceptions each agent will respond to, and the actions that will be taken. The output of this phase is a diagram that identifies which agents exist, the types of communication for each agent type, how they relate to each other, and how the shared information is stored in the system, as shown in Figure 2.

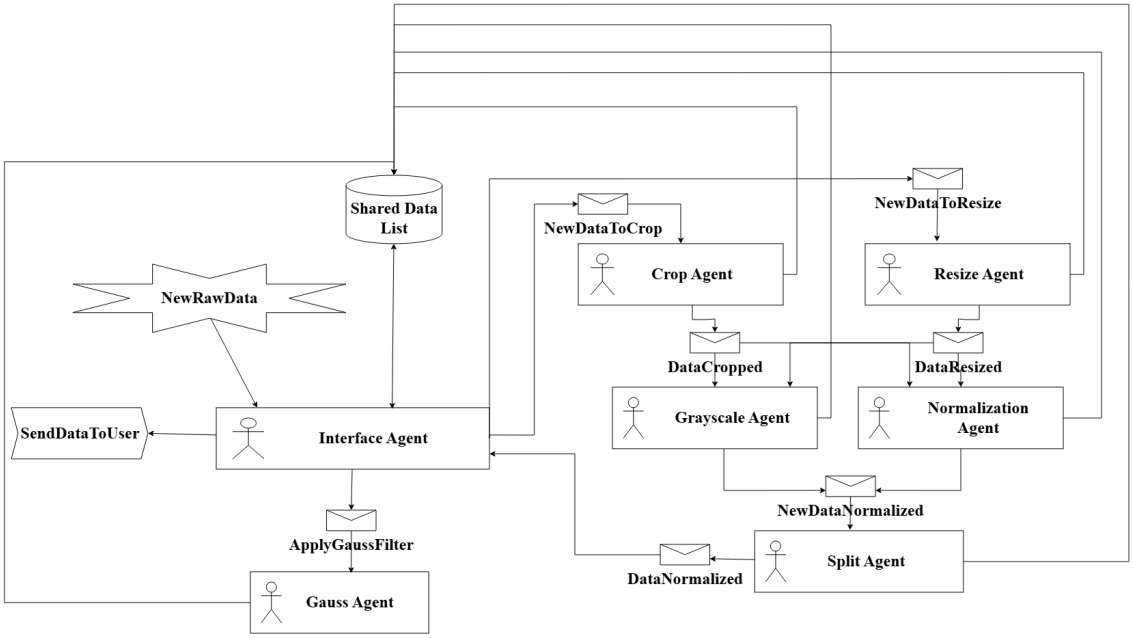

Figure 2. Architectural Diagram

To define the agents, we used a data coupling diagram to understand how data will be accessed between agents, as illustrated in Figure 3. Therefore, it was decided that a single shared memory would be used between the agents.

The content of the objectives diagrams (Figure 1), together with the coupling diagram (Figure 3), along with the use cases and system functionalities, defines seven agents:

i. Interface Agent

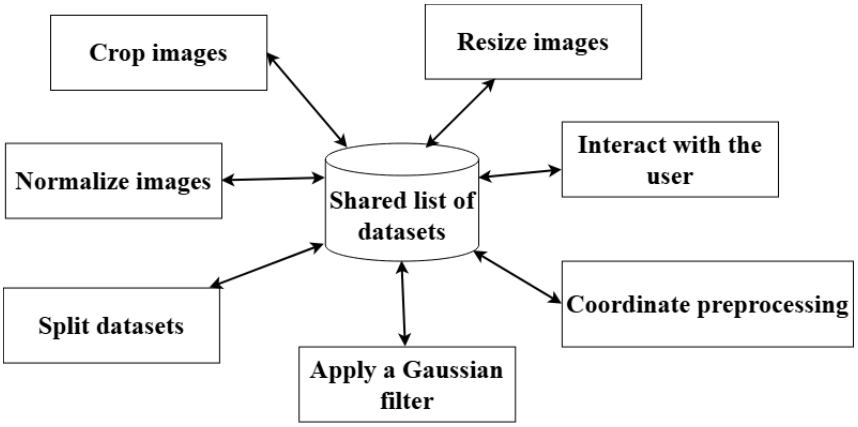

Figure 3. Coupling Diagram

ii. Normalization Agent
iii. Crop Agent
iv. Resize Agent
v. Grayscale Agent
vi. Gauss Agent
vii. Split Agent

In the proposed architecture, each preprocessing operation is encapsulated within a specialized agent. Agents are responsible for managing their own execution logic and can be activated independently according to the preprocessing workflow defined by the user. This design follows the fundamental principles of multi-agent systems, including autonomy, specialization, modularity, and cooperation among agents to achieve a common goal. Rather than implementing preprocessing as a monolithic pipeline, the proposed solution distributes responsibilities among specialized agents coordinated by a manager agent. To ensure clarity and objectivity in representing these interactions, a communication diagram was used, as shown in the Figure 4.

The Interface Agent acts as a coordination and orchestration agent within the multi-agent system. Its responsibilities include interacting with the user, interpreting preprocessing requirements, activating the appropriate specialized agents, and managing the execution flow. The remaining agents operate independently, executing their respective preprocessing tasks and returning the results to the Interface Agent, which is responsible for composing the final preprocessing pipeline. It will trigger the alarm so that the crop agent cuts the image to the defined size, using a specific section of the image, or the resizing agent, which compresses all the pixels of the image to adjust it to a new size. Regardless of which agent was triggered, if there is a need to convert the color image to grayscale, the dataset will be directed to the Grayscale Agent. Subsequently, the normalization agent will be activated, projecting pixel values from the RGB range [0,255] to a normalized interval, typically [0,1]. Finally, the Split Agent will separate the test and training sets.

With the agents and interactions defined, it becomes possible to define the internal details of how the agents will achieve their objectives and what tasks they will have within the overall context of the system. Thus, the last stage of the framework, the detailed design, is complete. In this context, Table 2 details how the agents work.

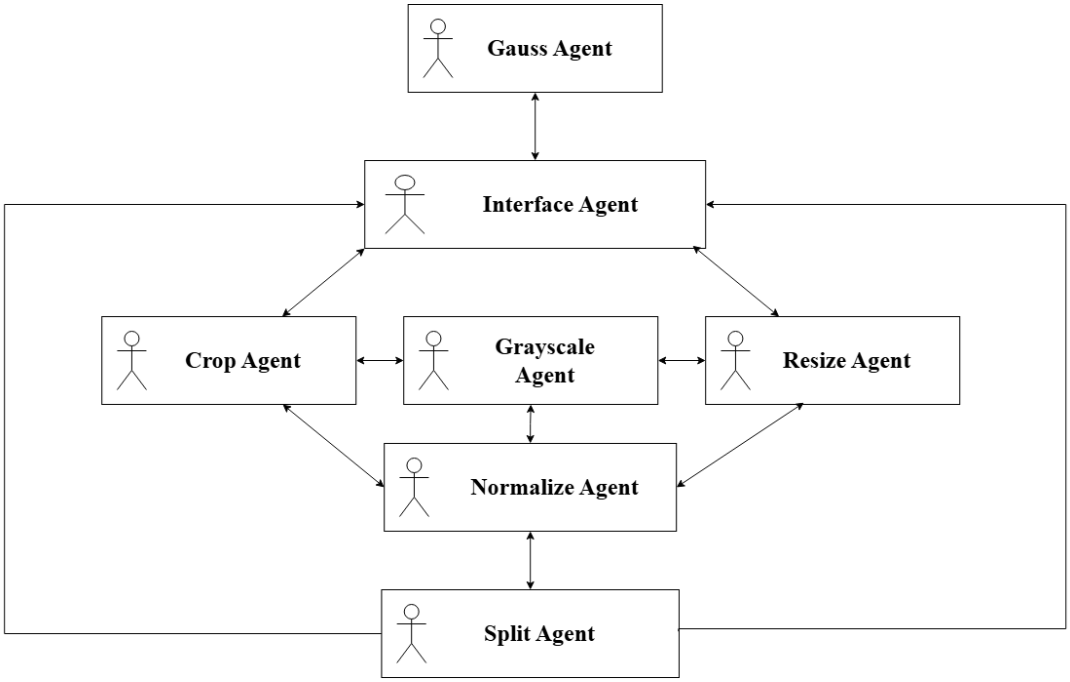

Figure 4. Communication Diagram

5. Implementation and Case Study

The code for the multi-agent system model was written in Python, with the aid of the NumPy, TensorFlow, PIL, and OpenCV libraries to perform data preprocessing methods, while the Matplotlib library was used to visualize the data as images. The dataset used to evaluate the method was the DIV2K [Agustsson and Timofte 2017]. The complete flowchart of the multi-agent system code presented in this work can be viewed at `https://github.com/YagoS2022/Multi_Agent_Image`

The multi-agent system was implemented in Python, where each preprocessing operation is encapsulated within a specialized agent implemented as an independent class. This design follows the architecture defined in Section 4, ensuring separation of responsibilities and facilitating the integration of new preprocessing capabilities through additional agents.

The agents responsible for preprocessing tasks do not communicate directly with one another. Instead, all interactions are coordinated by the Interface Agent, which acts as the central orchestrator of the workflow. This coordination mechanism allows preprocessing tasks to remain independent while enabling the dynamic composition of processing pipelines according to user requirements.

This approach reduces coupling among agents and improves maintainability, since modifications to a specific preprocessing technique can be performed without affecting the remaining agents in the system.

To use the model presented in this work, the user needs to create the Interface Agent class, indicate the folder where the images to be preprocessed are located, and finally, specify which pre-processing methods to execute. In this way, the Interface Agent will initiate the process of communicating with the other selected Agents and will return

Agent	Responsibilities
Interface Agent	Receive user input and send responses; Initiate preprocessing; Coordinate agents.
Normalization Agent	Normalize the dataset.
Crop Agent	Crop the dataset.
Resize Agent	Resize the dataset.
Grayscale Agent	Convert the dataset to grayscale.
Gauss Agent	Apply a Gaussian filter.
Split Agent	Split the dataset.

Table 2. Agent Roles in the Prometheus Methodology

the dataset with the preprocessing steps executed based on the selected Agents.

Figure 5 (a) illustrates an image before the application of the multi-agent system, while Figure 5 (b) shows it after its use, where the user chose to use the Crop Agent.

Figure 6 (a) illustrates the use of the Gauss Agent in the same image shown in Figure 5 (b), which smoothed the image by reducing its high-frequency details, resulting in the observed blurring effect. Figure 6 (b) shows the Grayscale Agent after the Crop Agent extracts another crop region of the image in Figure 5 (a), which applies the change of the three color channels (RGB) to a single channel, transforming a color image into grayscale.

To validate the efficiency of the system, experimental evaluations were conducted using the DIV2K dataset. The results showed that the parallelized execution of the Crop and Resize agents significantly reduced processing time, taking an average of 7 seconds compared to 42 seconds in the sequential implementation. Furthermore, comparative tests between the multi-agent framework and traditional preprocessing pipelines (without agents) revealed identical execution times, proving that the abstraction layer incurs negligible computational overhead. Finally, this approach eliminates the need for users to master complex underlying libraries or manage workflow execution order, as the system orchestrates the entire process in a single line of code based on the selected methods.

The results demonstrate that the proposed architecture successfully coordinates multiple specialized agents to execute distinct preprocessing operations in a modular workflow. The generated outputs confirm that each agent performs its assigned task independently while preserving the overall consistency of the preprocessing pipeline.

These applications demonstrate the applicability and flexibility of the proposed architecture, showing that independent, specialized agents can perform different preprocessing tasks. An inherent limitation of the current implementation is that the system is restricted to the specific preprocessing tasks for which its agents were originally programmed. However, this constraint is easily addressable due to the modular nature of the architecture. The system is designed to allow the integration of new preprocessing stages by simply instantiating additional specialized agents, ensuring the framework's continuous adaptability to evolving requirements.

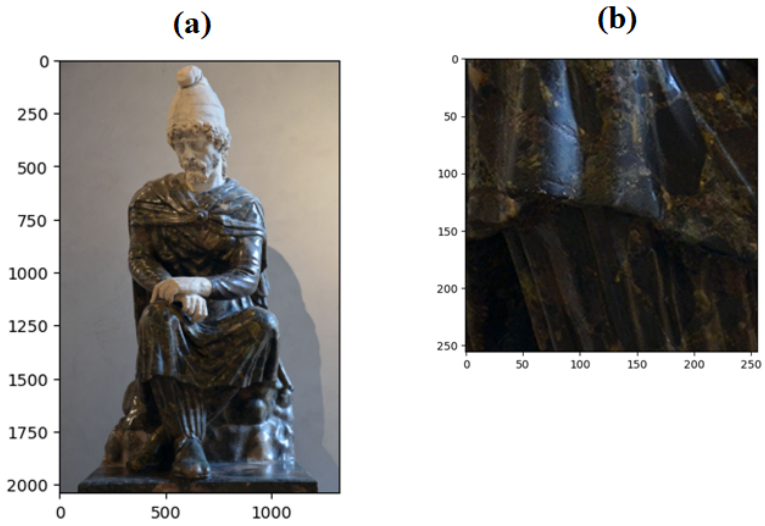

Figure 5. (a) Image from the raw dataset. (B) Image after the Crop agent.

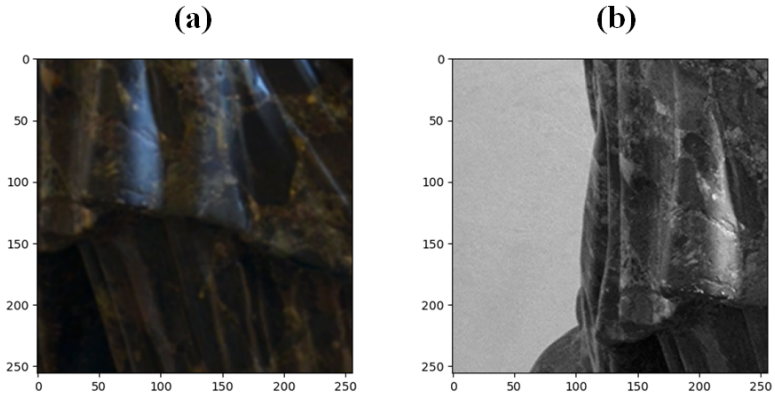

Figure 6. (a) Image after preprocessing with Crop Agent and Gauss Agent. (b) Another image after preprocessing with Crop Agent and Grayscale Agent.

5.1. Threats to Validity

The proposed architecture was evaluated through a proof-of-concept implementation and illustrative preprocessing examples. Therefore, some limitations should be considered when interpreting the results. First, the current study focuses on demonstrating the feasibility and modularity of the proposed multi-agent architecture rather than providing a large-scale quantitative evaluation. Consequently, no performance measurements regarding scalability, or resource consumption were conducted.

Second, the implemented agents represent only a subset of image preprocessing techniques commonly used in deep learning workflows. Although the architecture was designed to support extensibility through the addition of new specialized agents, the current implementation is limited to normalization, cropping, resizing, grayscale conversion, Gaussian filtering, and dataset splitting.

Finally, the case study was restricted to demonstrating the operation of the imple-

mented agents using representative image examples. Future work should include experiments with benchmark datasets, comparisons against traditional preprocessing pipelines, and evaluations of scalability and parallel execution capabilities in more complex scenarios.

6. Conclusion

The preprocessing step of a dataset for neural networks is fundamental to obtaining consistent and robust results. Implementing a multi-agent system capable of performing this stage helps the user execute preprocessing more efficiently, and adapt preprocessing to other application specific concerns.

Adopting an architecture where each agent performs a specific action promotes a high degree of system modularization. This characteristic enables low coupling, ensuring greater flexibility: system expansion is simplified, allowing the inclusion of new agents for different preprocessing stages without interfering with the existing structure. Furthermore, this division into smaller tasks facilitates work maintenance, since any modification or improvement can be made in isolation on just one agent. This avoids the need to alter or test the entire system again, ensuring that the program is more flexible and easier to expand as new preprocessing needs arise.

In general, a multi-agent system for image preprocessing allows greater adaptability to different neural network architectures and training workflows, as well as allowing the composition of customized preprocessing pipelines through specialized agents. The architecture also enables future parallel execution of independent preprocessing tasks. The presented architecture enables efficient, simplified system maintenance.

The proposed multi-agent system allows for several future extensions, leaving room for future work, such as the use of an LLM agent capable of chating with the user and better understanding the types of preprocessing needed for the user's problem. This new agent would then be able to understand the problem and identify which preprocessing agents would be used. Future works may include the integration of more preprocessing techniques, such as data augmentation pipelines, which are essential for robust deep learning models. Furthermore, the system could evolve into an open-source framework, enabling the continuous growth of its library of specialized agents. This collaborative model would allow the community to implement, test, and contribute new preprocessing modules, ensuring the architecture remains aligned with the state-of-the-art requirements of the computer vision field.

From a Multi-Agent Systems perspective, the proposed architecture demonstrates how image preprocessing workflows can be decomposed into specialized and cooperative agents. The separation of responsibilities among specialized agents and the coordination performed by the Interface Agent provide a flexible and extensible environment that simplifies the integration of new preprocessing capabilities while preserving system maintainability.

This work demonstrated the feasibility of employing a multi-agent architecture to organize image preprocessing workflows. The proposed approach provides modularity, extensibility, and clear separation of responsibilities, making it suitable for developing flexible computer vision preprocessing pipelines.

Acknowledgments

The authors acknowledge the financial support provided by Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), Process E-26/202.300/2025 (313381) and Process E-26/202.443/2026 (329002). The authors also thank the Universidade do Estado do Rio de Janeiro (UERJ) for the institutional support and research infrastructure that contributed to this study.

Declaration on the Use of AI Tools

ChatGPT (OpenAI), DeepL, and Grammarly were used exclusively to assist with language revision, translation, and grammar correction. The authors are fully responsible for the scientific content, analyses, interpretations, and conclusions presented in this manuscript.

References

[Agustsson and Timofte 2017] Agustsson, E. and Timofte, R. (2017). Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[DeVries and Taylor 2017] DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

[Ioffe and Szegedy 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR.

[Krizhevsky et al. 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25.

[Kumari et al. 2023] Kumari, N., Sharma, P., and Kansal, I. (2023). An analytical review on image enhancement techniques. In *2023 2nd International Conference on Disruptive Technologies in Multi-disciplinary Research (DELCON)*, pages 1–6. IEEE.

[Padgham and Winikoff 2004] Padgham, L. and Winikoff, M. (2004). *Developing Intelligent Agent Systems: A Practical Guide*. Wiley Series in Agent Technology. John Wiley & Sons, Chichester, UK.

[Shorten and Khoshgoftaar 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48.

[Soekarta and Ku-Mahamud 2025] Soekarta, R. and Ku-Mahamud, K. R. (2025). Recent facial image preprocessing techniques: A review. *Engineering Proceedings*, 84(1):39.

[Sun et al. 2025] Sun, H., Bian, H., Zeng, S., Rao, Y., Xu, X., Mei, L., and Gou, J. (2025). Datasetagent: A novel multi-agent system for auto-constructing datasets from real-world images. *arXiv preprint arXiv:2507.08648*.

[Tachibana et al. 2019] Tachibana, Y., Obata, T., Kershaw, J., Sakaki, H., Urushihata, T., Omatsu, T., Kishimoto, R., and Higashi, T. (2019). The utility of applying various image preprocessing strategies to reduce the ambiguity in deep learning-based clinical image diagnosis. *Magnetic Resonance in Medical Sciences*, 19:92–98.