**Article title**

*AgCV : An Agentic Framework for Automating Computer Vision Application*

**Authors**

*Arav Saxena1, Dr. Archana Y. Chaudhari1, Anilkumar Gupta2*

**Affiliations**

*1 Symbiosis Institute of Technology, Pune Campus, Symbiosis International (Deemed University) Pune*

*2 Centre for Development of Advanced Computing, Pune, India*

**Corresponding author’s email address and Twitter handle**

*archana.chaudhari@sitpune.edu.in*

**Keywords**

*Retrieval-Augmented Generation; LangChain; LangGraph; Groq Inferencing Engine; pipeline automation; NLP; LLM; computer vision; vision block system(AgenticAI)*

**Related research article**

*None*

**Changes Made:**

**A screenshot of a computer

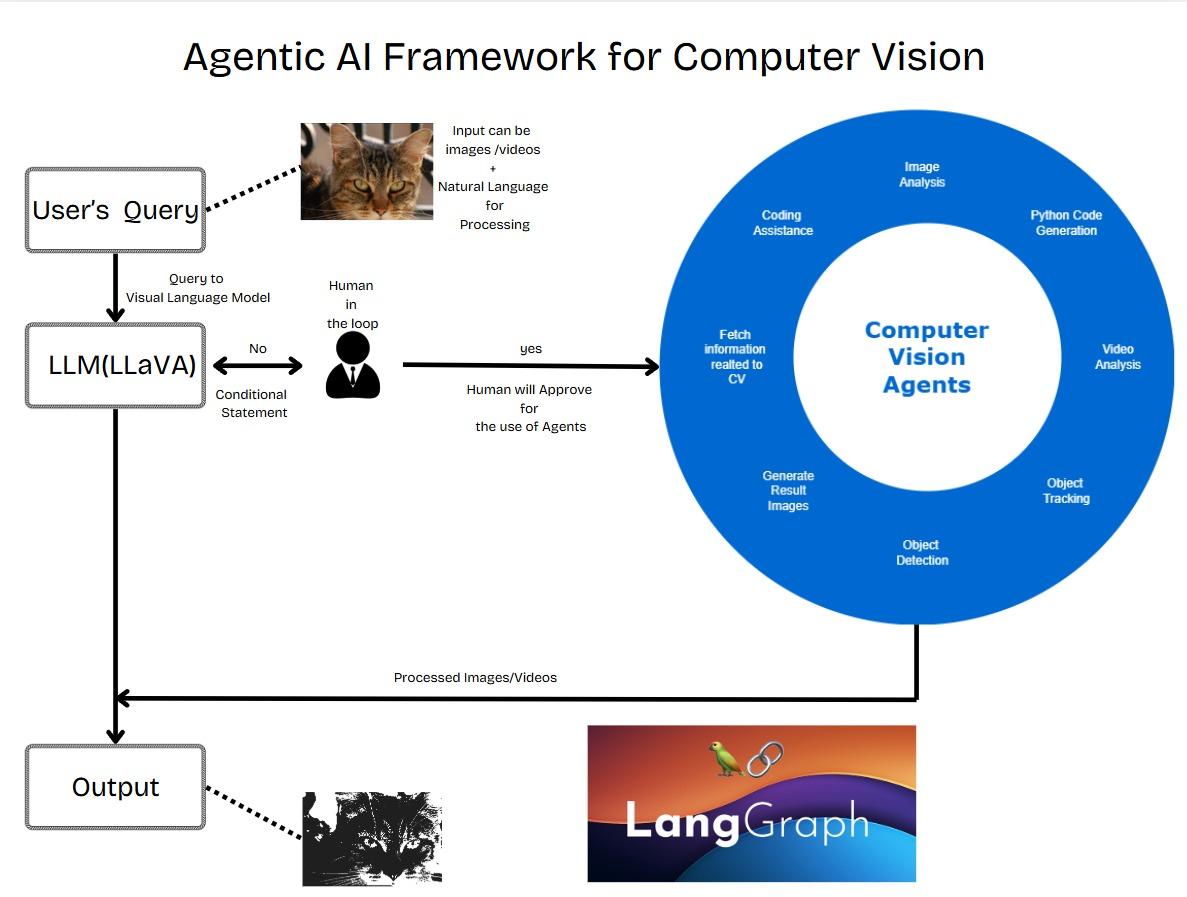
AI-generated content may be incorrect.**

**Abstract**

*This paper proposed the Agentic Computer Vision (AgCV) framework that enables autonomous interaction with computers through a Graphical User Interface (GUI), aimed at transforming human-computer interaction by automating complex, multi-step tasks. AgCV System uses LangGraph and integration of natural language processing, Deep Learning, and data science. AgCV creates flexible, user-centered pipelines for Computer Vision (CV) applications. In the proposed AgCV each Agent works on a particular task from object identification to object classification and image segmentation. It demonstrates how Retrieval-Augmented Generation and LangGraph enable fully automated pipelining through user interactions. The proposed Framework strategy reduces the need for technical expertise, allowing end-users to generate and configure CV operations using simple natural language processing and large language models. Comprehensive analysis highlights the effectiveness of individual components. AgCV promotes accessibility, scalability, and flexibility of CV applications in different domains.*

* *The proposed system allows users to create and configure CV operations using simple natural language, making it accessible even to those with limited technical expertise.*
* *The AgCV framework supports a wide range of CV tasks and can be easily adapted to different user needs and applications.*

**Graphical abstract**

**

**Specifications table**

*table provides general information on your method.*

|  |  |
| --- | --- |
| **Subject area** | *Engineering* |
| **More specific subject area** | *Machine Vision* |
| **Name of your method** | *AgCV : Agentic Computer Vision Framework* |
| **Name and reference of original method** | *A. Bendale, K. Chiu, K. Marwah and R. Raskar, "VisionBlocks: A Social Computer Vision Framework," 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, Boston, MA, USA, 2011, pp. 521-526.* |
| **Resource availability** | *https://www.kaggle.com/datasets/mrsimple07/injury-prediction-dataset* |

**Background**

*Computer vision[1] has grown rapidly and propelled improvements in various sectors such as surveillance, autonomous vehicles, retail, and healthcare. All these applications require Computer Vision (CV) systems to be adaptive and efficient enough to perform a wide range of tasks such as object identification, image classification, and segmentation. The technologies like Vision Blocks [2], Large Language Models (LLMs) [3], Natural Language Processing (NLP) [4], and LangChain [5] are newly popular in helping to create and implement CV kinds of systems. Basic tasks in CV include object detection, image segmentation, and classification. Traditionally, for each application, the process of selecting and fine-tuning the right models, preprocessing activities, and post-processing activities has been very labor-intensive and experience oriented. The Vision Block System would speed up this process by providing a modular approach to building CV pipelines. Vision blocks are block-based deep learning model components used for feature extraction, object detection, and segmentation in computer vision applications. Every "block" in the system represents a unique CV function, including a specific task, pretrained models, and relevant configurations. With such building blocks, different combinations of these can be mounted to come up with a unique process flexible enough for any range of CV applications. Modularizing such processes lessens the requirements for specialized knowledge and allows for fast deployment as complicated CV pipelines can be assembled easily. EfficientNet [6], ResNet [7], and Vision Transformers [8] popular models utilize vision blocks to enhance scalability and accuracy. Inception modules [9] and YOLO [10] models are optimized for real-time object detection using multi-scale feature learning. Utilization of vision blocks in autonomous cars, medical imaging, and industrial automation has enhanced model performance and interpretability.*

*The LangChain [11] framework, designed to facilitate interaction with LLMs and enable users to construct applications that mix the specialized activities like retrieval and production with heavy language processing responsibility. In paper [11], the developers utilized various components, including memory, retrieval, and document processing. LangChain makes it simple to integrate various sources of data, simplifies prompt engineering, and provides chaining mechanisms in a structured manner to develop complex workflows. LangChain coordinates workflows within the Vision Block System - dynamic blocks and mechanisms of triggering RAG are also managed; inputs from a user are further processed. The LangChain supports retrieval-augmented generation (RAG), thus proving particularly beneficial in applications requiring a lot of knowledge, including question answering and summarization. RAG, introduced by [12], RAG strengthens LLMs by first retrieving applicable documents or information from outside sources prior to response generation. This method successfully resolves the hallucination issue in LLMs by basing responses on verifiable facts. Research indicates that models utilizing RAG surpass conventional LLMs in knowledge-intensive applications, including open-domain question answering and summarization [13]. RAG is in widespread application in sectors such as healthcare, finance, and legal research, where accuracy in facts matters. RAG merges information retrieval techniques with generative AI models to improve response accuracy and contextual applicability.*

*LLMs models like OpenAI's GPT series, Google's BERT, and Meta's LLaMA utilize NLP techniques to transform ambiguous prompts into clear, actionable tasks by breaking down user inputs, processing language semantics, and extracting actionable details. The paper [14] proposed the transformer architecture, which is the foundation of contemporary LLMs based on the usage of self-attention mechanisms to achieve efficient understanding of language. LLM optimization has been the focus area recently with methods like parameter-efficient tuning [15] and human intent alignment [16]. Deployment of LLMs across diverse industries has brought about innovation in personalized AI assistants, auto-generated content, and sentiment analysis. Natural Language Processing (NLP) deals with allowing machines to understand, process, and generate human languages. Key developments in NLP are transformer-based models, transfer learning techniques, and pre-trained language models. The work conducted by Devlin [17] on BERT exposed bidirectional contextual embeddings, which dramatically enhanced performance on a range of NLP tasks, including named entity recognition (NER) and sentiment analysis. Contemporary NLP methods are extensively used in applications such as chatbots, machine translation, and speech recognition, showing the increasing influence of AI on human-computer interaction. The Table 1 depict the literature review of pretrained models used for CA application.*

*Table 1: Literature survey of pertained models useful for building a modular CV pipeline*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Model Name*** | ***Type*** | ***Description*** | ***Common Applications*** | ***Vision Block Use For*** |
| *CNN (Convolutional Neural Network) [18]* | *Feature Extractor* | *Extracts spatial features with convolutional filters* | *Image Classification, Object Detection* | *Core feature extraction for images* |
| *VGG [19]* | *Deep CNN* | *Deep CNN with uniform, small filters for detail features* | *Image Classification, Object Recognition* | *Transfer learning in classification* |
| *ResNet (Residual Network)[7]* | *Deep CNN with Residuals* | *Uses skip connections to enable deeper networks* | *Image Classification, Object Detection* | *Backbone for feature extraction* |
| *MobileNet[20]* | *Lightweight CNN* | *Optimized for mobile-enabled use with a smaller size* | *Mobile Image Classification, Embedded Vision* | *Low-power applications* |
| *EfficientNet [6]* | *Optimized CNN* | *Compound scaling for improved efficiency and accuracy* | *Image Classification, Object Detection* | *Efficient feature extraction* |
| *Inception [21]* | *Multi-path CNN* | *Multi-path convolution with different filter sizes* | *Image Classification, Object Detection* | *Multi-scale feature learning* |
| *YOLO (You Only Look Once)[22]* | *Object Detection CNN* | *Single-pass object detection for speed and efficiency* | *Real-time Object Detection* | *Real-time detection for vision applications* |
| *R-CNN (Region-based CNN) [22]* | *Region Proposal CNN* | *Generates region proposals for detection and classification* | *Object Detection* | *Detection and classification* |
| *Fast R-CNN [22]* | *Region Proposal CNN* | *Speeds up region proposals with selective search* | *Object Detection* | *Efficient object detection* |
| *Faster R-CNN [22]* | *Region Proposal CNN* | *Learns a region proposal network for speed* | *Object Detection* | *High-accuracy object detection* |
| *Mask R-CNN [22]* | *Instance Segmentation CNN* | *Adds instance segmentation to Faster R-CNN* | *Instance Segmentation, Object Detection* | *Higher instance segmentation* |
| *U-Net [23]* | *Semantic Segmentation Network* | *Encoder-decoder design for medical image segmentation* | *Medical Image Segmentation, Satellite Imagery* | *Segmentation in structured vision tasks* |
| *Transformers (Vision Transformers) [23]* | *Attention-based Network* | *Uses self-attention for large image dependencies* | *Image Classification, Image Generation* | *High-level feature representations* |
| *GAN (Generative Adversarial Network) [24]* | *Generative Network* | *Generator and discriminator for realistic image generation* | *Image Generation, Style Transfer* | *Synthetic data generation* |
| *Autoencoders [24]* | *Encoding-Decoding Network* | *Encodes and reconstructs for denoising or generation* | *Image Denoising, Compression, Generation* | *Encoding and feature compression* |

**Method details**

*The field of Computer Vision is growing rapidly currently so to automate the tasks related to the CV field. This research proposed an Agentic AI Pipeline for Automating Computer Vision Application (AgCV framework), which helps the user to do all CV tasks without getting much knowledge about Artificial Intelligence. This Automation will help the user to complete tasks even if they do not know about the context related to it, by using NLP. The proposed system will be allow user to perform actions without needing to write any complex codes using python libraries. This Automation helps in solving complex Workflows by automating the tasks involved in the process as shown in the figure1. The AgCV Framework uses pretrained Models like ResNet [2], VGG along with custom python tools for dynamic coding. Using these models and tools with the LLM provide the framework the strength to act on the tasks depending on the user’s query. For giving power to this framework this research uses the state of art module from Lnagchain like LangGraph. The task graph was based on which the LangGraph architecture was built, directed edges of graphs that indicated data dependence between the various tasks with nodes representing them as Vision Blocks. Because of this construct, the Graph can easily handle all dependencies automatically so that each Agent runs in the proper order relative to its respective input requirements. When possible, the LangGraph allows the execution of jobs in parallel by making the entire pipeline a Directed Acyclic Graph (DAG), maximizing efficiency and overall processing time.*

*LangGraphh helps in performing Agentic AI tasks and responsible for managing the tools properly. It helps in Orchestrating the Agents with the tools and modify the workflow according to the user’s Input. The LangGraph system is a key element in AgCV Framework, in charge of controlling and automating data flow and task execution in CV pipelines. The LangGraph gives a composite workflow whose chunks link many nodes as tools, these being modular pieces that justify a specific CV task, such as object identification, segmentation, and picture classification, all taking into user's mind for sophistication in utility goals.*

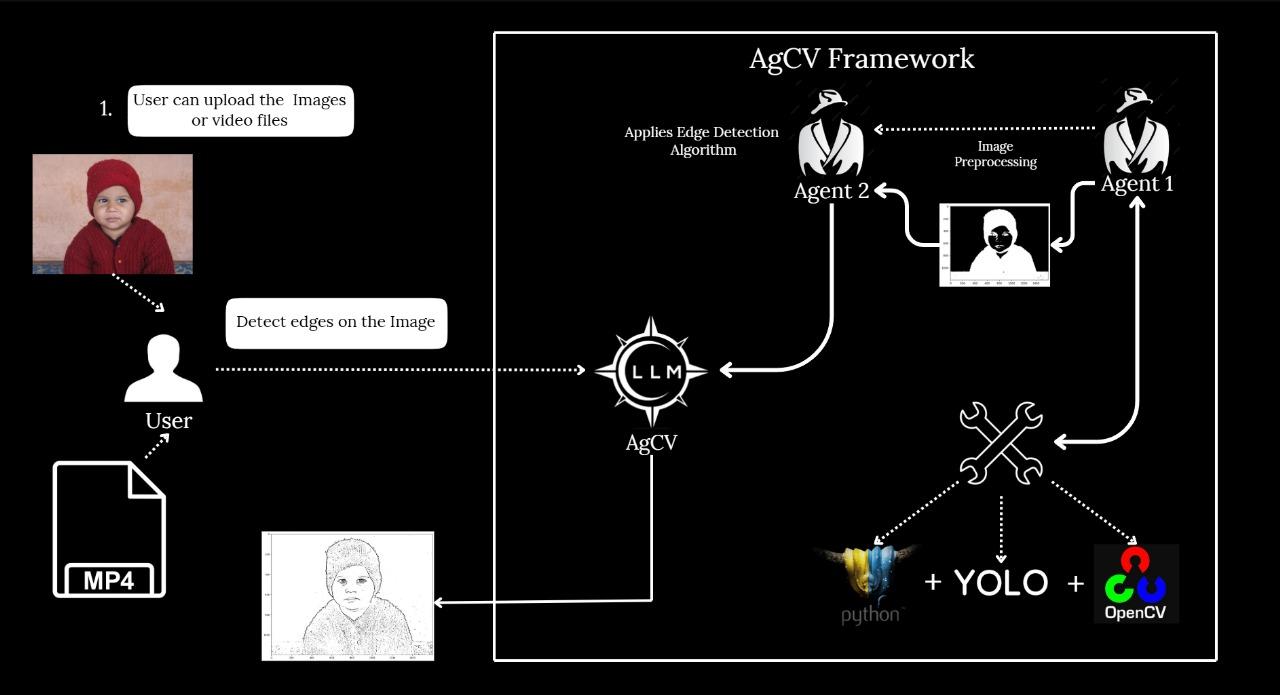
*In the AgCV framework, we designed multiple Agents, each specializing in a distinct CV task, to collaboratively execute complex, multi-agent workflows. These Agents, orchestrated by LangGraph, operate as modular Vision Blocks within a DAG, ensuring efficient task coordination. Below, we detail the primary Agents and their interactions, as outlined in our methodology:*

* ***Object Identification Agent****: This Agent employs models like YOLO to detect objects in images or video frames, generating bounding boxes or region proposals. It is optimized for real-time or high-accuracy detection, depending on the application.*
* ***Object Classification Agent****: Utilizing architectures like ResNet,, this Agent classifies detected objects into predefined categories. It leverages transfer learning to achieve robust classification performance across diverse datasets.*
* ***Image Segmentation Agent****: This Agent performs pixel-level segmentation using models like Mask R-CNN, enabling precise delineation of object boundaries. It is particularly effective for applications like medical imaging or satellite imagery analysis.*
* ***Feature Extraction Agent****: Responsible for isolating relevant image features, this Agent uses CNNs or traditional methods (e.g., SIFT, HOG) to extract low-level (edges, textures) and high-level (object shapes) features, supporting downstream tasks.*
* ***Data Preprocessing Agent****: This Agent handles image preprocessing tasks, such as resizing, thresholding, normalization, and augmentation, ensuring that input data is compatible with other Agents.*

***EXAMPLE WORKFLOW:***

***Interaction of AGCV Agents for Multi-Step Tasks****: LangGraph orchestrates these Agents by mapping them to nodes in a DAG, with edges representing data dependencies. For a task like "Segment and classify objects in an image," the workflow proceeds as follows:*

1. *The* ***Data Preprocessing Agent*** *normalizes and augments the input image.*
2. *The* ***Feature Extraction Agent*** *extracts relevant features, passing them to the* ***Image Segmentation Agent****.*
3. *The* ***Image Segmentation Agent*** *generates pixel-level masks, which are then used by the* ***Object Identification Agent*** *to refine bounding boxes.*
4. *The* ***Object Classification Agent*** *classifies the identified objects.*
5. *The* ***Model Evaluation Agent*** *assesses the pipeline’s performance, and results are visualized for user review.*

**

*Fig 1: Proposed AgCV framework for Automating Computer Vision Application*

*The Agentic Computer Vision (AgCV) framework leverages the synergistic integration of LangGraph, Natural Language Processing (NLP), and Deep Learning (DL) to create a robust, user-centric system that transforms human-computer interaction for computer vision (CV) applications. In our work, we designed LangGraph as the orchestrator of the CV pipeline, utilizing a Directed Acyclic Graph (DAG) structure where nodes represent modular Vision Blocks (e.g., object detection, segmentation) and edges denote data dependencies. This architecture ensures automated task sequencing, dependency management, and parallel execution, significantly enhancing pipeline efficiency and reducing processing time.*

*NLP, powered by Large Language Models (LLMs) such as those referenced in our literature survey (e.g., GPT series, BERT), enables the framework to interpret natural language inputs from users. By processing ambiguous or high-level prompts, NLP translates user intent into actionable CV tasks, eliminating the need for coding expertise or familiarity with CV algorithms. For instance, a user can input "Identify objects in this image," and the NLP component will parse the semantics to trigger the appropriate Vision Blocks.*

*Deep Learning models as AI Agent in DAG, includes fine-tuned pre-trained architectures like ResNet, VGG, and YOLO, form the computational backbone of the AgCV framework. These models execute complex CV tasks with high accuracy, leveraging their pre-trained capabilities for feature extraction, object detection, and segmentation. LangGraph seamlessly integrates these models into the pipeline, ensuring that they are invoked in the correct order based on user requirements.*

*This integration enhances user interactions by enabling a Graphical User Interface (GUI) where users can issue natural language commands to configure and execute CV tasks. The framework supports dynamic visualization of results using Python libraries ( Plotly, ), allowing users to review outputs and provide feedback in plain language. By abstracting technical complexities, the AgCV framework ensures that users, regardless of expertise, can efficiently interact with and customize CV pipelines*

*The AgCV Framework is performing the automation task in phases. AgCV includes feature extraction, data preprocessing, and model evaluation, on CV applications. This way, reliable systems of CV may be built with the ability of automatically analyzing, interpreting, and reacting to visual data for practical applications across broad categories of industries from retail and health diagnostics to self-driving cars and surveillance, among others. Main Elements of Proposed System is as follows:*

1. ***Data collection and annotation:*** *CV Data Science relies on the availability of quality, labeled datasets. Application domain requires huge databases of images or video frames that can be heterogeneous and related. The datasets in question may also be obtained by direct acquisition with cameras and sensors, artificially created, or produced in the public domain. Given that proper annotation is critical in as much as good learning from examples, because of labeled data-that can take a very heterogeneous form, like pixel-level labels for segmentation or bounding boxes for object detection-is quite feasible.*
2. ***Data Preprocessing and Augmentation* Feature Extraction:** *Feature extraction is the isolation and separation of useful features or patterns from image data. Among the traditional techniques, edge detection, HOG, and SIFT are usually used in the application of CV, though deep learning models automatically perform feature extraction. These models, just like CNNs, actually determine their features at different levels of abstraction ranging from low-level features like edges and textures up to higher abstraction levels concerning object shapes and structures.*
3. **Machine Learning and Deep Learning Models:** *The ML and DL models are the heart of CV Data Science. These are typically applied for image processing as CNNs can recognize spatial hierarchies in images. It encompasses architectures such as YOLO, ResNet, and EfficientNet. The parameters of each of these differ for tasks like object recognition or image segmentation or classification. Other more complex models that employ mechanisms such as self-attention, include Vision Transformers; they appear quite promising for high-demanding applications related to computer vision.*
4. **Train and Test Your Models:** *During the training phase, preprocessed data is fed into the model so that it learns the patterns and structures in the visual data. Accuracy, precision, recall, F1 score, IoU for segmentation, and mAP [25] for detection are the metrics used in assessing the performance of the model on test data. Evaluation findings guide the modification and improvement of the model so that the final system satisfies accuracy and efficiency standards.*
5. ***Deployment and Real-time Processing:****The trained CV models are used in real-world applications to process visual data in real-time. For example, self-driving cars identify objects in real-time and drive through safely. The data science techniques which are deployed along with the deployment of the model are model optimization and pruning, so that it maintains low latency and efficient inference on edge devices.*

*The proposed AgCV Framework provides the user the automated creation of results in a very efficient way. The proposed method stands out with the existing technologies like RapsAI (by google) where node graph editor is used for making pipeline and user have to manually connect all the pretrained blocks which requires pre knowledge so as to create a pipeline. Using our methods as an alternative will allow users to address problems related to Computer Vision without having any prior knowledge on the topic , user can ask the model to explain the steps involved with in the process of any task . For more understanding , user can ask the model for the information related to any topic related to Computer Vision as we have implemented a RAG(Retrieval Augmented Generation) pipeline from which the relevant data will be processed based on the users query as shown in figure 2.*

*A diagram of a person's hand

Description automatically generated*

*Fig 2. AgCV Framework Prompt*

*AgCV have fine tuned pre-trained State-of-Art models like VCG16 , ResNet , i3d(Inception). These models act as a tool in our framework to provide strength to the LLM for doing complex tasks, which it will not be able to do alone .This integration empowers our system with custom tool such object Detection, Image Analyzer ,Python REPL() etc to provide the accurate results to the user and also make them understand the task performed in the simple language using Natural Language Processing.*

*AgCV Framework involves dynamical visualization of the resultant output using python libraries (plotly/matplotlib) which will allow users to handle the result and give suggestion to the model if certain changes should be made to the output or it is satisfying.*

*AgCV involves Agentic RAG’s integration with LangGraph and NLP enabled fully automated pipelining, with vector database(ChromaDB) for efficient retrieval of knowledge stored in the database to keep the LLM updated with CV*

*The Agentic* ***Retrieval-Augmented Generation (RAG)*** *mechanism is a pivotal component of the AgCV framework, enhancing automation by combining information retrieval with generative capabilities. As outlined in our background section, RAG strengthens LLMs by retrieving relevant external information before generating responses, addressing the hallucination problem and ensuring fact-based outputs.*

***Contribution to Automation****:*

* ***Contextual Task Configuration****: RAG enables the framework to retrieve relevant CV knowledge (e.g., model specifications, task requirements) based on user queries. For instance, when a user requests “Segment objects in this image,” RAG retrieves information about segmentation models (e.g., Mask R-CNN, U-Net) and their configurations, allowing the LLM to select and configure the appropriate Vision Block automatically.*
* ***Enhanced User Interaction****: As shown in Figure 3, the RAG pipeline processes user queries to provide explanations or additional context. This allows users to understand the steps involved in a CV task without manual intervention, streamlining the automation process.*
* ***Error Reduction****: By grounding responses in verifiable data, RAG minimizes errors in task planning and execution. For example, when configuring a pipeline for object detection, RAG ensures that the selected model (e.g., YOLO) aligns with the task’s requirements, reducing planning errors (noted at 35.02% in Table 2).*
* ***Dynamic Workflow Adaptation****: RAG supports LangGraph in dynamically adjusting workflows by retrieving task-specific information. This ensures that the pipeline adapts to varying user inputs, such as switching from classification to segmentation, without requiring manual reconfiguration.*

*A diagram of a flowchart

AI-generated content may be incorrect.*

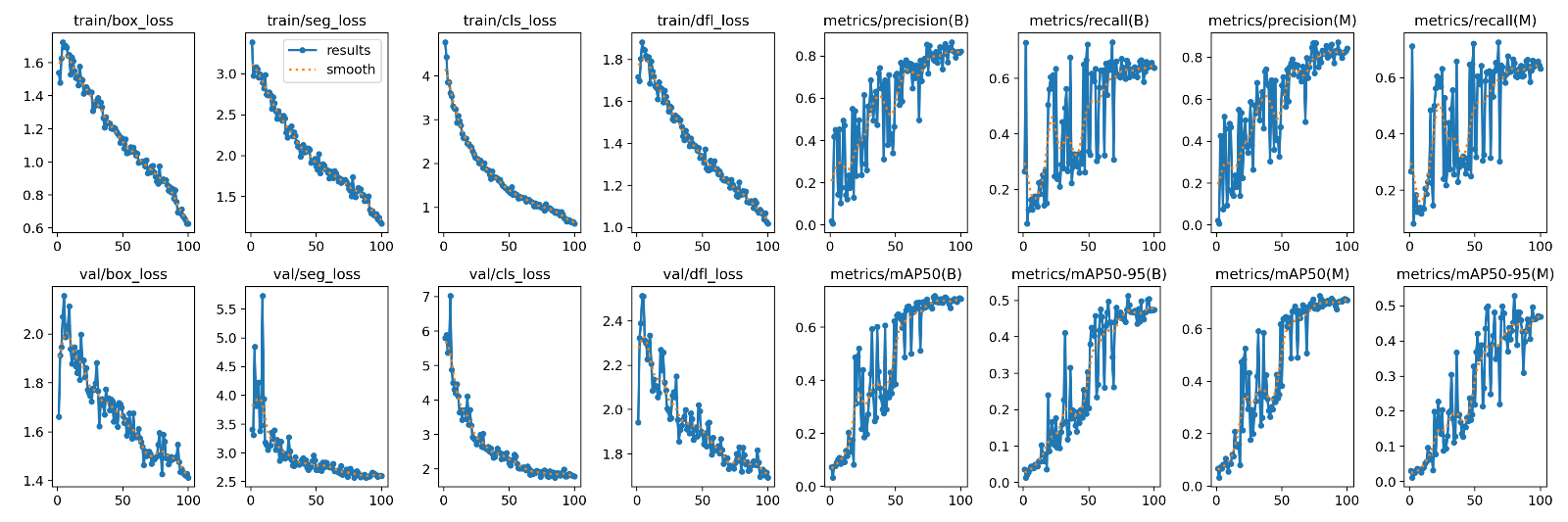
*Fig 3. AgCV Agentic RAG Workflow*

**Method validation**

*Our findings emphasize the AgCV framework’s accessibility for users with limited technical expertise, a core objective of our research. The evidence supporting this claim includes:*

* ***Natural Language Interface****: The integration of NLP allows users to interact with the framework using simple natural language prompts (e.g., “Detect objects in this image”). As described in the "Method Details" section, the LLM processes these prompts to configure CV pipelines, eliminating the need for coding or understanding CV algorithms.*
* ***Automated Pipeline Creation****: Unlike traditional CV systems requiring manual model selection and configuration, AgCV automates the entire pipeline using LangGraph and pre-trained models (e.g., ResNet, VGG). This abstraction reduces the technical barrier, as users do not need to fine-tune models or manage dependencies.*
* ***RAG Pipeline for Explanations****: The RAG mechanism enables users to query the system for explanations of CV processes or concepts, as shown in Figure 2. For example, a user can ask, “What is image segmentation?” and receive a tailored response, enhancing understanding without requiring prior knowledge.*
* ***Dynamic Visualization****: The framework provides visualized outputs using Plotly or Matplotlib, allowing users to review results and suggest modifications in plain language. This interactive feedback loop, detailed in the "Method Details" section, ensures that non-experts can engage with the system intuitively.*

*The proposed AgCV framework evaluated on Injury detection dataset [26] for testing the multimodal agents’ capability of executing a wide range of computer tasks in a prediction environment. The figure 3 depict the training and validation losses, as well as key performance metrics over 100 epochs of AgCV Framework on Injury Detection Dataset.* The AgCV framework effectively optimizes injury detection by reducing losses and improving precision, recall, and *Mean Average Precision (*mAP)[27].



*Fig. 4: training and validation loss of AgCV framework for Injury Detection Dataset*

***Loss Trends:***

*Train Losses (fig. 3. Top Row - Left): train/box\_loss, train/seg\_loss, train/cls\_loss, and train/dfl\_loss all show a steady decrease, indicating that the model is learning effectively.The smooth decline suggests that the model is converging well without severe overfitting.*

*Validation Losses (fig. 3. Bottom Row - Left): val/box\_loss, val/seg\_loss, val/cls\_loss, and val/dfl\_loss also decrease, but with more fluctuations compared to training losses. The presence of fluctuations suggests that some level of regularization might be needed to smoothen learning.*

***Performance Metrics:***

*Precision and Recall (fig. 3. Middle-Right): Precision and recall curves for both B (bounding box) and M (mask/segmentation) show a steady increase. The increasing trend indicates an improvement in model performance over epochs. However, fluctuations in recall suggest that while precision improves steadily, recall may require more optimization.*

*mAP (fig. 3. Rightmost Graphs): mAP@50(B), mAP@50-95(B), mAP@50(M), and mAP@50-95(M) all show continuous improvement. The model achieves better localization and segmentation accuracy over time.*

*The relatively smooth increase suggests that the model is effectively generalizing.*

*Figure 4 shows the four different performance evaluation curves for an injury detection model, assessing the classification of first-degree, second-degree, and all classes. Fig. 4(a) shows tThe F1 score peaks at a certain confidence threshold and then declines as confidence increases. First-degree injuries (light blue line) show a sharp peak and drop,*

|  |  |
| --- | --- |
|  |  |
|  | (b) |
|  |  |
| (c) | (d) |

*Fig. 5. Performance curve of AgCV framework on injury detection dataset: a)* F1-Confidence Curve b) Precision-Confidence Curve c) Recall-Confidence Curve d) Precision-Recall Curve

*indicating that their classification is sensitive to confidence levels. Second-degree injuries (orange line) exhibit a lower F1 score overall, suggesting more classification difficulty. All classes (dark blue line) have a moderate peak and gradual decline, showing a more stable performance across different injury types. Fig. 4(b) depict the precision confidence curve. Precision generally increases with confidence for all categories. The first-degree injuries (light blue) and all classes (dark blue) show better precision at higher confidence levels. Second-degree injuries (orange) have lower precision, indicating that these injuries may have more misclassifications. The dark blue line (all classes) reaches a high precision of ~1.00 at a confidence threshold of 0.965, meaning the model is very certain when it makes a prediction at this confidence level. Fig. 4(c) depicts recall decreases as confidence increases, which is expected since stricter confidence thresholds lead to fewer detections. All classes (dark blue) maintain a better recall than second-degree injuries but still show a decline at higher confidence thresholds. Fig. 4(d) shows all classes (dark blue line) have the highest precision across recall levels, meaning the model balances precision and recall well for overall detection.*

***Error Analysis***

*The error analysis on the tasks that AgCV framework failed to do. Table 2 shows there are three types of errors that observed: (1) Planning Error: A planning error occurs when the agent generates unsuitable plans for a task, including inaccuracies in the plan, misleading subtask information, or misalignment of subtask sequence with task requirements. (2) Grounding Error: A grounding error arises when the agent fails to accurately interact with target elements despite their visibility and the application of correct reasoning. This includes incorrect element selection or inaccurate coordinate selection due to the inherent limitations of our action. (3) Execution Error: An execution error emerges when the agent makes incorrect decisions or fails to adjust its behavior during task execution. This includes repetitive actions, diverging from subtask goals, delays in transitioning between subtasks or violating established protocols by combining multiple actions into one.*

*Table 2: Table Analysis*

|  |  |
| --- | --- |
| *Error Metric* | *Error Rate (%)* |
| *Planning Error* | *35.02%* |
| *Grounding Error* | *38.49%* |
| *Execution Error* | *40.26%* |

***Comparison to Existing Systems:***

*To validate the effectiveness of the AgCV framework, we conducted a comprehensive evaluation using the Injury Detection Dataset from Kaggle, focusing on the framework’s ability to execute multimodal CV tasks. Our validation, detailed in the "Method Validation" section, compared AgCV to existing human-computer interaction systems, such as Google’s RapsAI, which relies on a node graph editor requiring manual configuration of pretrained blocks.*

*Evaluation Metrics and Results:*

* *We trained the YOLO Model using AgCV framework over 100 epochs, monitoring training and validation losses (box, segmentation, classification, and distribution focal loss) and performance metrics (precision, recall, F1 score, mAP). As shown in Figure 3, training losses decreased steadily, indicating effective learning, while validation losses showed fluctuations, suggesting areas for regularization.*
* *Figure 4 illustrates performance curves (F1-confidence, precision-confidence, recall-confidence, and precision-recall) for injury detection, demonstrating high precision (~1.00 at a 0.965 confidence threshold) and balanced precision-recall for all classes, particularly for first-degree injuries.*
* *The framework achieved continuous improvement in mAP@50 and mAP@50-95 for bounding box and segmentation tasks, indicating robust generalization****.***

*Unlike RapsAI(Google), which demands prior knowledge to manually connect pretrained blocks, AgCV automates pipeline creation using natural language inputs. We highlighted that AgCV’s Retrieval-Augmented Generation (RAG) pipeline allows users to query the system for explanations of CV processes, enhancing accessibility. Our error analysis (Table 2) identified planning (35.02%), grounding (38.49%), and execution (40.26%) errors, providing insights into areas for improvement but also demonstrating competitive performance compared to manual systems requiring technical expertise.*

*This validation underscores AgCV’s superiority in automating complex CV tasks with minimal user intervention, making it more user-friendly and efficient than existing systems like RapsAI.*

**Limitations**

*AgCV face challenges in maintaining confidentiality and privacy, smooth data flow between highly modular blocks, and handling latency in real-time processing.*

*Confidentiality and Privacy: The framework faces challenges in maintaining data confidentiality and privacy, particularly when processing sensitive data (e.g., medical images). Ensuring compliance with data protection regulations and securing data flow between modular blocks is critical for applications with stringent privacy requirements, potentially limiting its adaptability in such domains.*

*Smooth Data Flow Between Modular Blocks: The highly modular nature of Vision Blocks, while enhancing flexibility, can introduce complexities in ensuring seamless data flow. Misalignments or inefficiencies in data transfer between blocks may affect performance, particularly for complex workflows, potentially limiting adaptability for users with unique pipeline requirements.*

*Latency in Real-Time Processing: Real-time applications, such as autonomous driving, require low-latency processing. The framework’s current implementation may face latency challenges, especially when handling large datasets or complex tasks, which could restrict its suitability for time-sensitive user.*

**Recommendations for improvements**

*While our paper does not explicitly list recommendations, our comprehensive analysis, particularly the error analysis (Table 2) and limitations section, implicitly suggests several areas for improving the AgCV framework to overcome its limitations and enhance its performance. Based on these insights, we propose the following recommendations for future iterations:*

* *Enhanced Privacy Mechanisms: To address confidentiality and privacy concerns, future iterations should incorporate robust encryption protocols and anonymization techniques for data processing. Integrating secure multi-party computation or federated learning could ensure compliance with privacy regulations, making AgCV suitable for sensitive applications like healthcare.*
* *Optimized Data Flow: To improve smooth data flow between modular blocks, we recommend developing standardized data exchange protocols and optimizing inter-block communication. Techniques like data caching or streamlined API integrations could reduce inefficiencies, enhancing adaptability for complex user workflows.*
* *Latency Reduction: To mitigate latency in real-time processing, future research should focus on model optimization techniques, such as quantization, pruning, or hardware acceleration (e.g., using GPUs or TPUs). Additionally, enhancing LangGraph’s parallelization capabilities could further reduce processing times, making AgCV viable for time-critical applications.*
* *Error Mitigation: Our error analysis identified planning (35.02%), grounding (38.49%), and execution (40.26%) errors. To address these, we suggest:* 
  + *Planning Errors: Improve the RAG pipeline’s retrieval accuracy and enhance LLM prompt engineering to generate more precise task plans.*
  + *Grounding Errors: Refine interaction mechanisms, such as improving coordinate selection algorithms or integrating visual feedback loops to ensure accurate element targeting.*
  + *Execution Errors: Implement adaptive decision-making algorithms to prevent repetitive actions and ensure smooth subtask transitions, possibly through reinforcement learning techniques.*
* *User-Centric Enhancements: To further enhance accessibility, future iterations could include interactive tutorials or guided workflows within the GUI, helping non-expert users navigate the system more effectively. Expanding the RAG pipeline to cover a broader range of CV topics could also improve user understanding.*
* *Scalability Improvements: To support larger-scale applications, we recommend integrating distributed computing frameworks or cloud-based processing to handle massive datasets. This would enhance AgCV’s scalability for enterprise-level CV tasks.*

*By addressing these areas, future iterations of the AgCV framework can overcome its current limitations, further improving its accessibility, scalability, and adaptability for diverse user needs in the rapidly evolving field of computer vision.*

**Ethics statements**

*In this work data collected from online platforms, confirming that participant data has been fully anonymized.*

**CRediT author statement**

*Arav Saxena: Methodology, Data curation, Visualization, Dr. Archan Y. Chaudhari: original draft writing, editing and reviewing, Anilkumar Gupta: editing and reviewing, Project administration.*

**Acknowledgments**

*This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.*

**Declaration of interests**

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**References**

1. Stockman, G., & Shapiro, L. G. (2001). *Computer vision*. Prentice Hall PTR.
2. *A. Bendale, K. Chiu, K. Marwah and R. Raskar, "VisionBlocks: A Social Computer Vision Framework," 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, Boston, MA, USA, 2011, pp. 521-526*
3. Shanahan, M. (2024). Talking about large language models. *Communications of the ACM*, *67*(2), 68-79.
4. Chowdhary, K., & Chowdhary, K. R. (2020). Natural language processing. *Fundamentals of artificial intelligence*, 603-649.
5. Mavroudis, V. (2024). LangChain.
6. Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. Proceedings of the International Conference on Machine Learning, 6105–6114.
7. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778.
8. Alexey Dosovitskiy and Lucas Beyer and Alexander Kolesnikov and Dirk Weissenborn and Xiaohua Zhai and Thomas Unterthiner and Mostafa Dehghani and Matthias Minderer and Georg Heigold and Sylvain Gelly and Jakob Uszkoreit and Neil Houlsby (2021). An image is worth 16x16 words: Transformers for image recognition at scale. International Conference on Learning Representations.1-21
9. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
10. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779–788.
11. Chase, H. (2022). LangChain: Building applications with LLMs. Retrieved from <https://www.langchain.com>
12. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv preprint arXiv:2005.11401.
13. Izacard, G., & Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. arXiv preprint arXiv:2007.01282.
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.
15. Lialin, V., Fedus, W., & Hinton, G. (2022). Scaling laws for parameter-efficient fine-tuning. arXiv preprint arXiv:2203.15556.
16. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... & Leike, J. (2022). Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155.
17. Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
18. Deokate, S. T., Chaudhari, A. Y., Bongale, A. M., & Dharrao, D. (2023, August). A comprehensive survey and bibliometric analysis of convolution neural network. In *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)* (pp. 1-9). IEEE.
19. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., & Sun, J. (2021). Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13733-13742).
20. Sinha, D., & El-Sharkawy, M. (2019, October). Thin mobilenet: An enhanced mobilenet architecture. In *2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON)* (pp. 0280-0285). IEEE.
21. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).
22. Du, J. (2018, April). Understanding of object detection based on CNN family and YOLO. In *Journal of Physics: Conference Series* (Vol. 1004, p. 012029). IOP Publishing.
23. Jia, X., Bartlett, J., Zhang, T., Lu, W., Qiu, Z., & Duan, J. (2022, September). U-net vs transformer: Is u-net outdated in medical image registration?. In *International Workshop on Machine Learning in Medical Imaging* (pp. 151-160). Cham: Springer Nature Switzerland.
24. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.
25. Chaudhari, A., & Mulay, P. (2019). Algorithmic analysis of intelligent electricity meter data for reduction of energy consumption and carbon emission. *The Electricity Journal*, *32*(10), 106674.
26. Injury prediction dataset <https://www.kaggle.com/datasets/mrsimple07/injury-prediction-dataset> (Accessed on 13 Jan 2025)
27. Chaudhari, A. Y., Mulay, P., & Chavan, S. (2025). The Role of Smart Electricity Meter Data Analysis in Driving Sustainable Development. *MethodsX*, 103196.