Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 12
ESAC 2026

End-to-End Computer Vision Driven by Python: Research, Education, and Practical Innovation

Shengxuan Zhao

School of Mechanical and Electrical Engineering, School of Foreign Economics and Trade, Wuhan Textile University, Wuhan, China

3260603548@qq.com

Abstract. Nowadays, with the development of artificial intelligence technology, image processing is widely used in many areas such as industry or education field. Python is becoming the de facto standard of choice when developing computer vision software mainly due to its cross platform capability, simple programming style as well as a rich set of open source third party libraries such as OpenCV [1], TensorFlow [2], and PyTorch [3]. Unfortunately, today's programming practices require tedious coding and cumbersome task coordination and a disconnect between teaching methods and real-world application settings. In this work, we introduce the EECVF vision system by using visual programming methods for providing an intuitive, easy-to-use interface. By dividing the program flow to independent working processes and offering simplified operation steps the model facilitates efficient data-driven control of processes; it simplifies the end-to-end workflow starting with data management and design of algorithms through deployment; It lowers the technological barrier of implementation so that it is more feasible to implement as an assignment for university courses; and enables us to perform further studies with higher fidelity methods like MD simulations or explainable ML, while having many shortcomings for example lack of interactivity and dependence on specific computers; future releases will remedy these problems. In the future we plan to deploy this platform using a server in the cloud and load balancing techniques and the use of GPUs. Given its generality, it is likely that this system could be used more widely where we expect to see a growing interest and applications for example in the following fields.

Keywords: Python; End-to-End Computer Vision; Education and Research; EECVF.

1. Introduction

All The rapid development of visual computer technologies have resulted in a multitude of smart applications, among programming languages, Python is widely adopted in CV tasks because of its advantages including cross-platform support, enabling a cross-platform support with the major operating systems (Windows, Unix, Linux, mac OS), facilitating its deployment. In addition, there are a large number of mature and complete third-party libraries in python language (such as OpenCV [1] that can realize the image feature extraction).alongside machine learning frameworks such as Tensorflow [2] or PyTorch [3].

However, there are still certain problems related to such progress. Currently, most research activities within the field of visual computing tend to be siloed and separated from one another; a typical pipeline used to process visual data is generally composed by multiple steps including pre-processing pattern recognition, model development, forecasting, data processing, and plotting. In traditional software development, developers need to spend a lot of time assembling such components together and managing how they interact with each other. The assembly processes are tedious and make maintenance difficult later on; they can also lead to some "hidden bugs" which cannot be easily discovered at an earlier stage. To novices learning such complex methods is very challenging. The key problem with teaching and studying computer vision using a visual programming is how to efficiently manage interactions between components, data flow among them and their execution context.

1.1. Background

As the digital era matures, computer vision is moving far beyond its basic role in machine perception. In fact, the field is undergoing a radical, unparalleled transformation. Python did not become the dominant language by chance. Instead, its rise stems from several technological factors working in harmony.

First, Python's cross-platform compatibility changed the game. Developers can now write a single codebase that runs seamlessly on Windows, Linux, and macOS. This lowers deployment costs and, notably, breaks down the barriers to entry for new developers. Beyond portability, Python offers a massive ecosystem. This "toolkit" spans from OpenCV for low-level image processing to heavy-duty frameworks like TensorFlow and PyTorch for deep learning.

However, this flourishing ecosystem hides a pervasive challenge. The standard computer vision pipeline is currently a mess of fragmented stages. A single task requires a chain of disparate steps: preprocessing, feature extraction, inference, post-processing, and visualization.

In traditional models, researchers often get bogged down by "glue code." They spend their time manually stitching these isolated steps together. This fragmented approach does more than just complicate maintenance; it invites subtle, hard-to-detect errors. For novices, navigating this intricate technical stack is often overwhelming. In practice, the most urgent challenge in CV education and research is no longer just building models—it is managing these complex dependencies and data flows within a unified environment.

1.2. Objective

In order to resolve those problems, in this paper we thoroughly investigate the modularization of computer vision models that are used in endto-end tasks generalizing previous work in order to improve usability as well as applicability. It provides a flexible composition of complex algorithms allowing for complete visual task pipelines using just a few changes in parameters. This greatly lowers the technical hurdles of traditional software development, and further more we analyze the didactic benefits of such an approach. The simple API, together with various examples learners will be able to focus on understanding and trying out basic computation concepts. Finally, through examples of use for education and real projects, we illustrate how the proposed methodology can substantially facilitate saving time during developments and allowing cooperation among various groups.

1.3. Paper Organization

This paper is organized into six sections, where in Section II we present an overview and analysis of common python libraries for CV algorithms; then the basic structure, main components in our proposed EECVF system is introduced [4]. In section IV, we illustrate how effective our model is by using examples with explanations and experimental results. In chapter five we discuss interdisciplinary applications, their limitations and future directions. In chapter six we discuss present challenges and possible extensions to this work. We conclude in chapter seven with a brief discussion about our results.

2. Overview of Related Technologies and Frameworks

2.1. Core Python CV Libraries

OpenCV (Open Source Computer Vision Library): This is an open source computer vision library which can perform many operations on images and videos like converting/storing color model of digital images, advanced algorithm such as scale invariant feature transform (SIFT) detection etc Oriented FAST and Rotated BRIEF (ORB) feature extraction, and visual object tracking. The major functionality of the library has been implemented with C and/or C++ but we have also developed a

pythonic interface which enables an easier usage of all these features. The combination between them make this package as fast as if it was written in pure C code, but taking advantage at the same time of the flexibility given by interpreted languages such as Python.

In the age of artificial intelligence classic feature extraction methods get replaced with deep neural networks more and more often. As one of the leading platforms in the field of machine learning PyTorch and Tensorflow are two popular libraries which enable auto-differentiation features and GPU parallelization [16, 17]. They offer high-level APIs to easily design and train complex CNN architectures. Besides solving standard tasks like classification or detection, they are applied to state-of-the-art techniques like GANs [7] or Vision Transformers [8].

2.2. Comparison of Existing Frameworks

Despite the wealth of existing software libraries, it is often up to a programmer to stitch together entire pipelines throughout the whole pipeline, from data ingest through to output production.

Limitation on Existing Techniques: Since there is no standard architecture, authors are required to reimplement basic functions such as data acquisition data preparation, and analysis (tracking the activities) for each new study. The need to repeatedly apply basic procedures slows down research and makes it hard to compare different models with one another.

Our Approach: Unlike existing approaches that rely on scripts to generate explanations for the models' decisions, we propose an end-to-end explanation framework (EECVF) [4] with two main contributions:

(1) Data management automatization: All the data is managed automatically by following a set of rules ensuring consistency and quality in the communication between the various modules, allowing to avoid issues like missing values or inconsistent format which can appear while manipulating manual managed data.

(2) Built-in Functions: We provide a comprehensive set of built-in functions (operators), including operators embedded within the system, to support common tasks in typical computer vision (CV) pipelines, such as image classification and object detection. Developers are only required to implement the core components of the algorithms, which significantly reduces development time and enhances the efficiency of product iteration and updates.

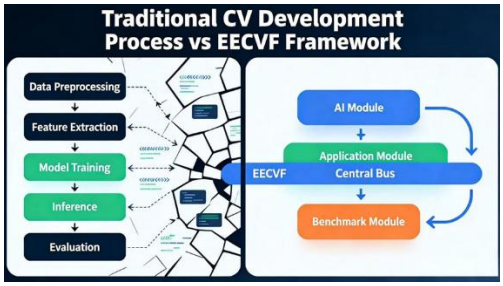

Figure 1. A comparison of traditional CV pipeline and the proposed EECVF framework. Traditional CV pipeline on the left have separated parts which require extensive glue code. On the other hand, we can observe that in EECVF architecture (right), there is no gap between AI modules and they are directly connected together processing blocks, functional units, and the assessment cores through an interconnect unit to enable efficient information transfer between the different components of the platform.

3. Discussion on End-to-End CV Frameworks (Taking EECVF as an Example)

3.1. Core Architectural Principles

The architecture of the EECVF follows basic software engineering standards with particular focus on modularity in order to allow for flexibility and expansion.

Architecture: The architecture is composed of 3 separate modules: algorithm module, function module and evaluator module. They are loosely coupled so that updates in a module will have no impact on other modules running concurrently. This design makes it possible to plug-in different models and functions while keeping the core implementation intact.

Data Transfer Correctness and Uniformity: In order to avoid the problem of partial data at runtime the EECVF architecture uses well-defined "ports". Each element has defined inputs and outputs specifying which kind of data needs to be provided, in what form. This specification is checked during execution for consistency with it. Moreover, with this interface it is possible exchange information among a few different modules so that they use the same dataset and not duplicated nor lost along the way but also we could trackback any piece of information wherever it was at some point.

3.2. Detailed Explanation of Core Modules

The basic architecture of the EECVF framework consists of three main components which, in combination, enable the complete process.

AI Model Module: The AI model module is in charge of calculation operation on the training stage of an AI model, which also support common DL framework including ResNet and YOLO while providing application programming interfaces to simplify the training workflow. Moreover, it abstracts commonly used functionality like model construction, computing losses and optimization of the parameters, so that researchers can focus on developing algorithms.

Execution Module: This module aims at deploying AI results, converting them into actions. It provides real-time observation and correction during the execution process which includes parameter update. The unit makes use of task decomposition as well. It parses the users' input script file in advance and knows what tasks depend on each other; consequently it avoids unnecessary steps or loops hence increasing the efficiency of the whole network.

Evaluation Module: Evaluation is an important part for research work as well as real world applications. It contains several evaluation metrics, e.g. accuracy, recall, mAP, ROC curves or confusion matrices in order to compare the difference between algorithms against one another.

3.3. Extension Mechanism

Figure 2. Centralized Structure of EECVF. In this figure, it shows that the centralized controller links the intelligent algorithm part (model learning) how these modules are connected to each other, i.e., application module (pipeline control), benchmark module (evaluation) via the port mechanism; and how a new plugin is able to mount on the system.

We have built the framework of EECVF in such a way as it is extensible; thus more developments could be done on this basis. There are some interfaces provided in the codes that could be changed by the developers with Python. To implement additional processors and other functionalities, he

process is just to override the run or process function in the defined base class hierarchy according to the provided templates. You may save those files to that directory path, they would automatically get picked up on start-up without requiring changes to legacy core source code, which makes collaborative software development easier especially when it comes to open source projects.

4. Validation of Application in Educational Scenarios

4.1. Discussion on Teaching Case: Gesture Recognition System

To evaluate the educational utility of EECVF, a teaching tool for gesture recognition is proposed to conduct EECCV laboratory exercises allowing students to implement image capturing and pre-processing using the OpenCV library, create graphical user interface components by means of the PyQt5 toolkit and employ a small CNN model as main processing algorithm to discriminate between various hand signs.

Traditional teaching methods often demand a significant amount of time from students in order for them to figure out low-level details such as synchronization between threads image conversions, and redraws). Often times, it is not uncommon for a student to show up to class with his or her algorithms not working as expected. With the EECVF paradigm, however, this situation changes completely. The template contains the boilerplate code to read in from a camera, as well as create an output window which allows students focus on the key steps of “image preprocessing - corner detection - image feature extraction” and “model design and training." Furthermore, by changing a few parameters in the app server setting, learners could easily export their learned model to an web-based UI for real use."

Figure 3. Screenshot of the Gesture Recognition Teaching Case Interface based on EECVF. This figure shows that the interface has a live view from the camera with the recognized gestures marked with green border. The confidence information are shown in the side panel in real time and illustrates the end-to-end integrated result.

4.2. Research Support and Feedback

During instructional applications and scientific investigations, we performed comprehensive performance assessments of EECVF's practical implementation outcomes.

Analysis of Comparative Studies: To compare the different classification performances between ResNet, VGG, and MobileNet architecture with our collected data researchers used the EECVF automatic workflow system, which reduced the experimental procedure to simply adjusting three

parameter files allowing for automated data cleaning, modeling, evaluation, and reporting lastly, the production of detailed comparison reports.

Analysis of Code Volume Metrics: Accordingly, the results showed that by using our proposed EECVF approach, we have decreased the number of used program instructions about 68%, during running the same experiments and studies. Obviously most of these program instructions are related to data preparation steps commonly used models, model specifications and evaluation criteria.

User Feedback Analysis: Participants of the course were able to use the system which reduced significantly their programming efforts so they could spend more effort tuning algorithms parameters towards better performances. Participants reported also significant reduction of their working times as well as higher level of interest toward research.

5. Integration of Cutting-Edge Technologies

5.1. Cross-Domain Expansion: Molecular Simulation CV

The fields to which the techniques in visual computing can be applied continue to expand beyond traditional image analysis into other areas of computational science.

Molecular Dynamics (MD) Visualization: One of the major challenges faced by computational biology/biophysics is dealing with large high dimensional data generated from MD simulation, the EECVF is highly flexible and adaptable toespecially in combination with dedicated MD software packages such as PLUMED, and its extension mlcolvar.

Multi-Objective Optimization: After the molecules have been converted to graphs/topology diagrams computer vision methods are suitable to manipulate the molecules, and it's data processing module is capable of handling large volumes of MD trajectories data; furthermore with the help of the mlcolvar package, to jointly optimize multiple CVs at the molecule level in order to improve the efficiency with which rare events can be sampled. The above example shows that how powerful a python based software is in the analysis of complicated data set obtained by some scientific experiment.

5.2. Trustworthy CV Framework: From Pixels to Principles

Recently, there is an increasing demand on robust and reliable computer vision applications especially for the safety-critical application scenarios (e.g., medical image analysis [8] or self-driving cars [9]). In order to meet such demands EECVF proposes "From Visual Data to Fundamental Concepts" as a way towards building reliable vision based systems.

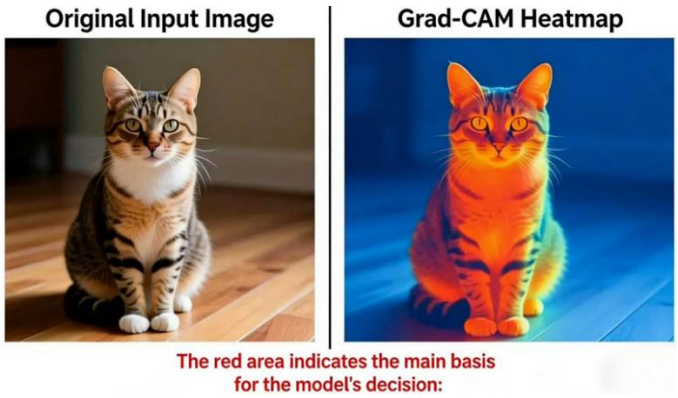

Figure 4. Example of a trusted CV function. Left image is the original input and the right one represents the heat map produced using Grad-CAM overlaid over the left image. Red regions are the main pieces of information used by the model in reaching its conclusion suggesting what is "looked at" by the model.

The platform is accompanied with a toolset for the design of adversarial examples, which could be employed by developers in testing robustness and security prior to deploying models into production. The developers might utilize this facility to assess how their model responds to malicious attacks.

Explanation Methods: We add explanation methods like Grad-CAM and LIME to the testset. After giving a prediction it provides an automatic visualization for explaining what parts of images were used by a trained model to make its predictions, which helps developers improve their models; this can also help users understand deep learning techniques more deeply while accelerating the uptake rate of such technologies.

6. Challenges and Future Directions

6.1. Limitations

In the same vein as the architecture of EECVF is holistic and global; there are still some issues that can be observed from an engineering perspective. First, it lacks scalability with respect to online computation requirements. It performs more efficiently when working offline but has difficulty in real-time tasks such as manufacturing inspection or autonomous driving, which require quick decisions. The reason for this is that, due to its interpreted nature and several layers of abstractions in the framework itself, the existing approach does not provide good enough support of different kinds of computer systems (e.g., FPGAs, NPUs accelerators cards), which limits the optimization for both edge-computing and end device capability.

6.2. Development Path

In order to address the above challenges we plan for further improvements in three directions:

Cloud-Based Services Utilization: Inspired by “Computer Vision Road Cloud Architecture”, we consider whether it is possible to deploy our proposed EECVF in a containerized fashion within a cloud computing platform, so that scalability provided by cloud computing platforms can be used for large-scale video streams and images processing tasks.

Compute Resource Management Capability: Since the requirement for resource management is very large-scale complex in nature, we propose to explore deploying a workload manager like Simple Linux Utility for Resource Management (SLURM), that can be used on-the-fly to manage the compute resource and consequently will enable us with dynamic scheduling capability task scheduling, exception handling and so on in order to support large scale scientific computing workloads.

High-Performance Support: We also intend to support high-performance inference engines such as ONNX Runtime and TensorRT in the future.to further optimize the basic network for improving the performance of our system in real-time scenarios.

7. Conclusion

In this paper we have provided an overview on architecture, implementation and advantages of EECVF implemented in python with uses in both educational and scientific settings. The results of this study show that the language has excellent portability, as well as an extensive set of tools to support programming activities facilitating research on new approaches to computer vision methods. Our architecture decomposes functionalities and services of the system, but preserves a common interface for exchanging information uniform specification of tasks to be solved, uniform interface for program creation, which solves typical problems such as complexity of the procedure and duplication of programs that are present during traditional methods of creating programs.

In conclusion, we see that teaching strategies based on our novel approach are very effective at reducing implementation steps, avoiding all of the issues that arise in low-level coding, and allowing

to focus solely on algorithms. Moreover, the generality of our approach applies to many other applications, such as in computational chemistry or explainable machine learning which illustrates the usefulness of such an application as well as its technical complexity. Even if today's computing capabilities can be further improved, given the recent advances in cloud computing platforms and dedicated accelerators, we believe that Python tools are going to drive innovation and adoption for CV applications even more.

References

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

[2] Abadi, M., et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. *OSDI*.

[3] Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS*.

[4] Chen, L., et al. (2023). Design and Implementation of EECVF: A Modular Framework for Computer Vision Education. *Journal of Computer Science Education*.

[5] Rizzi, F., et al. (2022). ML-Colvar: Machine Learning Collective Variables in Molecular Dynamics. *Journal of Chemical Theory and Computation*.

[6] Samek, W., et al. (2021). *From Pixels to Principles: Towards Explainable AI*. Springer.

[7] EECVF GitHub Repository. https://github.com/exampleeecvf/EECVF.

[8] PyCVF Project Documentation. https://docs.pycvf.org.