D.I.A.G.R.A.M.: Development of Image Analysis for Graph Recognition And Modeling

Filippo Garagnani, Saverio Napolitano, Nicola Ricciardi
'Computer Vision and Cognitive System' course
Università di Modena e Reggio Emilia

***Abstract*—This report describes D.I.A.G.R.A.M., a system for diagram recognition and generation. We present its architecture, the logic behind deterministic algorithms and the training of its deep learning components. The project aims to transform visual diagram input into structured representations.**

I. Introduction

Diagrams are crucial in education, documentation, and many other fields. Developing a system that automatically understands and generates diagrams may reveal helpful for having high-quality, easily editable representations. This, however, poses challenges involving classification, shape detection, structure interpretation and symbolic representation. Our project proposes a modular architecture to tackle these tasks.

II. System Architecture

Figure 1 illustrates the full pipeline, composed of four main modules: the Classifier, Extractor, Transducer, and Compiler. Each module is designed to process and transform the handwritten diagram image progressively toward a structured output.

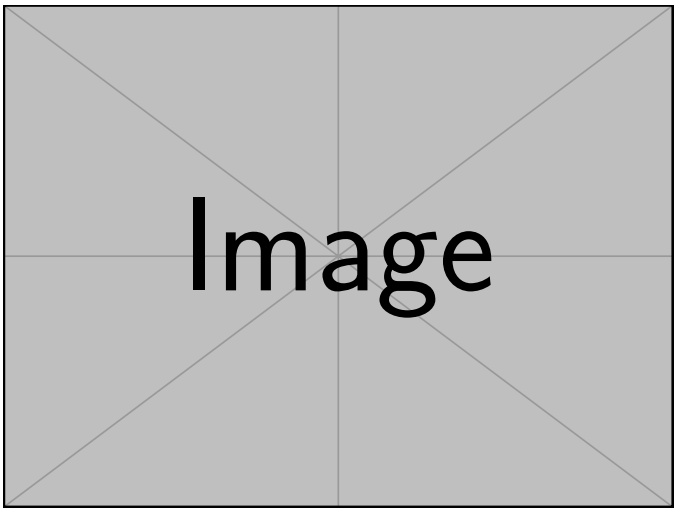

Fig. 1: Overview of the D.I.A.G.R.A.M. system architecture.

The Classifier is able to recognize the category of the hand-written diagram that the user submitted as input. This is necessary in order for the system to know to which Extractor pass the data. This module is able to apply object detection and semantic recognition to represent the diagram in a unified way. After that, the representation is sent to the Transducer, which converts the data in a Markup language of choice. The Markup language content is therefore sent to the Compiler, thus generating the high-quality and editable diagram in .png format.

III. Classifier Module

TODO

A. Preprocessing

TODO

B. Model

TODO

IV. Extractor Module

TODO

A. Preprocessing

TODO

B. Bounding Box Detection

TODO

C. Content Recognition

TODO

V. Transducer Module

TODO

VI. Compiler Module

The Compiler module is the final stage of the D.I.A.G.R.A.M. pipeline. Its primary role is to take the structured representation of the diagram, expressed in a markup language (e.g., Mermaid.js), and generate a high-quality, editable diagram in a visual format such as PNG. The Compiler is designed as an abstract class to ensure flexibility and extensibility. It provides a generic interface for compiling diagrams from various markup languages into visual outputs.

VII. Experiments

A. Dataset

TODO

B. Metrics

TODO

C. Results

TODO

VIII. Discussion

TODO

IX. Conclusion and Future Work

TODO