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

Filippo Garagnani, Saverio Napolitano, Nicola Ricciardi

April 7, 2025

Summary

This project proposes the development of a system for analyzing different types of handwritten diagrams and converting them into well-rendered images through a textual syntax. The goal is to create a tool capable of analyzing scanned or photographed sketches of diagrams and automatically generating code that can be rendered into the same diagrams digitally, to eventually integrate or modify them.

1 Introduction

Diagrams are a powerful means of representing complex systems, ideas, and relationships. Writing diagrams in a textual format is extremely efficient to easily edit the diagram and to display it in a better-formed format. This project aims to automate the conversion of handwritten diagrams into code, streamlining the process of diagram creation and visualization.

2 Objective

The main objective of the D.I.A.G.R.A.M. system is to:

- Develop a computer vision model to recognize handwritten diagrams.
- Extract relevant features from the diagrams (nodes, edges, labels).
- Generate equivalent code in supported languages for the recognized diagrams.
- Automatically parse the code to generate the formatted image of the input diagram.

3 System Architecture

The project will be composed of different modules that will be stacked together in an uniform pipeline [**Figure 1**]. These components are:

- **Preprocessor**: Applies some operations (like denoising, rotation...) in order to obtain a more suitable input image for the following modules.
- **Classifier**: In order to manage more than one type of diagrams, a classifier that identifies the category of the input diagram is needed.
- **Extractor**: By employing feature extraction, image segmentation and static analysis over the recognized components, the extractor will output a robust structure to contain all the relevant information about the diagram. This key component takes the image as input and will output the entities found, their relations and more useful information.
- **Transducer**: From the data output by the extractor, a certain transducer will give out a formatted code relative to the input diagram. We envision a single transducer for every supported language.
- **Compiler**: Given the code, the compiler will convert it into an output image - hopefully, as close as possible as the input diagram.

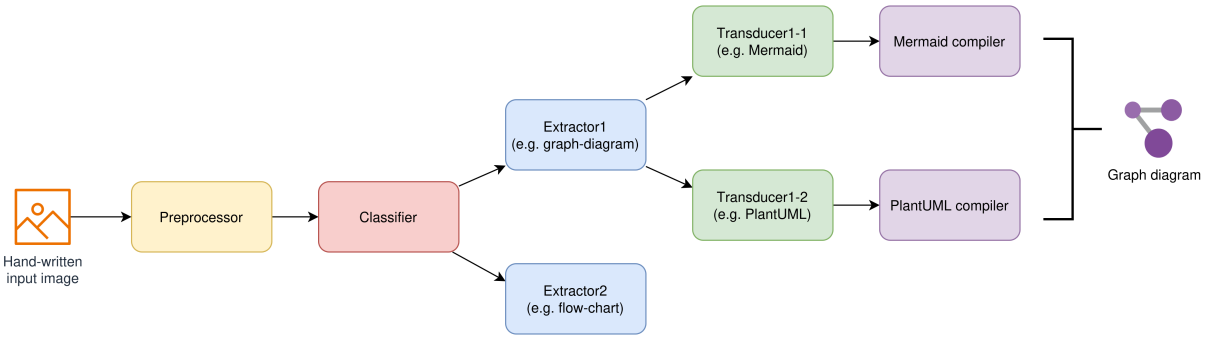

Figure 1: The D.I.A.G.R.A.M. pipeline.

4 Constraint

While the whole project idea could be applied to a wide variety of diagrams and languages, in order to manage the complexity we decided to impose a few theoretical constraints:

1. **Types of diagrams**: We restrict the types of diagrams that the system is able to correctly recognize and reconstruct (*e.g.* Graph diagrams, Flowchart diagrams...)

2. **Languages**: Even if there are various languages in which the diagrams could be formatted, we limit the languages that the system could translate the diagrams to (*e.g.* Mermaid, D2...)