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.

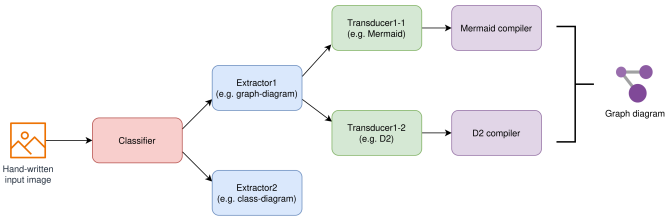

Fig. 1: Overview of the D.I.A.G.R.A.M. system architecture.

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.

A. Classifier Module

The Classifier is the first component of the pipeline. It is able to tell to which category a given image of an handwritten diagram belongs to. This is necessary in order to later know the Extractors that can process the input image.

B. Extractor Module

The Extractor is the key component of the D.I.A.G.R.A.M. system, able to transform the raw diagram image into a structured, category-agnostic representation. The Extractor recognizes the diagram's components, such as shapes, lines, and text, through an object detection network, and organizes them into a unified format. This representation serves as the foundation for the subsequent Transducer module, which converts the structured data into a domain-specific markup language.

C. Transducer Module

The Transducer is responsible for converting the unified, agnostic representation of a diagram into a domain-specific markup language. This transformation enables the subsequent Compiler module to generate high-quality visual outputs.

D. 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 diagram in a visual format such as PNG.

III. Graph Diagram Recognition

A. Internal Representation

In order to pass data through the components of the pipeline in a unified way, it was necessary to decide upon an internal representation of the graph diagram. It consists of **elements** and **relations**. The latter are linked to at least one element. Every element has a *category* and possibly some *inner* and *outer text*. Every relation has a *category* too, either a *source* or *target element* or both, and possibly *text* related to it.

This internal representation is generated from the image provided by the user by the Extractor module, while it is parsed and converted to a markup language of choice by the Transducer.

B. Classifier Module

1) Preprocessing: TODO

2) Model: TODO

C. Extractor Module

1) Preprocessing: TODO

2) Bounding Box Detection: TODO

3) Content Recognition: After having determined all the bounding boxes of the diagram image, in order to correctly create the diagram internal representation, there's the need to link together text, elements and relations.

4) Text Digitization: Since the task of converting an image of text into the relative string has already been solved in various ways, we settled upon the Microsoft *Trocr Handwritten* Transformer model; more precisely, the *base* version of it. Every bounding box containing text that has been found is later passed to the model, and the resulting string generated by the model is linked to it.

5) Text Association: It can be assumed that every piece of text is linked to one and only one element or relation. So, every bounding box containing text is linked to the nearest element - it being either a relation or an element. The associations that require too long a distance are eliminated further in the algorithm, and the relevant text is discarded.

6) Element-Text Association: After that a text box has been assigned to an element, there's the need to understand if the text is either inside or outside the element and whether the text is actually referring to the given element or not. In order to do this, the overlap between the two bounding boxes is computed; if the overlap is over a certain threshold, heuristically decided, the association is kept. Otherwise, the distance between the two bounding boxes is computed - as the distance between their central point. If the distance is over a certain threshold, the association is kept; otherwise, it's discarded.

7) Relation-Text Association: After a text box is assigned to a relation, some computation is needed to fully understand how the text is associated to the relation itself. [TODO: Come trova source e target]. The relation bounding box is then split into three different sections. The overlap between the text bounding box and each of the three sections boxes is computed - then, the section with the highest overlap value is then assigned the text under consideration.

IV. Experiments

A. Text Digitization

In order to choose an appropriate model, some tests were done over our specific dataset employing some different, pre-made models. The average results over the dataset are shown in Table I.

Some examples of the different models applied to our graph diagram dataset can be seen in Figure 2 and in Figure 3.

TABLE I: Text Distance Metrics

Model	Hamming	Cosine	Euclidean
microsoft-trocr-small-printed	1.392	0.150	7133
microsoft-trocr-small-handwritten	1.346	0.100	3947
microsoft-trocr-base-handwritten	1.549	0.057	3003

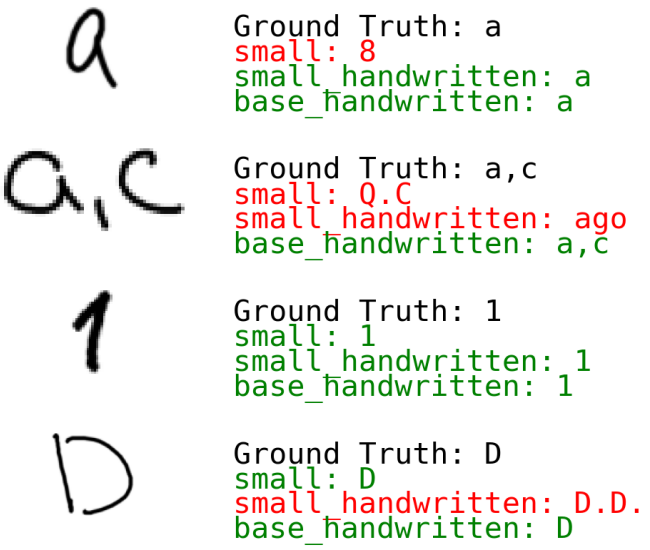

Fig. 2: Some random results of the text digitization over the proposed models.

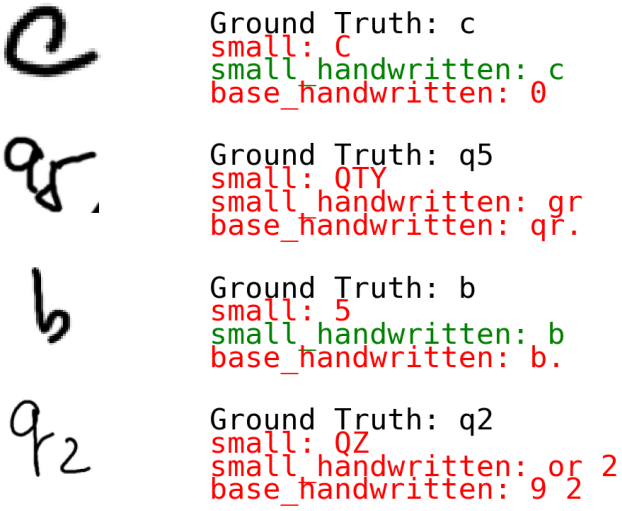

Fig. 3: Some results in which the *trocr-base-handwritten* fails.

V. Discussion

TODO

VI. Conclusion and Future Work

TODO