Automatic Construction of a Quantum Circuit Image-to-Text Dataset from Scientific Papers

Tejaswi Duptala
MAI Program
Ostbayerische Technische Hochschule Amberg-Weiden
Amberg, Germany
Supervisor: Prof. Dr. Patrick Levi

Technical Report, Winter Semester 2025

***Abstract*—The image-to-text models are widely used to give a description of an image in natural language, they can be ineffective in scientific schematic diagrams. A crucial application of that would be the description of the quantum circuit diagram, which consists of a structured layout with a set of specialized symbols that cannot be found in the standard image datasets. In this project, a corresponding dataset of quantum circuit images is developed to contribute to the domain of future image-to-text tasks within the context of quantum computing. Figures of quantum circuits are automatically retrieved from scientific papers published under arXiv with the "quant-ph" category. For each valid image of a circuit, more metadata can be obtained, such as the arXiv identifier, the page number where the figure appears, the figure number, the set of quantum gates that the circuit uses, the corresponding quantum algorithm if determinable from the context of the circuit, and a brief description fetched from the scientific paper. The dataset can be assembled by a completely reproducible pipeline involving PDF data processing, rule-based image filtering techniques, optical character recognition routines, and a set of keywords. Ultimately, it is possible to compile such a dataset given reasonable efforts.**

***Index Terms*—Quantum circuits, dataset creation, image-to-text, arXiv papers, quantum computing, OCR, scientific diagrams, natural language processing.**

I. Introduction

Image-to-text models are known to be a significant application area of artificial intelligence, where machines can explain what is perceived in a given photo or picture in terms of language. The models are trained using data sets that contain images of everyday objects, for example, people, animals, and so on. Despite this, these models are unable to grasp a technical diagram, specifically a scientific diagram, such as a quantum circuit.

Quantum circuit diagrams are a unique form of diagrams used for research work in quantum computing. They contain organized designs, a unique set of symbol representations for gates, and related technical details, which cannot be appropriately encoded within standard image datasets. It becomes a challenge to train image-to-text models for handling such a specialized form of content unless a suitable dataset is provided.

The purpose of this project is to provide a small, high-quality dataset of quantum circuit images and their corresponding descriptions. Additionally, all circuit images are obtained from recent scientific literature in the "quant-ph" (Quantum Physics) category on arXiv.org. This was done to provide all of the circuit images with a clean, straightforward PNG format and to tie each image to its corresponding data, such as gate information, quantum algorithm information (if applicable), and image descriptions from the literature.

A dataset like this can serve as assistance in the future completion of specialized image-to-text tasks in quantum computing, which are currently not done well by present models. A data set like this, as it contains consistent examples, can assist in evaluating models that understand scientific schematic images.

II. Related Work

Image to text models are known to be a major use case for artificial intelligence, where computers are able to describe what is perceived in a given picture in terms of language. The models are trained using data sets which contain images of various objects in a day-to-day life, for example, people, animals, and so on. Despite this, these models are unable to understand a technical diagram, specifically a scientific diagram, for example, a quantum circuit.

Quantum circuit diagrams are a special type of diagrams for which the task of research work related to quantum computing can be done. These include the structured diagram, a unique set of symbol representations for gates, as well as some additional technical information, which cannot really be encoded in general image datasets. There might be a challenge when training image-to-text models to handle this specialized type of content. The objective of this project is to offer a small but high-quality set of images of quantum circuits and their descriptions. In addition, all of the circuit images are taken from recent literature within the "quant-ph" (Quantum Physics) category at arXiv.org. This approach ensured that all of the circuit images were of a clean and simple PNG format and that each image is linked with its corresponding information, including gate information, information about quantum algorithms (if applicable), and image descriptions from literature.

Such a data set can always be of use in the future processing of specialized image-to-text jobs in quantum computing, which cannot currently be done effectively by existing models. Such a data set, as it consists of relevant exemplars, can always Aid in assessing models capable of comprehending scientific images.

III. METHODS

A. *Project Setup and Tools*

This project was implemented entirely using a **Jupyter Notebook**, which allowed for interactive development and easier debugging. All code for the dataset extraction, classification, and annotation was written and executed in a single notebook file.

1) Project Folder Structure: The following folder structure was used to organize the project:

```
NLP_examid_5_Tejaswi_Duptala/

Project_code_file/
paper_list_5.txt/
data/
  pdfs/ # Downloaded PDFs from arXiv

images_5/ # PNG images of extracted circuits

outputs/
  dataset_5.json # Annotated dataset in JSON
  paper_list_counts_5.csv # Circuit counts per
       paper
  logs/ # Error logs

documentation_5.pdf # Final project report (
    this document)
```

2) Tools and Libraries: All components of the project were developed in Python, using libraries that are available on the lab machines (DC1.07). The table below summarizes the main tools used:

Library	Purpose
PyMuPDF (fitz)	Extract images, text, and layout from scientific PDFs
Pillow (PIL)	Image manipulation and PNG saving
OpenCV	image processing (e.g., for line detection)
pytesseract	OCR for extracting gate labels directly from images
re	Regular expressions for figure numbers, gate names, etc.
json / csv	Output writing for the dataset and paper counts
pathlib / os	File handling and directory management
logging	Recording errors, missing data, and skipped figures

TABLE I
PYTHON LIBRARIES USED IN THE NOTEBOOK-BASED IMPLEMENTATION

No deep learning models or external APIs were used. All methods are fully rule-based and reproducible without internet access apart from arXiv paper downloading.

B. *Paper Collection and PDF Download*

The dataset construction starts with collecting scientific papers from **arXiv**. A predefined list of paper identifiers was provided in the file `paper_list_5.txt`. This list contains paper IDs from the *quant-ph* category and must be processed **strictly in the given order**, as required by the project instructions.

Paper Processing Strategy:

- All paper IDs are loaded into memory at the beginning of the program.
- Papers are processed **one by one in sequence**.
- The pipeline stops automatically once the target number of images (`MAX_IMAGES`, set to 250 for final runs) is reached.
- Papers that are not reached due to early stopping remain unprocessed and are left blank in the final CSV file.
- If a paper is processed but no valid circuit images are found, it is recorded with a count of `0`.

This approach guarantees reproducibility and full compliance with the exam requirements.

PDF Download Mechanism: For each paper ID, the corresponding PDF is downloaded directly from arXiv using the URL pattern : https://export.arxiv.org/pdf/arxiv_id.pdf

The downloaded files are stored locally in the directory: data/pdfs/

To ensure efficiency and robustness, the download process includes several safeguards:

- **Caching:** If a PDF file already exists locally and its file size exceeds a minimum threshold, it is reused instead of being downloaded again.
- **Retry mechanism:** Each PDF download is attempted up to three times in case of temporary network issues.
- **Rate limiting:** A fixed delay is added between requests to avoid overloading the arXiv servers and to comply with good usage practices.
- **File integrity check:** PDFs that are too small (likely incomplete or invalid) are rejected and logged as errors.
- **Error handling and logging:** Failed downloads, HTTP errors, and unexpected exceptions are recorded in a dedicated log file (`download_errors.log`) for later inspection.

If a paper cannot be downloaded successfully after all retry attempts, it is skipped, and the pipeline continues with the next paper. Such papers are marked as inspected with zero extracted images.

Outcome of This Step: At the end of this stage, all successfully downloaded PDFs are available locally and ready for text extraction and image analysis. This ensures a stable and repeatable foundation for the subsequent steps of image extraction and circuit classification.

C. *Image Extraction from Papers*

After the download of the PDF file from arXiv is complete, the following step involves navigating through each and every page of the file and extracting the images from the file. The

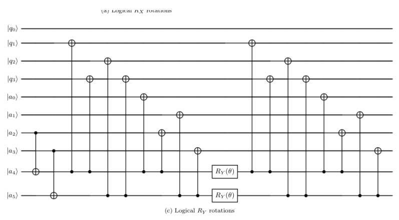

Fig. 1. An example of a raster based quantum circuit diagram extraction

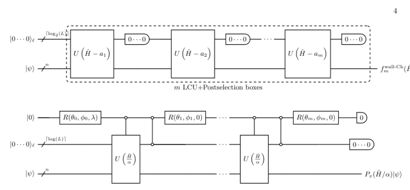

Fig. 2. An example of a vector based quantum circuit diagram extraction

extracted images may comprise a variety of figures like graphs, tables, photographs, and quantum circuits.Depending on how the image is stored inside the PDF, we use two separate methods:

A) Raster Image Overlay (Pixel Overlay): Some images are saved in the form of raster images, meaning they are saved in pixel formats such as PNG or JPEG within the PDF. Function: extract_images_from_pdf(pdf_path)

- We employ the `PyMuPDF (fitz)` function for scanning pages and extracting these raster images each page of the PDF and lists all embedded images.
- Raster figures are stored as temporary Pillow images for additional processing.
- These raster images are then passed to a classification function to check whether the images correspond to a quantum circuit (for example, according to the aspect ratio and line structure).
- Filtering small images (e.g., icons or equations) with help of minimum area threshold (MIN_IMAGE_AREA).
- And Then valid quantum circuit images will be saved in the `images_5/` folder in PNG format.

B) Vector Diagram Extraction (Drawn Figures, TikZ, etc.): In some cases, rather than being recorded in a form that can be seen as a two-dimensional pixel image, the circuits are represented in the PDF using vector graphics, for example, by using the TikZ package in LaTeX equations to produce graphics directly in the PDF. Function: extract_vector_circuit_images(pdf_path, full_text, page_texts)

- These vector diagrams are not included in the list of images that are embedded in this document.
- Instead, we render the entire PDF page as a high-resolution raster image (think screenshot).
- We then use heuristics to detect if the page contains a circuit, including:
  - Figure captions using regex (fig., figure, etc.).
  - OCR for the names of the gates
  - Geometry and layout checks
- Once a valid circuit has been detected, we crop the relevant part of the page and save it as a PNG image.

Why use two methods?: Raster images can be extracted easily, but most quality quantum circuits have been stored as vector diagrams; these require special handling.

We support both methods, so that we do not miss any valid circuits, however they might be represented in the paper.

In the end, the processing of both raster and vector figures is completed, after which candidate circuit images are saved, waiting for classification and annotation.

D. Circuit Classification

To make it possible to distinguish quantum gate circuit diagrams from other images, such as graphs or text pages, the pipeline relies on a number of helper functions. Each function checks a certain characteristic of the image. All of these functions together make up a reliable filter system.

1) Curve-Based Analysis: **curve_score**

Purpose:

This function determines to what extent the image is made up of curved shapes rather than straight lines.

Why this matters:

In quantum computations, the circuit consists largely of horizontal lines and rectangle-shaped gates.
Plots and graphs typically involve curves like sine waves, curves, or smooth trajectories.

Functioning (High-Level View):

- Finds all edges in the source image.
- This line detection algorithm identifies straight lines.
- Estimates the number of edge pixels that lie outside straight lines.
- Increases if there are lots of curved edges.

Interpretation:

- Low curve score $\rightarrow$ likely a circuit
- High curve score $\rightarrow$ likely a plot or graph

2) Text vs. Object Ratio: compute_text_object_ratio(pix)

Purpose:

This service compares the amount of the image that contains text with the amount with graphical structure.

Why it matters:

Quantum circuits contain some text (labels of the gates), but lines and boxes clearly dominate.
Text-heavy images (slides, paragraphs, equations) should be rejected.

How it works:

- Uses OCR to detect the text and approximate the total text area.
- Shape and contour recognition for determining object area.
- Computes the ratio:

text_area/object_area

Interpretation:

- High ratio → image dominated by texture (bad candidate)
- Low ratio → image dominated by structure (good candidate)

3) Graph and Plot Detection: is_definitely_graph(img, caption_text)

Purpose:
The function aggressively discards figures that are unmistakably graphs or plots, even if they are structured in any sort of way.

Why this is necessary:
The plots can become circuit-like in appearance (horizontal plus vertical lines) when performance graphs may appear similar in.

Signals used:

- **OCR keywords**
  Detects words like:
  "plot", "graph", "accuracy", "loss", "spectrum", "histogram"
- **Caption analysis**
  Ditches captions containing phrases such as:
  "as a function of", "versus (vs.)", "timetrace", "probability distribution"
- **Geometry checks**
  Presence of both powerful vertical and horizontal axes
  High line density
  Very few rectangular boxes unlike gate symbols

Interpretation:
If more than one graph-like signals are present → Image is discarded.

4) Text-Only Image Detection: **is_text_page(img)**

Purpose: Excludes images where the content is mostly paragraph text, equations, or section titles.

Why this matters: In vector page cropping, rather than images, text areas or equations may be extracted.

Checks used:

- OCR word count (many words → text)
- Definition:
- Very few boxes or wires
- Image with medium image density (text has many small edges)
- Small image height (typical for headers)

is_texts_strip(img)

Purpose: It rejects the broad and thin images of the caption, title, or a single line of text.

Typical examples excluded:

- "FIG. 2. Results of ..."
- Section headings
- Equation-only lines

Checks Used:

- Very high width-to-height ratio
- OCR recognizes words
- Nearly no geometric structure (no boxes or wires)

5) Supporting Geometry Functions: Functions that highlight the structural characteristics of the image. These functions are shared between all the classifiers.

horizontal_line_score(img)
Indicates the dominance of horizontal lines.
High score → strong indicator of quantum wires.

vertical_line_score(img) Indicates vertical lines.
In circuits, there are few lines which run vertically.
There are often many (axes) on a plot.

line_density_score(img)
Determines how "filled" a picture is.
There are sparse connections, which make the circuits sparse.
Heatmaps and graphs are densely packed.

horizontal_band_count(img)
Detects repeating parallel horizontal lines.
Contains multiple qubit wires laid out in a vertical stack.

strong_horizontal_wire_count(img)
Counts long horizontal lines.
Direct indicator of qubit wires.

small_box_count(img) Counts small rectangular shapes.
These often correspond to quantum gates.

Why So Many Functions?
There is no single rule to properly define a quantum circuit.
There's a different function to check each characteristic of the image:

- Unite
- Text
- Geometry
- Density
- Captions

The images that are qualified through various checks are recognized as valid quantum circuits. The process improves the data and reduces errors.

6) Example Classification Outcome: To show the effectiveness of the filtering pipeline:

- **Figure 3** shows a valid quantum circuit diagram that was successfully accepted by the pipeline. It contains parallel qubit lines, gate-like boxes, and a structured layout.
- **Figure 4** shows a scientific plot with curves and axes, which was correctly rejected by the pipeline as a non-circuit figure.

This shows the benefit of combining geometry-based heuristics, OCR keyword checks, and caption analysis to distinguish actual circuit diagrams from unrelated figures such as performance plots, heatmaps, or charts.

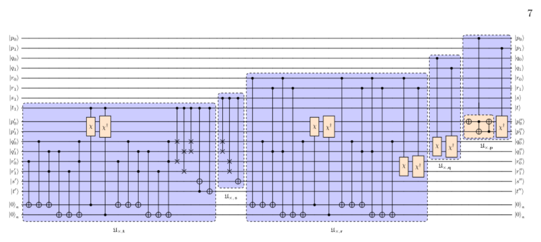

Fig. 3. Accepted quantum circuit diagram. The structure contains horizontal qubit wires, rectangular gates, and consistent layout typical of quantum circuits.

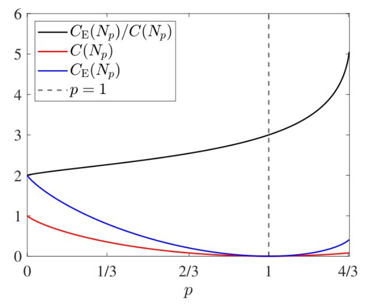

Fig. 4. Rejected scientific plot. This is a typical performance curve with axes, labels, and a high curve ratio, which triggered rejection.

E. *Saving Valid Circuit Images*

Once the image has cleared the filtering checks and has been declared a valid quantum circuit image, the image is then stored as a PNG file in the `images_5/` directory.

This applies in both cases of diagrams:

- Raster Images (Existing Image Files)
- Vector diagrams from converted high-resolution pages of PDFs

Each saved image will be given a unique filename corresponding to:

`image_0001.png,image_0002.png,image_0003.png,`

This naming scheme guarantees that:

- No overwriting of files
- Matching easily with associated metadata later in the pipeline.

Why this step matters?
By saving only the valid circuit images, it ensures that the dataset is kept clean and relevant. Further, it facilitates the subsequent processes of extracting metadata and annotation, as it has already been determined that these circuits are probably valid ones.

If the image does not pass the tests (for example, if it appears to be a plot or a text-heavy figure), the image is not saved, and the reason may be recorded for diagnostic purposes.

The difference between valid and rejected figures thus ensures that only the highest quality examples are selected in the final data pool.

F. *Metadata Extraction*

After an image is accepted as a valid quantum circuit and saved, the next step is to collect extra information about it. This is called metadata — data about the image that helps us understand what it represents. Metadata improves the quality and usefulness of the dataset. For each image, we try to answer questions like:

- Where in the paper was this image found?
- What gates are shown in the circuit?
- What is the purpose of this circuit?
- Can we map text labels to specific areas in the image? All this information is collected through the following substeps.

1) Caption and Description Extraction: In scientific research articles, each figure is always accompanied by a caption that describes the content. For example:

> Figure 4: Quantum circuit for implementing the quantum Fourier transform on 3 qubits.

This caption provides important information such as the name of the algorithm, names of the gates, and even the circuit structure.
We identify captions by scanning text regions around the image based on the page layout, which is provided through its PDF format. Once a figure caption is identified, it is then stored in a list of text lines.

Such descriptions are employed subsequently for:

- Gate detection (if it involves the Hadamard gate in the caption)
- Detection of the problem (e.g., if the caption reads “VQE circuit”)
- Label filtering (to exclude figures that contain terms such as “fidelity curve”)

Without a caption, while it will still be accepted if it passed previous tests, there will be less information available about that particular photograph.

2) Gate Type Detection: Quantum circuits consist of gates like:

- H (Hadamard)
- X, Y, Z (Pauli gates)
- CX (Controlled-X)
- T, S (phase gates)

To identify the gates in each picture, we employ two approaches:

a) OCR (Optical Character Recognition)::

- The image is then processed by `pytesseract`, which is a tool for reading text from images.
- The text is examined to find recognized gate identifiers such as “H”, “CX”, or “RZ”.

b) Caption Analysis::

- The extracted caption is scanned for gate-related keywords, such as “Hadamard” or “CNOT.”
- If detected, they are then added to the list of detected gates.

The combined result is stored as a list of gate names (e.g., `["H", "CX", "RZ"]`) within the metadata of the image. This helps in understanding the complexity levels of the circuits.

3) Quantum Problem: In addition to knowing what the gates are, we also attempt to discern what the circuit is doing, or what quantum algorithm/problem it represents.

for getting this competitive value requires scanning the caption for keywords related to common algorithms or topics in the field of quantum computation, such as: AOA, Adiabatic.

This is done by scanning the caption for keywords related to common quantum algorithms or problems, such as:

Keyword	Inferred Problem
QFT	Quantum Fourier Transform
VQE	Variational Quantum Eigensolver
Teleportation	Quantum Teleportation
Grover	Grover’s Search Algorithm
QAOA	Quantum Approximate Optimization

4) Text Position: When OCR software detects the labels on the gates inside the image containing the circuit, the software not only displays the text but also the position where the words or symbols appear inside the image.

For instance:

```
[
  { "word": "H", "x": 85, "y": ... },
  { "word": "CX", "x": 130, "y": ... }
]
```

We store such text positions for every image. This is useful because:

- It enables future models to learn where the labels are in relation to the wires and the boxes.
- It supports features such as gate layout recovery.
- It assists with the task of training image-to-text models that describe image parts.

Everything that can be considered metadata – gates, problem, caption, and position – is packaged together in a structured dictionary in Python and saved as `dataset_5.json`.

G. Dataset Construction (JSON and CSV)

After acquiring the images of the quantum circuits successfully, the final step is to compile all the relevant info into structured files. The structured files are actually the final dataset that can be used for further research purposes, training machine learning models, or for the performance analysis of the image-to-text models in the field of quantum computing.

During this process, two types of files are produced:

- A JSON file that holds information about the metadata of each circuit image
- The CSV file containing an outline of the number of circuits per paper.

1) dataset_5.json: Full Dataset along with Metadata: This is the main output file, in which all information concerning each circuit image is contained. For each valid circuit, there is a dictionary in a standard Python format, consisting of:

- `image_filename`: Image filename (for example, ‘image_0034.png‘)
- `arxiv_id`: The arXiv ID of the paper (e.g., 2401.98765)
- `page_number`: The page in the PDF where the circuit was found
- `figure_number`: The detected figure label (for example, “Figure 2.1” or “Fig.3”)
- `gates`: a list of the gate labels in the image determined by OCR, such as ‘H‘, ‘CX‘, ‘RX‘
- `problem`: If possible, the name of the quantum algorithm from the circuit (for example, “QAOA”, “VQE”)
- `descriptions`: Sentences about the circuit that could be found in the paper – often extracted from captions. This can be the only description that can be relied on.
- `positions`: This list contains dictionaries that hold the position (x and y values) of the detected labels

An example entry in the JSON file looks as follows:

```
{
  "image_filename": "image_0034.png",
  "arxiv_id": "2401.98765",
  "page_number": 6,
  "figure_number": "3",
  "gates": ["H", "CX", "RY", "MEASURE"],
  "problem": "Quantum Teleportation",
  "descriptions": [
    "This figure illustrates the
    quantum teleportation
    protocol.",
    "The circuit begins with a
    Hadamard gate followed by
    an entangling CX operation."
  ],
  "positions": [
    { "word": "H", "x": 95, "y": 210 },
    { "word": "CX", "x": 142, "y": 210 },
    { "word": "MEASURE", "x": 200, "y": 220 }
  ]
}
```

This file will continue to grow as the pipeline reads through increasingly more papers. This file will be machine-readable, as well as human-readable, meaning that it can easily be used to train or test a model, or reviewed by hand.

2) paper_list_counts_5.csv: Although the JSON file contains extensive information, the CSV file offers a snapshot view on the number of valid circuits that have been successfully retrieved for each paper.

A CSV file has two columns for each row in it:

- **arxiv_id**. It uniquely identifies the paper (e.g., 2307)
- **circuit_count**: Number of valid circuit images saved from that paper.

example what the file looks like:

```
arxiv_id,circuit_count
2301.23456,0
2304.45678,2
2307.98765,5
```

This file is particularly useful for:

- How productive were the papers?
- Researching whether some papers were missed or contained no viable circuit
- Checking the pipeline quickly to make sure it's operating as expected

In these cases, if a paper was indeed processed, but there are no images from the circuits, that paper will still show up in the CSV file, but with a count of 0.

H. Quality Checks and Manual Validation

Even though the creation of this dataset was completely automated, the outputs needed to be checked for quality. To ensure that the extracted and saved images are indeed valid quantum circuit diagrams, we performed some manual inspection steps after running this pipeline.

a) Why Quality Checks Were Necessary::

- Some figures may pass the filters by accident: for example, charts or tables that happen to resemble circuits.
- Incorrect captioning or the inability of OCR techniques may result in incorrectly cropped or missed valid circuits.
- Some values may be missing or wrong - this is because the original paper had inconsistent formatting that prevented accurate metadata extraction.

b) Steps done to validate::

- Manually reviewed the images in the folder `images_5/`.
- Cross-validated the image with its associated metadata in `dataset_5.json`.
- Identified if there were any recurring patterns of failure, for example, plots accepted, circuits missed etc.
- Verified gate detection and the identification of quantum problems were working as expected.

c) Adjustments Made After Checking::

- Filter thresholds tuned by filtering (for example, image density or word count in OCR).
- Improved text and caption extraction features.
- Changed classification rules in order to make them a bit more flexible or stricter according to findings.

This manual validation ensured that this final dataset was cleaned, useful, and reliable for further downstream tasks like training or evaluating image-to-text models.

IV. Results and Discussion

A. Dataset Summary

After having completed the entire process of extraction and filtering:

- **Total number of papers processed:** The system then proceeded to scan a list of papers on quantum physics, which was predetermined, supplied by arXiv, and was read from the file `` `paper_list_5.txt'' ``.
- **Number of valid circuit images found:** Of all the pages and figures that were scanned through, only those images that were able to pass all filters successfully were extracted. In this case, the total number of valid quantum circuit diagrams obtained (you will enter the number that you got from `dataset_5.json`) is **N**.
- **Images distribution among papers:** Some papers had many figures to use, while some had no figures at all. A file named `paper_list_counts_5.csv` assists in noting the most active papers.
- **Common quantum gates/models or problems tackled:** From the metadata information, one can identify the set of quantum gates (for example, "H", "CX") and quantum algorithms (for example, "QFT", "VQE") that are most commonly used. Thus, it is possible to understand the realm of quantum computing that is most commonly depicted.

B. Examples of Accepted and Rejected Figures

To gain deeper insight on what kind of images passed and failed the filter, we proceeded to manually examine samples of images that passed and failed the filter:

- **Accepted:** Images should clearly display horizontal wires and gates. Also accepted were images with clean layouts. Diagrams are usually made through TikZ.
- **Rejected:** Those that looked like performance graphs, bar graphs, heatmaps, or tables that might have involved lines or boxes (but not circuits) were successfully rejected. These usually involved keywords such as 'accuracy' or had axes.

A selection of image examples could be added here; one accepted and one rejected, with brief explanations such as "Accepted – good wire layout" and "Rejected – line graph incorrectly identified because of its structure."

C. Analysis of Filtering Performance

The filtering system employed a combination of OCR, Geometry, and heuristics. On the whole, the pipeline worked well, but there were some corner cases that included:

False Positives: A few false positives for non-circuit images passed. These were usually graphs with a circuit-like structure but were flagged too late in the system. For example, graphs with many horizontal lines and designated x/y axes.

False Negatives: There were some true images of circuits rejected, typically for:

- Extremely light or noisy graphs
- OCR text confidence

- Caption not provided or not easily identifiable with the graphic

Heuristic Effectiveness

- The color score assisted in eliminating plots based on their density and color/structure diversity.
- Gate + wire heuristics (small boxes and long lines) performed best with high-quality vector diagrams.
- The key role of text filters through keywords in captions and/or through Optical Character Recognition.

D. *Limitations*

Despite all this, some limitations exist:

a) OCR Quality: Sometimes the poor scanned image quality or the low resolution vector-image renderings resulted in inaccuracies during the OCR process. This hampered the detection of gates and the analysis of captions.

b) Caption Ambiguity: This is a condition that occurred when some of the captions lacked clarity or proximity to images, so that one may not identify a description with a specific picture.

c) Circuit Complexity: In cases where there were complex or dense circuits, with wires or gates crossing one another, there may have been some mistakes in

d) Dataset Bias: The dataset relies upon a closed list of papers. Perhaps the dataset is biased toward certain areas of quantum computing or certain styles of drawing circuit diagrams.

V. Conclusion & Future Work

This task has successfully produced a clean and well-organized dataset containing quantum circuit images and meaningful metadata like gate information, problem types, and text locations. This dataset has been produced solely through the help of an automated system that downloaded research papers containing images from arXiv. It also filtered out images that were not quantum circuit images and finally stored the valid quantum circuit images.

The dataset is very important in overcoming the gap that exists in the field of image to text studies, particularly when it comes to scientific images. The models that were being used were not very effective when dealing with specialized images such as those used in the field of quantum circuits, which are considerably different from the usual photos used in everyday life.

This dataset is also useful in some practical applications, for instance in:

- Training Models to Explain Quantum Circuits in Words This work aims to train
- Quantum gates and layout extraction from images
- Constructing learning aids on the subject of quantum computing using visuals.

Future Work and Ideas for Improvement

Although the rule-based system performed satisfactorily, in several ways, this project can still be extended or enhanced by:

- **Deep Learning Classification:** Rather than being limited to predefined rules, the next step would be to train a deep learning network to classify circuit images versus non-circuit images. This would enable the computer to identify patterns based on more than geometry or keywords.
- **Better OCR/LA/Text Extraction:** The existing OCR playes a good complementary role in the search. This can break down at lower resolution images or images with difficult-to-read text. An improved model in this area will contribute towards better metatada and lower false negatives.
- **Construction of a Larger Dataset:** In the study, the number of papers that could be considered would be limited to those indexed at arXiv. However, when proper computing power is applied, one would be able to generate a larger dataset of thousands of papers.
- **Metadata Enhancement:** Incorporation of more detailed information such as the quantum circuit depth and the connectivity patterns may be done in the future.

In conclusion, it was found that it is indeed possible to produce a high-quality scientific data set using an entirely automated process.This is a strong step toward supporting more intelligent systems that can understand and describe quantum computing concepts from visual input.

Acknowledgment

I would like to express my sincere gratitude to Prof. Dr. Patrick Levi for his valuable guidance, support, and feedback throughout the course and this project. His insights into natural language processing and scientific data understanding were instrumental in shaping the direction and quality of this work.

References

[1] Cornell University, *arXiv.org e-Print archive*, Available: https://arxiv.org
[2] PyMuPDF Documentation, *MuPDF Python bindings (fitz)*, Available: https://pymupdf.readthedocs.io/
[3] Pillow Documentation, *Python Imaging Library (PIL)*, Available: https://pillow.readthedocs.io/
[4] Tesseract OCR Engine, *Tesseract Open Source OCR*, Available: https://github.com/tesseract-ocr/tesseract
[5] OpenCV, *Open Source Computer Vision Library*, Available: https://opencv.org/
[6] Python Regular Expressions, *re module documentation*, Available: https://docs.python.org/3/library/re.html