Robust Line Detection and Association in Piping & Instrumentation Diagrams

Msc. Thesis Robotics

Xavier Fung-A-Jou

TU Delft · MCDERMOTT · VIK TOR

Robust Line Detection and Association in Piping and Instrumentation Diagrams

Msc. Thesis Robotics

by

Xavier Fung-A-Jou

To obtain the degree of Master of Science in Robotics at the Delft University of Technology.
To be defended publicly on Wednesday, December 20, 2023, at 14:30

Student Number:	4697596	
Project Duration:	May 2023 - December 2023	
Primary supervisor:	Prof. Dr. Ir. J. Hellendoorn	Delft University of Technology
Daily supervisors:	Ir. M. Slootweg	Viktor
	Ir. R. Banotra	McDermott
Graduation Committee:	Dr. Ir. M. Kok	Delft University of Technology
	Ir. N. Ibrahimli	Delft University of Technology
Place:	Faculty of Cognitive Robotics, Delft	

Faculty of Mechanical, Maritime and Materials Engineering (3mE), Delft University of Technology

TU Delft

Abstract

Piping and Instrumentation Diagrams (P&IDs) are graphical representations utilized in chemical engineering plants. Due to confidentiality reasons and legacy drawings, these diagrams are sent in PDF format. Piping engineers need to make a material take-off (MTO), a document containing all the components of a P&ID from these drawings. Today, this is done manually, which proves to be time-consuming and laborious. A piping engineer spends approximately 36 hours per 10 P&IDs, with an average of 500-1000 P&IDs per project. Given the expertise and value of process engineers, this manual counting process incurs substantial costs and a repetitive workload. Consequently, there is a growing motivation to automate this process.

In response, this thesis introduces an innovative deep learning model, PandID-Net, designed specifically for P&IDs. PandID-Net uniquely integrates symbol detection, line detection, and text recognition into a single model, diverging from previous methods that relied on separate models and rule-based techniques. It is the first method that uses deep learning for the line detection task in P&IDs. This all-in-one approach not only simplifies the processing pipeline but also enhances computational efficiency in detecting and pinpointing symbols, lines, and text, as well as their interrelationships.

The optimal configuration of PandID-Net is found by an ablation study where the performance of individual components is tested in isolation. This optimized configuration is then evaluated and benchmarked against a prior study by Paliwal et al. [25] on the same dataset. PandID-Net achieves a performance in F1 scores of 92.89 and 94.48 for line detection and keypoint detection respectively.

Acknowledgements

I extend my thanks to all who have supported me during the course of this project. Special appreciation is due to my supervisors, Hans Hellendoorn, Marcel Slootweg, and Richa Banotra, for their guidance and constructive discussions which were highly beneficial and appreciated.

I am also thankful to the companies Viktor and McDermott Inc. for providing the necessary resources for this research. The mental support of my colleagues at Viktor has been invaluable.

Additionally, I would like to acknowledge my family, friends, and housemates for their support, with particular mention of my sister, Zola, for her encouragement and advice.

Xavier Fung-A-Jou
December 12, 2023

Contents

List of Figures

List of Tables

1 Introduction

1.1. Background

In the landscape of modern engineering, the drive for automation using Artificial Intelligence (AI), has become a necessity rather than just a trend. This is particularly evident in the engineering, procurement, and construction (EPC) sector, where companies of all sizes are actively integrating AI to stay competitive. EPC companies are either leveraging off-the-shelf AI solutions or collaborating with external entities and hiring skilled resources like data scientists and software developers to create tailor-made AI applications. These efforts are aimed at addressing the existing gap in deploying AI-based tools on live projects at an industrial scale.

Engineering drawings hold a central position in the EPC sector and various other engineering domains. Consider the Piping and Instrumentation Diagrams (P&IDs), which are standard in process plant designs. These diagrams provide a detailed visual representation of the connections between various process equipment and their associated instrumentation controls. These digital diagrams are composed of symbols representing different components, such as equipment, piping, and instrumentation. These symbols are interconnected by lines that depict either flow or signals, and the diagrams are often supplemented with additional features like flow indicators, titles, and tables. An excerpt of a P&ID can be found in Figure 1.1.

The P&IDs are used to create a material take-off document (MTO). This is an inventory list containing piping equipment, their quantities, sizes, and other relevant information. MTOs play a crucial role in budgeting, as they inform cost estimation processes, and ultimately, they function as the definitive procurement list for equipment. Given their significance, accuracy in MTOs is paramount since errors can lead to significant financial repercussions. An example of an MTO can be seen in Figure 1.2

In their digital form, P&IDs are structured in such a way that they can be easily processed by computer systems. The digital files contain metadata that can be used to create an MTO automatically almost without error. However, a challenge presents itself. A significant number of P&IDs are rendered in image formats. This choice often stems from concerns about intellectual property or specific contractual terms for new facilities. Plants with a long operational history might also predominantly use image-based P&IDs, with some only having paper copies available. Transitioning from these image formats to digital ones requires a methodical approach to identify objects, extract vital information, and then restructure the entire diagram for computational use. Currently, this transition largely hinges on manual intervention by domain experts. This method is not only time-intensive but also susceptible to errors and inconsistencies based on individual expertise. The integration of AI solutions in this realm holds immense potential, promising a blend of efficiency, precision, and standardization in the digitization process.

Nonetheless, the task is not without its complexities. Many drawings are dense with symbols, with only subtle differences between them, making them challenging to process. Moreover, there exists a class imbalance challenge where certain symbols overwhelm the data, and others are scarcely present. There are differences in how the symbols are presented between different projects and legacy drawings can contain noise due to being scanned.

Many organizations are now realizing the transformative potential of machine learning (ML) and

deep learning (DL) tools. Unlike their rule-based predecessors designed for specific P&ID formats, ML and DL systems bring a nuanced approach. They have the ability to learn and adapt, which can lead to enhanced accuracy and operational efficiencies. A particularly promising application of these technologies is in automating aspects of engineering that were previously labor-intensive.

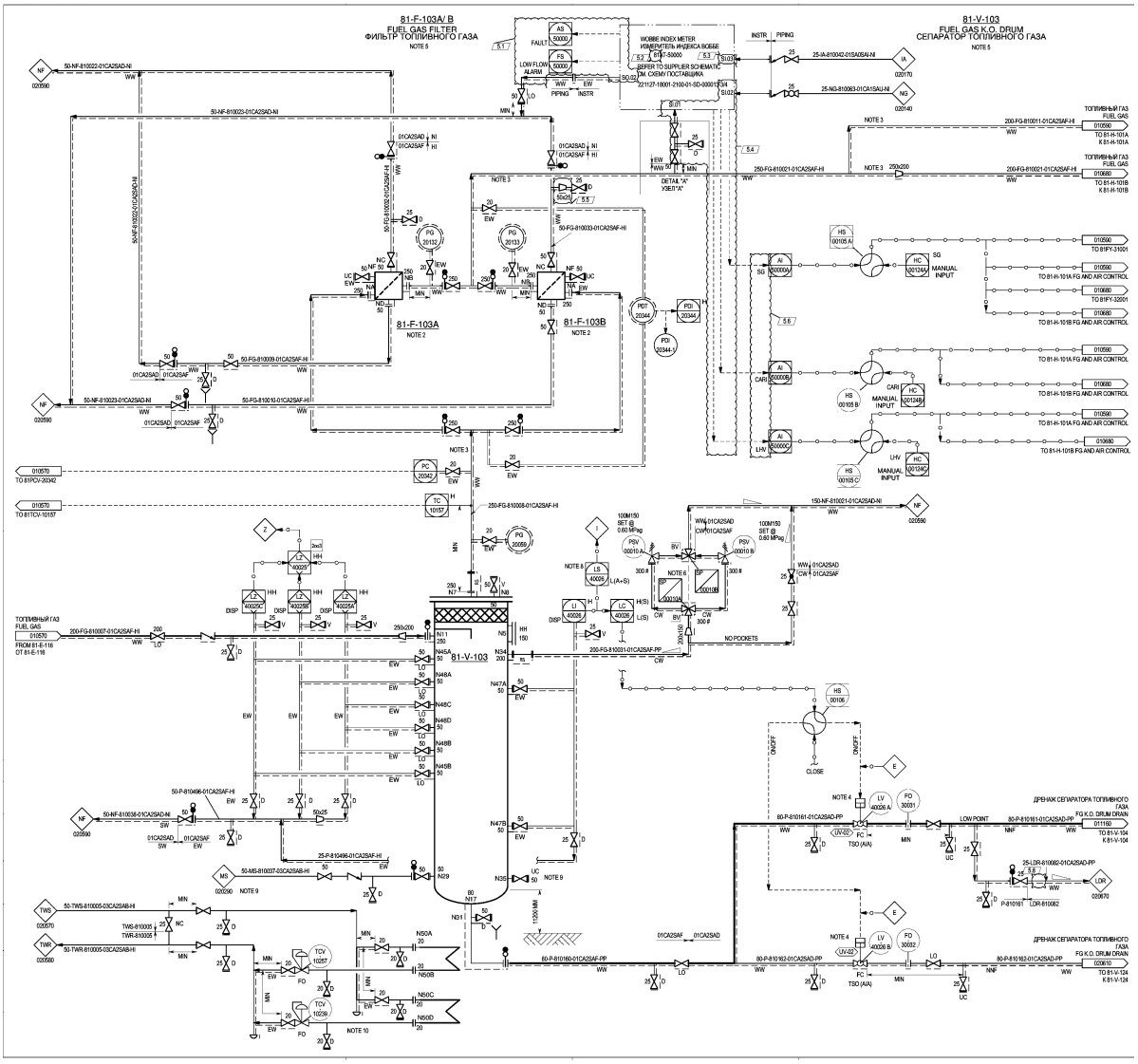

Figure 1.1: Excerpt of a P&ID

1.2. Problem statement

P&IDs are commonly used in process plants. An example is shown in Figure 1.1. Due to the aging of plants and confidentiality regulations, P&IDs are frequently archived as image files, presenting challenges in extracting and analyzing their content. An MTO is a comprehensive spreadsheet capturing crucial information about piping components. This includes details such as part numbers, part sizes, and the associated pipe numbers. They are an essential part of the project estimation process. An example is depicted in Figure 1.2. MTOs are currently constructed manually by process engineers and this is a tedious task. All the piping components need to be counted and associated with the correct pipe diameter and line number. This is done by looking visually at the P&IDs and following the pipelines. Then all the relevant information needs to be written down in the MTO. During the course of a project, many revisions of the P&IDs are made. This causes the MTOs to also need to be revised. The number of drawings per project falls in the range between 500 to 1000 [5] and currently it takes approximately

36 hours to create MTOs for 10 P&IDs. Due to the immense scale of the project, the additional cost each error causes is significant. Therefore, any error mitigated represents a valuable improvement.

PROJECT NR	PID NR	LINE NUMBER	LINE NR	SHORT CODE	HEADER SIZE	QUANTITY
210192	V-71079	2"-PR-1A3A-711503/0-NI	711503	FLG	2	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	FLG	2	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	VC	2	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	BRA	10	2
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	BRA	10	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	INS	10	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	BRA	10	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	RE	12	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	FLG	12	1
210192	V-71079	10"-PR-2A5B-711343/0-NI	711343	FEF	12	1
210192	V-71079	10"-PR-2A5B-711497/1-LST	711497	BRA	10	1
210192	V-71079	10"-PR-2A5B-711497/1-LST	711497	FLG	2	1
210192	V-71079	10"-PR-2A5B-711497/1-LST	711497	MTW	2	1

Figure 1.2: Example Material Take-Off

Industries are increasingly embracing AI technologies to leverage the automation of repetitive tasks to their advantage. Automating the task of constructing an MTO can reduce the cost and error margin of the MTO creation process. That is why McDermott Inc. came up with the following task:

> *Develop a tool that will automate the construction of MTOs. The tool will combine text recognition, symbol recognition, and line recognition to extract crucial information from P&ID images, such as part numbers, sizes, and pipe numbers.*

The principal aim of this research is to create and implement an AI-based P&ID recognition model to streamline the creation of MTOs, enhancing the efficiency of engineers and reducing costs. The P&ID is usually in PDF format lacking embedded metadata. To create an MTO, the method will need to extract information about text, symbols, and lines from the P&ID images, such as part numbers, sizes, and pipe numbers. After extraction, it will need to associate this data to reconstruct the relationships and connections depicted in the diagram. The task of automatically creating an MTO is typically divided into three distinct processes: symbol recognition, text detection, and line detection. Each of these processes is self-contained, and their results are combined with logic to construct the MTO. A more in-depth explanation can be found in Section 3.1. This study, in particular, focuses on the aspect of line detection within this pipeline.

To deploy the model McDermott has set a minimum for F1-score of 70% for the symbol detection. This target is based on two factors. Firstly, the model is an aid to engineers, not a replacement; it will assist in producing initial MTOs for projects, which will subsequently undergo multiple reviews and corrections by engineers before being finalized for clients, mirroring the current practice with manually produced MTOs. Secondly, recognizing that human error is a natural part of the manual MTO generation process, McDermott's estimation department traditionally incorporates an error margin in their MTO usage. Therefore, applying a similar error margin to the model's output aligns with existing practices and acknowledges the model's role as a supportive tool rather than a standalone solution.

1.3. Research Plan

The goal of this research is to improve line detection in P&IDs. Most of the research previously done on line detection in P&IDs was aimed at improving the performance and robustness of line detection techniques the solution was to develop ever more complicated programs to keep up with the many conventions and exceptions. For that reason, several research papers propose the utilization of deep learning techniques [31, 21, 20, 14] for line detection. However, the challenge lies in the scarcity of publicly accessible annotated datasets and standardized labels, which hinders the implementation of Deep Learning techniques [14]. Because both paths have their own limitations, this research will first

look into the possibilities to improve the rule-based methods, whereafter it will explore the options for deep learning methods.

The main research question for the thesis derived from the task assigned by McDermott is:

In what ways can the processing pipeline effectively be improved for line detection and association in Piping and Instrumentation Diagrams?

To answer this question adequately the following four sub-questions are constructed:

1. *What are the limitations and challenges in the existing line detection and association process?*
2. *In what ways can deep learning techniques be effectively applied to line detection and association in Piping and Instrumentation Diagrams?*
3. *What metrics can be used to evaluate the effectiveness of the improved processing pipeline for line detection and association?*
4. *Could the proposed enhancements to the processing pipeline be applied to improve diagram interpretation across different industries? If so, how?*

1.4. Contributions

The primary contribution of this thesis is the introduction of an innovative deep learning model designed for P&ID images called PandID-Net. The model is capable of detecting and pinpointing symbols, lines, and text while also determining their interrelationships and class. To the best of my understanding, this represents the first application of deep learning specifically for line detection in P&IDs.

Contrary to previous methods PandID-Net is an all-in-one model that will combine line recognition, symbol recognition, and text detection into one model without using rule-based techniques to combine separate models. This methodology illustrates the model's capability in simultaneously addressing two primary challenges: the detection of symbols and lines, along with their associated relationships. This also increases computational efficiency and allows for multitask learning.

Moreover, this approach demonstrates superior scalability in comparison to conventional rule-based line detection algorithms, because the model can be trained on project-specific data. The model is capable of learning a range of line types with the flexibility to expand its range according to the requirements of the final objective.

1.5. Outline

Following this introduction, Chapter 2 delves into related works, discussing both rule-based line detection methods for P&IDs and machine learning line detection methods such as wireframe parsers, including the Line CNN, F-clip, HAWP, and HT-HAWP. In Chapter 3, a general pipeline is explained and a rule-based method is explored. The rule-based method gets discontinued and Chapter 4 explains the workings of our PandID-Net. Furthermore, Chapter 5 will describe the test setup and chosen hyperparameters. Chapter 6 is dedicated to the experiments conducted and their results. Here, details of the dataset, evaluation metrics, and details of each experiment are provided, culminating in a synthesis of the results obtained including a comparison with previous work. Lastly, Chapter 7 offers a discussion on the implications of the findings, concluding remarks about the research, and answers to the research questions are given. Followed by recommendations for future endeavors in this domain.

2

Related Work

In this Chapter, the related work is presented. This includes methods that are used in the digitization of P&IDs such as line detection in Section 2.1. There is extra emphasis on the work of Paliwal et al. [25] and Moon et al. [21] in Sections 2.1.2 and 2.1.1 because of their inspiration for this work. An introduction to deep learning techniques is given in Section 2.2. The field of research that uses DL for line detection in images called wireframe parsers is covered in Section 2.3.

2.1. Rule-based line detection in P&IDs

Extensive research has been conducted on symbol detection in P&IDs, with a more limited focus on line detection. The existing methods for line detection are all rule-based. Implying that they don't utilize learned models to construct connections between the different types of components in the diagram. This section provides a summary of the techniques that have been previously explored. The described methods are listed in Table 2.1.

Fatasumata et al. [7] introduced a pipeline for line detection in P&IDs in 1990. It included vectorization of the image followed by classification based on height and width to differentiate text characters, line candidates, and symbol candidates. The paper was one of the first on this subject and in only four pages it tried to cover text, symbol, and line recognition.

Arroyo et al. [1] employed a raster-based document analysis for P&IDs and used a filter method for line detection, with an emphasis on a common pipeline as shown in Equation 2.1. In succeeding papers, this processing pipeline is often used as a framework.

$$\textit{preprocessing} \rightarrow \textit{edge detection} \rightarrow \textit{line detection} \rightarrow \textit{association} \tag{2.1}$$

Moreno-Garcia et al. [22] implemented a four-step process that involved preprocessing, image resizing, detection of representative shapes, and text/graphics segmentation. Key techniques included thresholding, noise removal, and Canny edge detection [3]. The philosophy of the method is to remove all objects so the lines are the last thing that remains.

Kang et al. [12] utilized a sliding window approach for line detection, initiating from recognized symbols and following the direction of the line.

Another method is the pixel-wise traversal approach proposed by Yu et al. [31]. The line detection involves realignment, border removal, thickness compression, and line merging based on a predominant line thickness. However, this technique demonstrated limitations in accurately detecting diagonal and dashed lines.

Rahul et al. [26] applied the probabilistic Hough transform [10] for line detection, further refined with line-thinning techniques, specifically using a skeletonization method.

Mani et al. [19] innovatively used depth-first search (DFS) for line association in P&IDs, representing the image as a graph with each pixel as a node. The paths are along the lines and the lines are detected while the association is made. The solution does not seem to work on dashed lines or line crossings.

The first paper of Moon et al. [20] combined image processing with object detection, employing techniques like binarization, outer border removal, line thinning (Zhang-Suen [33]), pixel processing, and the Hough transform [10].

Paliwal [25] introduced a kernel-based ”Hit-or-miss” algorithm for line detection in P&IDs, designed to handle noisy data. Dashed lines were detected based on consistent gaps and then merged using the DBSCAN algorithm [6].

Stinner et al. [29] adopted a classical approach, converting the image to binary and detecting lines using the Hough transform [10], also identifying three types of line crossings.

Kim et al. [14] further developed a hybrid line recognition combining neural networks with image processing. Techniques used included a pixel-unit traversal method for horizontal and vertical lines and the Hough transform for diagonal lines.

The latest work by Moon et al. [21] focused on enhancing continuous line detection, employing a differential filter for horizontal/vertical lines and a Prewitt filter with the Hough transform for diagonal lines.

Moon et al. [20, 21] conducted the most research on the subject of line detection in P&IDs. In their latest research, they took several different lines into account such as continuous, diagonal, and dashed. Also, they used a detection model for line signs to give more context to the line-like class and flow direction. For these reasons, their approach currently represents the state-of-the-art and provides a starting point for further study. A more in-depth description of their method is written in Section 2.1.1.

Source	Year	Contents
Fatasumata [7]	1990	Development of an Automatic Recognition System for Plant Diagrams
Arroyo [1]	2016	Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams
Moreno-Garcia [22]	2017	Heuristics-Based Detection to Improve Text/Graphics Segmentation in Complex Engineering Drawings
Kang [12]	2019	A Digitization and Conversion Tool for Imaged Drawings to Intelligent Piping and Instrumentation Diagrams (P&ID)
Yu [31]	2019	Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network
Moreno-Garcia [23]	2019	New trends on digitisation of complex engineering drawings
Rahul [26]	2019	Automatic Information Extraction from Piping and Instrumentation Diagrams
Mani [19]	2020	Automatic Digitization of Engineering Diagrams using Deep Learning and Graph Search
Moon [20]	2021	Deep Learning-Based Method to Recognize Line Objects and Flow Arrows from Image-Format Piping and Instrumentation Diagrams for Digitization
Paliwal [25]	2021	Digitize-PID: Automatic Digitization of Piping and Instrumentation Diagrams
Stinner [29]	2021	Automatic digital twin data model generation of building energy systems from piping and instrumentation diagrams
Kim [14]	2022	End-to-end digitization of image format piping and instrumentation diagrams at an industrially applicable level
Moon [21]	2023	Extraction of line objects from piping and instrumentation diagrams using an improved continuous line detection algorithm

Table 2.1: Papers used in comparison

2.1.1. Analysis of Moon et al.’s Methodology

In the paper by Moon et al. [21], the methodology for extracting line objects from P&IDs using an improved continuous line detection algorithm involves several key processes.

The process begins with preprocessing, which includes transforming original P&ID images into binary format to simplify the images and facilitate easier processing. This step is followed by the removal of the diagram’s outline borders and heading areas, focusing the line detection on the actual content

of the P&IDs and avoiding misinterpretation of non-relevant lines or text.

The line detection process involves two main components: the detection of special signs and flow arrows, and the detection of continuous lines. Special signs and flow arrows are detected using a Deep Neural Network (DNN), specifically RetinaNet [17], which is adept at identifying these elements separately from lines. The detection of continuous lines, which represent the flow paths and connections in P&IDs, is performed using a combination of differential filters and the Hough Transform technique [10]. First-order differential filters are used for edge detection, with basic differential filters applied for vertical and horizontal lines and Prewitt filters for diagonal lines. The Hough Transform [10] is then applied to these detected edges to find lines. A unique approach is adopted for diagonal lines, where a segmented approach is implemented to improve the detection of short diagonal lines, which are commonly missed in traditional line detection methods. Postprocessing involves the integration of the information from the earlier detection of flow arrows and special signs. Detected line signs are used to modify continuous lines into different types as required, ensuring the final output accurately represents the P&ID, including both lines and other crucial elements like flow directions. The line information is merged with flow arrow detection to produce a cohesive and accurate representation of the P&ID diagram.

This method significantly improves the accuracy and speed of line extraction in P&IDs, addressing the challenges of previous methods, especially in detecting diagonal and short lines.

2.1.2. Analysis of 'Digitize-PID' by Paliwal et al.

The paper by Paliwal et al. [25] provides a solution for converting scanned P&ID diagrams into digital formats. It automates the recognition and association of lines, symbols, and text on diagrams, leveraging image processing and deep learning to overcome the challenges of manual digitization. Because of the lack of available data, they created a new comprehensive dataset. This is the same dataset on which PandID-Net is evaluated. The results of the paper are compared with the results of PandID-Net in Section 6.5

Line Detection

The line extraction module employs a morphological approach. To detect lines, a structuring element matrix is used, essentially a filter that defines the smallest recognizable line segment within the digital image's resolution. The method involves erosion and dilation operations, and mathematical morphology techniques that enhance structures within images that correspond to the shape of the structuring element. Erosion strips away pixels not matching the element, while dilation restores the structure to its original size. This helps to emphasize lines while diminishing the influence of noise. The result is a clear delineation of line contours within the P&ID, which are then encapsulated by their convex hull a minimal convex boundary that envelops the line segments. The endpoints of these lines are derived, offering the exact coordinates.

Dashed Line detection zeroes in on the characteristics that distinguish dashed lines, such as consistent segment lengths and gaps. By setting thresholds for these dimensions based on a cluster analysis of the lines, the system can identify and differentiate dashed lines from solid ones. Anomalies in the pattern, such as contiguous gaps, are filtered to maintain consistency. The method then utilizes clustering algorithms like DBSCAN [6] to group line segments that should be connected, effectively reconstructing the dashed lines' continuity.

Symbol Detection

The Basic Shape extraction module addresses the identification of primitive shapes, like circles and rectangles, which comprise part of the symbols in P&IDs. The ubiquitous circle, for example, is extracted using Hough Transforms [10]. By applying this transform across image patches and then aggregating the results, the system ensures that no circular symbols are overlooked.

For Complex Shape extraction, a two-step deep learning strategy is employed to discern the intricacies of more elaborate symbols, which often bear subtle differences. Initially, an FCN-based semantic segmentation model localizes the symbols by categorizing them into broad classes. Following this, the TBMSL-Net network [32], a deep learning model trained for fine-grained symbol recognition, classifies the symbols. The interplay of these networks allows for the distinction of symbols that may otherwise be indistinguishable by more traditional image-processing techniques.

Text Detection
Text Extraction is approached with a two-tier process. Initially, the P&ID image is divided into overlapping patches to ensure that texts at the borders are not missed. These patches are then processed using the Character Region Awareness for Text Detection (CRAFT) network [2]. This neural network is designed to localize text regions, predicting bounding boxes for each text instance. Through the Intersection over Union (IOU) metric, overlapping bounding boxes are amalgamated, providing a more comprehensive text localization. The merging process aims to minimize the occurrence of missing texts. These boxes then direct the extraction of text patches, from which single-lined texts are read using the OCR engine Tesseract [13], known for its proficiency in text recognition.

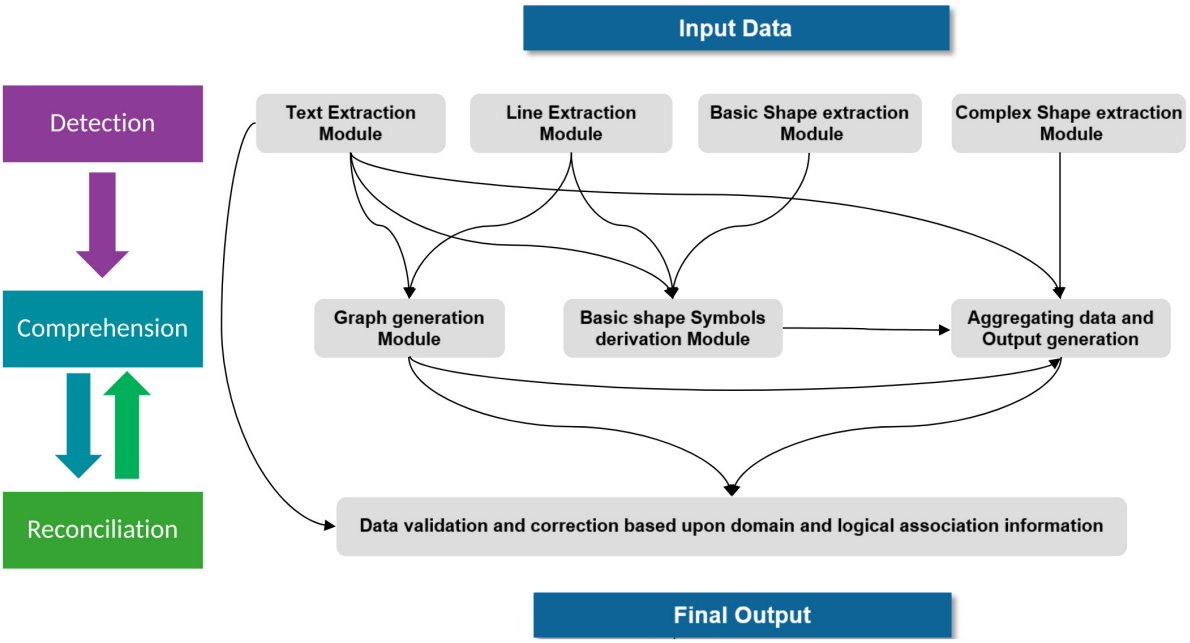

Figure 2.1: An overview of Digitize-PID which consists of 3 sequential modules with their corresponding sub-modules: Detection, Comprehension, and Reconciliation. [25]

2.2. Introduction to Deep Learning Techniques

In this section, deep learning techniques that are used for line detection are described. These techniques are the basis of processing images using Neural Networks (NN).

2.2.1. Introduction to Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized neural architecture specifically crafted for processing data with a grid-like topology, such as images. Derived from three foundational concepts: local receptive fields, shared weights, and spatial sub-sampling, CNNs primarily consist of convolutional, pooling, and fully connected layers. These layers collaboratively enable the network to recognize intricate patterns in input data. [8]

The core operation in a CNN is the convolution, which involves processing an input matrix (like an image) using a filter or kernel. This operation is mathematically defined as:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.2}$$

In this formula, x denotes the input (possibly multidimensional), w is the weight function (or kernel), t is the time index, and a symbolizes the age of measurement. During the learning process, these kernels, which are essentially multidimensional parameter matrices, are iteratively adjusted via backpropagation.

Through the application of local receptive fields, CNNs are capable of detecting elementary features such as edges and contours. As data progresses through the layers, the network learns to recognize

more complex features. The principle of shared weights ensures that kernel parameters remain consistent for all input values within a given layer. This promotes parameter sharing in the CNN, leading to a reduced memory footprint. The blend of sparse connectivity and weight sharing boosts the efficiency of convolutions, allowing CNNs to proficiently identify features across different sections of an image. One significant advantage of CNNs is their parameter efficiency. Due to weight sharing, the network requires fewer parameters, leading to compact models and a reduced risk of overfitting. Additionally, CNNs, with their local receptive fields, focus on small regions of the input, ensuring the extraction of localized features. Furthermore, they can recognize patterns regardless of their spatial position in the input, offering consistent recognition. Lastly, as input data flows through the network layers, features of increasing complexity are learned, providing a multi-level representation.

Given their intrinsic capabilities, CNNs have become indispensable tools in computer vision tasks, ranging from image classification to object detection and beyond [8].

2.2.2. Introduction to Residual Neural Networks (ResNets)

Residual Neural Networks [9], or ResNets, are an innovative type of neural network that address the challenge of training very deep networks. Traditional deep neural networks face issues like vanishing and exploding gradients when adding more layers, which make them difficult to train and can lead to poorer performance than shallower networks.

When considering a deep neural network with L layers, one might assume that the network's performance would continuously improve as more layers are added. However, in reality, there's a point beyond which adding more layers can cause the network to perform worse, which is surprising and goes against the initial hypothesis that deeper is better.

To tackle this problem, ResNets were developed with a key feature: the use of skip connections that jump over some layers. Unlike traditional layers that attempt to learn the output $H(x)$ from the input x, a ResNet layer aims to learn the difference between the output and the input, known as the residual:

$$F(x) = H(x) - x$$

This allows us to redefine the desired output as:

$$H(x) = F(x) + x$$

In this equation, $F(x)$ is the result from the network layers, and x is the original input to these layers. The advantage of this approach is that if the best solution is simply the input itself (identity mapping), the network can easily learn to make $F(x)$ very small, so that $H(x)$ becomes approximately equal to x. The architecture is illustrated in Figure 2.2. The ReLU (Rectified Linear Unit) is an activation function that outputs the input if it is positive and zero otherwise. It helps introduce nonlinearities into the model essential for the model to learn complex patterns.

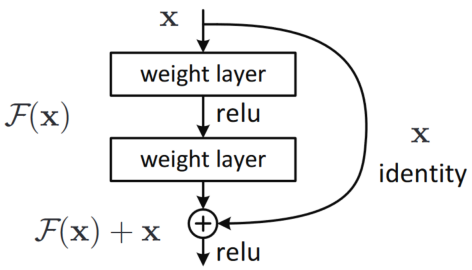

Figure 2.2: Architecture of a residual block [9]

Skip connections have proven to be beneficial by allowing gradients, a measure of how much change is needed in the model's parameters, to flow directly back to earlier layers without being diluted, which solves the vanishing gradient problem. This capability enables the training of networks that are much deeper than was previously possible without a drop in performance. Additionally, these connections can help prevent overfitting, a situation where a model learns the training data too well but does not generalize to new data.

ResNets represent a significant advancement in the field of deep learning, allowing researchers to train networks that are much deeper and perform better on a wide range of tasks.

2.2.3. Hourglass Pose

The hourglass network, introduced by Newell et al. [24] in 2016, is a seminal architecture tailored for human pose estimation tasks, although its applicability extends to other domains. Distinctly characterized by its symmetric, top-down-bottom-up structure, the hourglass module encapsulates the principle of capturing and processing information at various scales, thereby achieving a refined spatial understanding of images.

At a high level, the module's design can be visualized as an hourglass shape: it first downsamples the input through successive pooling or strided convolutions, reducing spatial dimensions while increasing feature depth. As it progresses deeper into the network, the resolution decreases, enabling the capture of more abstract and global contextual information. This process is then mirrored by a series of upsampling operations, restoring the spatial dimensions to their original resolution. Throughout this upscaling phase, lateral connections from corresponding downsampling layers are introduced, ensuring the fusion of high-resolution details with the abstracted features.

Mathematically, if $\mathcal{F}_d$ and $\mathcal{F}_u$ represent the downsampling and upsampling functions respectively, and $\mathbf{x}$ is the input to the module, the output $\mathbf{y}$ can be expressed as:

$$\mathbf{y} = \mathcal{F}_u(\mathcal{F}_d(\mathbf{x})) + \mathbf{x}$$

This design ensures the preservation and integration of multi-scale features, which is crucial for tasks like pose estimation where both global pose context and local joint details are essential. The modularity of the hourglass design allows for stacking multiple such modules sequentially, further enhancing the network's capacity to refine predictions iteratively.

Newell's hourglass network, with its innovative structure, has set benchmark performance in human pose estimation tasks and has paved the way for subsequent research and architectural evolutions in the field

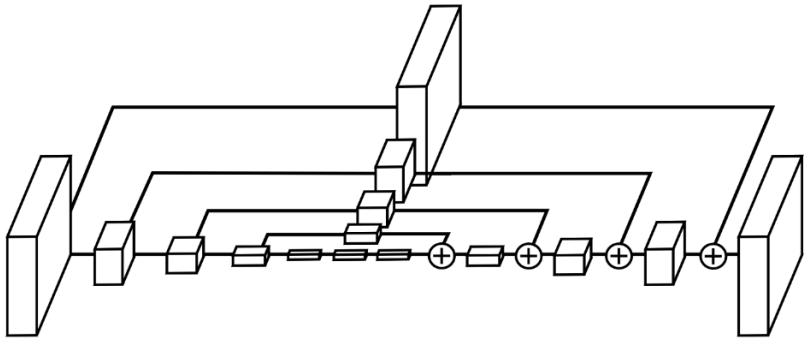

Figure 2.3: Illustration of a single "hourglass" module. Each block corresponds to a residual module. The number of features is consistent across the whole hourglass. [24]

2.3. Wireframe Parsers

This section is about a research field called wireframe parsing. Wireframe parsing is an emerging area of research within the fields of computer vision and graphics, focused on extracting structured wireframe representations from images or 3D data. Such wireframe models typically consist of junctions, edges, and their geometric relationships, capturing the skeletal structure and topological features of an object or scene. This research has numerous applications, from architectural design and urban planning to augmented reality and autonomous vehicle navigation. Traditional methods often involve edge detection and Hough transforms [10], but recent advances leverage deep learning techniques to improve accuracy and efficiency. As wireframe parsing becomes more sophisticated, it promises to bridge the gap between low-level pixel data and high-level geometric understanding, enabling machines to better comprehend and interact with complex visual environments. An example of the result of a wireframe is shown in Figure 2.4

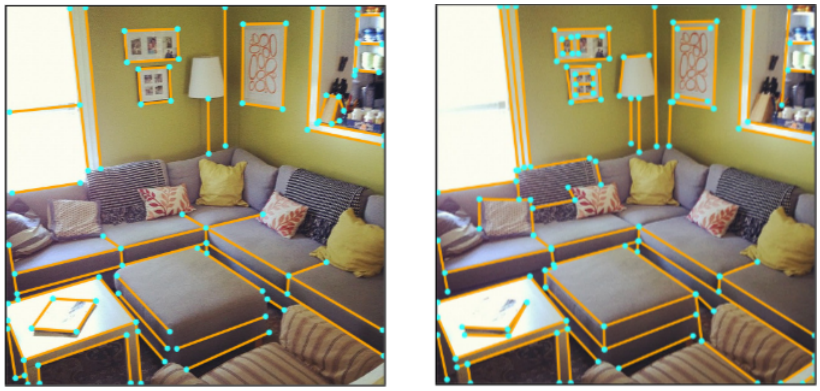

Figure 2.4: The objective of a wireframe parser is to find the lines in an image [34]

2.3.1. Line-CNN

In the 2021 study, ”End-to-End Wireframe Parsing” by Zhou et al. [34], a novel neural network architecture is presented for directly extracting wireframes from images. The core of this architecture consists of two components: the Junction Detection Network (JDN) and the Line Proposal Network (LPN). The architecture is depicted in Figure 2.5.

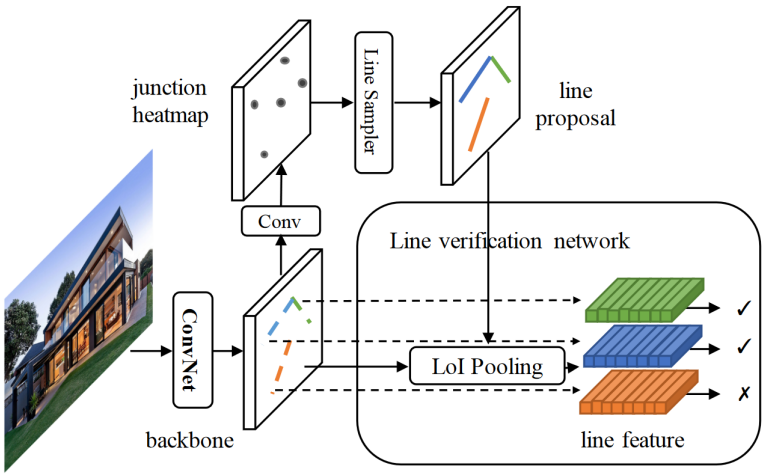

Figure 2.5: Line-CNN architecture [34]

The Junction Detection Network (JDN) takes the first step in the wireframe parsing process. It analyzes the input image to detect potential junction points that may contribute to the structure of a wireframe. The output from this network is a set of candidate junction points that are crucial in the subsequent construction of the wireframe.

Following the identification of junctions, the Line Proposal Network (LPN) comes into play. Its primary role is to propose potential lines by pairing the detected junctions. It starts by estimating the likelihood of a line existing between each pair of junctions, while a clever sampling strategy helps manage the computational burden by focusing only on the most promising pairs.

The LPN also incorporates geometric reasoning to refine these line proposals. It assesses the spatial relationships between junctions and the overall structure of the image. For instance, if two junctions are proximal and align well with an evident linear structure in the image, they are deemed more likely to be connected by a valid line. In contrast, if junctions are far apart or their alignment with visible structures is lacking, the likelihood of forming a valid line decreases.

To further improve the accuracy of line proposals, the LPN integrates deep learning techniques. The network has been trained on a substantial dataset of annotated wireframes, which enables it to rec-

ognize patterns and structures typical to wireframes. This knowledge is instrumental in distinguishing between plausible and implausible line proposals.

Finally, the process includes a post-processing step designed to ensure the output is a coherent and clean wireframe representation. This step filters out any redundant or conflicting lines that might have been proposed, resulting in a wireframe that is both accurate and aesthetically coherent.

By replacing traditional edge detection and Hough transform-based [10] methods with this combination of deep learning and geometric reasoning, Zhou et al.'s method [34] efficiently and accurately generates wireframe proposals.

2.3.2. Advancements in Wireframe Parsing

Wireframe parsing is a rapidly advancing field, with several new approaches improving upon the Line-CNN method. These newer methods, including F-Clip, HAWP, and HT-HAWP, introduce novel strategies to detect and parse lines more accurately and efficiently.

F-Clip [4] simplifies the approach of L-CNN by using a fully convolutional network that directly predicts the central position, length, and orientation of line segments. This streamlined method boasts a significant increase in speed, capable of running at 73 frames per second on a single GPU, making it well-suited for real-time applications

HAWP [30] differs from L-CNN by integrating a unique line segment reparameterization and end-to-end parsing pipeline. It generates a 4-dimensional attraction field map to represent wireframes, treating junctions as attraction basins. This allows for more coherent structure formation and outperforms L-CNN in both accuracy and speed, improving the mean structural average precision (msAP) and offering higher frame rates on the same hardware.

HT-HAWP [18] takes a different tack by embedding a trainable Hough transform [10] within a deep network, providing a blend of classical geometric line priors and learned local features. Leveraging this combination, enhances data efficiency, meaning the network requires less data to learn about line parameterizations. This method has shown effectiveness in line segment detection tasks, confirming the benefit of integrating geometric priors into deep learning frameworks. The workings of the Hough transform block are further explained in Section 5.3.2

2.4. Take-aways

This Chapter started with describing the landscape of line detection in P&IDs. It can be concluded that it has been researched since 1990 and multiple papers are published every year making it a relevant research topic. Noticeably, only rule-based or hybrid methods are explored and there has been no research on deep learning methods for line detection. This leaves a gap since other sub-processes in the digitalization of P&IDs are advancing in the DL domain.

The research field of wireframe parsing has shown that lines can be extracted from far more complex images using deep learning, suggesting promising directions for P&ID line detection. Notably, the approach by Zhou et al. [34], which employs CNNs for junction detection, could potentially be adapted for symbol detection in P&IDs. This makes the L-CNN a fitting starting point for exploring the line detection in P&IDs with NNs.

3

Rule-Based Approach

In this chapter, we'll describe a pipeline we created using rule-based techniques. It starts with an outline of the general process of converting a PDF into an MTO followed by a developed rule-based pipeline. Since we ultimately decided to discontinue this approach, the reasons for this decision are discussed towards the end of the chapter.

3.1. General Pipeline

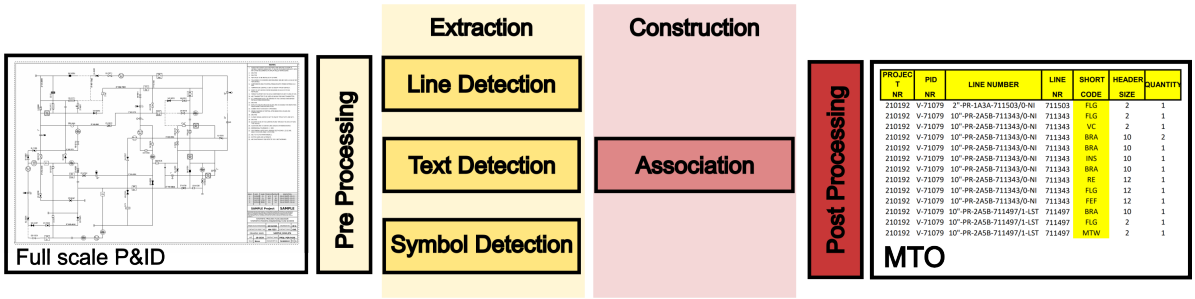

Figure 3.1: This is a schematic overview of a pipeline to construct MTOs from PDF P&IDs

As explained in Chapter 1 this study is about going from a P&ID in PDF format to an MTO. To achieve this information about the text, symbols, and lines is required. This needs to be extracted from the PDF since it has no embedded meta-data. After extracted data is acquired this needs to be associated to reconstruct the relations that are depicted in the P&ID. The steps of the pipeline are shown in Figure 3.1. This pipeline is a generalization of the methods described in Chapter 2.

This study focuses on the extraction and association of the data in the P&IDs. The data extraction can be divided into three different steps. The detection of lines, the detection of symbols, and the detection of text. This can be done in various ways as explained in Chapter 2. The main division can be made between learned methods and rule-based methods. The trend in the last years has been that rule-based methods are being replaced with learned methods. This has been the case for symbol recognition and text recognition but no study has researched the use of deep learning for line detection.

To understand the rule-based methods and the obstacles that they present we explore the usage of the rule-based methods for line detection. The pipeline depicted in Figure 3.2 is developed. It is largely based on the paper of Moon et al. [21] since this is the state of the art.

3.2. Rule-based Pipeline

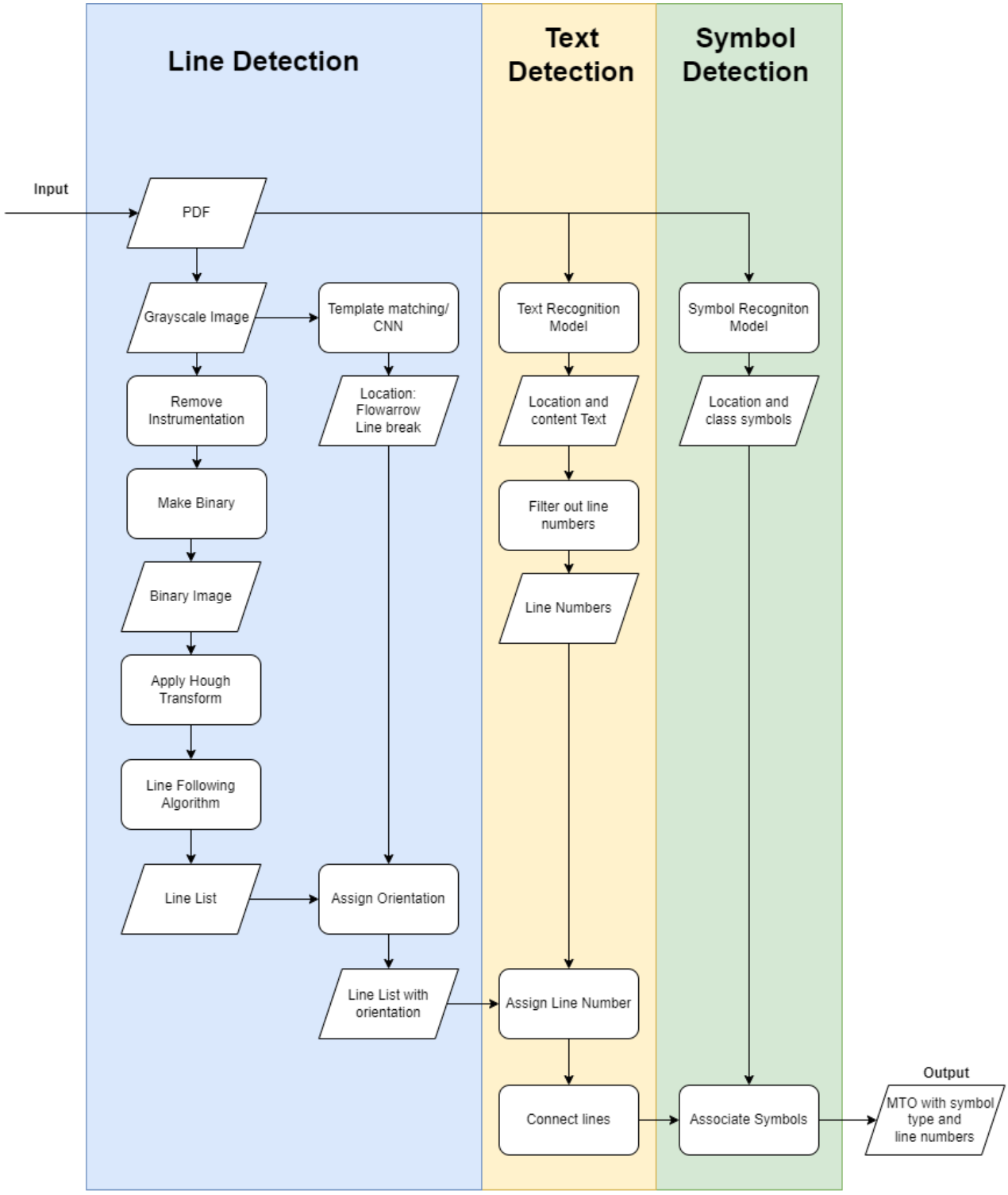

Figure 3.2: Pipeline of the rule-based method

We developed a rule-based method to extract lines from P&IDs to see if current methods could be improved. The pipeline is depicted in Figure 3.2. The starting point was the method of Moon et al. [21] which was already described in Section 2.1.1. This method is created around a technique called filtering, where basic filters (Equation 3.1) are applied on lines to subtract horizontal and vertical lines. This

method solves a problem that occurs with edge detection where every line has two edges and where the detection with Hough lines [10] would recognize double lines. This could be solved with thinning like Zhang-Suen [33] algorithm but that is computationally expensive and thereby time-consuming.

$$\text{filter vertical} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{filter horizontal} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.1}$$

Since the paper of Moon et al. [21] did not provide any code we made an adaptation based on their method. The script is described by the pseudo-code shown in Algorithm 1. This is our interpretation of the method. The edges are detected using the filters whereafter the Hough transform [10] is applied for the horizontal and vertical lines separately. These Hough lines are compared with the original picture. From top to bottom for horizontal lines and left to right for vertical lines, the Hough lines are traced saving the location of the first black pixel it encounters. If the consecutive amount of black pixels is larger than a threshold the line is added to the line set.

Algorithm 1 Continuous Line Detection

```
1:  procedure DetectVerticalHorizontalLines(image)
2:      Detect edges using basic differential mask
3:      Detect edges in the vertical direction
4:      Apply Hough transform to search for straight lines
5:      for each straight line found do
6:          Compare with the original image
7:          Identify starting points where black pixels begin
8:          Identify ending points where black pixels end
9:      Note: Repeat the same method for horizontal continuous lines using horizontal differential
10:     Note: Basic differential mask is effective for vertical and horizontal lines
11:     Note: For diagonal lines, use Prewitt differential mask for edge detection
12: Extract continuous line detection information
```

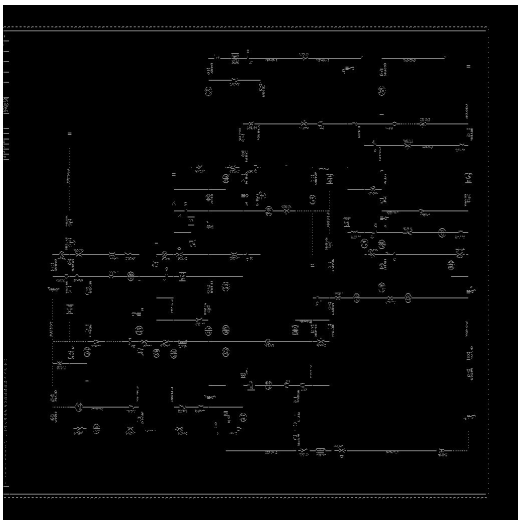

Figure 3.3: Filtered Horizontal Image

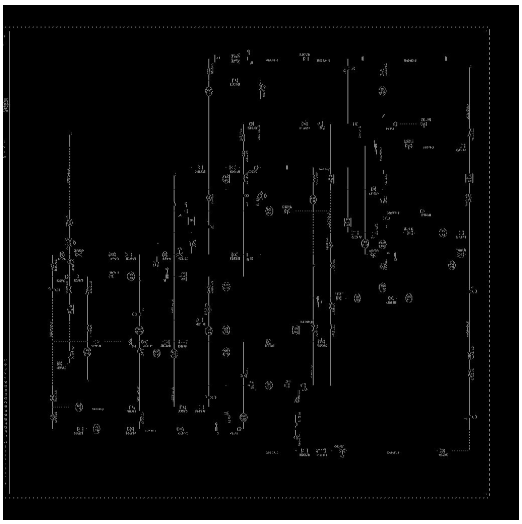

Figure 3.4: Filtered Vertical Image

In Figures 3.3 and 3.4 the results of the horizontal and vertical filtering are shown. Figure 3.6 shows the results of the line detection with the detected line plotted in red in the inverted image. On the left of it is the original P&ID in Figure 3.5.

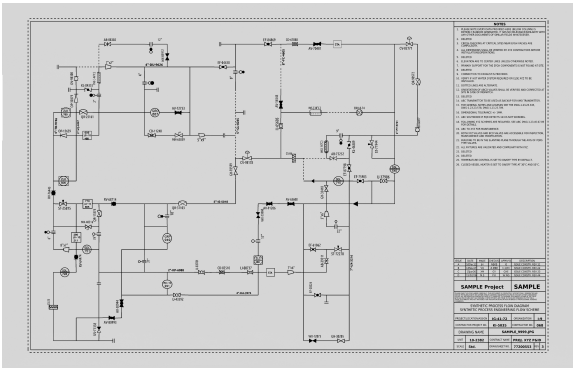

Figure 3.5: Original P&ID

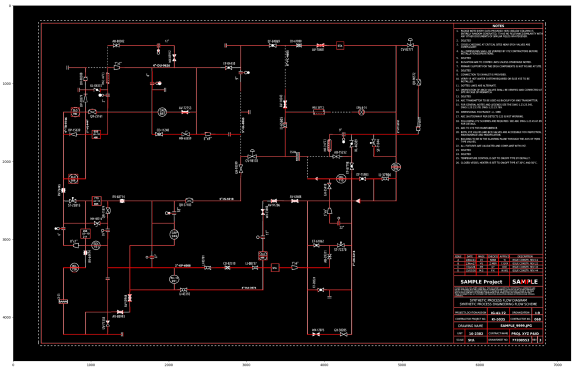

Figure 3.6: The output of the rule-based line detection

After the line segments were found, specific line symbols like flow arrows were detected with template matching. These are assigned to the line using the Hungarian association method [16]. A symbol recognition detection network and a text recognition network are applied. Both were already developed by McDermott. The different components were connected using an extensive set of rules. All the information was then put into the NetworkX format. Every line, flow arrow, text, and symbol was represented by a node, and the relations by edges.

The resulting method was a merger of large existing scripts that did not fit easily together. Because there were multiple networks with distinct detection tasks. The integration of these networks was confined to their outputs, neglecting the potential synergies that might arise from merging intermediate features or results. This approach led to inefficiencies, as there was a redundant learning of similar features across different models.

3.3. Rule-Based Limitations

Rule-based methods require an extensive set of logic to account for the many exceptions. These rules are dependent on the scale, quality, and format of the P&ID. It requires a lot of manual tuning and this needs to be done over for every project. These many exceptions can be learned by a generalized model. Also, the academic value of deep learning methods is higher. There is a growing consensus among the reviewed papers, emphasizing the need for the utilization of Deep Learning techniques in line detection in P&IDs. Notably, several papers explicitly call for further research in this specific application [31, 21, 20, 14]. Collectively, these factors prompted the discontinuation of the rule-based approach and initiated the pursuit of a model capable of integrating subtasks.

4

Machine Learning-Based Approach

In this chapter, a novel learning-based approach is presented. This developed model called PandID-Net is used in the experiments of this study. The first Section 4.1 explains how the pipeline is constructed. After which the main components are explained in Section 4.2. The methods to post-process the results are explained in section 4.3. In the last section, the chapter is summarized.

4.1. Learning-Based Pipeline

Since the rule-based approach did not give the desired results we explore a novel approach using deep learning for line detection in P&IDs. The total pipeline is shown in Figure 4.1. In this updated approach, the tasks of line detection, symbol detection, and text detection are now handled by PandID-Net, a model we have developed. An optical character recognition (OCR) module and some logic are required to construct the actual MTO but this is out of the scope of this research.

The pipeline accepts a high-resolution P&ID image. This image is then transformed into a 1024x1024 pixel image. The image is made binary to remove any noise inherent in the dataset and real-world P&IDs. Utilizing Pandid-Net the system identifies and localizes lines, symbols, and text within the binary image. The detected text locations are subjected to the OCR module, a process that digitizes the text, enabling the extraction of crucial data such as component numbers. These numbers are instrumental for the subsequent rule-based MTO creation. The method outputs, next to the MTO, a visual representation of the results plotted in the original P&ID, which serves as confirmation of the system's accuracy.

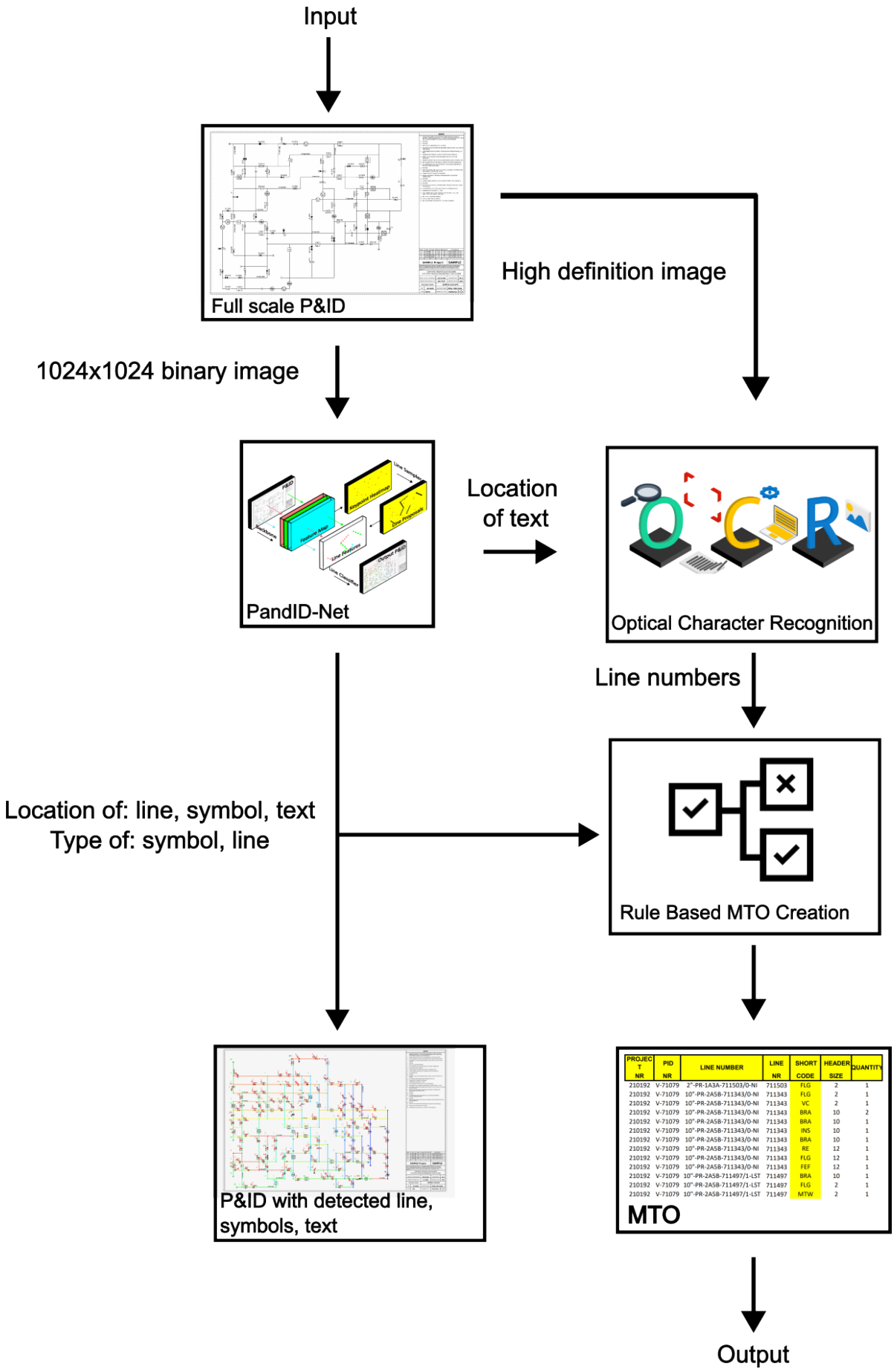

Figure 4.1: This is the pipeline where PandID-Net will be integrated. Since the model does not have an OCR module built-in, this needs to be added. Also, some rule-based code is needed to create the MTO.

4.2. PandID-Net

The PandID-Net is a two-step line, symbol, and text detection model. It processes complete scaled P&IDs and outputs the location and classification of both lines and symbols, along with the positions of text elements. It is the first model to combine symbol recognition and line recognition tasks into one trainable model. It is capable of detecting three different types of lines: dashed, continuous, and text-symbol relation lines, and 34 different types of keypoints: junctions, text locations, and 32 distinct symbols. The different classes for lines and keypoints are depicted in Figure 4.2. The workings principles of the model are explained in this section.

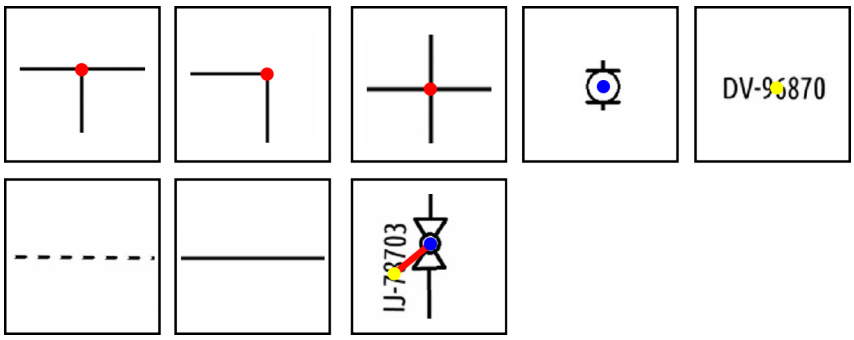

Figure 4.2: The top row shows the keypoint classes: junction, symbol, and text. In the bottom row the three line classes: dashed, continuous, and relational are depicted

4.2.1. Architecture

The PandID-Net architecture is largely based on the work of Zhou et al. [34] and is illustrated in Figure 4.3. It can be split into four parts: First the feature extraction backbone. This will output a feature map (depicted by the multicolor block) that is the input for the other modules. Secondly, the keypoint detection module outputs the detection of symbols, junctions, and text locations. Its outputs feed into the line sampler module, which will construct line proposals from sets of two keypoints. It is represented by two arrows indicating the extraction of line proposals and line features. Lastly, the line classification module will decide using the line features to which class the line belongs. In the figure, the arrows with text represent the different modules, and the blocks represent the intermediate results. The four modules are explained in Sections 4.2.2 to 4.2.5.

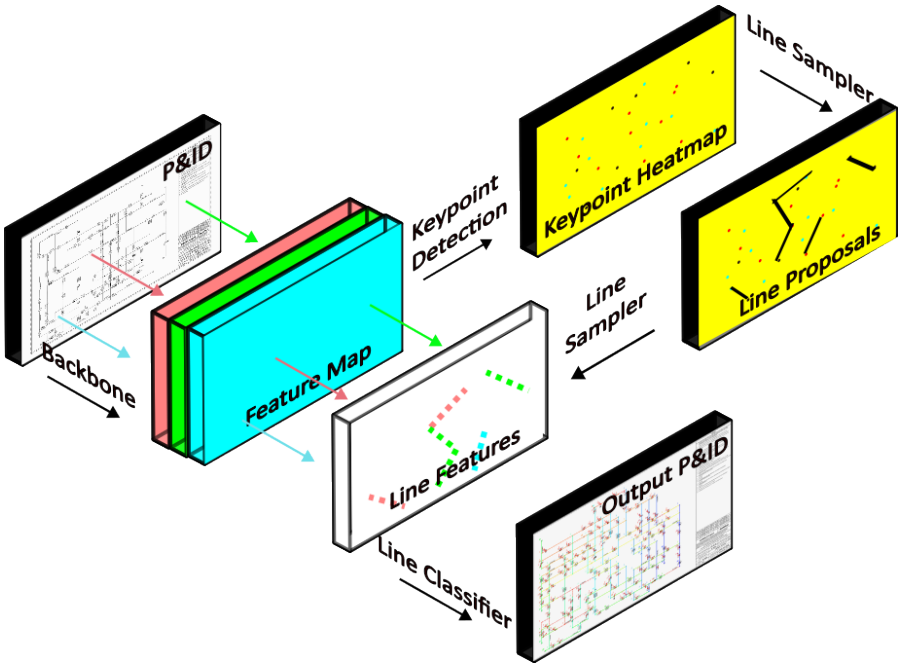

Figure 4.3: Schemetic overview of the PandID-Net architecture

4.2.2. Feature Backbone

When an image is processed through a CNN, each layer extracts certain features from the input. Early layers might capture basic features like edges and textures, while deeper layers can identify more complex patterns. These sets of features, represented as multi-dimensional arrays, are what we call feature maps.

A backbone in neural network architectures is a pre-trained network used primarily for feature extraction. This backbone network processes the input image and generates feature maps that contain important information about the visual characteristics of the image. In the specific case of the PandID-Net, the feature backbone is responsible for extracting feature maps from P&ID images. These feature maps are then utilized by other modules within the network to predict keypoint locations and classify lines explained in Section 4.2.3, and 4.2.5 respectively.

PandID-Net employs a stacked hourglass backbone [24], a network structure renowned for its effectiveness in human keypoint estimation and corner-point detection. This network is also explained in Section 2.2.3. The image is first downscaled to smaller dimensions in this case, 256 by 256 pixels from an initial size of 1024 by 1024 pixels. The hourglass modules are named for their hourglass shape, these modules involve a process of downsampling followed by upsampling, allowing the network to capture the contextual and global features of the image. During this upsampling process, the network refines the feature maps, enhancing details and spatial accuracy. This structure allows the hourglass network to capture features at multiple scales effectively, making it well-suited for tasks requiring precise localization, such as keypoint detection and line classification in P&ID images.

4.2.3. Keypoint Detection

The keypoint detection module in our approach, inspired by the L-CNN method for junction detection in images [34], aims to identify keypoints on a heatmap. We process an image with dimensions $W \times H$ by dividing it into smaller sections, or bins, each sized $W_b \times H_b$. The network predicts whether each bin contains a keypoint.

Two maps are created during this process: the keypoint mask map, denoted as K, and the 2-D offset map, denoted as O. We further break down the image into a grid of $H' \times W'$ for a simpler, lower resolution. In each grid section b, if there is a point p in K, then $K(b)$ is set to 1 and $O(b)$ records the difference between the actual point location x_b and the point p. If there's no point in K within that section, both $K(b)$ and $O(b)$ are set to zero. The values in $O(b)$ are adjusted relative to the size of the grid section, and their range is limited to $[-0.5, 0.5] \times [-0.5, 0.5]$. Since there is a separate heatmap for each class C, the dimensions of the heatmap tensors K and O are $C \times 256 \times 256$.

The method faces the challenge of class imbalance, arising from each class having its unique heatmap. This issue is exacerbated when some keypoints appear only once in the image. In such cases, there are $256 * 256 - 1 = 65535$ instances of "non-keypoint" locations and just a single keypoint location. To address this imbalance, we test with two approaches in the ablation study: focal loss and weighted cross-entropy loss. More about loss functions can be found in Section 5.1.

Furthermore, our method involves predicting the keypoint offset. This offset represents the difference between the junction's actual location in the image and its corresponding position on the scaled map. For this prediction, we employ l_1 loss function which is the absolute difference between the predicted value and the targets.

4.2.4. Line Sampler

The line sampler module in the system is divided into two components: a static line sampler and a dynamic line sampler. This is because the lines are constructed between keypoint pairs. In cases where the keypoints detections in the initial forward passes of the network are not sufficiently accurate, the dynamic line sampler struggles to effectively sample lines. To mitigate this issue, the static line sampler steps in, ensuring that line sampling can still occur efficiently even when initial keypoint detection is suboptimal.

Static Line Sampler

The static line sampler works by extracting certain types of line samples from each image, based on the ground truth labels of the image. These include continuous lines, dashed lines, relational lines, and negative lines (lines that do not exist in the image). These are called static samples because they are

selected without considering the positions of keypoints predicted by the model and are only based on the labels of the training data.

The positive line samples (continuous, dashed, and relational) are always taken from the actual lines present in the image, along with their corresponding keypoint coordinates. However, there are usually far more negative samples than positive ones. To effectively choose negative samples, a technique is used where all real lines are drawn onto a low-resolution bitmap. Each potential but non-existent line between two ground truth keypoints is then scored based on how densely its path is populated on this bitmap. The most 'difficult' lines – the ones that might be mistaken for real lines – are chosen as negative samples.

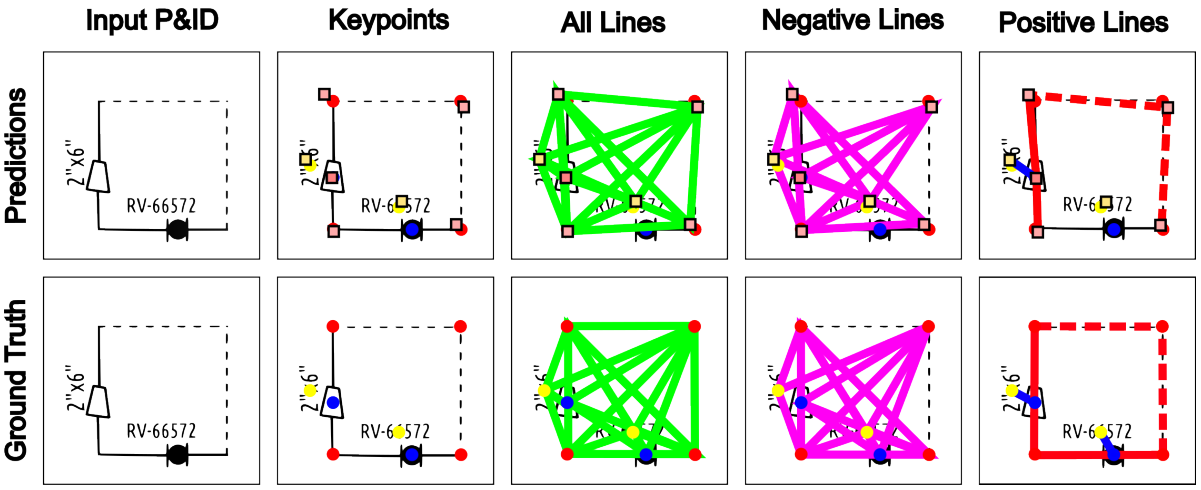

Figure 4.4: This is an illustration of the dynamic line sampling method. The most left images are a simplification of the input P&ID. The bottom row is the ground truth where the green lines represent all possible lines, and the pink lines represent the set of negative lines. The red continuous lines, dashed lines, and blue lines are the ground truth lines. On the top row, the squares are the predicted symbols, junctions, and text locations. The green lines represent the set of all possible lines, the pink lines are the negative lines and the last image represents the set of predicted lines in the three different classes.

Dynamic Line Sampler

The dynamic line sampler, on the other hand, selects lines based on the keypoint predictions made by the keypoint detection module from Section 4.2.3. It involves a matching process where each predicted keypoint is compared with the actual keypoints from the ground truth. If a predicted keypoint is close enough to a real keypoint (within a certain distance threshold), and they have the same class, it's considered a match.

Using these matched keypoints, the sampler then forms different categories of line samples. If both endpoints of a proposed line match real keypoints and correspond to an actual line in the image, it's classified as a positive line. If they match but don't correspond to a real line, the line is considered a hard negative sample. Additionally, all possible lines formed from the predicted keypoints are collected, regardless of whether they match the ground truth or not.

From these categories, a certain number of lines are randomly selected to form the set of dynamic line samples. This sampler is particularly useful because it adjusts the endpoints of the lines to match the predicted keypoints, thereby refining the line detection performance. The method is illustrated in Figure 4.4

The static sampler is especially valuable early in training, providing reliable positive and challenging negative samples when the dynamic sampler's predictions are less accurate. As training progresses and the dynamic sampler's predictions improve, it contributes increasingly to the line detection performance. Together, these samplers offer a comprehensive approach to learning and detecting lines in images.

4.2.5. Line Classification Module

The line verification network in our approach takes a set of candidate lines from the line sampler of the previous section and the feature maps from the backbone network. Its primary function is to determine

whether each line exists in the P&ID.

Each candidate line segment is processed to produce a fixed-length feature vector. This process, inspired by object detection techniques, involves a Line of Interest (LoI) pooling layer that takes the coordinates of the line's endpoints and extracts relevant features. The LoI pooling layer operates by first calculating 32 uniformly spaced points along each line using linear interpolation. It then extracts feature values at these points from the backbone's feature map using bilinear interpolation, reducing quantization errors. The resulting feature vector is then downsized using a 1D max pooling layer. The final output of the LoI pooling layer is a flattened feature vector, which is used to predict the presence of the line. These features are then concatenated and fed into a network head consisting of two fully connected layers, which produce a logit. The presence of the line is determined by comparing this logit to the line's label using a loss function.

4.3. Post-processing

The PandID-Net will give a set of lines and a set of keypoints. The results of the model are post-processed with some logic to improve the results. The following operations are performed:

The keypoints are associated with lines. Only the keypoints that are connected with lines are included in the filtered set. This is done with Algorithm 2 that is shown below:

Algorithm 2 Keypoint Filtering

procedure filter_keypoints(kps, $kscore$, $lines$, $ktype$, $angle\_threshold$, $distance\_threshold$, $tolerance$)
    Define get_keypoint_indices($line$, kps) to find indices of kps that form $line$
    for each $line$ in $lines$ **do**
        for each kp_index in get_keypoint_indices($line$, kps) **do**
            if kp_index not in $unique\_indices$ **then**
                Add kp_index to $unique\_indices$
                Append $kps[kp\_index]$ to $filtered\_keypoints$
                Append $kscore[kp\_index]$ to $filtered\_scores$
                Append $ktype[kp\_index]$ to $filtered\_ktypes$
    return $filtered\_keypoints$, $filtered\_scores$, $filtered\_ktypes$

Algorithm 3 is a filter for the line data. It starts by calculating the lengths and angles of lines to assess spatial relationships. The algorithm then removes lines that are too similar in terms of length and orientation, prioritizing longer and more significant lines.

Algorithm 3 Process Lines for Enhanced Line Detection

procedure process_lines($lines$, $scores$, kps, $ktype$, $angle\_threshold$, $distance\_threshold$, $tolerance$)
    Define function to calculate line length
    Define function to calculate the angle between two lines
    Define function to check the proximity of two lines
    Define function to remove redundant lines based on length, angle, and proximity
    Define function to establish the canonical form of a line
    Define function to check if the line is horizontal or vertical
    Define function to get keypoint indices for a given line
    Initialize seen lines and best lines for keypoints
    Obtain unique lines by removing double lines
    Filter lines based on type and orientation
    Keep lines with the highest score for each keypoint
    Compile final filtered predictions and their scores
    return filtered lines and scores

Lines are standardized to a uniform format and filtered based on their orientation, keeping only those that are almost perfectly horizontal or vertical if they belong to the continuous or dashed line class. The

algorithm further refines the selection of relational lines by making sure that every text only has one line connected. this is done by considering the confidence scores of detected lines and choosing the line with the highest scores for each text point.

4.4. Take-aways

This chapter introduced a learning-based pipeline for constructing MTOs from P&IDs. The method is centered around the PandID-Net which was explained in Section 4.2. This is a novel approach to detect lines in P&IDs and the first method to use deep learning for this task. The next chapter will describe the test setup. This includes an explanation of the loss functions used and what experiments are conducted.

5

Experimental Setup

In this chapter, the test setup for PandID-Net is explained. The compared model functions for the ablation study are explained and chosen hyperparameters are defined. We begin by defining the concept and purpose of an ablation study. The main objective of an ablation study is to evaluate the effectiveness of each system component in achieving the desired results. This involves methodically modifying or removing different parts of the system and assessing how these alterations affect overall performance. This approach helps to identify the most effective functions.

In the context of PandID-Net, our goal is to establish the most efficient configuration before drawing comparisons with previous studies. This is done using an ablation study were we conduct tests on PandID-Net using two different loss functions for both the keypoint detection module (see Section 4.2.3) and the line classifier (see Section 4.2.5). Additionally, we experiment with three different configurations of the feature extraction backbone (see Section 4.2.2) and compare two varying learning rates.

5.1. Loss Functions

Neural networks utilize loss functions to measure the difference between the predicted output and the actual data, guiding the network in learning accurately. These functions are crucial as they provide a quantifiable metric that the network aims to minimize during training, adjusting the weights of the network accordingly. By iteratively updating these weights based on the loss function, the neural network optimizes its predictions, progressively improving its performance on the given task.

Both the keypoint detection module and the line classifier are tasks of multiple classes with a large class imbalance. That is why this study will compare the weighted cross-entropy loss and the focal loss. Both are designed for handling large class imbalances.

5.1.1. Weigthed Cross-entropy Loss

The traditional cross-entropy loss, used in binary classification, is formulated as follows:

$$CE(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{if } y = 0 \end{cases}$$

In this equation, p represents the predicted probability of an instance being classified as positive, and y is the actual label (1 for positive, 0 for negative).

In scenarios with significant class imbalances, where negative samples far outnumber positive ones, traditional cross-entropy loss may become dominated by these numerous negative instances. This dominance can skew the model training towards these negative samples, often at the expense of the critical, yet infrequent, positive instances.

To counteract this imbalance, weighted cross-entropy loss is introduced. This variant adjusts the loss function by applying different weights to the positive and negative instances. It's a modification of the standard cross-entropy formula, adding a weighting factor that increases the loss contribution from the minority class (the positive instances) and decreases it from the majority class (the negative instances). This adjustment helps the model to pay more attention to the rare positive instances, thereby balancing the influence of both classes during the training process.

5.1.2. Focal Loss

Another method for addressing significant class imbalances is the Focal Loss [17]. This approach modifies the traditional cross-entropy loss by incorporating a modulating factor. This factor reduces the loss from easy, correctly classified examples, allowing the model to concentrate more on difficult, misclassified cases.

Focal Loss is expressed as:

$$FL(p, y) = -(1-p)^{\gamma} \log(p) \text{ if } y = 1 \text{ else } -p^{\gamma} \log(1-p)$$

Here, γ is a focusing parameter that controls how much the loss is adjusted for easy examples. A γ value of 0 means the Focal Loss is the same as standard cross-entropy loss. As γ increases, the influence of the modulating factor strengthens.

In the context of heatmap-based keypoint detection, Focal Loss proves particularly advantageous. The majority of negative pixels in the heatmap, which are easily classified, contribute less to the loss, thereby allowing the model to prioritize the more challenging, positive pixels. The Focal Loss introduces an α term to balance the disparity between classes. While α is typically a constant, our method dynamically calculates it based on class frequency in the dataset.

The α value for each class is inversely related to its frequency, meaning less common classes receive a higher α value. The calculation of α is as follows:

$$\alpha_c = \frac{1}{\text{frequency}(c) + \epsilon}$$

Here, c represents a class, frequency(c) is its occurrence rate in the dataset, and ϵ is a small constant to avoid division by zero.

Thus, the Focal Loss with this dynamic α component is:

$$FL(p_t, y) = -\alpha_t \times (1-p_t)^{\gamma} \times \log(p_t + \epsilon)$$

In this formula, p_t is the probability of the actual class, α_t the α value for that class, and ϵ ensures numerical stability.

5.2. Mutual Exclusivity

Because the keypoint detection module has a keypoint map for every class, and since the loss is computed individually for each map, mutual exclusivity between keypoints is not guaranteed. To address this we took two measures. First, we introduced a form of non-maximum suppression (NMS) after the keypoint detection. Additionally, an extra loss function will penalize lines where the endpoints do not have the correct class. The loss function and the NMS are explained in this section.

NMS is a post-processing algorithm used primarily in object detection tasks to prune redundant and overlapping predictions. When applied to keypoint predictions where each keypoint class occupies a distinct layer, NMS enforces class-specific mutual exclusivity that is not guaranteed otherwise. The basic idea behind NMS is to retain only the most confident prediction while suppressing all other predictions that have a high overlap with it.

Algorithm 4 3D Non-Maximum Suppression (NMS)

function nms_3d(a)
  $original\_shape \leftarrow$ Shape(a)
  if $original\_shape[0] = 1$ **then**
    return nms_2d(a)
  $a \leftarrow$ View($a, 1, original\_shape[0], original\_shape[1], original\_shape[2]$)
  $ap \leftarrow$ MaxPool3D($a, (original\_shape[0], 3, 3), stride = (1,1,1), padding = (0,1,1)$)
  $keep \leftarrow (a == ap) * 1.0$
  return Squeeze($a * keep, 0$)

The newly introduced loss function in the line sampler component serves two main purposes related to mutual exclusivity. Firstly, it calculates the loss based on the distance between the detected keypoints

and their corresponding ground truth counterparts. This part of the loss function essentially quantifies how close the model's predictions are to the actual keypoint locations in the data, with a focus on minimizing this spatial discrepancy. Secondly, an additional penalty is incorporated for any keypoints that the model fails to match. This aspect of the loss function penalizes the model for missing keypoints, encouraging it to improve both its detection accuracy and its ability to correctly identify all relevant keypoints.

5.3. Feature Extractor Backbone

We also test three different feature extraction backbones to evaluate which performs best on P&ID datasets. The different variants of the backbone are a ”deep” hourglass backbone, a ”shallow” hourglass backbone, and the HT-backbone.

5.3.1. Hourglass

The hourglass backbone developed by Newell et al. [24] and simplified for junction detection by Zhou et al. [34] uses downsampling and upsampling modules with interconnections. This shape and interconnections will keep features of different resolutions and spatial properties.

The ”deep” and ”shallow” configurations of the model indicate the depth, number of stacks, and the number of blocks. The sizes of the configurations are in Table 5.1

	”Deep”	”Shallow”
Depth	5	4
Stacks	2	2
Blocks	2	1

Table 5.1: Number of stacks, blocks, and the depth of the hourglass modules

Depth is a parameter of the Hourglass module within each stack. It defines the number of times the process of downsampling and then upsampling is repeated within each individual Hourglass module. The Stack refers to the sequential arrangement of hourglass modules. Each stack comprises a series of these modules placed end-to-end, allowing for repeated processing of features across scales. The architecture benefits from multiple stacks to refine and reassess predictions iteratively. Figure 5.1 depicts multiple stacked hourglasses. Blocks denote the number of residual units (or groups of layers) within each downsampling or upsampling stage of the Hourglass module. A visualization of the blocks and depth is depicted in Figure 2.3 in Chapter 2.

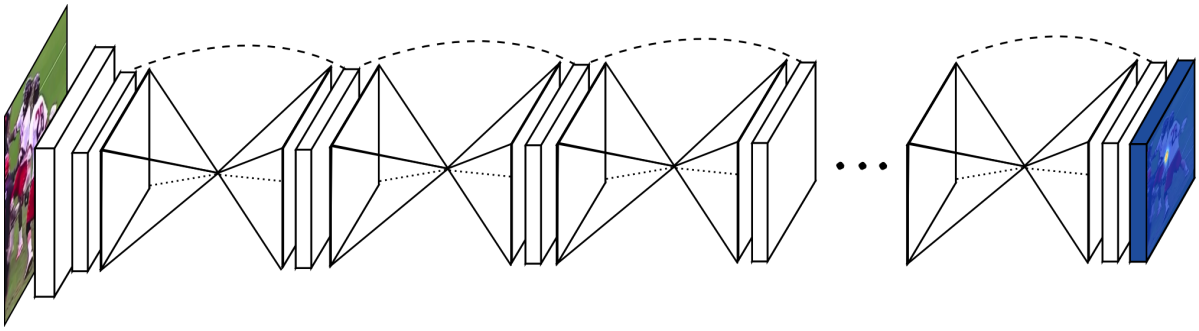

Figure 5.1: Example of multiple stacked hourglass modules from Newell et al. [24]

5.3.2. HT-IHT Block

The Hough Transform-Inverse Hough Transform HT-IHT block from Lin et al. [18] incorporates the Hough Transform into a deep network, allowing the combination of local image features with global line priors. The HT layer transforms input feature maps to the Hough domain, followed by local convolutions in this domain, equivalent to global operations in the image domain. The IHT layer then inverts these results back to the image domain. This method effectively combines learned local appearance with global geometric line structures, enhancing the network's ability to detect lines efficiently and reducing reliance on extensive labeled data

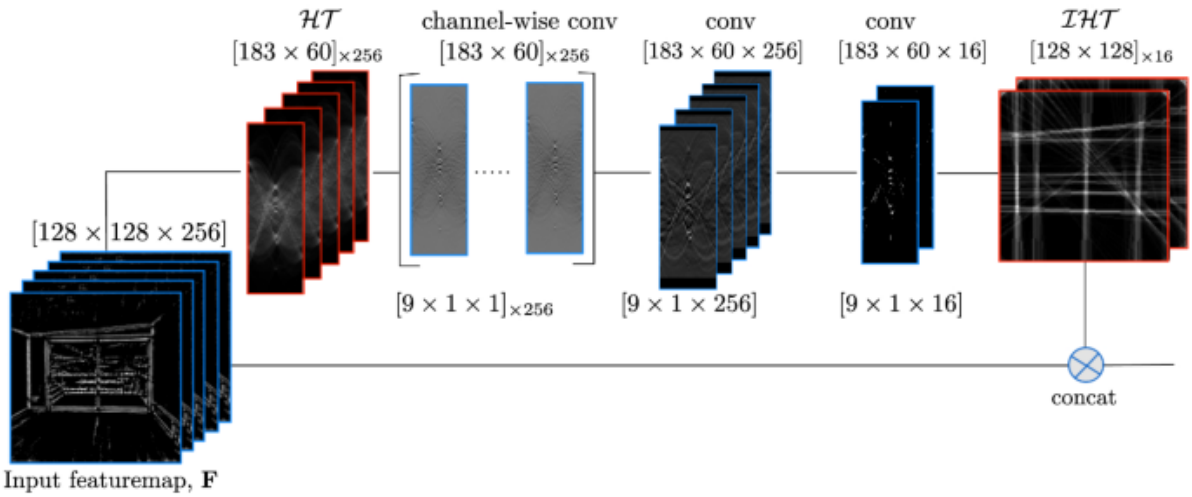

Figure 5.2: The Hough Transform - Inverse Hough Transform block from Lin et al. [18]

The hourglass modules in the network, which involve sequences of downsampling and upsampling for feature processing, are replaced with HT-IHT modules. Each HT-IHT module is a combination of a residual block and an HT-IHT block which applies the Hough Transform and its inverse to process features. The block is depicted in Figure 5.2. This integration leverages the HT-IHT's capability to capture and utilize both local and global image features, enhancing the network's ability to detect lines.

5.4. Learning Rate

The learning rate is the size of the weight adjustments the model can make during the update process. Choosing the right learning rate involves balancing the need for fast convergence with the risk of overshooting the optimal solution or getting stuck in local minima.

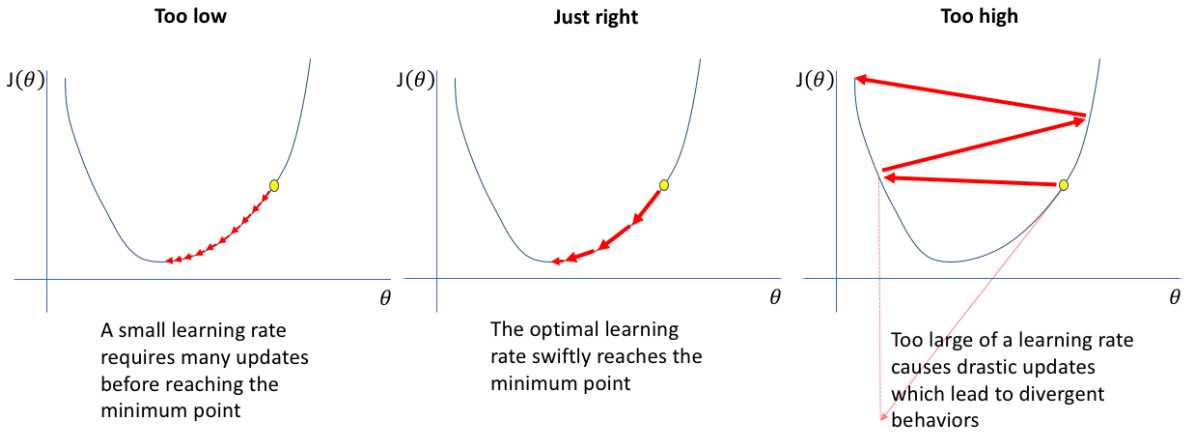

Figure 5.3: Visualization of different learning rates [11]

In Figure 5.3, the impact of different learning rates on the convergence behavior of an optimization algorithm is illustrated. As depicted, a learning rate that is too low leads to a gradual and potentially suboptimal convergence path, possibly stagnating in local minima and resulting in a protracted training process. Conversely, a learning rate that is too high may cause the algorithm to oscillate erratically and overshoot the minimum, failing to converge. An optimally chosen learning rate strikes a balance, providing a steady and efficient trajectory toward the global minimum. Notably, an excessively high learning rate not only overshoots the minimum but may also diverge, as the updates become too large for the algorithm to maintain a trajectory towards the minimum.

To find the correct learning rate for PandID-Net multiple small runs of only two epochs were done to

see how quickly it would diverge. This resulted in the conclusion that we would try two learning rates for a long running time. Additionally, a learning rate decay is added which helps prevent overfitting by reducing the learning rate during training.

5.5. Optimizer

An optimizer is an algorithm that adjusts the network's weights and learning rate to minimize the loss function. It determines how the network updates its parameters, such as weights, in response to the loss function's output, thereby guiding the network towards more accurate predictions.

Gradient Descent is a basic method for this, involving the iterative adjustment of weights to minimize the loss. However, Gradient Descent can be time-consuming for large datasets, so Stochastic Gradient Descent (SGD) is often used instead. SGD updates weights using a single data sample at a time, making it more efficient. To further enhance SGD, a momentum term is added to stabilize and direct the optimization process. SGD with and without momentum are depicted in Figures 5.4, and 5.5.

Figure 5.4: SGD without momentum [28]

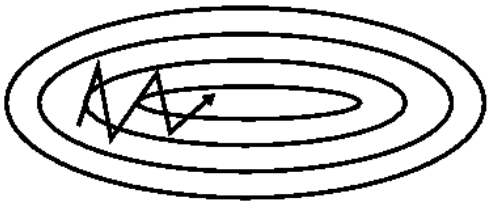

Figure 5.5: SGD with momentum [28]

As optimizer in PandID-Net, we used the Adam optimizer [15]. This is an adaptive learning rate optimization algorithm that has been designed specifically for training deep neural networks. It stands for ”Adaptive Moment Estimation”. The key feature of Adam is that it maintains two different learning rates which are adaptively changed for each parameter. The first moment estimate, which is the mean of the gradient, helps to accelerate the optimizer in the relevant direction, while the second moment estimate, the uncentered variance of the gradient, aids in adapting the learning rate to the topology of the error surface. This dual approach allows Adam to handle sparse gradients on noisy problems, making it exceptionally versatile for a wide array of tasks and models. Adam also includes a bias correction mechanism to counteract the biases in the first and second moment estimates toward zero, making it effective right from the initial iterations.

In addition to the standard Adam optimizer, PandID-Net incorporates ”AMSGrad” [27] a variant of Adam. AMSGrad addresses an issue in the original Adam optimizer related to the learning rate decay. This modification stabilizes the learning rate updates, preventing them from becoming too small as training progresses.

5.6. Hyperparameters

In this section, we detail the supplementary hyperparameters utilized in our model. For training, we have set the batch size to two, and for evaluation, it is one, optimizing GPU memory usage. The loss weights are adjusted to ensure consistency across all loss functions, as outlined in Table 5.2. In the LoIPool layer, 32 points are selected along each line as features and reduced via stride-4 max pooling from 32 to 8 spatially. All tests were conducted on Google Colab's T4 GPU, maintaining a consistent environment with a fixed seed. The models were trained for a fixed length of 24 epochs

Parameter	Weight
keypoint map	0.01
line map	0.5
keypoint offset	0.25
line class 0	10
line class 1	10
line class 2	10
line negative	10
keypoint type	0.1

Table 5.2: Loss Weights for Different Parameters

The model utilizes two distinct samplers: a static sampler and a dynamic sampler, each with its own set of parameters. The static sampler's configurations are in Table 5.3, and the dynamic sampler in Table 5.4.

Static Sampler Parameter	Value
Number of positive lines for dashed class (n_stc_posl0)	100
Number of positive lines for continuous class (n_stc_posl1)	200
Number of positive lines for relational class (n_stc_posl2)	200
Number of negative lines (n_stc_negl)	200

Table 5.3: Static Sampler Configurations

Dynamic Sampler Parameter	Value
Number of keypoints (n_dyn_kpoint)	400
Number of positive lines for dashed class (n_dyn_posl0)	400
Number of positive lines for continuous class (n_dyn_posl1)	400
Number of positive lines for relational class (n_dyn_posl2)	400
Number of negative lines (n_dyn_negl)	400
Number of random lines (n_dyn_othr)	400

Table 5.4: Dynamic Sampler Configurations

Table 5.5 specifies the maximum number of keypoints and lines that the model outputs. This cap is implemented to control the computational demands. The limits are set based on the quantity of ground truth keypoints and lines present in the P&ID.

Parameter	Value
Number of keypoints in the output (n_out_keypoint)	350
Number of lines in the output (n_out_line)	800

Table 5.5: Output cap for number of lines and keyponts

5.7. Take-aways

This chapter outlined the testing setup and delved into the specifics of the loss functions employed in this study. It also provided descriptions of other notable methods used in the research. The final section offered insights into the various hyperparameters that can be adjusted in the configuration file. More information about software packages and versions can be found in Appendix A.2. The subsequent chapter will present the results of our experiments, starting with a description of the dataset and the metrics utilized.

6

Experiments And Results

This chapter presents the experiments conducted and the results obtained in this study. It begins by describing the dataset used for the experiments. Following this, the chapter delves into the evaluation methods, partitioned into quantitative and qualitative analyses, to offer a thorough examination of the data. The quantitative section focuses on statistical measures and numerical data analysis, while the qualitative part offers a narrative interpretation of the results, providing deeper insights into the findings. An ablation study is performed to find the optimal configuration for PandID-Net. This is a study that tests the performance of individual components by conducting the same experiment twice with only one component changed. This optimal configuration is then evaluated and compared to a previous study on the same dataset from Paliwal et al. [25].

6.1. Dataset

The model is trained on a synthetic dataset, referred to as Digitize-PID, originally created and made publicly available by Paliwal et al. [25] for training and evaluation in the absence of a publicly available standard dataset for P&ID sheets. This dataset is comprehensive, containing 500 annotated P&ID sheets.

Digitize-PID includes 32 distinct symbols, depicted in Figure 6.1, and these symbols are uniformly plotted over varying graph structures. These structures are specifically generated to resemble authentic P&ID sheets, and the dataset introduces several types of noise such as pixelation, blurring, and salt and pepper noise to simulate real-world conditions.

The creators of the dataset assigned text labels to symbols and pipelines, adhering to the standards prevalent in real-world P&ID sheets. The ground truth provided with the dataset encompasses spatial information related to symbols, connected pipelines, and associated text labels. Additionally, sets of horizontal and vertical lines along with their begin and end coordinates and a list of all texts present in the P&ID sheets, together with their spatial positions, are provided.

We changed the annotation to align with the model. This adjustment includes the transition from using bounding boxes to center points. If these points do not align with a line, they are projected onto the nearest line to ensure accurate association. The process includes extending lines from junctions to symbols, with an emphasis on maintaining these lines for as long a stretch as possible. For lines, we used begin and end coordinates as annotation. Additionally, the annotations for text have been converted from bounding boxes to center points. These text center points are then linked with the nearest symbols. Subsequently, both these points, symbol and text, are integrated into the set of lines. All lines are annotated with seven attributes: $(class_{line}, x_1, y_1, class_1, x_2, y_2, class_2)$

This dataset to train and validate the PandID-net models, leveraging the varied and realistic conditions provided within. The diverse symbol representation and the intricate detailing in labeling and noise introduction will aid in assessing the robustness and efficacy of the models in development. The availability of this dataset is a substantial advantage, allowing for a more thorough and nuanced exploration and evaluation of methodologies applied to P&ID sheets.

The dataset is split into three sets where the validation and test set both have 50 samples and the train set includes 400 pictures that are augmented by turning them 90 degrees making it a set of

1600 samples. The images are cropped to remove the table and create a square and rescaled to the 1024×1024 format that the model can process. The images are made binary to remove any noise inherent in the dataset and real-world P&IDs.

The rescaling to 1024×1024 is necessary so that the model can process the entire P&ID on one GPU. It was the biggest size that could still be processed on one GPU and with a batch size of two. However, a drawback of this rescaling is that thin horizontal and vertical lines in the image can become less visible or disappear. This happens because reducing the resolution results in fewer pixels being available to represent these fine lines. More about this problem is written in Chapter 7.1.

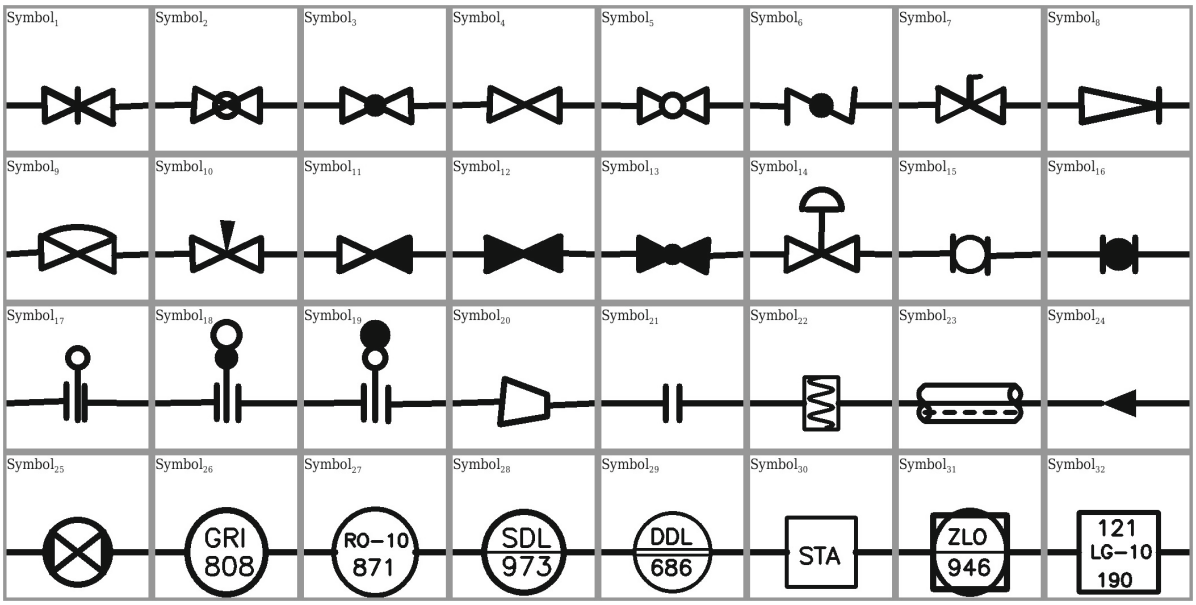

Figure 6.1: Figure showing a set of 32 different symbols used for Dataset-P&ID. Symbol 1 to Symbol 25 are complex symbols. The remaining Symbol 26 to Symbol 32 are considered based symbols since they are constructed from squares and circles. [25]

6.2. Evaluation

In this section, the evaluation methods and metrics are explained. It gives insight into the priorities of the models and the framework used for the evaluation of the results. The section is split into two parts, a quantitative part where the numerical metrics are explained and a qualitative part that explains the evaluation of the visual results. Both the subsections are again split into a part for line detection and keypoint detection.

6.2.1. Quantitative

In evaluating the performance of classification models, several metrics are commonly employed, each offering insights into different aspects of the model. Precision (P) measures the accuracy of positive predictions, quantifying the ratio of true positives (TP) to the total number of positive predictions, which includes both true positives and false positives (FP). This metric is particularly important in contexts where the cost of a false positive is high. Recall (R), in contrast, assesses the model's ability to correctly identify all relevant instances, calculated as the ratio of true positives to the total number of actual positives (the sum of TP and false negatives, FN). High recall is crucial in scenarios where missing a positive instance carries significant consequences. The formulas for precision and recall can be found in Equation 6.1.

$$\textit{Precision (P)} = \frac{TP}{TP + FP} \qquad \textit{Recall (R)} = \frac{TP}{TP + FN} \tag{6.1}$$

$$\textit{Accuracy (Acc)} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \textit{F1-Score} = 2 \times \frac{P \times R}{P + R} \tag{6.2}$$

Accuracy (Acc) offers a general measure of model performance, representing the proportion of true results (both true positives and true negatives, TN) in relation to the total number of cases examined.

This metric, while providing an overall effectiveness, may not be reliable in situations of class imbalance. The F1-Score addresses this by combining precision and recall into a single metric, offering a balance between the two by calculating their harmonic mean. It is particularly useful in contexts with an uneven class distribution, as it equally considers the impact of both false positives and false negatives. In Equation 6.2 the formulas of the F1-score and accuracy are shown.

$$sAP = \frac{1}{|C|} \sum_{c \in C} \int_0^1 p_c(t)\, dt \qquad mAP = \frac{1}{|C|} \sum_{c \in C} \int_0^1 p_c(t)\, dt \tag{6.3}$$

Additionally, in fields such as object detection or structure recognition, more specialized metrics like Structural Average Precision (sAP) and Mean Average Precision (mAP) are utilized. The sAP, introduced by Zhou et al. [34], derived from mAP but tailored for structural elements such as lines, calculates the mean area under the precision-recall curve for each class, providing a nuanced view of a model's effectiveness across the full range of confidence thresholds. The mAP, similarly, averages the area under the precision-recall curve across different classes (AP) and is widely applied in object detection and segmentation tasks. A high sAP/mAP reflects a model's stability and consistency across a range of confidence thresholds, highlighting its robustness in performance. It encapsulates the model's effectiveness in balancing precision and recall across various thresholds, rather than focusing on a singular point.

In Equation 6.3 the equations for sAP and mAP are shown. The $p_c(t)$ denotes the precision at a given threshold t for class c, and the integration across the range from 0 to 1 represents the area under the curve across different score thresholds.

Line Detection

In our methodology, we employ the Structural Average Precision (sAP) metric, like Zhou et al. [34] and the F1-score to evaluate line performance. For the ablation study, the sAP is used to get a robust single metric comparison between the different models. For the evaluation of the best-performing configuration of PandID-Net, the main focus is the F1-score. This is because the output of the model should have as little error as possible and it does not matter whether this is FN or FP.

The sAP metric calculates the area under the precision-recall curve. This curve is derived from a scored list of detected line segments across all test images. A detected line segment is considered a true positive if it can be associated with a ground truth line. The association of these lines with their ground truth counterparts is determined by calculating the Euclidean distances between the endpoints of the predicted and ground truth lines. For every predicted line, the algorithm selects the nearest ground truth line, provided it falls within a predefined Euclidean distance limit, which, for the purposes of this study, is set at a threshold value θ_L of 1, 5, and 10 in a 256×256 resolution setting. In the tables, the metrics are denoted as $sAP^{1.0}$, $sAP^{5.0}$, and sAP^{10}. The mean of the classes combined is given in the row "Total," and in the other rows the values are the AP for the classes separately.

$$\min_{(i,j)} \|x_1 - \hat{x}_i\|^2 + \left\|x_2 - \hat{x}_j\right\|^2 \leq \theta_L \tag{6.4}$$

Equation 6.4 represents the condition where the sum of the squared Euclidean distances between two points x_1 and x_2, and their respective estimates $\hat{x}_i$ and $\hat{x}_j$, must be less than or equal to a threshold θ_L. The $\min_{(i,j)}$ notation indicates that this condition is evaluated over all possible pairs of i and j, ensuring that the closest match within the threshold is chosen for each predicted line. This matching process is essential for determining true positives, which are then used to construct the precision-recall curve. The curve's area, quantified by the sAP metric at different θ_L values, provides a comprehensive evaluation of the model's performance in detecting and correctly associating line segments across various strictness levels.

Keypoint Detection

For keypoints, next to the F1-score, the metric mAP is used to evaluate the quality of the detections. Again the mAP will give a robust single metric to compare the variations of the model in the ablation study. The F1-score is used to evaluate the best-performing model since it will hold false positives and false negatives in equal significance. This is because the output of the model should have as little error as possible and it does not matter whether this is FN or FP.

A keypoint is deemed accurate if the Euclidean distance between this keypoint and its nearest ground-truth is within a threshold θ_K. In this study, averages are taken over 0.5, 1.0, and 2.0 in a 256×256 resolution setting. In the tables, these metrics are denoted as $AP^{0.5}$, $AP^{1.0}$, and $AP^{2.0}$, where for each row the AP for the class across the samples of the test set is shown and the mAP is shown at the row denoted by "Total". To ensure a one-to-one correspondence, keypoints are matched with ground truth keypoints using the Hungarian method [16], guaranteeing that each ground truth keypoint is paired only once.

The cost matrix is constructed to facilitate the matching process using the Hungarian algorithm. For each predicted keypoint, the Euclidean distance to each ground truth keypoint is calculated. This is formalized as:

$$\text{Cost}_{ij} = \begin{cases} \sqrt{\sum_{k=1}^{n}(p_{ik} - g_{jk})^2}, & \text{if } \sqrt{\sum_{k=1}^{n}(p_{ik} - g_{jk})^2} \leq \theta_K \\ \infty, & \text{otherwise} \end{cases} \tag{6.5}$$

In Equation 6.5, p_{ik} represents the k-th coordinate of the i-th predicted keypoint, and g_{jk} is the k-th coordinate of the j-th ground truth keypoint, with n being the dimensionality of the keypoints. The cost matrix is initially filled with infinity values, signifying no association. For each predicted keypoint, if the computed distance to a ground truth keypoint is less than or equal to the threshold θ_K, this distance replaces the corresponding infinity value in the cost matrix. The Hungarian algorithm then uses this matrix to find the optimal one-to-one matching between predicted and ground truth keypoints, minimizing the overall cost while ensuring that each ground truth keypoint is matched at most once.

6.2.2. Qualtiative

Qualitative evaluation plays a crucial role in complementing the quantitative metrics, providing insights into aspects of line and keypoint detection that are not immediately apparent through numerical analysis. This includes examining the alignment accuracy to ensure that detected lines correspond closely with the true line features in the image. Evaluating the continuity of lines is also important, especially in complex diagrams where line breaks are common. Assessing the straightness and angle of detected lines is vital to accurately represent the correct lines. Furthermore, it's necessary to check for offsets or discrepancies in the positioning of detected lines relative to their actual locations.

In the case of keypoint detection, the qualitative assessment involves inspecting the precision of keypoint placements relative to their intended positions in the diagrams. Additionally, the use of a confusion matrix in keypoint detection is instrumental in detecting causations. In this matrix, each column represents the instances of a predicted class, while each row represents the instances of an actual class. The diagonal elements show the number of correct classifications for each class, where the predicted class matches the actual class. The off-diagonal elements, on the other hand, indicate misclassifications, showing how often a particular class was predicted as another class. This matrix is especially useful in complex classification tasks, as it provides a detailed breakdown of the performance for each class, highlighting not only the overall accuracy but also specific areas where the model might be confusing one class for another.

6.3. Ablation Study

This section is dedicated to examining the individual impact of various components on the overall performance of the model. This is done by conducting an experiment twice with only one or two changed functions. This way we can isolate and understand the contribution of each component. The analysis focuses on evaluating different models using sAP and mAP metrics, chosen for their robustness across a broad spectrum of thresholds. The findings from this ablation study will serve as substantiation for the final configuration of PandID-Net.

The models all have been trained according to the training setup from Chapter 5, and tested on the test dataset unless specified otherwise.

6.3.1. Keypoint Detection Loss Function

In our keypoint detection study, we compared the performance of two loss functions: weighted cross-entropy (CE) loss and focal loss with a dynamic alpha parameter. The application of weighted CE

loss resulted in notably poor performance, with keypoint confidence scores consistently below 0.008, indicating an inability to effectively detect keypoints, likely due to the significant class imbalance present.

Focal loss, tailored to address class imbalance, regulates the cross-entropy criterion by reducing the loss contribution from easily classified examples, thereby emphasizing the more challenging instances. Its dynamic alpha parameter further enhances class balance. In contrast, weighted CE loss, despite its intent to mitigate class imbalance by assigning different weights to classes based on their frequency, fails to prioritize learning from challenging instances.

The effectiveness of focal loss in our experiments can be attributed to its nuanced approach to handling class imbalance. This task of detecting keypoints in P&IDs is characterized by a disproportionate number of negative or background instances compared to positive or keypoint instances. Focal loss, by focusing on the rare but critical positive examples, ensures that the model remains sensitive to key keypoints.

6.3.2. Line Detection Loss Function

The line detection module has been fitted with a focal loss and a cross-entropy loss with an extra penalty. The penalty is specific for predictions with the ground truth class of no-line and that are classified as dashed lines. This was implemented to increase the performance on the dashed line class.

The AP is calculated for the three classes separately. For the total calculation, the total amount of TPs, FPs, and FNs from the subclasses are used to calculate the mAP.

	Lines					
	Focal Loss			Cross Entropy Loss		
Line Type	$sAP^{1.0}$	$sAP^{5.0}$	sAP^{10}	$sAP^{1.0}$	$sAP^{5.0}$	sAP^{10}
Dashed	87.7	88.4	90.0	77.3	78.1	79.1
Continuous	89.0	90.3	90.4	80.8	82.1	82.1
Relational	92.2	93.0	93.8	89.3	90.6	91.7
Total	**89.6**	**90.6**	**91.4**	82.5	83.6	84.3

Table 6.1: This table presents the average structural precision of line detection, comparing the efficacy of different loss functions. The depicted values are the means computed from the test set outcomes. Analysis was conducted using three distinct distance thresholds, with each test implemented at a resolution of 256×256.

	Keypoints					
	Focal loss			Cross entropy loss		
Keypoint Type	$AP^{0.5}$	$AP^{1.0}$	$AP^{2.0}$	$AP^{0.5}$	$AP^{1.0}$	$AP^{2.0}$
Junction	88.4	88.9	88.9	88.6	89.3	89.4
Text	93.9	94.8	94.9	93.8	95.0	95.0
Symbol	68.2	84.2	84.2	60.5	74.9	75.0
Total	**83.5**	**89.3**	**89.3**	81.0	86.4	86.5

Table 6.2: This table summarizes the mean average precision (mAP) scores for keypoint detection across a selection of loss functions. The reported values represent the average performance on the test dataset. Evaluations were carried out at a fixed resolution of 256×256 pixels, applying three varied distance thresholds.

The performance comparison of loss functions at varying distance thresholds, as detailed in Table 6.1, reveals that the focal loss function outperforms the cross entropy loss function in the line classification module in terms of structural average precision (sAP). Furthermore, Table 6.2 presents the keypoint detection results with the average precision (AP) per class and the mAP in the row ”Total”. The focal loss function similarly enhances the performance of the keypoint predictor. In both tables, the highest values for mAP and sAP are written in bold typeface.

6.3.3. Feature Extractor Network

For the feature extractor network configuration, three different models were tested: an hourglass backbone with depth 4, stack 2, and one block (referred to as ”shallow”); a variant with depth 5, stack 2, and block 2 (referred to as ”deep”); and the third model, which incorporates an HT-block from Lin et al. [18], also with ”shallow” parameters. The models employ a focal loss for both the line detection module

and the keypoint detection module and are trained for 24 epochs before being evaluated on the test dataset.

	Backbone Configuration				Lines		
Line Type	Hourglass	HT-resnet	”Shallow”	”Deep”	$sAP^{1.0}$	$sAP^{5.0}$	sAP^{10}
Dashed	✓		✓		68.93	71.48	72.69
Continuous	✓		✓		86.01	87.44	87.57
Relational	✓		✓		92.16	93.17	93.91
Total	✓		✓		82.37	84.03	84.72
Dashed	✓			✓	87.69	88.40	89.99
Continuous	✓			✓	88.99	90.27	90.36
Relational	✓			✓	92.24	92.98	93.82
Total	✓			✓	**89.64**	**90.55**	**91.39**
Dashed		✓	✓		92.39	92.53	92.87
Continuous		✓	✓		82.23	82.95	82.86
Relational		✓	✓		86.24	86.91	88.21
Total		✓	✓		86.95	87.47	87.98

Table 6.3: Ablation study for selection of the backbone. Three different configurations are tested on the sAP for line detection. The configurations are a ”shallow” hourglass, a ”deep” hourglass, and the Hough transform module from Lin et al. [18]. The lines are split into three classes: Dashed, Continous, and Relational, and the mAP is given for three different thresholds on a scale of 256×256

	Backbone Configuration				Keypoints		
Keypoint Type	Hourglass	HT-resnet	”Shallow”	”Deep”	$AP^{0.5}$	$AP^{1.0}$	$AP^{2.0}$
Junction	✓		✓		88.69	89.18	89.33
Text	✓		✓		93.92	94.75	94.92
Symbol	✓		✓		67.79	83.79	83.83
Total	✓		✓		83.47	89.24	89.36
Junction	✓			✓	88.44	88.86	88.91
Text	✓			✓	93.86	94.84	94.89
Symbol	✓			✓	68.24	84.20	84.21
Total	✓			✓	**83.51**	**89.30**	**89.34**
Junction		✓	✓		88.78	88.87	88.87
Text		✓	✓		92.61	93.82	93.90
Symbol		✓	✓		66.48	82.15	82.22
Total		✓	✓		82.62	88.28	88.33

Table 6.4: Ablation study for selection of the backbone. Three different configurations are tested on the mAP for keypoint detection. The configurations are a ”shallow” hourglass, a ”deep” hourglass, and the Hough transform module from Lin et al. [18]. The keypoints are split into three classes: Junctions, Text, and Symbols, and the mAP is given for three different thresholds on a scale of 256×256

The highest results within Tables 6.3 and 6.4 are indicated in bold typeface. These results were achieved using a ”deep” hourglass backbone for both line detection and keypoint detection. The difference is more profound in the line detection result than in the keypoint detection. Due to the ”shallow” hourglass backbone surpassing the HT-module in performance and the GPU memory reaching its capacity, further exploration of the HT-module was deemed impractical and unnecessary. Consequently, the ”deep” hourglass backbone was selected for continued use in this study.

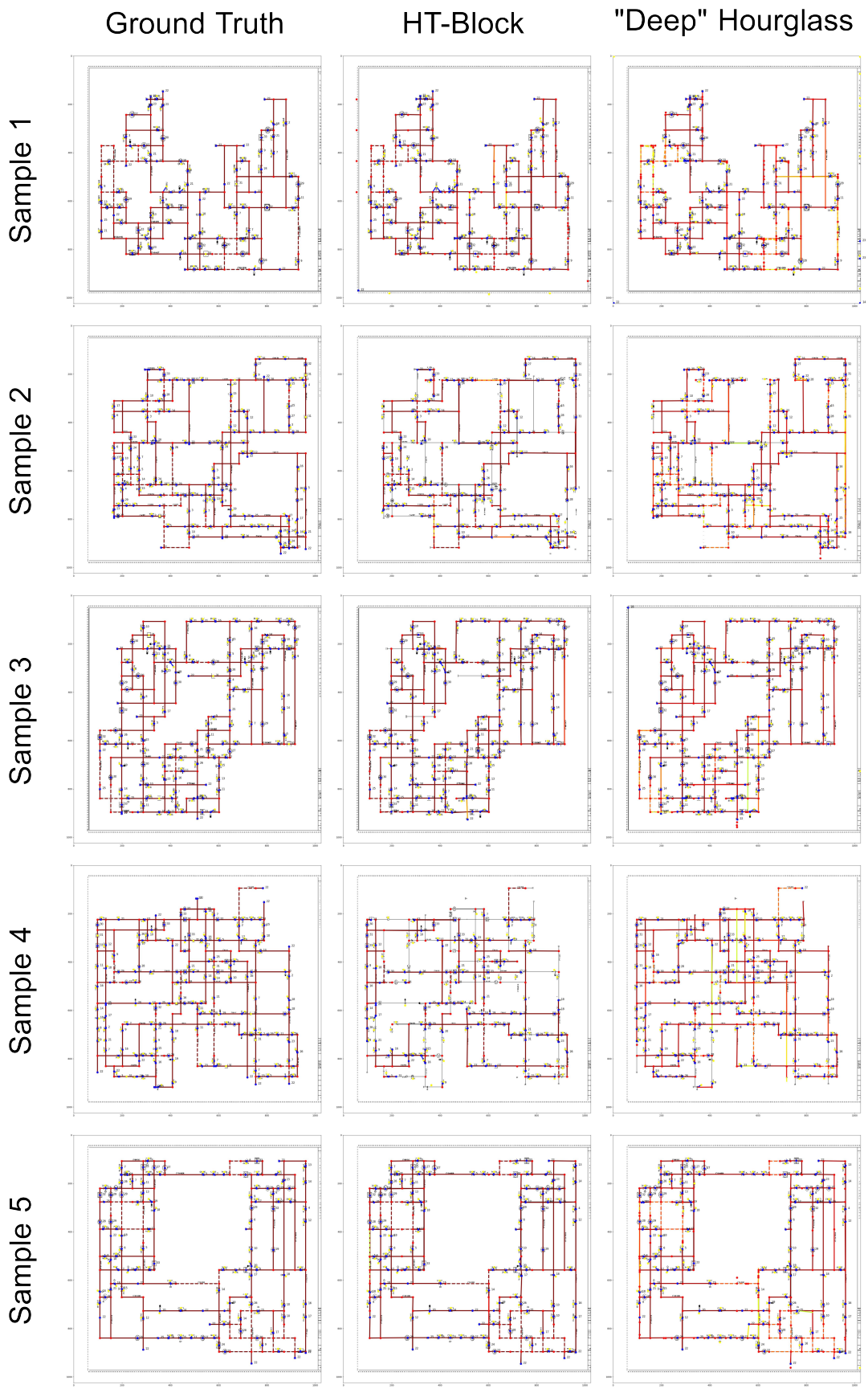

Figure 6.2: This Figure includes the results from the ”Deep” Hourglass model (right), the HT-backbone model (middle) next to the ground truth (left)

In Figure 6.2, we observe the performance of the HT-backbone in line detection, which yields notably ”clean” results. The predictions for lines are characterized by high confidence scores, as indicated by the dark red color, and a minimal presence of false positives. However, it should be noted that there are instances of missed lines. Conversely, the predictions obtained from the ”deep” backbone display a more chaotic pattern, with an increased number of false positives that have low confidence scores. This way it missed fewer lines. The false positive lines with low confidence scores can be filtered out with a confidence threshold resulting in an overall higher mAP.

6.3.4. Learning Rate

For the learning rate, there were multiple small experiments conducted that concluded that the learning rate was not higher than $4 * 10^{-4}$ (the rate used by Zhou et al. [34]) This experiment with two learning rates, a constant learning rate of $4 * 10^{-5}$ trained for 50 epochs and a learning rate of $4 * 10^{-4}$ trained for 24 epochs is meant to exclude a lower learning rate. Both of the models use a cross-entropy loss for the line detection.

	Lines					
	Learning Rate: 4*10^{-5}			Learning Rate: 4*10^{-4}		
Line Type	sAP$^{1.0}$	sAP$^{5.0}$	sAP^{10}	sAP$^{1.0}$	sAP$^{5.0}$	sAP^{10}
Dashed	75.65	77.14	78.59	77.34	78.13	79.09
Continous	74.72	81.17	81.18	80.82	82.13	82.07
Relational	78.70	85.93	88.05	89.34	90.58	91.73
Total	76.36	81.41	82.61	**82.50**	**83.62**	**84.30**

Table 6.5: The sAP scores for line detection at different learning rates and three different distance thresholds on a scale of 256×256

	Keypoints					
	Learning Rate: 4*10^{-5}			Learning Rate: 4*10^{-4}		
Keypoint Type	AP$^{0.5}$	AP$^{1.0}$	AP$^{2.0}$	AP$^{0.5}$	AP$^{1.0}$	AP$^{2.0}$
Junction	88.20	90.04	90.37	88.62	89.29	89.36
Text	82.90	93.62	94.63	93.77	94.99	95.04
Symbol	10.51	12.87	12.87	60.46	74.93	75.01
Total	60.54	65.51	65.96	**80.95**	**86.40**	**86.47**

Table 6.6: The AP and mAP scores for keypoint detection at different learning rates, and three different distance thresholds on a scale of 256×256

The results for the two learning rates can be found in tables 6.5 and 6.6. The highest scores, indicated by bold lettering, belong to the model trained with a learning rate of $4 * 10^{-4}$. The difference is quite profound, especially for the keypoint detections.

6.4. Optimal Configuration

The optimal configuration for the task of recognizing lines, text, and symbols in P&IDs is a ”deep” hourglass backbone with the focal loss for both line and symbol classification, and a learning rate of $4 * 10^{-4}$ including a learning rate decay of $1 * 10^{-4}$ after the tenth epoch. Now we can evaluate this model on the metrics of precision recall and F1-score. The results are shown for different confidence thresholds. This approach is necessary because a higher threshold typically enhances precision by reducing false positives, yet may lower recall by missing true positives. Ultimately, the optimal model configuration aims to strike a balance between precision and recall to maximize the F1-score. Furthermore, post-processing, that was explained in Section 4.3, is applied to fine-tune for optimal results.

	F1-score No-postprocessing				F1-score Postprocessed			
Score Thres	Total	Dashed	Continous	Relational	Total	Dashed	Continous	Relational
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.95	88.68	0.00	88.53	95.33	88.67	0.00	88.65	95.17
0.9	92.44	52.89	**92.41**	95.84	92.40	58.44	**92.42**	95.64
0.85	**92.89**	**85.48**	91.83	**95.86**	**92.85**	**90.24**	91.79	**95.68**
0.8	91.71	77.26	90.70	95.66	91.64	82.17	90.63	95.47
0.7	90.75	70.99	89.71	95.60	90.61	77.18	89.63	95.25
0.6	90.74	70.99	89.71	95.58	90.60	77.18	89.63	95.24
0.5	90.74	70.99	89.71	95.58	90.60	77.18	89.63	95.24
0.4	90.74	70.99	89.71	95.58	90.60	77.18	89.63	95.24
0.3	90.74	70.99	89.71	95.58	90.60	77.18	89.63	95.24
0.2	90.74	70.99	89.71	95.58	90.60	77.18	89.63	95.24
0.1	90.74	70.99	89.71	95.58	90.60	77.18	89.63	95.24
0	90.74	70.99	89.71	95.58	90.60	77.18	89.63	95.24

Table 6.7: These are the F1-scores per class and in total of the line predictions per confidence score threshold. The left side of the table is without post-processing and the right is with post-processing, The distance threshold was set on 5 on a resolution of 256×256

The results of the line detection per confidence threshold are shown in Table 6.7 where the right side of the table contains the filtered results and the highest scores per class are indicated with a bold typeface. The model finds it most difficult to classify the dashed lines. The text symbol connection line has the highest F1-score, and the continuous lines have almost the same F1-score as the total. The post-processing does have a limited effect on the results but results in slightly higher scores. The highest performance is at a confidence threshold of 0.85.

	F1-score Low keypoint Out				F1-score High Keypoint Out			
Score Thres	Total	Junction	Text	Symbol	Total	Junction	Text	Symbol
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.95	19.99	47.50	8.90	0.00	20.79	51.25	9.31	0.00
0.9	50.68	79.58	63.55	1.21	52.34	84.64	65.46	1.21
0.85	61.21	86.37	77.41	13.72	63.17	91.58	79.56	14.00
0.8	70.49	89.22	83.05	38.69	72.75	94.49	85.28	39.41
0.7	82.68	91.39	89.12	69.80	85.22	96.61	91.39	70.95
0.6	88.84	92.42	93.01	82.93	91.51	97.57	95.23	84.34
0.5	91.79	93.01	95.99	87.86	94.54	98.16	98.20	89.46
0.4	92.75	93.40	96.96	**89.32**	95.63	98.52	99.12	91.28
0.3	**92.81**	93.67	97.17	89.15	96.21	**98.89**	**99.33**	92.32
0.2	92.55	**93.81**	**97.21**	88.45	**96.69**	98.54	99.21	93.79
0.1	92.03	93.28	97.05	87.71	95.60	94.68	98.20	**94.48**
0	82.66	73.60	90.84	85.78	82.79	68.61	89.63	92.48

Table 6.8: These are F1-scores for the keypoint detections for different confidence score thresholds. On the left side of the table are the scores for a max keypoint out of 350 and the right side of the table is for a max keypoint out of 600. The distance threshold was set at 2.0 on a scale of 256×256

To increase the performance of the keypoint predictions it was beneficial to increase the number of outputted keypoints from 350 to 600. This parameter limits the number of keypoints while only keeping the predictions with the highest confidence score. This lower limit decreases the computational load.

Keypoint prediction outcomes are enumerated in Table 6.8. On the left side of the table the results of keypoint prediction when the max output of keypoints is 350 and on the right side the max keypoint output is 600. It is evident from the data that by increasing the maximum of outputted keypoints the F1-score increases up to 4%. Detection of text locations yields a robust F1-score, while symbols yield the lowest. This is likely attributed to the diversity of symbols and the inherent class imbalance present within the dataset. Collectively, these factors culminate in an optimal F1-score of 96.69 at a confidence threshold of 0.2.

The post-processing had a negative effect on the model's keypoint detection results. For that reason, the results are not included in the overview. Increasing the line output did not affect the line detection results.

The model's performance reaches its peak when each line and keypoint class is precisely adjusted to its respective optimal confidence score threshold. Notably, this threshold is discernibly lower for keypoints than for lines. The performance distribution for symbol detection is illustrated in the confusion matrix provided in Figure 6.3, offering a visual representation of the model's classification accuracy across different symbol categories.

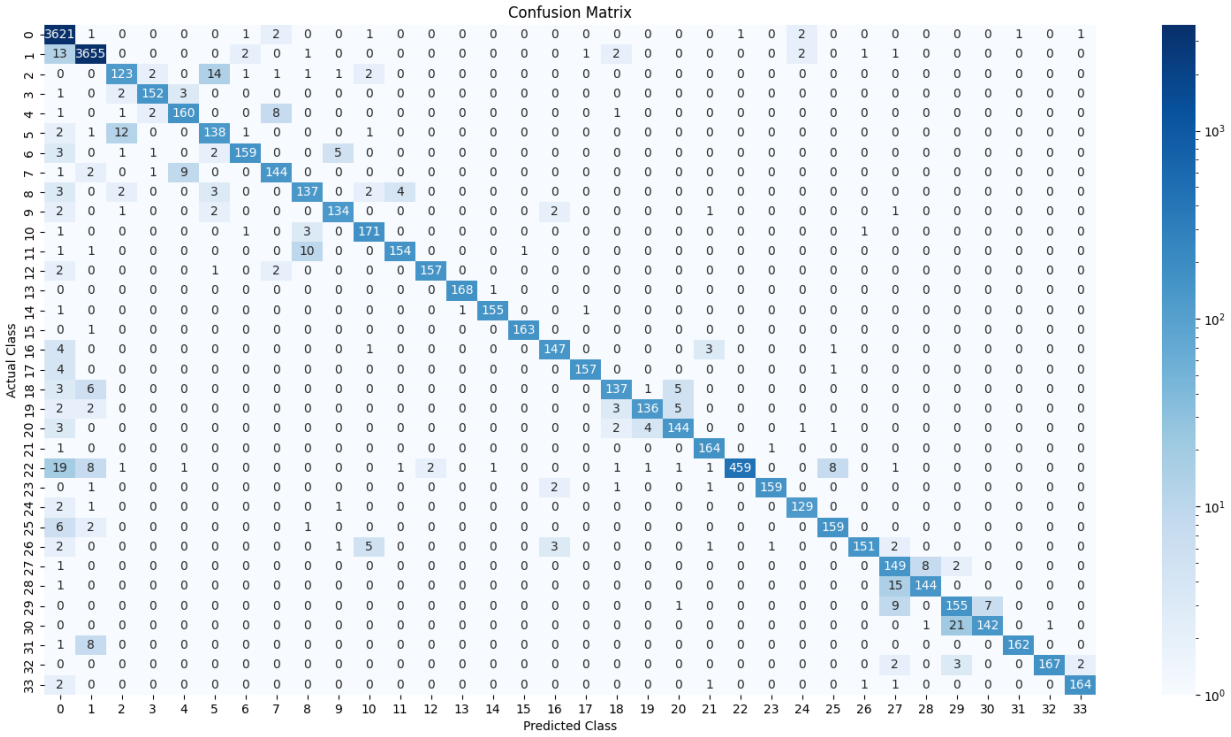

Figure 6.3: The confusion matrix for keypoint detection in the test set

6.5. Comparison

The model has been configured and fine-tuned for optimal performance. Now, PandID-Net will be benchmarked against the work of Paliwal et al. [25], focusing on comparisons in line detection and keypoint detection. The method of Paliwal et al. [25] is described in Section 2.1.2. The line detection is performed with a rule-based method and for symbol detection, they use a combination of a learned model and image processing techniques.

The metrics used are determined by what was published in the paper and differ per category. First, the line detection results are compared, and later the keypoint detection. Paliwal et al. [25] tested on 100 images from the Digitze-PID dataset and our method on 50 images of that dataset.

	PandID-Net				Paliwal et al. [25]	
Line Type	Precision	Recall	F1-score	Accuracy	correct	Accuracy
Dashed	86.67	95.28	90.24	**82.93**	20620/24848	82.91
Continuous	94.26	90.76	92.41	86.06	90774/91416	**99.34**
Relational	98.60	93.36	95.86	**92.13**	-	-
Total	93.19	92.69	92.89	86.81	111394/116264	**91.1**

Table 6.9: Comparison of line detection between PandID-Net and the method of Paliwal et al. [25]. Results are for a distance threshold of 5 on a scale of 256×256 and a confidence threshold of 0.85

In Table 6.9, we present the line detection performance of our PandID-Net and the approach by

Paliwal et al. [25]. Noticeable in the results is the high F1-score for relational lines. Our model finds it most challenging to classify the dashed lines correctly but the results are slightly better than the previous method. Although underperforming compared to the rule-based method the continuous lines still have an F1-score of 92.41.

The methodology employed by Paliwal et al. [25] quantifies accuracy by the ratio of correctly detected lines to the total number of ground truth lines. While this yields high accuracy figures, it fails to account for the rate of false positives or the instances of misclassification. The comparison would have been more relevant if the F1-score of the line detection had been published. A visual comparison of the line detection outputs from both methods is illustrated in Figure 6.4. The figures are from different P&ID images.

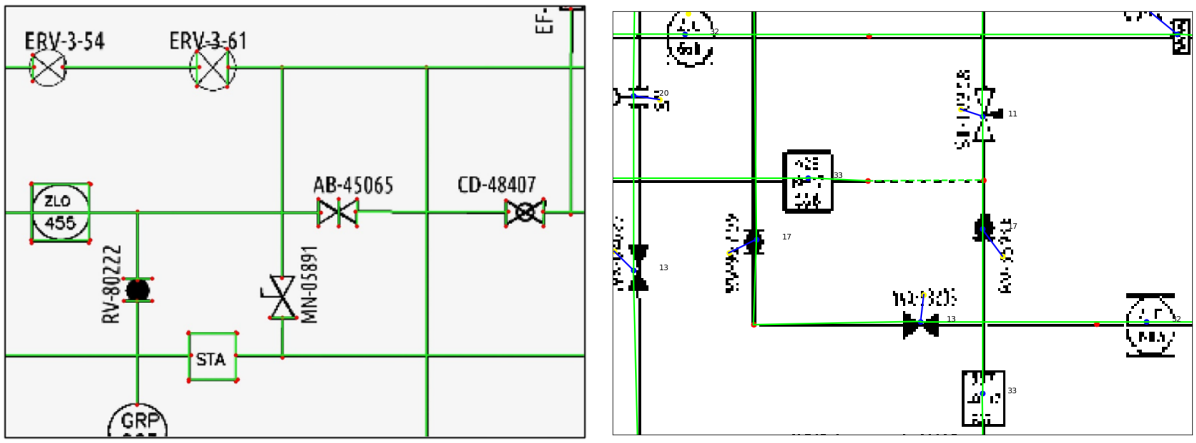

Figure 6.4: These are two snippets of the results of the two methods. The snippets are from different P&IDs. The left image shows the output using a structuring element [25] and the right image shows the line detection using PandID-Net.

	PandID-Net			Paliwal et al. [25]			
Keypoint Type	Precision	Recall	F1-Score	Precision	Recall	F1-Score	accuracy
Junctions	99.68	98.15	98.89	-	-	-	-
Text	99.59	99.09	99.33	-	-	-	87.2
Symbol	95.31	93.69	94.48	92	93	92.2	-
Total	97.50	95.91	96.69	-	-	-	-

Table 6.10: The comparison of the keypoint detections between our method and the method of Paliwal et al. [25]

In Table 6.10, the evaluation of keypoint detections in terms of precision, recall, and F1-score is presented. A noteworthy observation looking at the results of our model is the high precision associated with junctions and text locations, indicating a minimized occurrence of false positives. Furthermore, an improved performance in text detection is evident when compared to prior work. Since the work of Paliwal et al. [25] does not make use of the junction detection this could not be compared.

The performance metrics for individual symbol classes, including precision, recall, and F1-score, are detailed in Table 6.11. The highest values in each metric are highlighted in bold. The total scores are derived from the sum of true positives, false negatives, and false positives of all the symbol classes combined, essentially representing a form of weighted average. It is important to note that Paliwal et al. [25] did not report these weighted scores, hence the usage of average scores for comparative analysis.

In 18 of the 32 individual symbol classes, the method by Paliwal et al. [25] demonstrates better F1-scores and slightly exceeds PandID-Net in average F1-score with 1.2%. Comparing symbols 26-32 where the method of Paliwal et al. [25] uses image processing techniques our method has a higher F1-score for 4 out of the 7 symbols.

When examining the total precision, PandID-Net achieves a higher score. This outcome is indicative of a more precise yet conservative approach in symbol prediction, characterized by higher precision but reduced recall. Such a pattern suggests that while PandID-Net is adept at accurately identifying symbols, it is more prone to overlook actual symbols, thereby incurring a higher false negative rate.

	PandID-Net			Paliwal et al. [25]		
Symbol	Precision	Recall	F1-score	Precision	Recall	F1-score
1	0.858	0.840	0.833	0.932	0.882	**0.906**
2	0.957	0.934	0.939	0.968	0.968	**0.968**
3	0.895	0.909	0.894	0.965	0.847	**0.902**
4	0.846	0.882	0.851	0.974	0.904	**0.938**
5	0.937	0.892	0.909	0.986	0.973	**0.979**
6	0.857	0.847	0.846	0.978	0.967	**0.972**
7	0.903	0.868	0.876	0.971	0.911	**0.940**
8	0.906	0.886	**0.892**	0.823	0.963	0.888
9	0.869	0.881	**0.872**	0.772	0.986	0.866
10	0.976	0.941	0.953	0.974	0.958	**0.966**
11	0.933	0.916	**0.923**	0.741	0.991	0.848
12	0.953	0.944	**0.947**	0.875	0.793	0.832
13	0.910	0.907	0.907	0.972	0.938	**0.955**
14	0.975	0.966	**0.970**	0.916	0.961	0.938
15	0.897	0.901	0.895	0.947	0.997	**0.971**
16	0.972	0.961	**0.965**	0.979	0.941	0.960
17	0.883	0.855	0.861	0.813	0.979	**0.888**
18	0.900	0.899	0.888	0.946	0.993	**0.969**
19	0.918	0.871	**0.883**	0.946	0.724	0.820
20	0.939	0.944	0.941	0.962	0.929	**0.945**
21	0.985	0.877	0.924	0.876	0.988	**0.929**
22	0.971	0.957	**0.962**	0.936	0.946	0.941
23	0.952	0.941	**0.944**	0.881	0.956	0.917
24	0.943	0.893	0.912	0.977	0.965	**0.971**
25	0.963	0.894	**0.916**	0.927	0.743	0.825
26	0.843	0.878	0.849	0.893	0.937	**0.914**
27	0.937	0.917	**0.922**	0.864	0.903	0.883
28	0.866	0.893	0.872	0.961	0.975	**0.968**
29	0.880	0.816	0.838	0.977	0.984	**0.980**
30	1.000	0.951	**0.971**	0.890	0.912	0.901
31	0.996	0.967	**0.979**	0.904	0.892	0.898
32	0.951	0.945	**0.947**	0.923	0.948	0.935
Total	**0.953**	**0.937**	**0.945**	-	-	-
Average	**0.925**	0.906	0.910	0.920	**0.930**	**0.922**

Table 6.11: The comparison of the symbol recognition between PandID-Net and Paliwal et al. [25]. The results of PandID-Net are with a threshold of 2 and a confidence threshold of 0.3. The total of the results of Paliwal et al. [25] are the average over all the symbols. The total of the results of PandID-Net is the average weighted by occurrence

In summary, PandID-Net shows very constant performance for line detection across different classes with a total F1-score of 92.89. Compared to the rule-based line detection of Paliwal et al. [25] the continuous line detection needs to be improved. The high performance on dashed and connection line detection indicates that the line classification model can be extended with all the different types of lines that real-world P&IDs contain.

The keypoints are detected by PandID-Net with a total F1-score of 96.69 The locations of the text are detected with a respectable F1-score of 99.33. Paliwal et al.'s [25] methodology establishes a strong baseline for symbol detection, particularly with its higher F1-scores across many symbol classes. Nonetheless, the high total precision of PandID-Net in symbol and text detection it positions itself as a formidable contender. The weighted scores further accentuate the efficacy of PandID-Net, giving a more accurate view of the overall performance.

Chapter 7.1 will dive deeper into the performance of the model, hypothesizing explanations for its specific behaviors.

7

Discussion and Conclusion

The aim of this thesis is to provide a combined solution for line recognition, symbol recognition, and text detection for the generation of MTOs. First, the state-of-the-art work was explored and a rule-based method was developed. This was a limited method due to the ever-increasing logic of dealing with the large diversity of P&ID image formats. A novel route is taken to replace the different detection components into a single architecture. This resulted in a network called PandID-Net. The experiments were conducted to explore the optimal configuration. In the comparison, the method turned out to be a promising alternative for analyzing P&IDs. But not without its challenges and limitations.

In Section 7.1 we will discuss the advantages and weaknesses of PandID-Net based on the results from the previous chapter. To begin, the general limitations of the research are discussed. The conclusion and answers to the research questions that were formulated in Chapter 1 will be given in Section 7.2. The last section will give recommendations for future work.

7.1. Discussion

In this section, the results of the PandID-Net are discussed. The advantages and challenges are covered together with some solution proposals. The section is split into a line detection and a keypoint detection part. It starts by discussing the general limitations of experimental research.

7.1.1. Research Limitations

This section addresses the general limitations of the experiments. They are primarily stemming from the use of a synthetic dataset, which lacks the complexity of real-world data. The dataset's simplified nature is evident in several aspects: it encompasses only basic line styles such as dashed, and continuous, and features standard line orientations such as horizontal, and vertical However, it fails to represent the diversity of line styles and orientations found in real-world P&IDs. An example of a legend sheet for lines can be found in figure 7.1. Additionally, the dataset does not account for common real-world challenges such as the partial obscuration of line numbers and symbols by "clouds". These are wavy lines that indicate changes made in the P&ID (depicted in Figure 7.2).

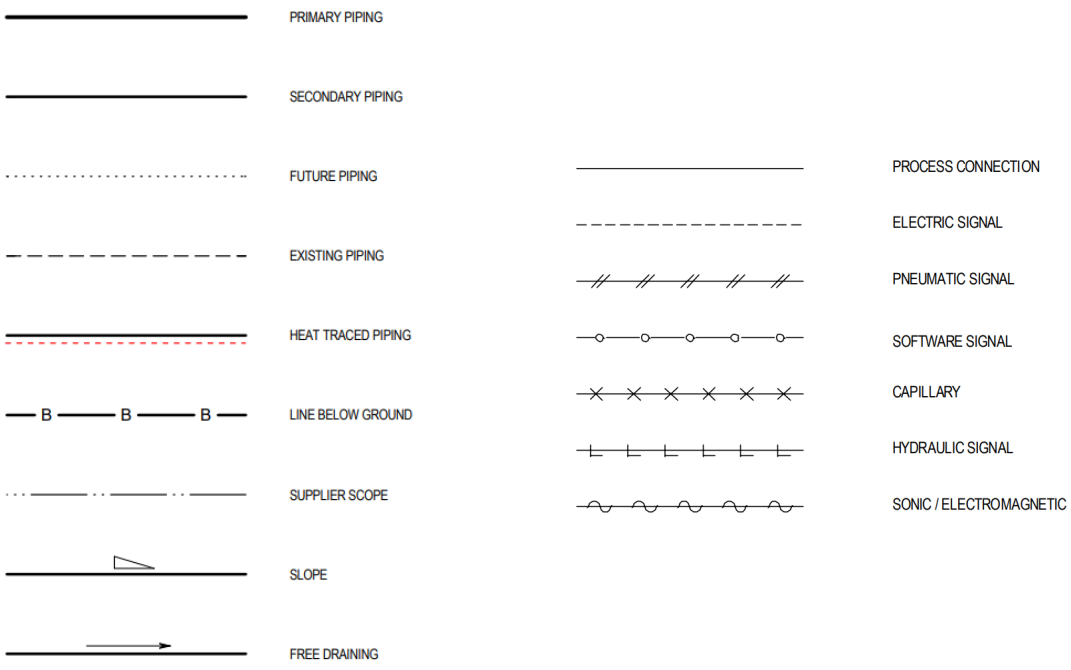

Figure 7.1: Legend of line types found in real-world P&IDs

Another challenge is the variations in line numbers and the variability in rules and conventions across different projects. A critical aspect not covered by the dataset is the interpretation of line interruptions, which can indicate either a crossing or a discontinuous line in real-life data. Consequently, while the synthetic dataset offers a controlled testing environment, its lack of real-world complexities limits the generalizability of our findings.

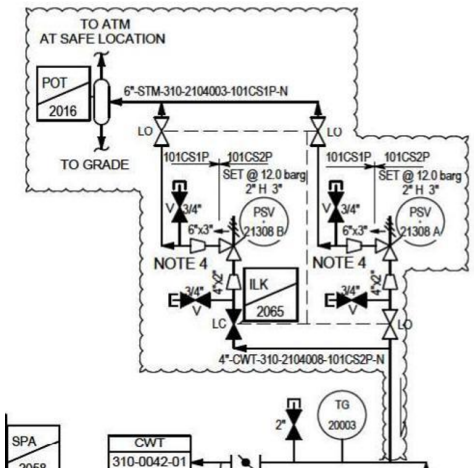

Figure 7.2: Example of cloud-like lines that indicate changes in real-world P&IDs

7.1.2. Analysis of Keypoint Detection Preformance

The keypoint detection in PandID-Net, with an overall F1-score of 96.69, demonstrates promising results. Despite some variability in class-specific results, particularly in symbol recognition with a lower F1-score of 94.48, the performance significantly surpasses McDermott's established minimum threshold of 70.

The weaker performance on symbol recognition could be attributed to the large class imbalance of the keypoint dataset. Additionally, the size of the Digitize-PID dataset is relatively small. Expanding the size of the dataset by more advanced data augmentation or adding more samples would probably

decrease the problem.

The change from a ”shallow” hourglass backbone to a ”deep” backbone did not significantly increase performance. However, further research could explore whether employing a more complex backbone, such as ResNet50 [9], could enhance symbol classification in P&IDs.

An interesting observation from the results is that the performance for predicting keypoints is better when the focal loss is used for the line classifier than when the cross-entropy loss is used. The results are shown in Table 6.2. This is interesting because the keypoint detection precedes line classification in the process flow. This suggests that by optimizing the line classification model with a different loss function, the network might learn better or more relevant features that indirectly enhance keypoint prediction accuracy. This indicates that PandID-Net benefits from multi-task learning, which adds to its advantages.

7.1.3. Analysis of Line Detection Performance

The line detection achieves an F1-score of 92.89. This leaves room for improvement. The use of focal loss for line detection had a significant improvement. Also, the deeper hourglass backbone showed an increase of 6% in average precision. The learning rate seemed to have a less significant effect.

A possible cause of suboptimal learning behavior is the resolution of the dataset. A problem that occurred when creating the dataset is that some lines disappeared when rescaling the image to 1024×1024. When downsizing an image with thin black lines against a large background, the reduction in available pixels can reach a point where there are not enough black pixels left to represent a line.

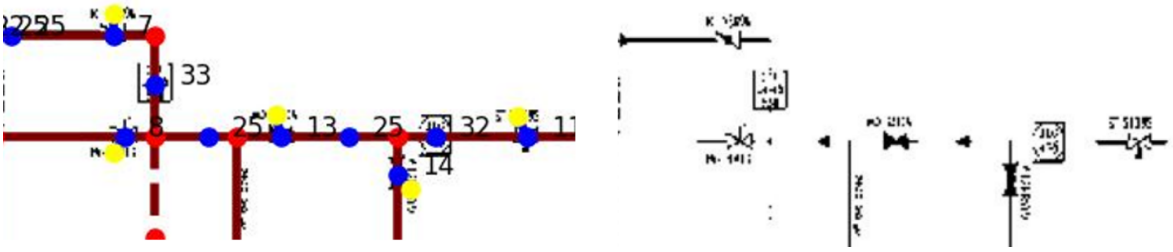

Figure 7.3: Example of a disappeared line due to resizing. On the left are the ground truth lines and on the right a snippet of the input image.

The option of increasing the input image was explored. The results can be found in Appendix A.3. The increased size of the images to 1792 ×1792 prevented the lines from disappearing which increased the performance of the continuous line detection. However, it did increase the overall complexity of the image. This caused the performance of symbol recognition to plummet. To resolve these problems more research is needed to find a balance between GPU load, complexity, and performance.

Another option is a technique called tiling where the image is kept at a high resolution and is divided into smaller chunks that are processed separately. This does require some logic to split and reassemble the annotations and results.

The highest detection results among all lines were observed for relational lines between text and symbols, achieving an F1-score of 95.86. This is an interesting result given that this is the only line, next to the no-line class, that does not have pixels in the P&ID image to indicate it is a line. This can maybe be attributed to the workings of the hourglass net. Since the backbone keeps the spatial information, the contention lines can be classified based on the relative proximity of the two associated keypoints. Another explanation could be that this class is the most consistent. Since the continuous lines and dashed lines have disappeared in some cases, the ground truth feature representation may not always be correct. For relational lines, there is no line to disappear and is thereby more consistent. A third reason could be that relational lines can be diagonal in contrast with dashed and continuous lines that are always either vertical or horizontal.

The high performance of the relational lines could indicate that there is no need to have an underlying line to learn a relation between two points. To create an MTO the line number and text near the symbol are the most important features of each symbol. If these connections can be learned without

using the line then a less annotated P&IDs with correct MTOs can serve as a dataset. This can save resources because annotating a P&ID dataset is costly.

Statistical analysis of the line detection results revealed abnormally small lines among the list of false negatives. After a thorough examination of the ground truth lines, it appeared that due to an error in the code that created the dataset, small lines were created between the text within a symbol and the symbol location. The lines are only a few pixels long and can barely be seen in the plots. This happened to some of the symbols from 25 to 32 of Figure 6.1. The results of the statistical analysis are in Appendix A.5.

This error caused noisy data, affecting the training of all models. The model was evaluated with these lines removed from the ground truth dataset. This increased the performance by a few percent. Without this noisy data, the model could potentially perform better.

Advantages of PandID-Net

PandID-Net is one model designed for three tasks that previously were performed separately. This is an advancement by making the model computationally efficient. The model is also capable of adapting to a dataset with more line classes. In real-world P&IDs there could be many different line types for piping and instrumentation.

The unified architecture also appears to benefit from multi-task learning, increasing the performance of subprocesses simultaneously. The shared feature map promotes the knowledge transfer between line and keypoint detection.

Moreover, it demonstrates consistent and high performance across different line classes with a total F1-score of 92.89 and excels in detecting dashed and connection lines. Its high precision in symbol and text detection makes it suitable for further investigation with real-world datasets.

Challenges

The way PandID-Net is programmed with some legacy subtasks and the large memory usage are points of improvement. Also, the model inherited has some legacy code such as a model that will also predict a heatmap for the lines. Several functions rely on for-loops. Rewriting the code to make the model more efficient could save GPU memory usage. Furthermore, there is room for improvement in continuous line detection compared to rule-based methods. The results for symbol detection are not yet on the level of the state-of-the-art DL methods.

7.2. Conclusion

To conclude this thesis the research questions of Chapter 1 are answered. First, the subquestions are answered in chronological order. The main question is the last question that is answered in this section.

What are the limitations and challenges in the existing line detection and association process?

In the field of research on line detection and association in P&IDs, a primary challenge is the diversity and complexity of P&ID image formats. Traditional rule-based have limitations in handling the wide variety of designs and styles present in P&IDs. These methods lack flexibility and scalability, and they struggle to adapt to different formats. Leading to systems that are either non-generalizable or have to deal with ever-increasing and complex logic. Maintaining and updating rule-based systems is complex and labor-intensive, especially as new variations in data require additional rules or modifications. Unlike machine learning models, these methods do not learn or improve over time with new data. Furthermore, rule-based approaches often fail to handle the variability and subtleties in real-world data, this can especially be a problem when working with scanned P&ID images. With the rapid advancements in deep learning providing more robust and efficient solutions, the reliance on rigid, rule-based systems is being re-evaluated. This shift highlights the need for more adaptable and sophisticated approaches in the research and application of line detection and association in P&IDs.

In what ways can deep learning techniques be effectively applied to line detection and association in Piping and Instrumentation Diagrams?

Deep learning can be applied for line detection and association by implementing PandID-Net. This comprehensive model integrates symbol recognition, line recognition, and text detection into a unified framework. It streamlines the processing pipeline, reducing the complexity of handling separate models

for each task. Thereby utilizing the benefits of multitask learning. The advantages of using deep learning for line detection are that the set of different line types can be large and the model can be generalizable across different projects.

What metrics can be used to evaluate the effectiveness of the improved processing pipeline for line detection and association?
Standardization of evaluation metrics, with a focus on the F1-score, is advocated in this thesis. The F1-score is recommended as the main metric. Its balanced approach to precision and recall makes it ideal for aiming at the lowest possible error rate, without overly focusing on either false negatives or false positives.

Also, this thesis vouches for open datasets and promotes competition. Together with common metrics, this can stimulate progress in the field. While respecting IP concerns, companies should recognize the mutual benefits of sharing and standardizing data. This collaborative approach would enable a more objective comparison of methodologies and bring advancements in P&ID analysis, ultimately contributing to the development of more effective, efficient, and accurate processing models suitable for industrial applications.

Could the proposed enhancements to the processing pipeline be applied to improve diagram interpretation across different industries? If so, how?
The advancements in line detection and association processing pipelines have potential applications beyond P&IDs. Adaptation of these improvements for other types of engineering diagrams, like wiring schematics, or engineering drawings is feasible. This potential extends the model's utility across various engineering domains, and when trained on a diverse range of datasets can possibly lead to improved performance. This improvement could come from the principles of transfer learning, where a model trained on diverse data acquires a more robust and generalized understanding, beneficial across multiple contexts.

In what ways can the processing pipeline effectively be improved for line detection and association in Piping and Instrumentation Diagrams?
The processing pipeline for the generation of MTOs from P&IDs can be improved by introducing Deep Learning for line detection and combining it with text and symbol detection to create a single model. This model called PandID-Net is trained on the Digitize-PID dataset from Paliwal et al. [25]. It achieves competitive results with an F1 score of 92.89 and 94.48 for line detection and keypoint detection respectively.

The integration of the different detection components not only simplifies the pipeline but also allows for multitask learning. Making the model computational more efficient. Furthermore, while the current results are promising, continued refinement of the model, particularly in handling diverse and complex real-world P&ID layouts, could further enhance its accuracy and reliability. Future enhancements could also explore the scalability of the model to accommodate larger datasets and its adaptability to different types of P&ID designs, ensuring its applicability across various engineering contexts.

7.3. Recommendations

This section presents recommendations for future research in this field. These first suggestions are designed to build upon the findings of the current study. The last suggestion is to explore alternative architectural approaches.

For future research, several practical advancements are proposed to enhance the functionality and real-world application of our model. The first step is to train the model using real-world P&IDs and evaluate how well it performs in an operational context. This study used the only dataset that was publicly available to train the model. Since this is a synthetic dataset it is not fully representative of real-world P&IDs

Secondly, the PandID-Net should be integrated into a pipeline, completing the development of an end-to-end system that can automatically generate MTOs from P&IDs. This includes incorporating a proven Optical Character Recognition (OCR) tool like Tesseract to interpret the text. Also, some logic is needed to create the MTO spreadsheet in the desired format.

Since there has been a lot of research in symbol detection in P&IDs, replacing the model's core architecture with well-established neural networks might also improve its ability to process and analyze complex images even further

Employing a technique called tiling, which breaks down large diagrams into smaller pieces, could help manage and analyze large P&IDs more at a higher resolution while also decreasing the model size. This could resolve the problem of disappearing lines.

The end goal is to create a zero-error process for creating MTOs. Achieving this is challenging due to technical and methodological limitations. This creates the need for a human-supervised process. An interactive application should be developed, facilitating human oversight and data collection for reinforcement learning. Which streamlines the process, and promises substantial time savings even with the need for manual review.

Lastly, there is a need to investigate the possibilities for a new model architecture that learns to associate structures in P&IDs directly with MTO, eliminating the reliance on annotated lines as a source of ground truth. The high performance of the relational lines without the necessity for an underlying physical line underscores the potential of this approach. It suggests that the mere knowledge of point locations and their relationships could be sufficient for the model to learn effectively. This learning process could be achieved using merely annotated text and symbols in conjunction with a verified MTO. This could provide a groundbreaking alternative to current methods and remove the need for on labor-intensive, detailed annotated datasets.

A

Appendix

A.1. Abbreviations

3D	Three Dimensional
AI	Artificial Intelligence
AP	Average Precision
CE	Cross-Entropy
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
DL	Deep Learning
EPC	Engineering, Procurement, and Construction
FCN	Fully Convolutional Network
FN	False Negative
FP	False Positive
GPU	Graphic Processing Unit
IP	Intellectual property
IOU	Intersect Over Union
JDN	Junction Detection Network
LPN	Line Proposal Network
LoI	Line of Interest
ML	Machine Learning
mAP	Mean Average Precision
MTO	Material Take-Off
NMS	Non-Maximum Suppression
NN	Neural Network
OCR	Optical Character Recognition
P&ID	Piping and Instrumentation Diagram
RAM	Random Access Memory
sAP	Structural Average Precision
SGD	Stochastic Gradient Descent
TN	True Negative
TP	True Positive

A.2. Software Framework and Libraries

The experimental phase was underpinned by a suite of software packages and libraries vital for the creation and evaluation of our neural network models. The following is a catalog of these tools, complete with version specifications, which constituted the foundation of our computational setup:

- TensorFlow 2.13.0: A comprehensive, flexible ecosystem of tools, libraries, and community resources that enables researchers to build and deploy machine learning applications.

- PyTorch 2.0.1: An open-source machine learning library based on the Torch library, used for applications such as natural language processing.

- Scikit-Learn 1.3.2: A simple and efficient tool for data mining and data analysis built on NumPy, SciPy, and matplotlib.

- NumPy 1.24.3: A library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- Matplotlib 3.8.0: A plotting library for the Python programming language and its numerical mathematics extension NumPy.

- OpenCV 4.8.0.74: An open-source computer vision and machine learning software library.

- Pandas 1.5.3: A software library written for the Python programming language for data manipulation and analysis.

These tools were carefully chosen for their reliability, acceptance in the broader machine learning sphere, and their seamless integration with our chosen hardware infrastructure.

A.3. Larger Input Images

To solve the problem of the disappearing lines we created a dataset of images with size 1792 by 1792. At this resolution, the lines did not disappear. The same heatmap dimension of 256 by 256 was used. To make this possible the legacy code of the lmap was removed to free up space.

The results of this model are in the following tables:

	F1-score 1024x1024				F1-score 1536x1536				F1-score 1792x1972			
Score Thres	Total	Dashed	Continous	Relational	Total	Dashed	Continous	Relational	Total	Dashed	Continous	Relational
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.95	88.67	0.00	88.65	95.17	85.69	0.00	84.23	94.77	85.08	0.00	83.44	94.47
0.9	92.40	58.44	**92.42**	95.64	89.64	37.44	88.89	**95.68**	90.00	45.30	89.12	**95.71**
0.85	**92.85**	**90.24**	91.79	**95.68**	**94.31**	**89.95**	**94.21**	95.45	**94.97**	**95.17**	**94.76**	95.50
0.8	91.64	82.17	90.63	95.47	93.83	86.98	93.91	95.15	94.76	93.90	94.62	95.34
0.7	90.61	77.18	89.63	95.25	93.41	85.18	93.55	94.88	94.48	92.91	94.33	95.22
0.6	90.60	77.18	89.63	95.24	93.41	85.18	93.55	94.90	94.48	92.91	94.33	95.22
0.5	90.60	77.18	89.63	95.24	93.41	85.18	93.55	94.90	94.48	92.91	94.33	95.22
0.4	90.60	77.18	89.63	95.24	93.41	85.18	93.55	94.90	94.48	92.91	94.33	95.22
0.3	90.60	77.18	89.63	95.24	93.41	85.18	93.55	94.90	94.48	92.91	94.33	95.22
0.2	90.60	77.18	89.63	95.24	93.41	85.18	93.55	94.90	94.48	92.91	94.33	95.22
0.1	90.60	77.18	89.63	95.24	93.41	85.18	93.55	94.90	94.48	92.91	94.33	95.22
0	90.60	77.18	89.63	95.24	93.41	85.18	93.55	94.90	94.48	92.91	94.33	95.22

Table A.1: These are the F1-scores per class and in total of the line predictions per confidence score threshold. The left side of the table is the resolution of 1024x1024 used in the rest of the thesis, in the middle is a resolution of 1536x1536, and on the right side is the resolution of 1792x1792. The distance threshold was set on 5 on a resolution of 256×256

	F1-score 1024x1024				F1-score 1536x1536				F1-score 1792x1972			
Score Thresh	Total	Juntion	Text	Symbol	Total	Juntion	Text	Symbol	Total	Juntion	Text	Symbol
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.95	20.79	51.25	9.31	0.00	33.04	73.59	19.41	0.00	16.60	47.18	1.29	0.00
0.9	52.34	84.64	65.46	1.21	51.43	91.27	54.58	0.04	39.19	81.84	27.87	0.00
0.85	63.17	91.58	79.56	14.00	58.95	95.41	71.14	0.54	50.51	89.95	53.80	0.19
0.8	72.75	94.49	85.28	39.41	63.45	96.91	79.48	4.42	57.34	93.45	68.49	2.02
0.7	85.22	96.61	91.39	70.95	72.80	98.50	89.17	27.13	66.84	96.62	83.05	14.73
0.6	91.51	97.57	95.23	84.34	82.10	99.17	94.75	54.10	77.87	98.08	92.13	43.62
0.5	94.54	98.16	98.20	89.46	89.79	99.53	98.21	75.17	87.98	99.03	97.36	70.85
0.4	95.63	98.52	99.12	91.28	93.41	**99.69**	99.31	84.90	93.00	99.48	99.07	84.06
0.3	96.21	**98.89**	**99.33**	92.32	**94.11**	99.62	**99.59**	86.77	**94.18**	**99.61**	**99.53**	86.98
0.2	**96.69**	98.54	99.21	93.79	93.98	99.25	99.42	**86.87**	94.08	99.01	99.41	**87.29**
0.1	95.60	94.68	98.20	**94.48**	92.45	96.53	97.35	86.37	92.99	96.92	98.08	86.98
0	82.79	68.61	89.63	92.48	80.86	79.74	83.03	80.44	81.31	82.48	80.17	81.80

Table A.2: These are the F1-scores per class and in total of the keypoint predictions per confidence score threshold. The left side of the table is the resolution of 1024x1024 used in the rest of the thesis, in the middle is a resolution of 1536x1536, and on the right side is the resolution of 1792x1792. The distance threshold was set on 2.0 on a resolution of 256×256

In Tables A.2 and A.1 the results for a high-resolution input and a resolution of 1024×1024 are shown. This test has the purpose of seeing if the disappearing lines were the reason that the continuous line detection underperformed. What can be seen is that at a higher resolution, the detection of continuous lines and dashed lines increases slightly with 1.79 % and 0.29% respectively.

However, the results of the keypoint detection plummeted with this higher resolution input. The results for the junction and text recognition stay roughly the same while symbol recognition f1 scores decreased enormously with 7.61 %.

These findings underscore the complexity inherent in optimizing neural network models for image analysis tasks, particularly when dealing with varied object types such as lines and symbols. They also highlight the importance of carefully considering the resolution in relation to the specific features and tasks being targeted by the model.

A.4. Results on real-world P&IDs

Figures A.1 and A.2 show the results of PandID-Net on real-world P&IDs given by McDermott. It shows a poor performance on this unseen and slightly different data. The P&IDs did not have the same scale. Interestingly the text locations of the sizes next to the symbols are recognized and the relational lines are constructed.

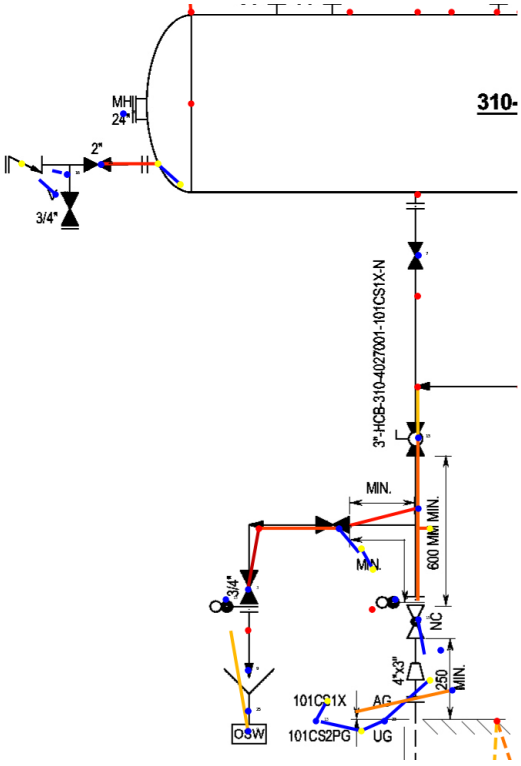

Figure A.1: Results of PandID-Net on real-world P&IDs

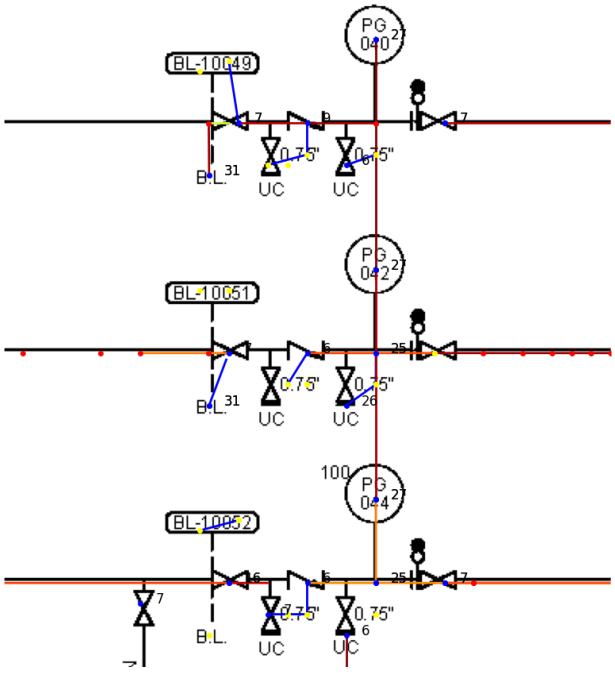

Figure A.2: Results of PandID-Net on real-world P&IDs

A.5. Statistical Analysis

In Figures A.3 and A.4 the distributions of the line length and the X, and Y locations of the lines are given per set. It can be seen that a large amount of lines in the FN set is very small. Further investigation

concluded that the dataset contained inaccurate ground truth lines. The histograms are the sum of the results of the test set using the optimal configuration of PandID-Net.

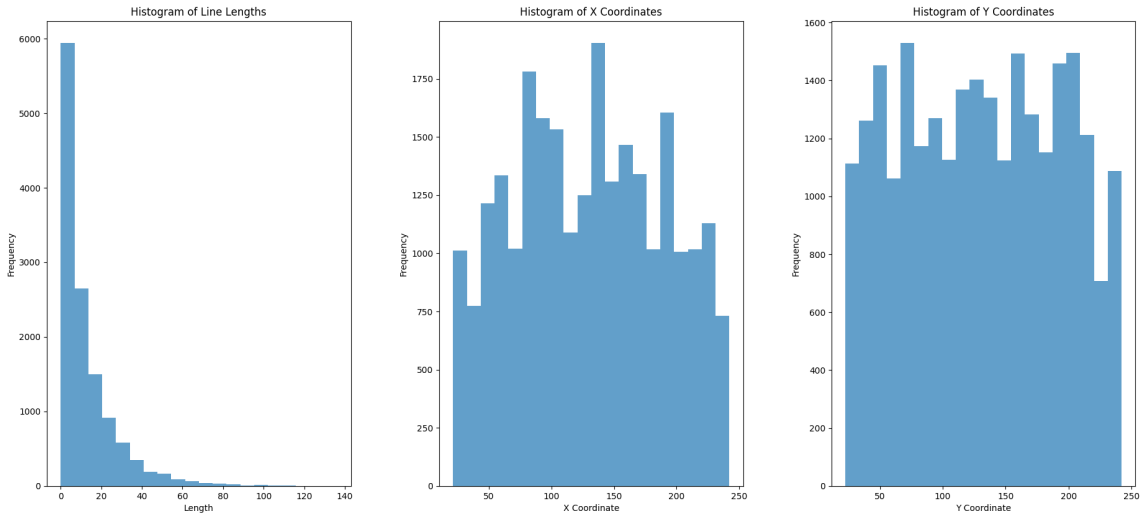

Figure A.3: The distribution of line length, x and y coordinate location of the ground truth lines.

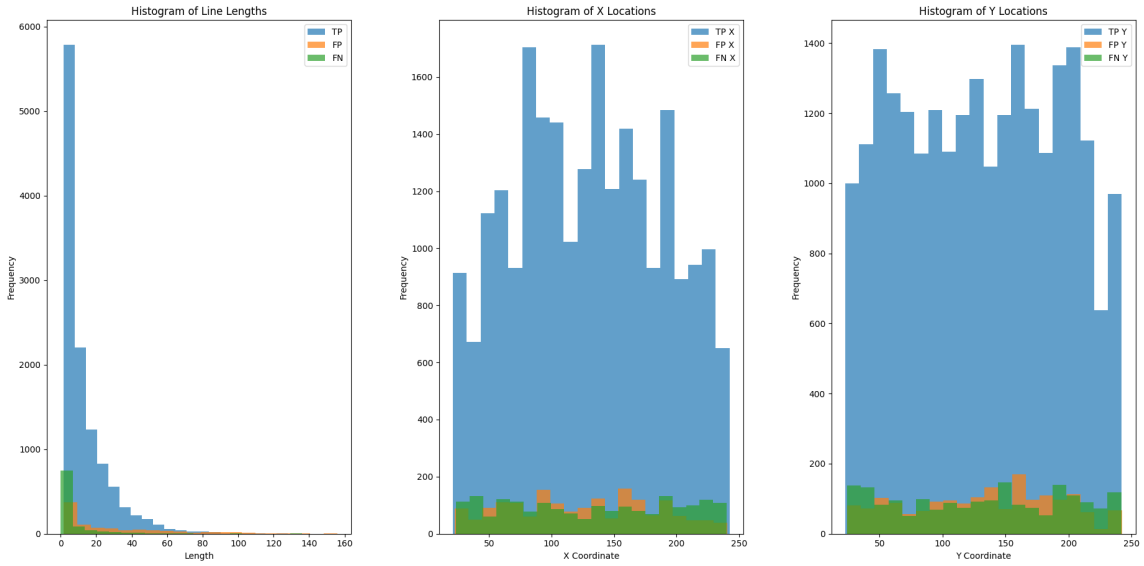

Figure A.4: The distribution of line length, x and y coordinate location of the TP, FN, and FP lines.

Bibliography

[1] Esteban Arroyo et al. “Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams”. en. In: *Computers & Chemical Engineering* 92 (Sept. 2016), pp. 112–132. issn: 0098-1354. doi: 10.1016/j.compchemeng.2016.04.040. url: https://www.sciencedirect.com/science/article/pii/S0098135416301363 (visited on 05/24/2023).

[2] Youngmin Baek et al. *Character Region Awareness for Text Detection*. arXiv:1904.01941 [cs]. Apr. 2019. doi: 10.48550/arXiv.1904.01941. url: http://arxiv.org/abs/1904.01941 (visited on 11/06/2023).

[3] John Canny. “A Computational Approach To Edge Detection”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-8 (Dec. 1986), pp. 679–698. issn: 9780080515816. doi: 10.1109/TPAMI.1986.4767851.

[4] Xili Dai et al. *Fully Convolutional Line Parsing*. arXiv:2104.11207 [cs] version: 3. Dec. 2022. url: http://arxiv.org/abs/2104.11207 (visited on 09/27/2023).

[5] Rimma Dzhusupova et al. “Using artificial intelligence to find design errors in the engineering drawings”. en. In: *Journal of Software: Evolution and Process* n/a.n/a (). _eprint: https://onlinelibrary.wiley.com/doi/pdf/1 e2543. issn: 2047-7481. doi: 10.1002/smr.2543. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2543 (visited on 05/01/2023).

[6] Martin Ester and Hans-Peter Kriegel. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. en. In: ().

[7] Takashi Futatsumata et al. “Development of an Automatic Recognition System for Plant Diagrams”. In: 1990. url: https://www.semanticscholar.org/paper/Development-of-an-Automatic-Recognition-System-for-Futatsumata-Shichino/da4ebf95426864fd1d625533e2de (visited on 05/24/2023).

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[9] Kaiming He et al. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385 [cs] version: 1. Dec. 2015. url: http://arxiv.org/abs/1512.03385 (visited on 10/13/2023).

[10] Paul V. C. Hough. “Method and means for recognizing complex patterns”. US3069654A. Dec. 1962. url: https://patents.google.com/patent/US3069654A/en (visited on 05/19/2023).

[11] Jeremy Jordan. *Understanding learning rates and how it improves performance in deep learning*. 2023. url: https://www.jeremyjordan.me/nn-learning-rate/.

[12] Sung-O. Kang, Eul-Bum Lee, and Hum-Kyung Baek. “A Digitization and Conversion Tool for Imaged Drawings to Intelligent Piping and Instrumentation Diagrams (P&ID)”. en. In: *Energies* 12.13 (Jan. 2019). Number: 13 Publisher: Multidisciplinary Digital Publishing Institute, p. 2593. issn: 1996-1073. doi: 10.3390/en12132593. url: https://www.mdpi.com/1996-1073/12/13/2593 (visited on 05/02/2023).

[13] Anthony Kay. “Tesseract: An Open-Source Optical Character Recognition Engine”. In: *Linux J.* 2007.159 (July 2007). Place: Houston, TX Publisher: Belltown Media, p. 2. issn: 1075-3583.

[14] Byung Chul Kim et al. “End-to-end digitization of image format piping and instrumentation diagrams at an industrially applicable level”. en. In: *Journal of Computational Design and Engineering* 9.4 (July 2022), pp. 1298–1326. issn: 2288-5048. doi: 10.1093/jcde/qwac056. url: https://academic.oup.com/jcde/article/9/4/1298/6611631 (visited on 05/02/2023).

[15] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980 [cs]. Jan. 2017. doi: 10.48550/arXiv.1412.6980. url: http://arxiv.org/abs/1412.6980 (visited on 11/22/2023).

[16] H. W. Kuhn. “The Hungarian method for the assignment problem”. en. In: *Naval Research Logistics Quarterly* 2.1-2 (1955). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109, pp. 83–97. issn: 1931-9193. doi: 10.1002/nav.3800020109. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109 (visited on 11/14/2023).

[17] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. arXiv:1708.02002 [cs]. Feb. 2018. doi: 10.48550/arXiv.1708.02002. url: http://arxiv.org/abs/1708.02002 (visited on 10/28/2023).

[18] Yancong Lin, Silvia L. Pintea, and Jan C. van Gemert. “Deep Hough-Transform Line Priors”. en. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 323–340. isbn: 978-3-030-58542-6. doi: 10.1007/978-3-030-58542-6_20.

[19] Shouvik Mani et al. “Automatic Digitization of Engineering Diagrams using Deep Learning and Graph Search”. en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA: IEEE, June 2020, pp. 673–679. isbn: 978-1-72819-360-1. doi: 10.1109/CVPRW50498.2020.00096. url: https://ieeexplore.ieee.org/document/9151021/ (visited on 05/05/2023).

[20] Yoochan Moon et al. “Deep Learning-Based Method to Recognize Line Objects and Flow Arrows from Image-Format Piping and Instrumentation Diagrams for Digitization”. en. In: *Applied Sciences* 11.21 (Oct. 2021), p. 10054. issn: 2076-3417. doi: 10.3390/app112110054. url: https://www.mdpi.com/2076-3417/11/21/10054 (visited on 05/04/2023).

[21] Yoochan Moon et al. “Extraction of line objects from piping and instrumentation diagrams using an improved continuous line detection algorithm”. en. In: *Journal of Mechanical Science and Technology* 37.4 (Apr. 2023), pp. 1959–1972. issn: 1738-494X, 1976-3824. doi: 10.1007/s12206-023-0333-9. url: https://link.springer.com/10.1007/s12206-023-0333-9 (visited on 05/02/2023).

[22] Carlos Moreno-García, Eyad Elyan, and Chrisina Jayne. *Heuristics-Based Detection to Improve Text/Graphics Segmentation in Complex Engineering Drawings*. Aug. 2017. doi: 10.1007/978-3-319-65172-9_8.

[23] Carlos Francisco Moreno-García, Eyad Elyan, and Chrisina Jayne. “New trends on digitisation of complex engineering drawings”. en. In: *Neural Computing and Applications* 31.6 (June 2019), pp. 1695–1712. issn: 1433-3058. doi: 10.1007/s00521-018-3583-1. url: https://doi.org/10.1007/s00521-018-3583-1 (visited on 05/10/2023).

[24] Alejandro Newell, Kaiyu Yang, and Jia Deng. *Stacked Hourglass Networks for Human Pose Estimation*. arXiv:1603.06937 [cs]. July 2016. doi: 10.48550/arXiv.1603.06937. url: http://arxiv.org/abs/1603.06937 (visited on 10/13/2023).

[25] Shubham Paliwal et al. “Digitize-PID: Automatic Digitization of Piping and Instrumentation Diagrams”. en. In: *Trends and Applications in Knowledge Discovery and Data Mining*. Ed. by Manish Gupta and Ganesh Ramakrishnan. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 168–180. isbn: 978-3-030-75015-2. doi: 10.1007/978-3-030-75015-2_17.

[26] Rohit Rahul et al. *Automatic Information Extraction from Piping and Instrumentation Diagrams*. en. arXiv:1901.11383 [cs]. Jan. 2019. url: http://arxiv.org/abs/1901.11383 (visited on 05/02/2023).

[27] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. *On the Convergence of Adam and Beyond*. arXiv:1904.09237 [cs, math, stat]. Apr. 2019. doi: 10.48550/arXiv.1904.09237. url: http://arxiv.org/abs/1904.09237 (visited on 11/23/2023).

[28] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. arXiv:1609.04747 [cs]. June 2017. doi: 10.48550/arXiv.1609.04747. url: http://arxiv.org/abs/1609.04747 (visited on 11/22/2023).

[29] Florian Stinner et al. *Automatic digital twin data model generation of building energy systems from piping and instrumentation diagrams*. arXiv:2108.13912 [cs]. Aug. 2021. doi: 10.48550/arXiv.2108.13912. url: http://arxiv.org/abs/2108.13912 (visited on 05/05/2023).

[30] Nan Xue et al. *Holistically-Attracted Wireframe Parsing*. arXiv:2003.01663 [cs]. Mar. 2020. url: http://arxiv.org/abs/2003.01663 (visited on 09/27/2023).

[31] Eun-Seop Yu et al. “Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network”. en. In: *Energies* 12.23 (Jan. 2019). Number: 23 Publisher: Multidisciplinary Digital Publishing Institute, p. 4425. issn: 1996-1073. doi: 10.3390/en12234425. url: https://www.mdpi.com/1996-1073/12/23/4425 (visited on 05/11/2023).

[32] Fan Zhang et al. *Three-branch and Mutil-scale learning for Fine-grained Image Recognition (TBMSL-Net)*. _eprint: 2003.09150. 2020.

[33] T. Y. Zhang and C. Y. Suen. “A fast parallel algorithm for thinning digital patterns”. In: *Communications of the ACM* 27.3 (1984), pp. 236–239. issn: 0001-0782. doi: 10.1145/357994.358023. url: https://dl.acm.org/doi/10.1145/357994.358023 (visited on 05/19/2023).

[34] Yichao Zhou, Haozhi Qi, and Yi Ma. *End-to-End Wireframe Parsing*. arXiv:1905.03246 [cs]. May 2021. url: http://arxiv.org/abs/1905.03246 (visited on 09/11/2023).