Automatic Digitization of Engineering Diagrams using Deep Learning and Graph Search

Shouvik Mani, Michael A. Haddad, Dan Constantini, Willy Douhard, Qiwei Li, Louis Poirier
C3.ai
{firstname.lastname}@c3.ai

Abstract

A Piping and Instrumentation Diagram (P&ID) is a type of engineering diagram that uses symbols, text, and lines to represent the components and flow of an industrial process. Although used universally across industries such as manufacturing and oil & gas, P&IDs are usually trapped in image files with limited metadata, making their contents unsearchable and siloed from operational or enterprise systems. In order to extract the information contained in these diagrams, we propose a pipeline for automatically digitizing P&IDs. Our pipeline combines a series of computer vision techniques to detect symbols in a diagram, match symbols with associated text, and detect connections between symbols through lines. For the symbol detection task, we train a Convolutional Neural Network to classify certain common symbols with over 90% precision and recall. To detect connections between symbols, we use a graph search approach to traverse a diagram through its lines and discover interconnected symbols. By transforming unstructured diagrams into structured information, our pipeline enables applications such as diagram search, equipment-to-sensor mapping, and asset hierarchy creation. When integrated with operational and enterprise data, the extracted asset hierarchy serves as the foundation for a facility-wide digital twin, enabling advanced applications such as machine learning-based predictive maintenance.

1. Introduction

Engineering designs, such as Piping and Instrumentation Diagrams (P&IDs), are used throughout the entire lifecycle of a facility. Created during the engineering and design phase, they are used to communicate requirements throughout the construction and operational phases of an industrial facility. In many jurisdictions, primarily for safety considerations, there are legal requirements to keep these designs up to date with any changes. Additionally, process engineers rely primarily on P&IDs to understand an industrial process and as a tool to plan and implement process changes. These factors make P&IDs a valuable, up to date ground-truth data source for the configuration of an industrial facility. However, P&IDs are often archived as hundreds of CAD files per facility with limited or no metadata describing the components, connections between components, or connections between P&IDs.

As IoT sensors have proliferated throughout industrial facilities they have become an important element of P&IDs. These sensors, commonly referred to as tags, provide critical, real-time operational data (e.g., temperature, pressure, flow, etc.) and are often connected to locally mounted instruments (LMIs) or installed directly onto equipment. This real-time operational data has become increasingly important for plant management, asset performance management, and process optimization. As machine learning techniques evolve to extract even more value from this data, contextualizing these tags within diagrams and referencing them to external sensor management systems is an increasingly critical pre-processing step. For an end user of an AI application to properly interpret machine learning predictions they must have a clearly defined asset hierarchy with sensor locations. For example, when implementing anomaly detection, a well-defined asset hierarchy will enable a user to localize an anomalous sensor within a facility and identify a particular component of the machinery.

Performing the contextualization step manually on P&IDs is time consuming and error prone. Our digitization pipeline aims to automate this step by combining state-of-the-art computer vision, graph search, and optical character recognition techniques to map the interconnectedness of equipment, pipelines, and sensors within P&IDs. The output of the pipeline, a high-fidelity asset hierarchy, will serve as the foundation for a digital twin used in a multitude of high-value business use cases for machine learning analytics.

P&IDs depict the configuration and properties of all equipment, components (such as valves and insulation), process lines, and instrumentation with standardized symbols related to an industrial process. Connections between

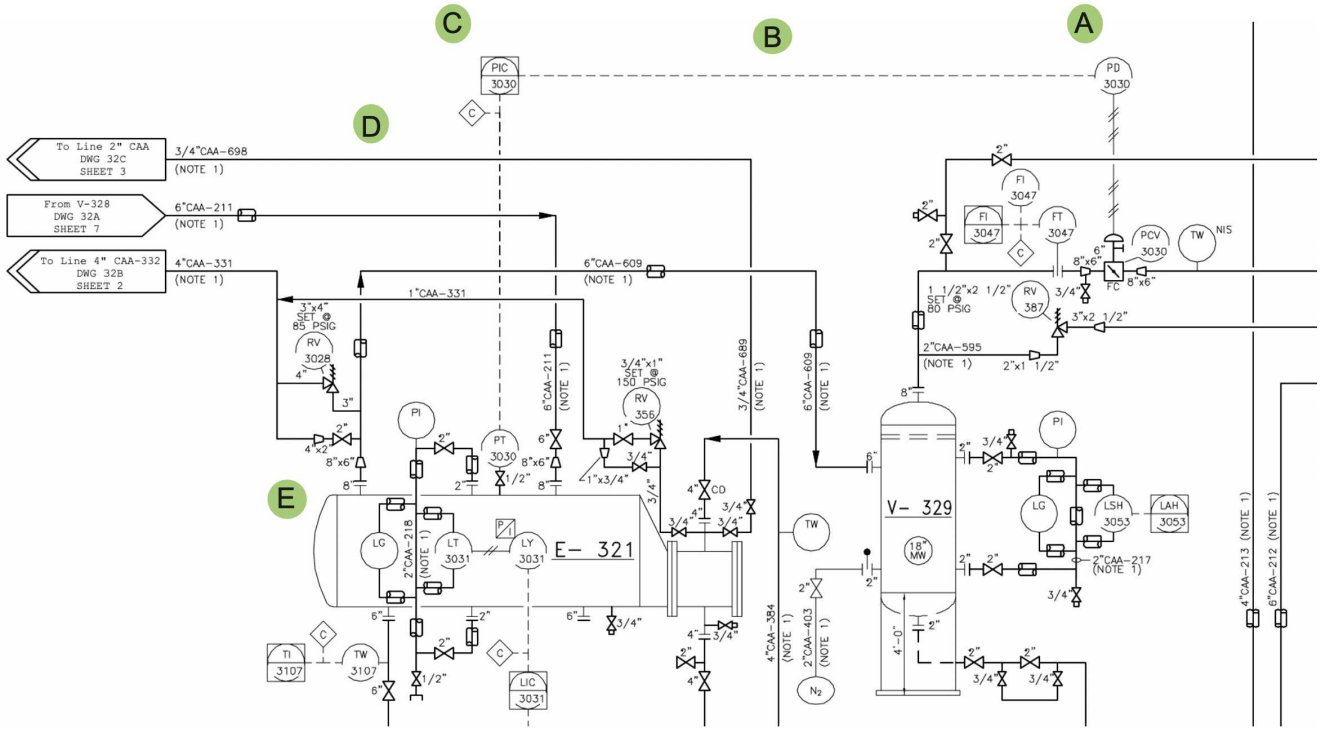

Figure 1. A section of a Piping and Instrumentation Diagram (P&ID). Instrumentation is represented by a circle. Text inside a circle indicates the instrument type and unique ID. Text adjacent to process lines and components specifies size and/or material.

P&IDs are marked as inputs or outputs to other equipment shown on separate P&IDs. Our digitization pipeline does not attempt to map all these connections and symbols, but rather focuses on a subset of equipment and instrumentation relationships that are of value for analytical applications. Symbols of interest for our digitization pipeline are shown in Figure 1: (A) locally mounted instrument (LMI) sensor for in-field readings, (B) electrical signal between instruments, (C) sensor (or tag) represented in a database, (D) process line, (E) equipment (such as a vessel, or pump).

2. Related work

Digitization of complex engineering diagrams has been an active area of research long before the recent adoption of deep learning techniques. In their survey paper, Moreno-García et al. describe the motivations and challenges in engineering diagram digitization and provide an extensive review of methods for the digitization problem [6]. The authors insist that the automatic analysis and processing of engineering diagrams is far from being complete and call for the application of deep learning in this domain. Additionally, they mention a set of contextualization challenges concerning the connectivity of symbols in diagrams which are not addressed by the current literature.

Most of the prior work on diagram digitization was prior to the popularization of deep learning and featured traditional computer vision techniques. The older methods achieved reasonable results but were limited in their application as they were inflexible across different diagram drafting standards and between companies. Our method is designed to be extendable to any diagram drafting standards and requires only a small number of labeled diagrams. Therefore, users can hand label a limited set of diagrams from any company and parse them effectively. Recent approaches have also leveraged deep learning for symbol and text detection with relatively high precision and recall. We extend upon these works by using a different CNN for symbol detection and a graph search approach to determine the interconnections between symbols.

While there exists a vast literature on symbol and text detection in engineering diagrams, there are relatively few examples that address the problem of connection detection. Some approaches use a simple Euclidean distance approach to assign symbols as connected symbols if they are within a predefined threshold distance of each other [3]. While this is sometimes a reasonable heuristic, it fails when applied to dense, complex P&IDs in which symbols may be close to each other but not connected via lines – the ultimate indicator of connectivity. To address this limitation, our pipeline features a graph search approach which traverses the diagram through its solid and dashed lines and only marks symbols as connected if there is a valid path between them.

A similar approach is implemented by Cardoso et al., who detect staff lines in musical scores by representing the music sheet image as a graph and identifying the shortest connected paths of black pixels in the image graph [1].

Most prior work in this field addressed a specific part of the digitization process such as symbol or text detection; an exception is the pipeline proposed by Rahul et al. which performs symbol detection, text detection, and symbol-to-line association [7]. Their pipeline uses state-of-the-art deep learning models such as the Connectionist Text Proposal Network (CTPN) for text detection and a Fully Convolutional Network (FCN) for symbol detection. While we share the common objective of completely digitizing P&IDs, our methods for symbol detection are noticeably different. Their pipeline uses an FCN which performs image segmentation to segment out symbols of interest using a 19-layer VGG-19 architecture which requires them 7000 epochs to train. Instead of doing segmentation, we classify symbols directly and use a much lighter CNN.

Other examples of pipelines include the system proposed by Kang et al., which uses relatively simple methods such as template matching to detect symbols and a sliding window approach to detect lines and text [4]. Although easy to implement, template matching approaches either fail to generalize well due to minor differences in visual appearances between symbols across diagrams or require a vast symbol library (i.e. the templates) to perform well. Finally, Daele et al. present a promising approach for similarity-based search in CAD drawings by developing a pipeline to extract components using image segmentation and object recognition, parse properties from tabular data in the drawings, and represent the extracted information in a feature vector to facilitate similarity comparisons [2].

3. Methodology

There are three steps in our diagram digitization pipeline: symbol detection, text recognition and association, and connection detection. The inputs to the pipeline are a diagram image and an optional set of manually-labeled symbols in the diagram. The manually-labeled symbols are ones that are of interest to the user but are not among the symbols automatically detected during the symbol detection step. An example of a snippet from an input diagram and a manually-labeled symbol is shown in Figure 2.

In this section, we describe each step of the pipeline and demonstrate how each step transforms the input diagram snippet in Figure 2 into intermediate outputs. The final output of the pipeline, presented in Table 1, is an asset hierarchy table containing all information extracted from the diagram snippet. See Appendix A for the asset hierarchy table on the entire diagram.

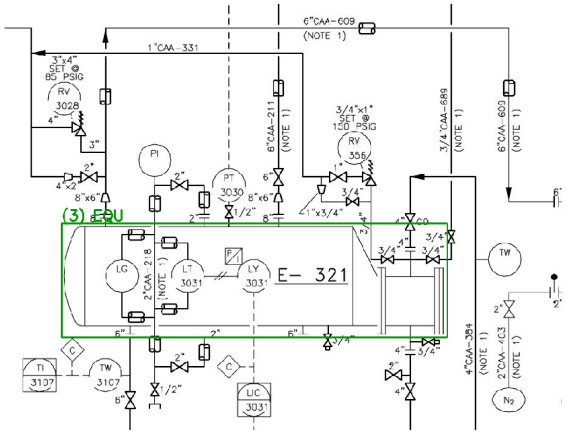

Figure 2. A snippet from an input diagram with a manually-labeled symbol (green).

3.1. Symbol detection

The goal of the symbol detection step is to automate the identification of frequently appearing symbols in diagrams. Two of the most common and useful symbols in P&IDs are tags and LMIs. In this set of diagrams, a tag is the digital representation of a sensor in the facility and is represented by a circle inscribed in a square. An LMI is a physical instrument such as a pressure gauge or temperature reading and is represented by a circle. Tags and LMIs are important because they often have corresponding entries in time series sensor databases. By identifying the tag and LMI symbols in a diagram we can localize sensor data to specific equipment or sections of a facility.

In order to train and validate a machine learning model to detect and classify tags and LMIs, we created a dataset of symbol crops from a collection of 18 P&IDs. Tags and LMIs across the diagrams had a constant size and fit within a 100 x 100-pixel window. So, we labeled all the tags and LMIs in the 18 diagrams using 100 x 100-pixel bounding boxes, resulting in 308 tag crops and 687 LMI crops. Additionally, we took 100 random crops that did not contain a tag or LMI symbol from each diagram, resulting in 1800 "not symbol" crops. Examples of crops in each of the three classes are given in Figure 3.

Using our dataset of symbol crops, we trained a Convolutional Neural Network (CNN) to perform a three-way classification task: to determine whether an input image contained a tag, LMI, or no symbol. We designed a simple CNN architecture with three convolutional layers (with ReLU activations and max pooling) and two fully-connected dense layers. The first hidden layer had 64 units with ReLU activations. The final output layer had three units with softmax activations to predict the probability that the input image belonged to each of the three classes. This

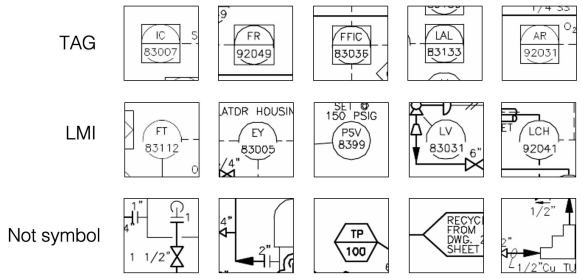

Figure 3. Examples of crops in each of the three classes.

architecture, shown in Figure 4, was inspired by the LeNet architectures popularized for digit recognition [5], as we expected the complexity of diagram symbols to be similar to that of handwritten digits. The network had a total of 437,923 parameters.

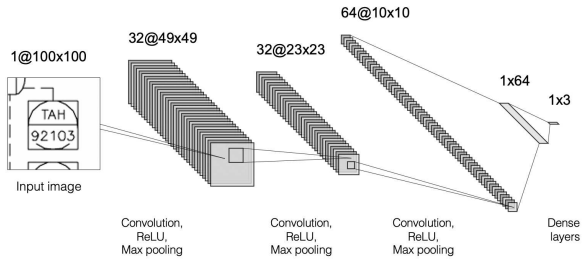

Figure 4. The CNN architecture used for symbol detection.

Given the limited number of training examples, we used data augmentation and dropout to achieve a more robust and generalizable model. We augmented the training examples by rotating, shifting, shearing, zooming, and flipping to make the network invariant to these transformations. Dropout was applied in the dense layers to add some regularization and improve the network's generalization performance. We randomly split the symbol crops dataset into training (60%) and validation (40%) sets and trained the network to minimize cross-entropy loss on the validation set. As shown in Figure 5, both the training and validation loss had converged by 100 epochs of training. We then combined the training and validation sets and trained a final model on the entire dataset for 100 epochs.

To apply the trained CNN to detect symbols in a new diagram, we first slide over the diagram image with a small stride length and generate all 100 x 100-pixel windows from the input diagram. Because most of the diagram is sparse, we filter out windows which are over 90% blank and automatically classify them as "not symbol." The remaining dense windows are processed through the CNN to produce predicted probabilities of belonging to each symbol class. Non-maximum suppression is applied to resolve

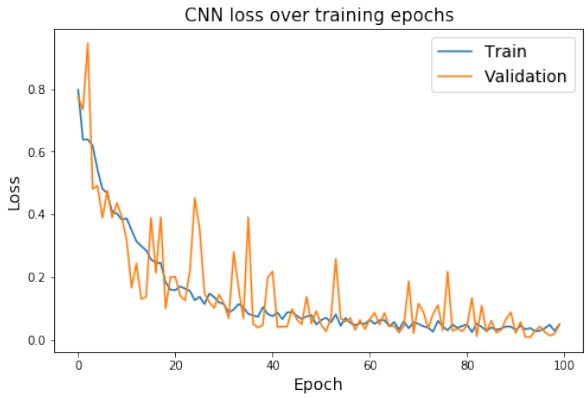

Figure 5. The training and validation loss converged by 100 epochs of training.

multiple symbols detected in overlapping windows to a single detected symbol. Finally, the predicted probabilities are thresholded, using a predefined threshold for each symbol class, to convert the probabilities into a discrete symbol classification. Windows with predicted probabilities less than all thresholds are classified as not symbol.

Figure 6 shows the results of symbol detection on the input diagram snippet. The detected tag and LMI symbols are displayed in red. Nearly all of the predicted symbols are correctly classified, except for symbol 39, which is not an LMI (LMI are circles, not ovals).

3.2. Text Recognition and Association

Text is a crucial element in P&IDs which identifies and describes elements in the diagrams. For instance, the text inside the tag symbols often serves as a key to fetch data for the tag from a time series sensor database. Text is also used to specify the length and diameter of pipes in the diagram. Because of the importance of text, an digitization of the diagram is not complete without recognizing and interpreting text, as well as associating it with detected symbols in the diagram.

To detect text in the diagram, we use Efficient and Accurate Scene Text Detector (EAST) [8], a state-of-the-art pipeline which uses a neural network to produce bounding boxes where text is present in an image. The text bounding boxes generated by EAST on the input diagram snippet are displayed in blue in Figure 6. For each symbol, we identify associated text based on the proximity of the symbol to text bounding boxes using a distance threshold. Associated text for each symbol is then interpreted using Tesseract OCR, and the results are added to the extracted information in the asset hierarchy in Table 1.

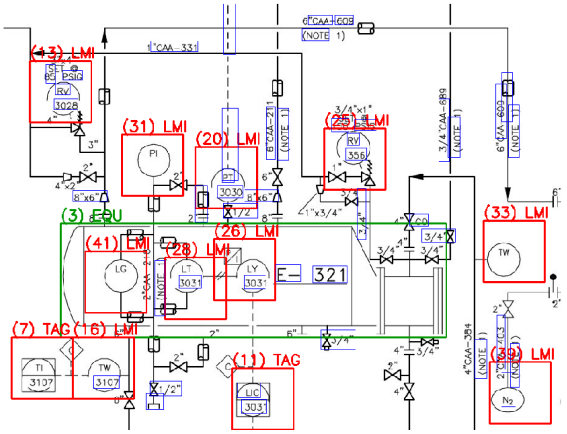

Figure 6. The diagram snippet with detected symbols (red), detected text (blue), and manually-labeled symbols (green).

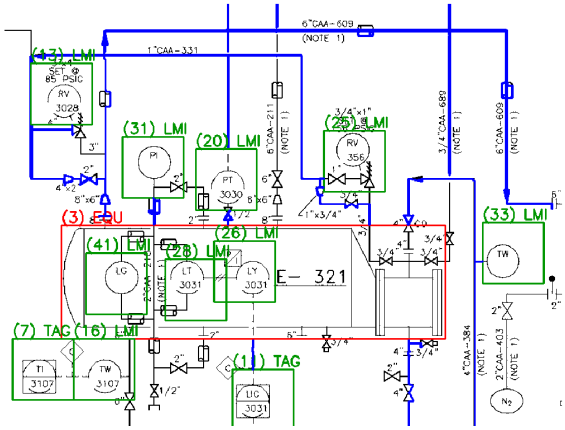

Figure 7. Connections detected between a source symbol (red) and connected symbols (green) through lines traversed by depth-first search (blue).

3.3. Connection Detection

Symbols in the diagram are connected to each other via a dense network of lines. While solid lines indicate physical connections such as pipes transporting fluids, dashed lines indicate digital connections such as equipment-to-sensor relationships. The connection detection step builds on the symbol detection step and determines which symbols are connected to each other via lines. This final step in the pipeline is essential for digitally reconstructing the relationships in the diagram and creating an asset hierarchy.

We use a graph search approach for connection detection. First, the thresholded diagram image is represented as a graph. In this diagram graph, nodes are individual pixels in the diagram. Each node contains information on whether it is black or white (based on its thresholded pixel intensity) and whether it is part of a symbol (and if so, which one). The graph's edges are links between neighboring pixels, with a maximum of eight edges per node. Symbols are represented in the graph as a collection of nodes corresponding to the pixels that form the symbol.

With this graph representation of the diagram, connections between symbols can be identified through a depth-first search (DFS). Specifically, for each detected symbol, a DFS is initialized from one of the nodes in the symbol. The DFS traverses the diagram graph along its black nodes, hitting (and keeping track of) connected symbols along its path. The search terminates once all valid paths are exhausted.

Figure 7 shows the result of running connection detection on the diagram snippet, with the source symbol in red, the connected symbols in green, and the paths traversed by DFS to reach the connected symbols from the source symbol in blue. Note that all detected symbols except for the oval symbol are identified as connected to the source symbol, which is correct as there are no continuous paths between those two symbols. We run connection detection starting at every symbol in the diagram and populate the entire list of connected symbols in the asset hierarchy in Table 1. In Table 1, we represent this information by listing the ids of connected symbols from each source symbol.

3.4. Diagram-to-Diagram Relationships

For large facilities or systems, multiple P&ID diagrams will collectively represent the components and flow of an industrial process. In these cases, in addition to the digitization of intra-diagram relationships, diagram-to-diagram relationships are also desired. Our approach is to convert the problem into a symbol detection task and a text recognition and association task. Diagram-to-diagram relationships are often indicated by reserved symbols as shown in Figure 8. The symbol representing the connection to a downstream diagram is a right-pointing arrow box and the symbol for the connection to an upstream diagram is represented as a left-pointing arrow box.

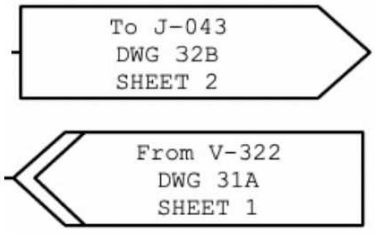

Figure 8. Inlet and outlet symbols represent connections to other diagrams.

Our methodology discussed in the previous Symbol De-

Symbol id	Method	Symbol type	Associated text	Connected symbols
3	Manually-labeled	EQU	3030, 321, CO, E“X6F, XVZ, E., XJA, BX6F, 303,...	33, 5, 7, 9, 41, 11, 13, 16, 18, 20, 25, 26, 2...
7	Detected	TAG	3I07	16, 3
11	Detected	TAG	7L?, XJA, 303J	3, 14
13	Detected	LMI	3023, RV, 1251C, 3%	25, 3, 20, 5
16	Detected	LMI	3I07, XVI, MOE.9, 1	3, 7, 41, 24, 26, 28
20	Detected	LMI	3030, XVZ, BX6F, UFOZ, E, J91, E, RIÉ.0O	3, 5, 13, 25, 31
25	Detected	LMI	P§IC, RV, CO, 355, 1533, —2M	5, 3, 20, 13
26	Detected	LMI	XVZ, E., 303, 303, E	16, 41, 3, 28
28	Detected	LMI	303, 303, MOE.9	16, 41, 26, 3
31	Detected	LMI	E“X6F, E	3, 20
33	Detected	LMI	347	18, 3
39	Detected	LMI	M.OE, MOVIÉO, JN, ZO5	34, 2
41	Detected	LMI	MOE.9	16, 26, 3, 28

Table 1. The output of the pipeline is an asset hierarchy table populated with information extracted from the diagram snippet in Figure 2.

tection section can be applied. Second, we realize that for most P&IDs, the diagram names follow convention and always appear with the diagram-to-diagram connection symbols. As shown in Figure 8, we see the connected P&ID diagram names (DWG Number) are very structured. The combination of our methodology discussed in the previous Text Recognition and Association section and some regular expressions can be applied to link diagrams by their names.

4. Results

We evaluated our symbol detection procedure on the 18 training diagrams and 11 diagrams in a held-out test set. Symbols were considered correctly classified if the detected symbol's class matched the ground truth label and the detected symbol's bounding box had an intersection over union (IOU) of at least 0.5 with the ground truth symbol's bounding box. In the 11 unseen test diagrams, tags were classified with 100% precision and 98% recall using a classification probability threshold of 0.95. LMIs were classified with 85% precision and 90% recall using a classification probability threshold of 0.95.

Precision-recall curves in Figure 9 summarize the classification performance for each symbol class over multiple probability thresholds in the training and test diagrams. Tag classification performance is considerably better than LMI classification performance since many circular-shaped symbols which are not LMIs are incorrectly classified as LMIs, driving down the precision of LMI classification. Tags, on the other hand, are consistently represented by a distinct circle-inside-square symbol and are less prone to getting confused for other symbols. A full comparison between the detected symbols and ground truth symbols in a diagram is provided in Appendix B.

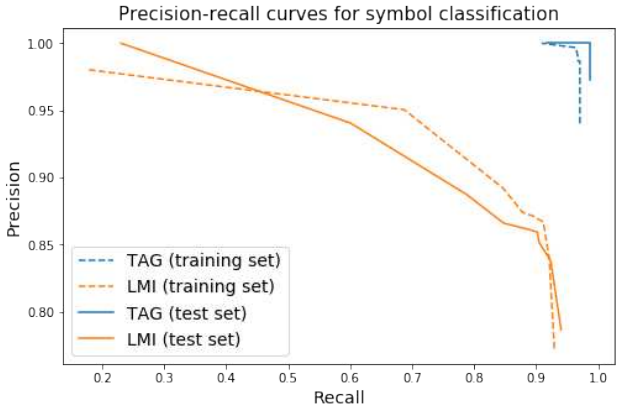

Figure 9. Precision-recall curves show the classification performance for each symbol class on the 18 training diagrams and 11 test diagrams.

5. Conclusion

In this paper, we have presented an automatic digitization pipeline for P&IDs. Our pipeline combines state-of-the-art computer vision methods to detect symbols, recognize and interpret text, and detect connections between symbols through lines. To the best of our knowledge, our pipeline is the first to apply graph search to detect connections between symbols in P&IDs. Moreover, our symbol detection CNN model achieves high levels of precision and recall and is easily extensible to additional symbol classes.

There are several future research directions which should be pursued to improve the accuracy and scalability of P&ID

digitization. First, leveraging a binary object detection CNN such as FasterRCNN to detect symbol/not-a-symbol would not only improve the performance of the end classifier but also reduce the overhead from the sliding window. Second, we need to develop sample-efficient symbol detection models which can learn to classify symbols from only a handful of examples instead of hundreds. Recent work in few-shot learning techniques may be relevant for this challenge and help scale the symbol detection capability to the myriad symbol classes that appear in P&IDs.

Text interpretation can also be made more accurate by providing a dictionary of known terms in the facility, which can help to resolve some of the mistakes made by OCR. Finally, the graph search approach for connection detection can be made more robust by integrating simple heuristics such as the maximum permissible distance between two connected symbols, as well as more complex constraints such as the direction of flow of the connecting pipes based on arrow symbols.

The structured asset hierarchy extracted from diagrams using our pipeline can be used to support a wide range of applications. We envision diagram search applications which can help to localize symbols across a large number of diagrams based on just their associated text or symbol type. Predictive maintenance applications for equipment can also be supported with valuable knowledge of equipment-to-tag connections, which can help to train better models for equipment failure. By automatically transforming unstructured diagrams into structured information, our pipeline can unlock the value of these diagrams for industries and significantly reduce the manual work necessary to digitize and understand them.

6. Acknowledgements

We would like to thank Varun Badrinath Krishna for motivating this work and his feedback. We thank Parthan Kasarapu and Samaneh Aminikhanghahi for their early advice and discussion on possible techniques. Finally, we thank Mehdi Maasoumy for his continued support and feedback on this project.

References

[1] J. S. Cardoso, A. Capela, A. Rebelo, and C. Guedes. A connected path approach for staff detection on a music score. In *2008 15th IEEE International Conference on Image Processing*, pages 1005–1008, 2008. 3

[2] Dries Van Daele, Nicholas Decleyre, Herman Dubois, and Wannes Meert. An automated engineering assistant: Learning parsers for technical drawings. *ArXiv*, abs/1909.08552, 2019. 3

[3] C Howie, J Kunz, T Binford, T Chen, and K.H Law. Computer interpretation of process and instrumentation drawings. *Advances in Engineering Software*, 29(7):563 – 570, 1998. 2

[4] 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). *Energies*, 12(13):2593, Jul 2019. 3

[5] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990. 4

[6] Carlos Francisco Moreno-García, Eyad Elyan, and Chrisina Jayne. New trends on digitisation of complex engineering drawings. *Neural Computing and Applications*, 31(6):1695–1712, 2019. 2

[7] Rohit Rahul, Shubham Paliwal, Monika Sharma, and Lovekesh Vig. Automatic information extraction from piping and instrumentation diagrams, 2019. 3

[8] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: an efficient and accurate scene text detector. *CoRR*, abs/1704.03155, 2017. 4