Digitization of chemical process flow diagrams using deep convolutional neural networks

Theisen, Maximilian F.; Flores, Kenji Nishizaki; Schulze Balhorn, Lukas; Schweidtmann, Artur M.

DOI
10.1016/j.dche.2022.100072

Publication date
2023

Document Version
Final published version

Published in
Digital Chemical Engineering

Citation (APA)
Theisen, M. F., Flores, K. N., Schulze Balhorn, L., & Schweidtmann, A. M. (2023). Digitization of chemical process flow diagrams using deep convolutional neural networks. *Digital Chemical Engineering*, *6*, 11. Article 100072. https://doi.org/10.1016/j.dche.2022.100072

Digital Chemical Engineering 6 (2023) 100072

Contents lists available at ScienceDirect

Digital Chemical Engineering

journal homepage: www.elsevier.com/locate/dche

Best practices and methods

Digitization of chemical process flow diagrams using deep convolutional neural networks

Maximilian F. Theisen, Kenji Nishizaki Flores, Lukas Schulze Balhorn, Artur M. Schweidtmann *

Delft University of Technology, Department of Chemical Engineering, Van der Maasweg 9, Delft, 2629 HZ, Netherlands

ARTICLE INFO

Keywords:
Process flow diagrams (PFD)
Flowsheet digitization
Object detection
Digitalization
Machine learning
Deep convolutional neural network

ABSTRACT

Advances in deep convolutional neural networks led to breakthroughs in many computer vision applications. In chemical engineering, a number of tools have been developed for the digitization of Process and Instrumentation Diagrams. However, there is no framework for the digitization of process flow diagrams (PFDs). PFDs are difficult to digitize because of the large variability in the data, e.g., there are multiple ways to depict unit operations in PFDs. We propose a two-step framework for digitizing PFDs: (i) unit operations are detected using a deep learning powered object detection model, (ii) the connectivities between unit operations are detected using a pixel-based search algorithm. To ensure robustness, we collect and label over 1000 PFDs from diversified sources including various scientific journals and books. To cope with the high intra-class variability in the data, we define 47 distinct classes that account for different drawing styles of unit operations. Our algorithm delivers accurate and robust results on an independent test set. We report promising results for line and unit operation detection with an Average Precision at 50 percent (AP50) of 88% and an Average Precision (AP) of 68% for the detection of unit operations.

1. Introduction

Engineering diagrams (EDs) are very important documents in the chemical process industry. In particular, there exist a variety of different EDs used during all engineering stages from early-stage process development to detailed engineering, construction, operation, and disassembly. The most relevant EDs in the chemical industry are block flow diagrams (BFDs), process flow diagrams (PFDs) (e.g., shown in Fig. 1), and piping and instrumentation diagrams (P&IDs). These EDs represent essential information of chemical processes (Nasby, 2012), such as process topology, major unit operations, control equipment, and piping information.

Despite the availability of advanced CAD software, EDs are still commonly stored and communicated as PDFs, image files, or printouts. Thus, the topological information of the ED can often not be directly read out from computer programs. Consequently, chemical companies have large amounts of legacy EDs which are not machine-readable. Likewise, there is a large number of BFDs, PFDs, and P&IDs depicted in figures in scientific literature and patents, which contradicts the FAIR (Findable, Accessible, Interoperable, Reusable) data principles.

The analog storage and communication of ED files leads to a number of practical issues. First, modifying and updating EDs is expensive and time-intensive. Changing ED is often necessary to maintain them up to date for safety or regulatory reasons. Second, analog EDs are not interoperable with other software applications, for instance Internet of Things (IoT) or digital twins. The lack of interoperability furthermore hinders the development of novel artificial intelligence (AI) applications in chemical engineering (Schweidtmann, 2022; Schweidtmann et al., 2021a; Weber et al., 2021; Wiedau et al., 2021). For instance, we have shown that AI algorithms can learn from the structure of flowsheets to facilitate process engineering (Vogel et al., 2022a; Vogel et al., 2022b). Third, information in EDs is not findable. Since analog EDs are not machine-readable, they can also not be integrated into knowledge bases and thus information about the processes is not findable. To overcome these drawbacks, it is important to digitize EDs for the chemical industry.

Digitization of EDs has been a prevalent research topic in the classical computer vision community for a long time (Moreno-García et al., 2019). In the 1980s, the digitization of electric circuit diagrams (Bunke, 1982; Fahn et al., 1988; Groen et al., 1985; Okazaki et al., 1988) and P&IDs (Furuta et al., 1984; Ishii et al., 1989) attracted attention. These early digitization strategies relied on classical computer vision

* Corresponding author.
E-mail address: a.schweidtmann@tudelft.nl (A.M. Schweidtmann).

https://doi.org/10.1016/j.dche.2022.100072
Received 17 October 2022; Received in revised form 29 November 2022; Accepted 1 December 2022
Available online 8 December 2022

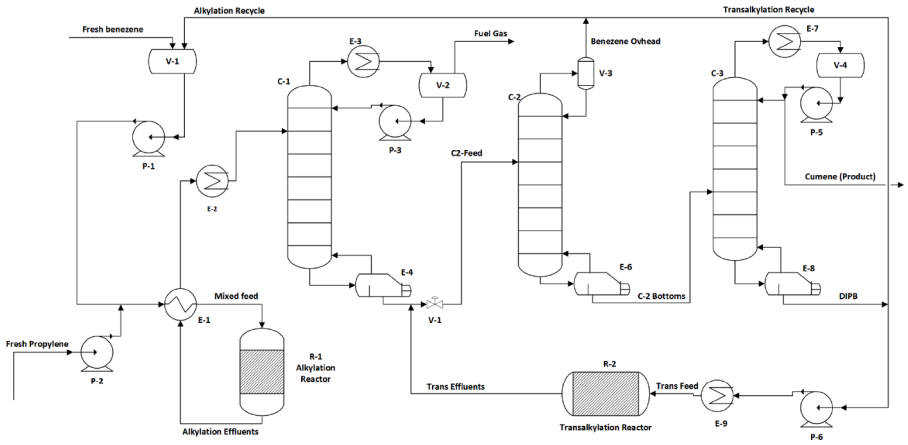


Fig. 1. Example flowsheet of a proposed Cumene production plant recreated from Norouzi et al. (2014). The illustration was slightly altered to introduce more optical variability. The flow structure however is kept.

algorithms, i.e., they were mostly rule-based, and relied strongly on manually-engineered feature extraction and pattern recognition (Mani et al., 2020). Several works have presented a semi-automatic digitization process where P&ID symbols were distinguished by template matching (Vanderbrug and Rosenfeld, 1977). Using template matching as well, Arroyo et al. (2016) derived plant simulations from P&ID diagrams. However, template-based detection approaches do not work well with heterogeneous data sources, where high intra-class variability occurs. More recently, Gellaboina and Venkoparao (2009) used a Hopfield model to recognize common symbols in P&IDs by recognizing patterns based on a memory system. While decent results were reported, the Hopfield neural network model presented problems with large datasets and class imbalances (Gellaboina and Venkoparao, 2009).

With recent advances in deep learning, digitization strategies for EDs in the process industry have improved significantly. Deep learning techniques and convolutional neural networks (CNNs) have significantly outperformed classical computer vision methods on a large variety of tasks. Flowsheets show a large diversity in styles and sizes of objects. This makes classical methods such as template matching difficult to apply. Also, CNNs can improve over time when more data is added. This has motivated several works to apply deep learning to flowsheet digization. Mani et al. (2020) trained a three-layer convolutional neural network to recognize symbols in P&IDs based on a dataset of 18 P&IDs. Similarly, Rohit et al. (2019) trained a CNN-based object detection model (Long et al., 2015) on 4 large industrial P&IDs which were sub-divided into 672 images. In 2021, Paliwal et al. (2021) improved upon that by training a two-stage detector, with a fully convolutional network (Long et al., 2015) for region proposals combined with a fine-grained image classification network (Zhang et al., 2021). For training, they used a dataset of 500 P&IDs, which was randomly generated using a CAD software. Jamieson et al. (2020) applied a state-of-the-art text detection model (Zhou et al., 2017) with an optical character recognition algorithm to detect and digitize text from 172 P&IDs from one source in the oil and gas industry. Yun et al. (2020) applied a R-CNN framework (Girshick, 2015) to symbol detection in P&IDs, training their model on 7 diagrams from an industry collaboration. They further introduced negative samples in training to increase model performance through clustering. With a custom CNN model inspired by AlexNet (Krizhevsky et al., 2012; Yu et al., 2019) approached the problem of symbol detection with a sliding window approach. Their dataset consisted of 68 P&IDs, provided by a Korean engineering company. To tackle the problem of tiny symbols in diagrams, Kim et al. (2021) trained two separated generalized focal loss (Li et al., 2020) networks for detection of small and large symbols on 82 P&IDs provided by an engineering company. Taking a few hundred P&IDs from Design Control Documents submitted to the US NRC, Gao et al. (2020) demonstrated the applicability of the Faster R-CNN framework (Ren et al., 2015) to symbol detection in P&IDs. Elyan et al. (2020) applied YOLOv3 (Redmon and Farhadi (2018) for symbol detection in P&IDs. Their dataset consisted of 172 P&IDs provided by an engineering company. To further extend the dataset, they applied generative adversarial networks. Concentrating on the detection of lines in P&IDs, Yoochan et al. (2021) introduced a framework to digitize flows. They detected line segments using Hough transformations and flow related symbols using deep learning based object detection. To test their framework, they applied it to 9 P&IDs provided by an engineering company. Very recently, Kim et al. (2022) presented a framework for P&ID digitization combining elements of previous work (Kim et al., 2021; Yoochan et al., 2021) to an end-to-end approach. They trained two GFL (Li et al. (2020)) models and a RetinaNet (Zlocha et al., 2019) to detect symbols, text and flows respectively on a dataset of 82 synthetic P&IDs. The training corpus consisted of 76 labels, of which 25 were related to unit operations, 33 to piping, and 18 to instruments. They further tested the trained model on 5 real P&IDs provided by a local engineering company.

Although promising results have been reported from previous studies, we identify four major limitations in previous literature approaches. First, while there has been a considerable effort towards digitizing P&IDs, to the best of our knowledge deep learning powered digitization approaches have not been applied to PFDs. PFDs have specific symbol illustrations and different flow depictions, mandating tailored approaches. Second, all ML models in the literature are trained on datasets from a single source or on a synthesized datasets based on one CAD software. Thus, the data has a lack of variety. Unsurprisingly, the accuracies of such models are often near perfection, as the data exhibits little variation. It needs to be acknowledged that retrieving P&IDs is not trivial, as companies rarely publish these highly confidential information. It is however doubtful if such models would generalize well to other data distributions, for instance diagrams generated with other CAD software or diagrams that use different styles to depict unit operations (Willemink et al., 2020). Third, as denoted previously by Kim et al. (2021), most symbol datasets only consist of a few categories. As a consequence, these categorizations do not apply to other PFDs or P&IDs as they are incomplete. Fourth, the datasets used for training are oftentimes very small. Deep learning models rely on large datasets in order to make generalizable predictions (i.e., big in volume) (LeCun et al., 2015). Thus, limited data can lead to poor generalization.

The main contribution of this manuscript is a novel flowsheet digitization framework that overcomes the four previously mentioned limitations. Following the first observation, we implement a framework for the digitization of PFDs. Second, we address the lack of diversity in data

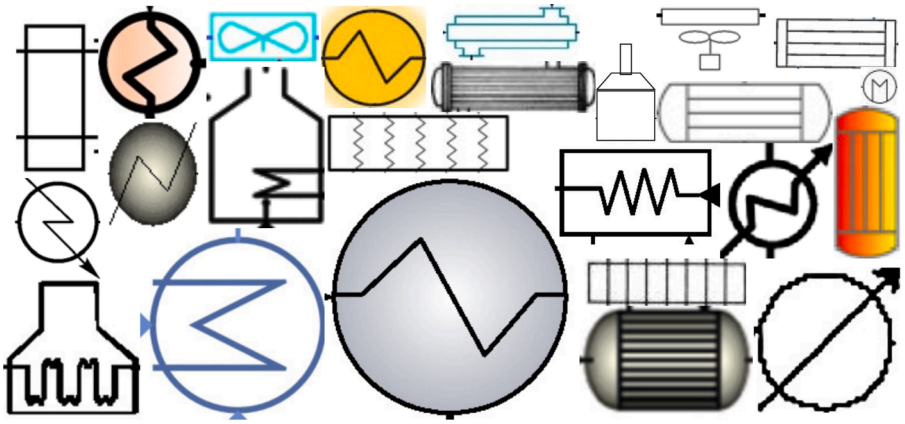


Fig. 2. There exist a large variety of symbols for the same unit operation. The figure shows 23 different representations of heat exchangers which are commonly found in PFDs. The variety also includes differences in technical solutions (e.g., heat exchanger furnace vs. plate heat exchanger), shapes, colors and shades (e.g., drop shadow used giving the impression that the object is raised above the objects behind it).

by collecting process flow diagrams from various sources including scientific literature, internet sources, and books. Third, in order to address the lack of complete data categorization commonly encountered, our data categorization is extensive with 47 categories. Fourth, we mine and label over 1000 flowsheets for training, which is significantly larger than previous work.

Our proposed digitization approach contains three main steps: (1) it recognizes objects in flowsheet images using computer vision, specifically deep learning-driven object detection, (2) it determines connectivities among unit operations using a pixel-based search algorithm, and (3) it converts the obtained information into a network graph representation of the flowsheet.

The remaining manuscript is structured as follows. Section 2 describes our training data, unit operation categorization, and data annotation procedure. Section 3 presents our approach for process flow diagram digitization. Section 4 presents the results of the object and stream detection algorithm. Finally, we draw conclusions from our work in Section 5.

2. Data

The development of deep learning algorithms through supervised learning methods usually requires large amounts of diverse data also referred to as big data (LeCun et al., 2015). In Section 2.1, we will discuss the sources of our data that we used for this study. In Section 2.2, we propose a unit operation symbol categorization for PFDs that is based on the topology of symbols and functionality in applications. In Section 2.3, we describe our data annotation process briefly before discussing challenges of the dataset in Section 2.4.

2.1. Flowsheet mining

There is currently no public dataset for chemical flowsheets. In our recent perspective paper, we describe the vision of mining flowsheets from scientific literature and patents in order to create a FAIR database of all chemical processes that have ever been published (Schweidtmann, 2022). Herein, we implement the first few steps towards our vision.

PFDs are retrieved using our recently developed flowsheet recognition algorithm (Balhorn et al., 2022). The algorithm downloads all full text papers from a given journal and extracts all images in said papers. Then, a CNN classifier detects if a figure is a flowsheet or not. The retrieved flowsheets are stored as images before being applied to our framework. We applied the algorithm to seven chemical engineering journals (i.e., Computers and Chemical Engineering, Korean Journal of Chemical Engineering, Brazilian Journal of Chemical Engineering, International Journal of Industrial Chemistry, Petroleum Science and Process Integration and Optimization for Sustainability). Next to journal publications, we also retrieve PFDs from the process engineering encyclopedia Ullmann's Encyclopedia of Industrial Chemistry (Ullmann, 2004). As an additional source we extracted images from Google and Bing search results based on related keywords such as "flowsheet" and "PFD". Overall, we mined 1005 PFDs using this approach. Using more journals, it would have been possible to retrieve more flowsheets. We however stopped at 1005 due to the time intensive annotation process. In accordance with the high accuracy reported by Balhorn et al. (2022), the overwhelming majority of extracted figures were correctly classified by the flowsheet image recognition algorithm.

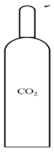
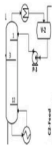
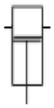
Our mined PFD dataset is diverse because of two main reasons. First, different authors of scientific publications often use different styles to illustrate unit operations and connectivities in PFDs. This reflects the diversity in personal preferences, backgrounds, demographics, and (inter-)national standards of the chemical engineering community. Second, we include many different sources, to avoid bias towards journal- or community-specific flowsheet conventions. The diversity of our data sources is imperative, as ML models regularly fail to extrapolate outside their trained data distribution (Schweidtmann et al., 2021b; Xu and Mannor, 2012). While our mined flowsheets are from international sources, their text labels are in always English.

2.2. Unit operation categorization

In order to train object detection algorithms, object classes need to be defined. These classes are used during data labeling and training. In the previous literature, common unit operations are usually defined as classes for object detection. For instance, Zhang et al. (2019) defined 12 common unit operations as classes in their object detection algorithm. However, this approach can lead to a large variety of different symbols for the same class. We refer to this as intra-class variations. For instance, Fig. 2 shows a number of different symbols for heat exchangers that are commonly used in PFDs. Such intra-class variations often result from different CAD software, personal preferences, or distinctions of different technical solutions. In the case of heat exchangers, there exist different symbols for heat exchanger furnaces, shell-tube heat exchangers, and plate heat exchangers. High intra-class variability can complicate the learning process as different drawing styles for the same class object could be considered as outliers in the dataset. This can lead to poor model performance in the light of limited data (Frid-Adar et al., 2017).

Class decomposition describes the method of splitting classes into different, more homogeneous sub-classes (Elyan and Gaber, 2016). In the context of PFD digitization, class decomposition has two main advantages. First, the decomposed classes exhibit more similar patterns within themselves and are more distinguishable from other classes.

Table 1
Examples of unit operations among the 47 categories with short acronyms used in this work.

Reactor (r)	Separator2 (sep2)	Gas tank (tank_gas)	CSTR (cstr)	Column (col)	Distillation System (dis)
					
Cooling tower (hex_cool)	Cooling fan (hex_fan)	Furnace (hex_fur)	Heat exchanger (hex)	In/Out (IO)	Mixer (mix)
					
Blower (blwr)	Pump1 (pp)	Pump2 (pp2)	Pump3 (pp3)	Compressor1 (comp)	Reboiler (reb)
					
Compressor2 (comp2)	Splitter (split)	Stream1 (strm)	Stream2 (strm2)	Stream3 (strm3)	Stream4 (strm4)
					
Tank1 (tank)	Tank2 (tank2)	Tank3 (tank3)	Tank4 (tank4)	Valve 3-way (v_3way)	Ball valve (v_ball)
					
Control valve (v_ctrl)	Global valve (v_glob)	Manuel valve (v_man)	Needle valve (v_ndl)	Standard valve (v)	Vessel (ves)
					
Mixer/Splitter (ms)	Filter (fil)	Mixer2 (mix2)	Controller (ctrl)	Separator (sep)	Check valve (v_chck)
					
Unknown/Unique (X)	Heat integration (hex_int)	Aspen column (col_aspen)	Aspen distillation system (dis_aspen)	Shell & tube HEX (hex_st)	

Thus, decomposition can improve classification accuracy. Second, the distinct class definitions allow to extract more detailed information about the unit operations. Taking again the example of heat exchangers, there exist many different types of heat exchangers with different functionalities, thus sub-classes contain more information about used equipment types.

We extended the original 12 categories from Zhang et al. (2019) to incorporate a larger variety of symbols and equipment types using class decomposition. Our final categorization consists of 47 classes that are illustrated in Table 1.

We arrived at the final classes from the original 12 classes based on three rules. First, we split classes that contain specific equipment subtypes. For instance, we split the original valve class into eight subclasses. Each subclass represents a different specific valve type, e.g., needle valves or globe valves. Second, we split classes based on different drawing styles for the same category. This class decomposition can help increase model performance. For instance, we split the tank category into five subcategories including four different drawing styles (tank1, tank2, tank3, tank4). Third, we add other flowsheet-specific symbols as additional classes (e.g., In/Out tags (IO), arrows, text, or stream tags).

2.3. Active learning for data annotation

Data annotation is a time-intensive and costly task. In order to accelerate the annotation process, we employed an active learning approach (Settles, 2012). First, all unit operations within a small batch of PFD were annotated manually. Then, a preliminary object detection model was trained and used for interference on a second batch of raw data (i.e., PFD images) to predict annotations for these. Then, the interfered annotations were manually corrected and used to retrain the model. This active learning loop was repeated multiple times. We found that this approach greatly accelerates the annotation, as the model quickly learns to detect the most common unit operations and human correction is only necessary for a small number of errors.

The mined flowsheets were labeled using domain expertise and contextual information. The open-source graphical annotation tool LabelImg (Tzutalin, 2015) was utilized. The quality of data annotation directly impacts the predicting performance of the object detection model (Su et al., 2012). Thus, correct and consistent annotation of objects in the data is essential. Using the active learning approach, we manually annotated 1005 flowsheets including 17,411 unit operations

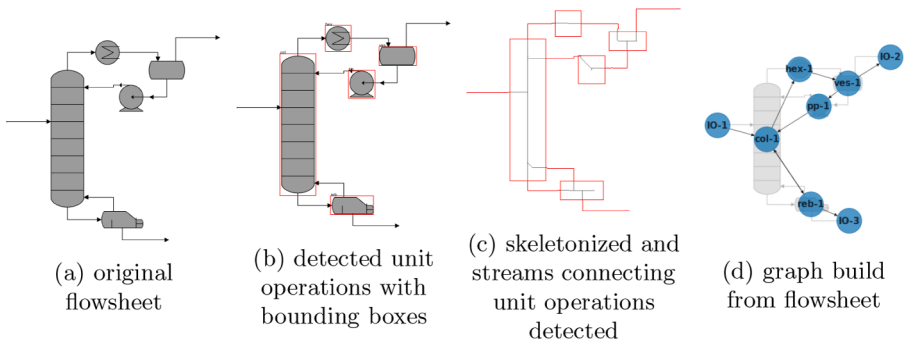


Fig. 3. The trained unit operation detection model is applied to the flowsheet (a). The object detection algorithm predicts bounding boxes and object classes (b). Then, skeletization is applied and streams connecting unit operations are identified (c). Finally, a graph representation is obtained (d).

in total.

2.4. Data challenges

The final dataset exhibits several challenges for deep learning. First, the sizes of unit operations vary greatly. This occurs commonly in PFDs reflecting the physical size of the unit operations, e.g., valve symbols are smaller than symbols for distillation columns. This can be a challenge for object detection algorithms, which are known to struggle with very small objects (Li et al., 2017). Second, the dataset size is small. Our PFD dataset size is already significantly larger than the one of previous works. However, common object detection datasets that are used for benchmarking are often much larger. For instance, the COCO dataset includes 220,000 images which is significantly larger than our dataset. As the performance of deep learning models typically scales with dataset size, the limited dataset size is still a potential challenge. Third, our dataset is imbalanced. In particular, the number of examples per class varies greatly. This imbalance arises naturally since some unit operations are used more frequently than others. For instance, in the entire dataset there are 35 instances of the class blower (blwr), while there are 2617 heat exchangers (hex). Such class imbalance can lead to poor model performance on underrepresented classes (Oksuz et al., 2021).

3. Methodology

In this section, we explain the three key steps of our proposed PFD digitization algorithm as illustrated in Fig. 3: First, object detection models detect and classify unit operations, text, and arrowheads (Section 3.1). Second, the connectivities between unit operations are identified using a pixel-based search and the process graph is generated based on the identified information (Section 3.2).

3.1. Unit operation detection

The object detection identifies a variety of objects that are commonly found on PFDs. This includes unit operations, arrowheads, and text (e.g., tags).

Object detection nowadays is mostly based on deep learning architectures. There exist two types of object detection algorithms, i.e., one-stage (Lin et al., 2017b; Liu et al., 2016; Redmon et al., 2016; Redmon and Farhadi, 2017; Redmon and Farhadi, 2018) and two-stage (Ren et al., 2015; Zou et al., 2019) algorithms. Both architectures first use fully connected convoluted networks for feature extraction, commonly referred to as backbone. Then, two-stage detection algorithms use two stages, i.e., a region proposal stage and a classification stage. Herein, the region proposal stage predicts possible bounding boxes while the classification stage predicts the object class labels. One-stage models on the other hand omit the region proposal stage and combine the two tasks.

Single detectors exhibit high detection speeds (Zou et al., 2019), which makes them a natural choice in time-critical tasks or high-throughput tasks such as object detection in videos. However, one-stage detectors have been reported to struggle more detecting small objects in images (Liu et al., 2021; Redmon and Farhadi, 2017). Two-stage detectors on the other hand regularly report higher accuracies on benchmark datasets such as COCO (Lin et al., 2014).

We use the state-of-the-art two-stage network called Faster R-CNN for object detection for three main reasons: (i) PDFs exhibit often small objects like valves, (ii) the digitization is not a computationally demanding high-throughput task (compared to object detection in videos), and (iii) we prefer a high prediction accuracy over prediction speed.

We chose a ResNet-50 as our backbone model. The choice of a backbone model is hereby one of the most crucial decisions for performance. ResNet has been reported to be especially suitable for the detection of small objects, which are one of the challenges of this dataset. In preliminary trials, deeper ResNet (Xie et al., 2017) architectures showed no further improvement.

We further employed Feature Pyramid Networks (FPNs) in our backbone model. FPNs are a set of deep CNNs which construct features at different scales while keeping computation feasible (Lin et al., 2017a). The main objective of feature pyramids in a model is to allow a neural network to learn high to low-level features and independently make predictions at each level. In contrast to vanilla predictors, FPNs do not rely on the highest level features only. This allows to consider a broader spectrum of features that are otherwise lost during up-sampling. For more details, we refer to Lin et al. (2017a). Feature pyramids are an important component in detection systems that facilitate the recognition of objects at different scales (Lin et al., 2017a). This makes them suitable to overcome the present size differences of objects in our dataset. PFDs come in various sizes. Faster R-CNN can in principle handle any image size. For practical reasons however, images are resized to be between 800 and 1333 pixels during preprocessing. This is the default setting in Detectron2.

We use transfer learning to improve model performance. Transfer learning refers to the improvement of model learning in one task by transferring knowledge from a related, previously learned task (Torrey and Shavlik, 2010). With transfer learning, a model can initiate the training process on new data distributions with pre-trained weights, shortening training time and possibly leading to superior performance. We considered this to be advantageous in the light of limited training data. In our algorithm, the Faster R-CNN model had been pretrained on the COCO dataset 2017 (Lin et al., 2014).

We mitigate the issue of our imbalanced dataset using two techniques. First, we apply repeat factor sampling (Wu et al., 2019). Repeat factor sampling allows training images with underrepresented categories more often to account for slower learning effects. Repeat factor training is especially important for our dataset as some unit operations are seldom found in PFDs. Second, we trained two individual object detection models for different tasks: (1) detection of unit operations and unknown units, and (2) detection of arrowheads and text. This is

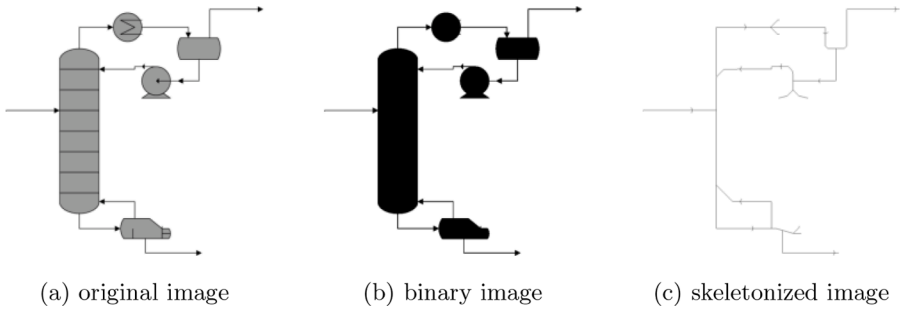


Fig. 4. Example of skeletonization applied to a common unit operation illustration (a). The image is thresholded and binarized (b). The skeletonization algorithm is applied, leaving a one-pixel thick representation of objects (c).

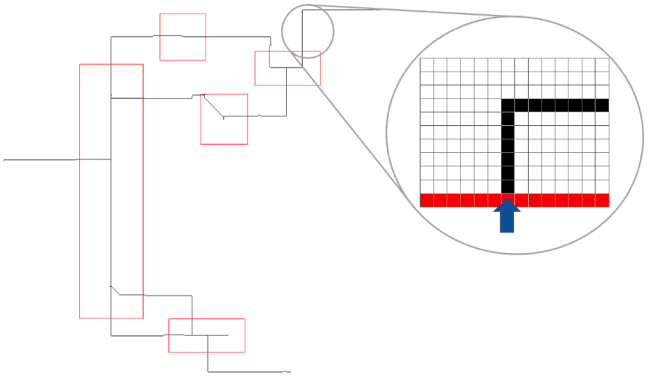


Fig. 5. Pixel-level representation of the connectivity search method.

advantageous because arrowheads and text are much more frequent in PFDs than unit operations. Next to these techniques, several other techniques have been described in literature to mitigate class imbalance. For example, different weights can be assigned to individual categories in the loss function during training (Oksuz et al., 2021). After unit operations are successfully detected, their bounding boxes are post-processed. Bounding boxes with significant overlap, measured as their intersection over union (IoU) (Rezatofighi et al., 2019), are compared and the ones with the lower confidence score are removed. This is necessary as Faster R-CNN has a tendency to detect the same objects several times (Rothe et al., 2015). Afterward, unit operations with low confidence scores are predicted as the unknown, i.e., category "X".

3.2. Connectivity recognition

Connectivity recognition is the second step of the framework. Connectivity recognition deals with finding connections between unit operations in the flowsheets. Connections in PFDs usually describe mass streams and are directed connections. Thus, the challenges of connectivity recognition are to first find connections among unit operations and second assign directions to the connections.

We employ a pixel search algorithm to identify the connectivity between unit operations. In order to explore connectives on a pixel level, the widths of all objects in the PFD are reduced into one-dimensional lines. Therefore, the flowsheet image is binarized and skeletonization is applied. Skeletonization produces a compact representation of objects in images by reducing them to their medial axis, effectively transforming shapes to curves of a 1-pixel thickness (Saha et al., 2016) while preserving their connectivities. Fig. 4 presents an example of distillation column skeletonization.

After skeletonization, a tailored connectivity search algorithm is applied. The connectivity search starts from the bounding box of a unit operation. For each white pixel neighboring the bounding box, the algorithm traverses along the path from the white pixel to the neighboring white pixel and so on. A graphical representation of this procedure is shown in Fig. 5. A connection between two unit operations is established when the algorithm reaches a pixel belonging to a new unit operation. If the exploration reaches a dead end, it creates an "In/Out" stream object, indicating an incoming or outgoing stream of the process. Once all the outgoing paths from a unit operation are explored, the algorithm moves to the next unit and repeats the search, storing information about all detected connections. A simplified pseudocode illustration can be found in Algorithm 1. Directions to paths are assigned according to detected arrowheads. After the paths are found, it is checked for each path if a arrowhead is on the path. The direction of the paths are then adjusted accordingly.

Finally, a graph representation of the flowsheet is constructed using the NetworkX open-source Python package (Hagberg, Aric et al., 2022). A graph is created where each unit operation, raw material, and product is represented as an individual node and the streams connecting them are represented as directed edges. Each edge and node in the graph also has attributes, such as unit operation type.

4. Results and discussion

In this section, we present the results of our flowsheet digitization framework on a comprehensive dataset. We first describe the training setup of the object detection algorithm as well as results of hyperparameter tuning (Section 4.1). Then, we show and discuss the results of training before showing the performance of the path exploration algorithm on an exemplary flowsheet (Section 4.2).

4.1. Training and hyperparameter optimization

The object detection model was trained on a NVIDIA GeForce RTX 3090. The training set includes 705 flowsheets, the validation set includes 150 flowsheets, and the independent test set includes 150 flowsheets.

The key hyperparameters of the model were optimized using Bayesian optimization. We investigated the influence of five hyperparameters on the prediction performance: the learning rate, the norm type for training ResNet, the momentum for ADAM optimization, the repeat threshold, and the freezing point for the pretrained backbone model. The hyperparameter optimization took approximately 3 days and involved 100 trials. The range of values considered as well as the best value found are shown in Table 2. These values are in alignment with the default values used in the Detectron2 for Faster R-CNN. This indicates robustness of the pretrained model hyperparameter towards our datasets.

In Fig. 6, the training loss curves of the model with the best hyperparameter configuration is depicted. The loss shown is the default loss as defined in the original Faster R-CNN work (Ren et al. (2015)). We trained the model for 20 epochs to determine the necessary number of training epochs. To train for 20 epochs took approximately three hours

```
connections = emptyList
for each start pixel do
    p = start pixel
    done = False
    while not done do
        done, p = NEXT PIXEL(p)
        if length(p) > 1 then:
            start pixels.add(p)
        end if
    end while
    connections.add([start pixel, current pixel])
end for

function NEXT PIXEL(current pixel)
    List neighbours = get_neighbours(current pixel)
    done = False
    Pixel next pixel = current pixel
    if neighbours = 0 then:
        done = True
        return done, next pixel
    end if
    if neighbours = 1 then:
        next pixel = neighbours[0]
        return done, next pixel
    end if
    if neighbours > 1 then:
        next pixel = neighbours
        done = True
        return done, next pixel
    end if
end function
```

Algorithm 1. Pseudocode of pixel search algorithm.

Table 2
Overview of hyperparameter studies with respective search space and best-found values. The norms investigated were Batchnorm (BN), frozen Batchnorm (FrozenBN), Synchronous Batchnorm (SyncBN) and Group norm (GN) (Ioffe and Szegedy, 2015; Wu and He, 2018).

Hyperparameter	Search space	Best found value
Freezing layer backbone model	[0;5]	1
Repeat factor	[0.05;0.5]	0.35
Norm	[BN; SyncBN; FrozenBN; GN]	FrozenBN
Momentum	[0.5; 0.99]	0.83
Base Learning rate	[0.0005; 0.05]	0.0083

on our computer. As it can be seen in Fig. 6, the model learns very quickly. This shows that the chosen architecture is well suited for unit operation detection. After 10 epochs the validation loss begins to flatten. Hence, we used the model after 10 epochs as our final model (i.e., following an early-stopping approach).

4.2. Object detection performance

The goal of object detection is to localize and classify objects within images. To evaluate the performance of object detection algorithms the placement of the bounding boxes around objects and the class predictions need to be measured.

A common performance evaluation metric for object detection is the Average Precision (AP) which considers correct, missed, and false predictions (Ren et al., 2015). The IoU is used to evaluate the correct bounding box placement. In particular, a minimal IoU threshold is chosen that corresponds to a correct placement. It is controversial what threshold value is considered correct (Everingham et al., 2010; Lin et al., 2014). For instance, the Pascal VOC AP metric, also known as AP50, is the AP calculated at an IoU threshold of 0.5. Another popular method is to take an average over an array of thresholds [0.5:0.05:0.95]. This

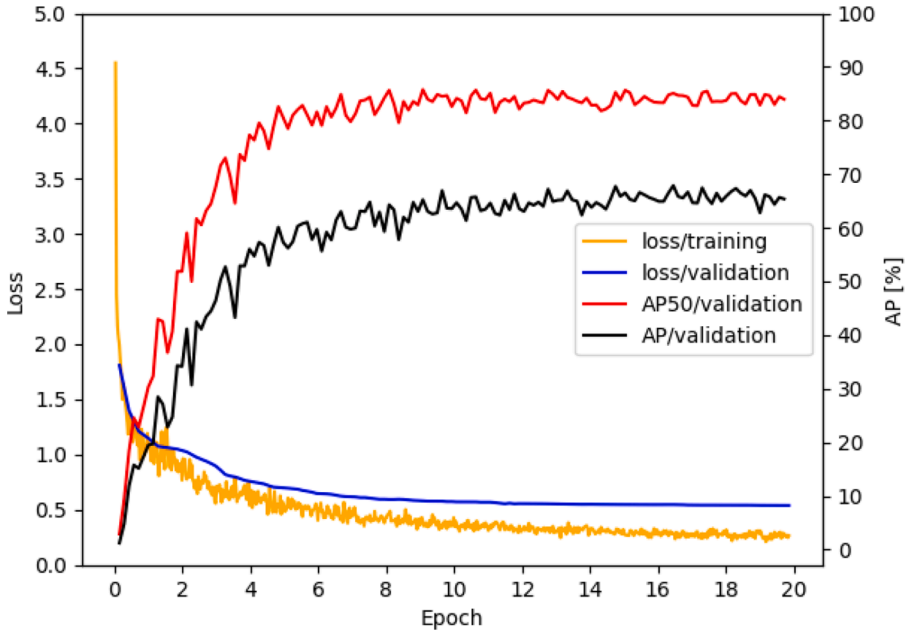


Fig. 6. Training curve of loss and AP metric for the model with the selected hyperparameter configuration. The loss curve for the validation set is smooth compared to the other documented metrics. This is because the validation set was not tested at every epoch. This leads to a smoother appearance of the validation loss.

Table 3
AP on the independent test set as calculated by COCO evaluation convention.

AP(%)	AP50 (%)	AP75 (%)	APs (%)	APm (%)	APl (%)
68.27	88.43	81.85	62.84	74.55	76.44

method is called AP and it is the primary metric for the well-known COCO challenge. Comparing the AP50 to the AP provides valuable insights into the performances of the classification and bounding box placement tasks. For example, a high AP50 and a low AP indicate that objects are correctly classified but the bounding box placement is imprecise. Moreover, additional metrics exist such as AP75 for an IoU of 0.75 and AP calculated separately for different object sizes.

The overall performance of the model on the test set is summarized in Table 3. The very high AP50 of 88.43% indicates that the model classifies most objects correctly when a small IoU of 0.5 is sufficient. At an IoU of 0.75, the model still shows a very high AP75 of 81.85%. The averaged AP is 68.27%. This is considerably lower than the AP50. Overall, the results indicate that the model classifies most unit operations correctly but fails to accurately localize some unit operations at a high IoU.

A comparison to state-of-the-art models on the COCO dataset as a large, real-world benchmark dataset provides further insights. The current best models on the COCO dataset achieve an averaged AP of 64.2% (Wei et al., 2022) and AP50 of 79.5% (Li et al., 2022). These performances are in line with Faster R-CNN performance on our dataset. This shows that the performance of our PFD digitization is comparable to the current state-of-the-art in computer vision and deep learning.

Table 3 also shows individual model performances for objects that have different sizes, i.e., large- (APl) medium- (APm), and small-sized (APs) objects. The results show that the model performs significantly better for large- and medium-sized objects than for small-sized objects. This is expected because the detection of small objects is challenging for two main reasons. First, object detection models often struggle to detect very small objects (Li et al., 2017). Second, annotation errors are more common for small objects as the correct positioning of small bounding boxes is challenging. Moreover, small absolute bounding box positioning errors can lead to large relative errors when considering small objects like arrows.

Table 4
Overview of AP per unit operation in ascending order for all 47 classes divided in below and above AP as indicated with red dashed line.

Unit operation	AP (%)	Unit operation	AP (%)	Unit operation	AP (%)
col_aspen	90.00	reb	75.35	v_glob	63.50
tank_gas	89.06	comp2	75.00	hex_st	62.07
blwr	87.62	fil	74.72	hex_fan	61.91
dist_aspen	87.62	ves	74.26	strm	60.11
tank1	85.55	r	72.56	tank3	60.00
hex_fur	85.08	pp	71.74	v_std	56.32
sep_2	85.05	ctrl	71.24	v_ndl	54.88
tank2	84.48	comp	71.10	v_man	52.68
col	81.43	pp3	70.74	X	46.24
sep	80.38	pp2	70.00	v_ball	45.35
mix	80.10	ms	69.63	IO	44.14
hex_cool	80.00	v_chck	66.80	strm3	40.97
splt	80.00	dist	65.52	tank	37.43
r_cstr	79.85	v_ctrl	65.19	strm4	35.69
strm2	77.89	hex_int	65.04	mix2	34.25
hex	75.46	v3_way	64.65		

In Table 4 the AP is further broken down into the 47 individual unit operation classes. The results show that most unit operations are detected very accurately. Unit operations detected with high accuracy are either frequently in the dataset or show little intra-class variability due to class decomposition. Examples of classes with high occurrence are furnace heat exchanger (hex_fur), columns (col), and mixers (mix). These classes can vary in color, shape, and size and are still detected accurately. Examples of classes with very consistent illustrations are aspen columns (col_asp), gas tanks (tank_gas), and blowers (blwr). While these classes are not commonly found, their illustrations are homogeneous due to the applied class decomposition.

The results further show that a few unit operations have a relatively low AP. In particular, the model struggles with three types of classes. First, classes with small symbols are an obstacle to object detection algorithms. Examples of this phenomenon are valves, certain mixers (mix2), and stream tags (strm4, strm3, strm). Second, classes with a low frequency of occurrence and inconsistent illustrations are challenging to correctly classify. Examples of such classes are tanks (tank, tank3) and In/Out tags (IO). Repeat factor sampling was hereby not sufficient to

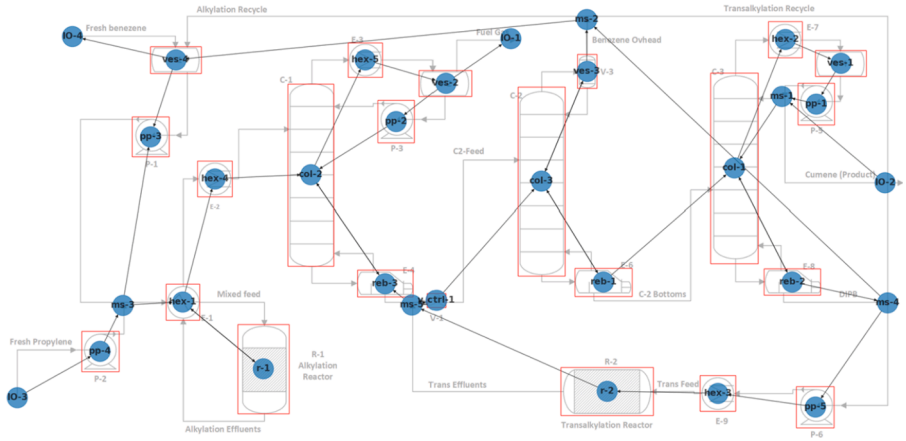


Fig. 7. Obtained graph from connectivity recognition laid over original flowsheet.

overcome the frequency imbalance. Third, the class of unknown unit operations (X) has a low AP of 46%. This is to be expected because the X class combines a large variety of uncommon unit operations. Thus, the class X has a particularly high intra-class variability.

4.3. Connectivity recognition

The quantitative evaluation of the connectivity recognition algorithm is challenging because it would require the manual labeling of all connections in the dataset. Moreover, the labeling of such graphs is uncommon and standard image annotation tools do not support the annotation of connectives in a graph format. Thus, we qualitatively assess the performance of the connectivity recognition algorithm on the illustrative example flowsheet shown in Fig. 1. The flowsheet was first processed by the object detection model to identify unit operations. Then, the connectivity recognition algorithm was applied to the image. The results of the unit operation detection and connectivity recognition are depicted in Fig. 7. The figure shows that all unit operations are classified correctly in the example. Moreover, all connectives between unit operations are identified correctly in the example. While all connections are correctly identified, wrong directions are assigned to some of the connections. Notably, the presented algorithm relies on arrowhead detection to determine the direction of streams. As pointed out before, object detection models can struggle to detect these very small objects, making identification of flow directions challenging. In future works, rule-based reasoning could potentially improve the automatic detection of flow directions based on the process context information. For example, we know the directions of the top and bottom outlet streams of distillation columns during normal operation.

Finally, it is worth mentioning that the connectivity recognition algorithm introduces new nodes for inlet and outlet streams. In addition, the algorithm adds new nodes for splitters and mixers such as the splitting of the Benezene overhead into a transalklynation and an alkylation recycle stream.

5. Conclusions

We proposed a two-step approach for the digitization of PFDs. Our proposed approach includes a unit operation detection with a Faster R-CNN and a pixel-based search algorithm for connectivity detection. The models are trained on a large dataset of over 1000 PFDs which are mined from scientific literature, books, and the internet. Our results show that the approach can robustly digitize PFD from these diverse sources. In the future, we envision to extend our approach and use it for the digitization of all PFDs that have ever been published in the scientific literature and patents. Also, data augmentation could help to improve object detection model performance in light of limited data. In addition, the identified text boxes on the PFDs could be processed by optical character recognition and could improve the digitization algorithm. Moreover, we see a great potential for the digitization of industrial PFDs. Ultimately, a database of PFDs is a potential enabler for future AI applications in the chemical engineering domain (Hirtreiter et al., 2022; Schweidtmann, 2022; Schweidtmann et al., 2021a; Vogel et al., 2022a; Vogel et al., 2022b; Weber et al., 2021; Wiedau et al., 2021).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Arroyo, E., Hoernicke, M., Rodríguez, P., Fay, A., 2016. Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams. Comput. Chem. Eng. 92, 112–132. https://doi.org/10.1016/j.compchemeng.2016.04.040.

Balhorn, L.S., Gao, Q., Goldstein, D., Schweidtmann, A.M., 2022. Flowsheet recognition using deep convolutional neural networks. Proceedings of the 14th International Symposium on Process Systems Engineering. Elsevier B. V., Kyoto, Japan.

Bunke, H., 1982. Automatic interpretation of lines and text in circuit diagrams. Pattern Recognition Theory and Applications. Springer, pp. 297–310.

Elyan, E., Gaber, M.M., 2016. A fine-grained random forests using class decomposition: an application to medical diagnosis. Neural Comput. Appl. 27 (8), 2279–2288.

Elyan, E., Jamieson, L., Ali-Gombe, A., 2020. Deep learning for symbols detection and classification in engineering drawings. Neural Netw. 129, 91–102. https://doi.org/10.1016/j.neunet.2020.05.025.

Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A., 2010. The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. 88 (2), 303–338.

Fahn, C.-S., Wang, J.-F., Lee, J.-Y., 1988. A topology-based component extractor for understanding electronic circuit diagrams. Comput. Vis. Graph. Image Process. 44 (2), 119–138.

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., Greenspan, H., 2017. Modeling the intra-class variability for liver lesion detection using a multi-class patch-based CNN. In: Wu, G., Munsell, B.C., Zhan, Y., Bai, W., Sanroma, G., Coupé, P. (Eds.), Patch-Based Techniques in Medical Imaging. Springer International Publishing, Cham, pp. 129–137. https://doi.org/10.1007/978-3-319-67434-6_15.

Furuta, M., Kase, N., Emori, S., 1984. Segmentation and recognition of symbols for handwritten piping and instrument diagram. Proceedings of the 7th IAPR International Conference on Pattern Recognition (ICPR), pp. 626–629.

Gao, W., Zhao, Y., Smidts, C., 2020. Component detection in piping and instrumentation diagrams of nuclear power plants based on neural networks. Prog. Nucl. Energy 128, 103491. https://doi.org/10.1016/j.pnucene.2020.103491.

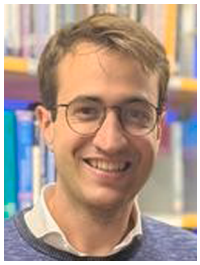
Gellaboina, M.K., Venkoparao, V.G., 2009. Graphic symbol recognition using auto associative neural network model. 2009 Seventh International Conference on Advances in Pattern Recognition. IEEE, pp. 297–301.

Girshick, R., 2015. Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV). IEEE, Santiago, Chile, pp. 1440–1448. https://doi.org/10.1109/ICCV.2015.169.

Groen, F.C., Sanderson, A.C., Schlag, J.F., 1985. Symbol recognition in electrical diagrams using probabilistic graph matching. Pattern Recognit. Lett. 3 (5), 343–350.

Hagberg, A., Chult, D. S., Swart, P., 2022. NetworkX. https://github.com/networkx/networkx.

Hirtreiter, E., Balhorn, L. S., Schweidtmann, A. M., 2022. Towards automatic generation of piping and instrumentation diagrams (P&IDs) with artificial intelligence. arXiv preprint arXiv:2211.05583.
Ioffe, S., Szegedy, C., 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on International Conference on Machine Learning, Vol. 37. JMLR.org, Lille, France, pp. 448–456.
Ishii, M., Ito, Y., Yamamoto, M., Harada, H., Iwasaki, M., 1989. An automatic recognition system for piping and instrument diagrams. Syst. Comput. Jpn. 20 (3), 32–46.
Jamieson, L., Moreno-Garcia, C.F., Elyan, E., 2020. Deep learning for text detection and recognition in complex engineering diagrams. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–7.
Kim, B.C., Kim, H., Moon, Y., Lee, G., Mun, D., 2022. End-to-end digitization of image format piping and instrumentation diagrams at an industrially applicable level. J. Comput. Des. Eng. 9 (4), 1298–1326. https://doi.org/10.1093/jcde/qwac056.
Kim, H., Lee, W., Kim, M., Moon, Y., Lee, T., Cho, M., Mun, D., 2021. Deep-learning-based recognition of symbols and texts at an industrially applicable level from images of high-density piping and instrumentation diagrams. Expert Syst. Appl. 183, 115337. https://doi.org/10.1016/j.eswa.2021.115337.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 25.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.
Li, J., Liang, X., Wei, Y., Xu, T., Feng, J., Yan, S., 2017. Perceptual generative adversarial networks for small object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
Li, L. H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.-N., Chang, K.-W., Gao, J., 2022. Grounded language-image pre-training. ArXiv:2112.03857 [cs] version: 2. 10.48550/arXiv:2112.03857.
Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., Yang, J., 2020. Generalized focal loss: learning qualified and distributed bounding boxes for dense object detection. NeurIPS.
Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125.
Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988.
Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft COCO: common objects in context. European Conference on Computer Vision. Springer, pp. 740–755.
Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. SSD: single shot multibox detector. European Conference on Computer Vision. Springer, pp. 21–37.
Liu, Y., Sun, P., Wergeles, N., Shang, Y., 2021. A survey and performance evaluation of deep learning methods for small object detection. Expert Syst. Appl. 172, 114602. https://doi.org/10.1016/j.eswa.2021.114602.
Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440.
Mani, S., Haddad, M.A., Constantini, D., Douhard, W., Li, Q., Poirier, L., 2020. Automatic digitization of engineering diagrams using deep learning and graph search. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 176–177.
Moreno-García, C.F., Elyan, E., Jayne, C., 2019. New trends on digitisation of complex engineering drawings. Neural Comput. Appl. 31 (6), 1695–1712.
Nasby, G., 2012. Using process flowsheets as communication tools. Chem. Eng. Prog. 108 (10), 36–44.
Norouzi, H.R., Hasani, M.A., Haddadi-Sisakht, B., Mostoufi, N., 2014. Economic design and optimization of zeolite-based cumene production plant. Chem. Eng. Commun. 201 (10), 1270–1293. https://doi.org/10.1080/00986445.2013.806312.
Okazaki, A., Kondo, T., Mori, K., Tsunekawa, S., Kawamoto, E., 1988. An automatic circuit diagram reader with loop-structure-based symbol recognition. IEEE Trans. Pattern Anal. Mach. Intell. 10 (3), 331–341.
Oksuz, K., Cam, B.C., Kalkan, S., Akbas, E., 2021. Imbalance problems in object detection: areview. IEEE Trans. Pattern Anal. Mach. Intell. 43 (10), 3388–3415. https://doi.org/10.1109/TPAMI.2020.2981890.Publisher: IEEE Computer Society
Paliwal, S., Jain, A., Sharma, M., Vig, L., 2021. Digitize-PID: automatic digitization of piping and instrumentation diagrams. Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2021 Workshops, WSPA, MLMEIN, SDPRA, DARAI, and AI4EPT, Delhi, India, May 11, 2021 Proceedings 25. Springer, pp. 168–180.
Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.
Redmon, J., Farhadi, A., 2017. Yolo9000: better, faster, stronger. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271.
Redmon, J., Farhadi, A., 2018. Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767.
Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. 28, 91–99.
Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S., 2019. Generalized intersection over union: a metric and a loss for bounding box regression. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
Rohit, R., Shubham, P., Monika, S., Lovekesh, V., 2019. Automatic information extraction from piping and instrumentation diagrams, 163–172. doi:10.5220/0007376401630172.
Rothe, R., Guillaumin, M., Van Gool, L., 2015. Non-maximum suppression for object detection by passing messages between windows. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (Eds.), Computer Vision – ACCV 2014. Springer International Publishing, Cham, pp. 290–306. https://doi.org/10.1007/978-3-319-16865-4_19.
Saha, P.K., Borgefors, G., di Baja, G.S., 2016. A survey on skeletonization algorithms and their applications. Pattern Recognit. Lett. 76, 3–12.
Schweidtmann, A. M., 2022. Flowsheet mining. In preparation.
Schweidtmann, A.M., Esche, E., Fischer, A., Kloft, M., Repke, J.-U., Sager, S., Mitsos, A., 2021. Machine learning in chemical engineering: aperspective. Chem. Ing. Tech. https://doi.org/10.1002/cite.202100083.
Schweidtmann, A.M., Weber, J.M., Wende, C., Netze, L., Mitsos, A., 2021. Obey validity limits of data-driven models through topological data analysis and one-class classification. Optim. Eng. https://doi.org/10.1007/s11081-021-09608-0.
Settles, B., 2012. Active learning. Synth. Lect. Artif. Intell. Mach. Learn. 6 (1), 1–114. https://doi.org/10.2200/S00429ED1V01Y201207AIM018.Publisher: Morgan & Claypool Publishers
Su, H., Deng, J., Fei-Fei, L., 2012. Crowdsourcing annotations for visual object detection. Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence.
Torrey, L., Shavlik, J., 2010. Transfer learning. Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. IGI global, pp. 242–264.
Tzutalin, 2015. Labelimg. https://github.com/tzutalin/labelImg.
Ullmann, F., 2004. Ullmann's Processes and Process Engineering. Wiley-VCH, Weinheim.
Vanderbrug, G.J., Rosenfeld, A., 1977. Two-stage template matching. IEEE Trans. Comput. 26 (04), 384–393.
Vogel, G., Balhorn, L. S., Hirtreiter, E., Schweidtmann, A. M., 2022a. SFILES 2.0: an extended text-based flowsheet representation. arXiv preprint arXiv:2208.00778.
Vogel, G., Balhorn, L. S., Schweidtmann, A. M., 2022b. Learning from flowsheets: a generative transformer model for autocompletion of flowsheets. arXiv preprint arXiv:2208.00859.
Weber, J.M., Guo, Z., Zhang, C., Schweidtmann, A.M., Lapkin, A.A., 2021. Chemical data intelligence for sustainable chemistry. Chem. Soc. Rev.
Wei, Y., Hu, H., Xie, Z., Zhang, Z., Cao, Y., Bao, J., Chen, D., Guo, B., 2022. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation. ArXiv: 2205.14141 [cs] version: 3. 10.48550/arXiv.2205.14141.
Wiedau, M., Tolksdorf, G., Oeing, J., Kockmann, N., 2021. Towards a systematic data harmonization to enable ai application in the process industry. Chem. Ing. Tech. 93 (12), 2105–2115.
Willemink, M.J., Koszek, W.A., Hardell, C., Wu, J., Fleischmann, D., Harvey, H., Folio, L. R., Summers, R.M., Rubin, D.L., Lungren, M.P., 2020. Preparing medical imaging data for machine learning. Radiology 295 (1), 4–15.
Wu, Y., He, K., 2018. Group normalization. ArXiv:1803.08494 [cs]. 10.48550/arXiv.1803.08494.
Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, R., 2019. Detectron2. https://github.com/facebookresearch/detectron2.
Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500.
Xu, H., Mannor, S., 2012. Robustness and generalization. Mach. Learn. 86 (3), 391–423. https://doi.org/10.1007/s10994-011-5268-1.
Yoochan, M., Lee, J., Mun, D., Lim, S., 2021. Deep learning-based method to recognize line objects and flow arrows from image-format piping and instrumentation diagrams for digitization. Appl. Sci. 11, 10054. https://doi.org/10.3390/app112110054.
Yu, Cha, Lee, Kim, Mun, 2019. Features recognition from piping and instrumentation diagrams in image format using a deep learning network. Energies 12 (23), 4425. https://doi.org/10.3390/en12234425.
Yun, D.-Y., Seo, S.-K., Zahid, U., Lee, C.-J., 2020. Deep neural network for automatic image recognition of engineering diagrams. Appl. Sci. 10 (11), 4005. https://doi.org/10.3390/app10114005.Number: 11 Publisher: Multidisciplinary Digital Publishing Institute
Zhang, F., Li, M., Zhai, G., Liu, Y., 2021. Multi-branch and multi-scale attention learning for fine-grained visual categorization. International Conference on Multimedia Modeling. Springer, pp. 136–147.
Zhang, T., Sahinidis, N.V., Siirola, J.J., 2019. Pattern recognition in chemical process flowsheets. AIChE J. 65 (2), 592–603. https://doi.org/10.1002/aic.16443.
Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J., 2017. East: an efficient and accurate scene text detector. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 5551–5560.
Zlocha, M., Dou, Q., Glocker, B., 2019. Improving RetinaNet for CT lesion detection with dense masks from weak RECIST labels. In: Shen, D., Liu, T., Peters, T.M., Staib, L.H., Essert, C., Zhou, S., Yap, P.-T., Khan, A. (Eds.), Medical Image Computing and Computer Assisted Intervention – MICCAI 2019. Springer International Publishing, Cham, pp. 402–410. https://doi.org/10.1007/978-3-030-32226-7_45.
Zou, Z., Shi, Z., Guo, Y., Ye, J., 2019. Object detection in 20 years: a survey. arXiv preprint arXiv:1905.05055.



Maximilian F. Theisen is a graduate student at Delft Technical University who is expected to graduate in 2023. He received his Bachelor of Science from RWTH Aachen University in 2021. During his studies, he spent the academic year 2019/2020 at University of California, Davis.

Kenji Nishizaki Flores is a chemical engineer who graduated from Delft Technical University in 2022. He received his Bachelor of Science from Texas Tech University in 2019.

Lukas Schulze Balhorn is a PhD student in the Department of Chemical Engineering at Delft Technical University since 2021. He received his Bachelor of Science from RWTH Aachen University in 2020 and his Master of Science from RWTH University in 2021. During his studies, he spent the academic year 2018/2019 at University of California, Davis.

Artur M. Schweidtmann is an assistant professor for chemical engineering at Delft Technical University and director of the Process Intelligence Research Lab. He received his Master of Science from RWTH Aachen University in 2017 and defended his PhD from RWTH in 2021, both in Chemical Engineering. During his studies, he spent the academic year 2013/2014 at Carnegie Mellon University as a visiting student via DAAD ISAP program. He performed his Master thesis at the University of Cambridge. His research focuses on the combination of artificial intelligence and chemical engineering.