

## Graph-to-SFILES

### Control structure prediction from process topologies using generative artificial intelligence

Schulze Balhorn, Lukas; Degens, Kevin; Schweidtmann, Artur M.

**DOI**

[10.1016/j.compchemeng.2025.109121](https://doi.org/10.1016/j.compchemeng.2025.109121)

**Publication date**

2025

**Document Version**

Final published version

**Published in**

Computers and Chemical Engineering

**Citation (APA)**

Schulze Balhorn, L., Degens, K., & Schweidtmann, A. M. (2025). Graph-to-SFILES: Control structure prediction from process topologies using generative artificial intelligence. *Computers and Chemical Engineering*, 199, Article 109121. <https://doi.org/10.1016/j.compchemeng.2025.109121>

**Important note**

To cite this publication, please use the final published version (if applicable). Please check the document version above.

**Copyright**

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.



# Graph-to-SFILES: Control structure prediction from process topologies using generative artificial intelligence

Lukas Schulze Balhorn<sup>1</sup>, Kevin Degens, Artur M. Schweidtmann<sup>1</sup>\*

*Process Intelligence Research Group, Department of Chemical Engineering, Delft University of Technology, Netherlands*

## ARTICLE INFO

### Keywords:

Control structure prediction  
Graph-to-sequence  
Process graph  
SFILES 2.0  
Generative artificial intelligence

## ABSTRACT

Control structure design is an important but tedious step in P&ID development. Generative artificial intelligence (AI) promises to reduce P&ID development time by supporting engineers. Previous research on generative AI in chemical process design mainly represented processes by sequences. However, graphs offer a promising alternative because of their permutation invariance. We propose the Graph-to-SFILES model, a generative AI method to predict control structures from flowsheet topologies. The Graph-to-SFILES model takes the flowsheet topology as a graph input and returns a control-extended flowsheet as a sequence in the SFILES 2.0 notation. We compare four different graph encoder architectures, one of them being a graph neural network (GNN) proposed in this work. The Graph-to-SFILES model achieves a top-5 accuracy of 73.2% when trained on 10,000 flowsheet topologies. In addition, the proposed GNN performs best among the encoder architectures. Compared to a purely sequence-based approach, the Graph-to-SFILES model improves the top-5 accuracy for a relatively small training dataset of 1,000 flowsheets from 0.9% to 28.4%. However, the sequence-based approach performs better on a large-scale dataset of 100,000 flowsheets. These results highlight the potential of graph-based AI models to accelerate P&ID development in small-data regimes but their effectiveness on industry relevant case studies still needs to be investigated.

## 1. Introduction

Developing piping and instrumentation diagrams (P&IDs) is currently a manual and tedious task (Toghraei, 2019). Supporting engineers in P&ID development with computer algorithms offers potential for cost and time reduction (Dzhusupova et al., 2022; Bramsiepe et al., 2014). Time savings can reduce a project's lead time, which is a critical economic success factor, particularly in the fine chemical and pharmaceutical industries (Bramsiepe et al., 2014). In addition, employing computer algorithms for tedious tasks can free up resources for innovation (Dzhusupova et al., 2022).

One essential step in P&ID development is designing a suitable control structure (Toghraei, 2019). The control structure aims to keep the process within a desired operating window. In decentralized control, the control structure consists of controllers that regulate manipulated variables based on measurements to follow the operating window (Morari et al., 1980). Typically, a suitable control structure is designed from a process flow diagram (PFD) depicting the conceptual design of a process. We refer to a Process Flow Diagram (PFD) that incorporates a decentralized control structure as a control-extended flowsheet (CEF). Control structure design involves, among other tasks,

analyzing the process degrees of freedom (i.e., the number of controllable variables) and defining control tasks, such as set point tracking or disturbance rejection, based on the operating window (Morari et al., 1980; Luyben et al., 1997; Ng and Stephanopoulos, 1998). Well-studied methods like dynamic simulations (Seborg et al., 2016), relative gain array (Van De Wal and De Jager, 2001), and further methods assist engineers.

Previous research on automatic control structure design has led to several promising approaches. The first approaches include knowledge-based expert systems (Williamson, 1989; Song and Park, 1990) and heuristic design procedures (Luyben et al., 1997). Despite these achievements, expert systems are generally expensive to develop and once set up, expert systems tend to be labor-intensive and expensive to maintain and extend (Venkatasubramanian, 2019). Recent advancements in machine learning, specifically in deep learning-based artificial intelligence (AI) have demonstrated remarkable improvements over traditional expert systems in several domains such as natural language processing (NLP) (Popel et al., 2020; Brown et al., 2020). Also in the natural sciences and engineering, deep learning has outperformed rule-based approaches. For example in organic chemistry, a transformer-based model can accurately predict reaction outcomes (Schwaller et al.,

\* Corresponding author.

E-mail address: [A.Schweidtmann@tudelft.nl](mailto:A.Schweidtmann@tudelft.nl) (A.M. Schweidtmann).

2019; Tu and Coley, 2023). Here, rule-based models used molecular templates selected by experts. Chemical engineering is another field where AI holds great potential to assist engineers with design tasks (Schweidtmann, 2024a). This motivates the investigation of AI-based control structure prediction in this research work. Recently, we proposed a generative AI model capable of extending a given PFD with a control structure (Hirtreiter et al., 2023). Hirtreiter et al. (2023) formulate the control structure design as a translation task, i.e., a sequence-to-sequence task, converting a PFD without a control structure into a CEF. The PFD and CEF are represented using the SFILES 2.0 string format (Vogel et al., 2023b). When trained on a synthetic dataset consisting of 100,000 PFD-CEF pairs, the generative AI model was able to predict the correct CEF for a given PFD with an accuracy of 48.6%.

An essential design choice for an AI model is the process information representation. The two most common process information representations are (i) sequences and (ii) graphs. First, generative AI models for chemical process development so far mainly represent processes as sequences, i.e., as strings (Nabil et al., 2019, 2023; Vogel et al., 2023a; Schulze Balhorn et al., 2024; Hirtreiter et al., 2023; Oeing et al., 2022; Azevedo et al., 2024; Li, 2024). The elements of the sequence can represent equipment and the order of the elements defines their connectivity (d'Anterrosches, 2005; Vogel et al., 2023b; Mann et al., 2024). Second, a process can be represented as a graph. A graph is a mathematical structure consisting of a set of nodes and a set of edges that describe relations between nodes. In the context of chemical processes, nodes can represent equipment and edges can represent pipes or signal lines between the equipment. The potential of graphs in AI for process development has already been shown for P&ID verification using a graph neural network (GNN) (Oeing et al., 2022, 2023) and process synthesis using reinforcement learning (Stops et al., 2023). The choice of information representation also influences what kind of generative models can be used. Graph-based approaches for generative AI commonly rely on Variational Graph Autoencoders (VGAEs) (Kipf and Welling, 2016), Graph Recurrent Neural Networks (GraphRNNs) (You et al., 2018), or Graph Generative Adversarial Networks (Graph GANs) (De Cao and Kipf, 2018), while sequence-based approaches rely on Recurrent Neural Networks (RNNs) (Hochreiter and Schmidhuber, 1997) or the transformer (Vaswani et al., 2017). It remains an open research question which form of information representation is most effective for chemical processes.

Graph-to-sequence models promise to combine the advantages of both sequence and graph representation. Graph-to-sequence models consist of two main components: (i) A graph encoder that learns a numerical representation of the input graph and (ii) a sequence decoder that, given the numerical representation, generates a sequence. Graph-to-sequence models were first proposed in NLP (Beck et al., 2018; Song et al., 2018; Xu et al., 2018). Also in the field of chemistry, graph-to-sequence-based models have been used for various tasks such as predicting the outcome of a reaction (Mao et al., 2021; Tu and Coley, 2023), developing new molecules with favorable properties (Wang et al., 2021; Dollar et al., 2023), or synthetic polymer design (Vogel et al., 2023c).

We hypothesize that, in the context of flowsheets, a graph encoder is more data efficient than a sequence encoder for two reasons: First, the sequence encoder captures the flowsheet topology explicitly. In the message passing of the graph encoder, information is only exchanged between nodes (here: flowsheet component) that are connected via an edge (here: pipe or signal line). The sequence encoder treats the input as a fully connected graph. In the sequence attention mechanism, every token (here: flowsheet component, among others) attends to every other token. Sequence-based approaches need to learn the grammar of the sequence (here: SFILES 2.0 notation) to infer the flowsheet topology. Second, the graph encoder is permutation invariant whereas the sequence decoder is permutation sensitive. A function is permutation invariant if permutations to the input do not affect

the output. Permutation invariance is desirable for creating a unique graph embedding (Tu and Coley, 2023). For sequence representations of graphs such as SMILES or SFILES, the model has to “re-learn” or “re-discover” that different sequences share the same underlying graph, which can demand more data or complex augmentations. Another advantage of the graph input is that it is more intuitive to include detailed process data, such as operating points, in a graph (Stops et al., 2023) than in a sequence. In contrast, generating a sequence representation such as SMILES or SFILES is often reported to be computationally more efficient than generating graphs, particularly in machine learning applications (Dollar et al., 2023). A possible explanation for this efficiency is that (i) the structured nature of sequences (here: the grammar of SFILES) simplifies autoregressive decoding in transformer-based models, and (ii) the search space is smaller when generating a linear representation (selecting a token) compared to generating a graph with arbitrary connectivity (selecting a source-target pair and a node type). Therefore, a beneficial inductive bias is to use a sequence decoder and represent the output (the generated CEF) as a sequence.

Previous studies have shown that graph-to-sequence models can improve accuracy compared to purely graph-based (Dollar et al., 2023) or purely sequence-based models (Tu and Coley, 2023). To the best of our knowledge, no graph-based, generative AI method for process development exists yet.

We propose a Graph-to-SFILES model for predicting control structures to aid P&ID development. Thereby, we use permutation invariant graphs to represent PFDs as the model input. To account for the variety of graph encoders available in literature, we compare three state-of-the-art GNN architectures: (i) The Graph Attention Transformer (GATv2) (Veličković et al., 2017; Brody et al., 2021), (ii) the Graph Convolution Transformer (GraphConv) (Shi et al., 2020), (iii) the Graph Transformer (Dwivedi and Bresson, 2020). These architectures are selected based on their relevance in the literature and on their effectiveness in learning graph representations through attention-based message passing (Min et al., 2022; Khemani et al., 2024). In addition, we propose a GNN architecture for the graph encoder that integrates key aspects of the selected GNNs. The representation of the output CEF is chosen as SFILES for efficient CEF generation. Inspired by the success in other domains, we hypothesize that for chemical processes a graph-to-sequence model yields better performance than a purely sequence-based model.

The remainder of this article is structured as follows: Section 2 introduces the information representations used in this work: Process graphs and SFILES 2.0. In Section 3, we describe the Graph-to-SFILES methodology for control structure prediction. Afterward, we explain the data used to train the Graph-to-SFILES model (Section 4). In Section 5, we discuss the model's results and demonstrate its performance in an illustrative case study.

## 2. Information representation of flowsheets

To leverage machine learning and learn from chemical processes and their data, we need to represent chemical processes in a machine-readable format. We represent processes as graphs for the input to our models (Section 2.1) and as SFILES for the output of our models (Section 2.2). The distillation column in Fig. 1(a) serves as an example to illustrate the information representations.

### 2.1. Process graphs

Chemical processes can be viewed as complex networks of unit operations connected through streams. An intuitive choice for the information representation is therefore a graph. We make use of the graph format defined by Vogel et al. (2023b) for PFDs. Here, nodes represent unit operations, control units, raw materials, and products, while directed edges represent streams and signals. Control units that refer to a stream, such as a temperature indicator, are placed in

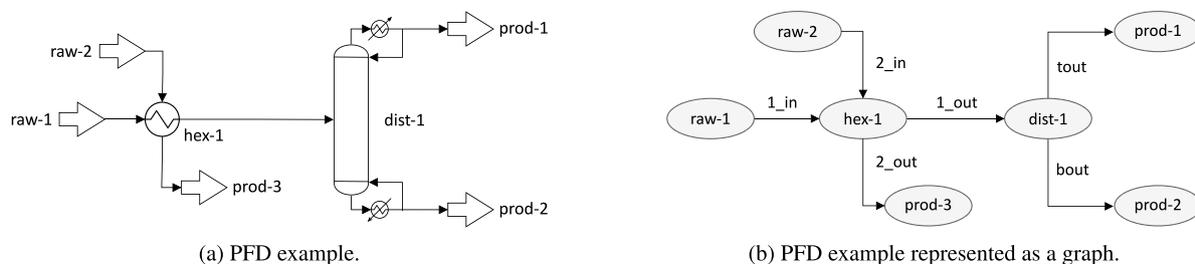


Fig. 1. Graph representation of flowsheets. The corresponding SFILES 2.0 notation is: (raw)(hex){1}(dist)[{bout}(prod)]{tout}(prod)n|(raw)(hex){1}(prod).

the stream by splitting the stream into two. In addition, edges have attributes to store further stream information: If multiple outlet streams exist, we define the outlet location as top outlet (tout) or bottom outlet (bout). For heat exchangers, we identify the incoming and outgoing streams that belong to the same mass train using the edge attribute `StreamID_{in/out}`. Fig. 1(b) shows an example process represented as a graph.

## 2.2. SFILES

An alternative information representation to graphs is the Simplified Flowsheet Input Line Entry System (SFILES). The SFILES was developed by d'Anterrosches (2005) and is inspired by the Simplified Molecular Input Line Entry system (SMILES) (Weininger et al., 1989) used to encode molecular graphs to strings. The SFILES encodes the topology of a process as a sequence of tokens, such as (raw) for the raw material or (r) for a reactor. The order of the tokens is determined by following a path from the raw materials to the products along the streams. In SFILES, square brackets indicate a side stream that diverges from the main process stream. At the place where the side stream diverges the brackets are opened, and the tokens of that side stream are added. When a side stream rejoins the main stream or ends the brackets are closed and the SFILES notation continues from where the stream diverged.

Recently Vogel et al. (2023b) introduced SFILES 2.0, which extends the original SFILES notation with control structures and topology details. Controllers are incorporated using the controller token (C) followed by a tag with the letter code (e.g., {PC}). The position of the controller token is based on its location in the flowsheet. The signal line is encoded similarly to recycles and differentiated by an underscore (\_#, <\_#). Topology details such as a stream leaving a column at the top are incorporated as tags in curly brackets (e.g., {tout}). By using an extra tag before a unit, the SFILES 2.0 encodes where the stream enters a unit, and similarly, by adding a tag in front of the stream leaving a unit it is encoded where the stream leaves. Independent material streams are appended to the SFILES, starting with an n|. Note that there can be multiple SFILES for the same flowsheet but there exists a unique, or canonical, way of writing them, defined by a graph invariant. The graph invariant provides a unique rank for each node that determines the order of the SFILES sequence. The graph invariant is obtained in two steps. First, the Morgan algorithm (Morgan, 1965) determines a node ranking. Second, a set of rules determines the order of the tokens in case the Morgan algorithm does not yield a unique rank for every node. These rules define an order based on (i) the node type (control, outlet, inlet, other), (ii) the number of successors in a depth-first search tree of the graph, (iii) the alphabetic order of the node name, and (iv) the node number (e.g., if the flowsheet contains three heat exchangers, they will be numbered from one to three). These rules are described in Vogel et al. (2023b) and made accessible via a

Python package.<sup>1</sup> To illustrate the notation, we provide the SFILES 2.0 of the distillation column from Fig. 1 in the figure caption.

## 3. Graph-to-SFILES methodology

We propose the Graph-to-SFILES model to predict the control structure of chemical processes. Fig. 2 provides an overview of the model. We start with a PFD that is represented by a graph as described in Section 2.1. Given the graph, the Graph-to-SFILES model generates an SFILES that represents the CEF (Section 2.2). The Graph-to-SFILES model is inspired by the Graph2SMILES model for molecules (Tu and Coley, 2023) and is based on a transformer architecture, as shown in Fig. 3. The model consists of two main components: (i) A graph encoder to encode a PFD given as a process graph and (ii) a sequence decoder that, given the encoded input, generates the CEF in the SFILES 2.0 format. In Section 3.1, we describe the graph encoder. The graph encoder aims to convert the input process graph into a continuous latent-space representation. The latent-space representation is essential for the sequence decoder to generate a suitable control structure. The sequence decoder and the corresponding decoding strategy are described in Section 3.2 and Section 3.3 respectively. Then, we provide details on the training process in Section 3.4.

### 3.1. Graph encoder

We implement the graph encoder as a GNN. GNNs are a type of neural network designed to operate on graph-structured data. The GNN iteratively aggregates information from neighboring nodes (and edges, depending on the architecture) to update the node (and edge) features. We define a sequence of GNN layers, where each layer updates the node features based on their direct neighboring nodes and edges.

A GNN layer generally consists of two basic steps, a message passing step and an update step. During the message passing step, each node sends a message, usually a linear transformation of its feature vector, to its neighboring nodes. The message passing is followed by an update step in which the incoming messages are merged into a single vector. This step uses a permutation invariant function, e.g., *sum*, *mean*, *min*, or *max*. Information is shared across the graph by using multiple iterations of these message-passing and update steps. The linear transformations using learnable parameters in the message passing and update steps allow the learning of complex relationships between the nodes and edges, making their feature vectors more valuable for downstream tasks. In the Graph-to-SFILES model, the individual component nodes get information about their neighboring components and streams.

We propose and compare four different GNN architectures for the generation of CEFs from PFDs, as shown in Fig. 4: (i) The Graph Attention Transformer (GATv2) (Veličković et al., 2017; Brody et al., 2021), (ii) the Graph Convolution Transformer (GraphConv) (Shi et al.,

<sup>1</sup> <https://github.com/process-intelligence-research/SFILES2>.

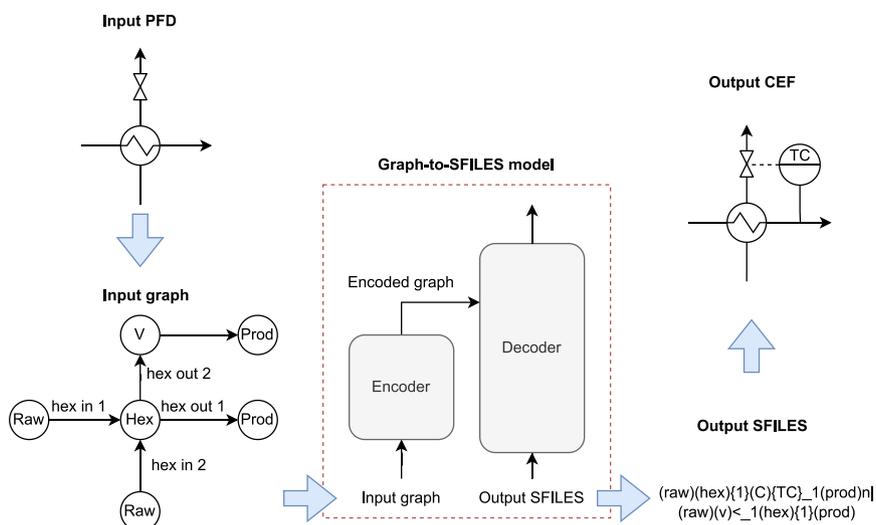


Fig. 2. Overview of Graph-to-SFILES methodology. The PFD is represented as a graph with node and edge features. The Graph-to-SFILES model translates the graph to an SFILES that describes a potential CEF for the given PFD. Note that the control structure shown here is one of multiple possible control options.

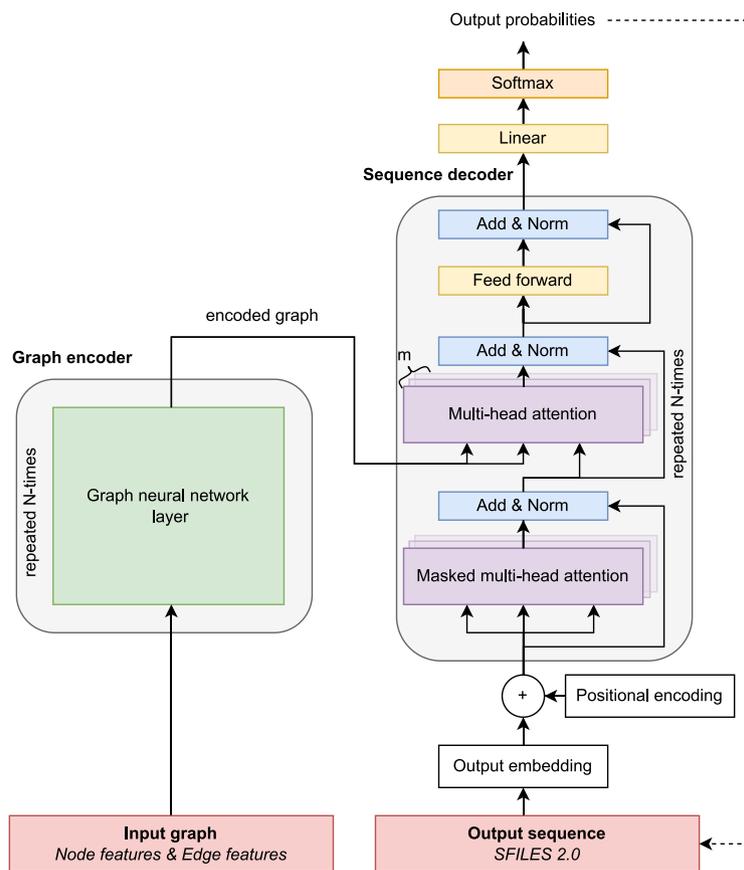
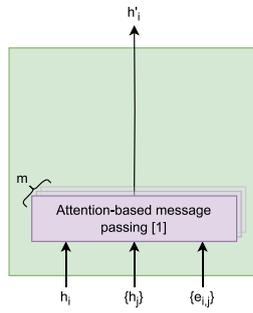


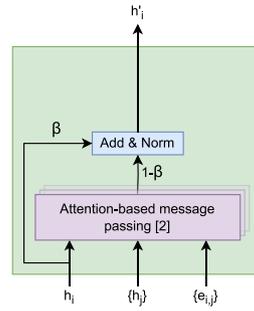
Fig. 3. Detailed Graph-to-SFILES model consisting of a graph encoder and a sequence decoder.

2020), (iii) the Graph Transformer (Dwivedi and Bresson, 2020), and (iv) a new GNN that we propose in this study, the Combined model. The GAT model (Veličković et al., 2017) was the first model to introduce the attention mechanism for GNNs and outperforms previous graph convolution architectures. Similarly to the GAT model, Shi et al. (2020) proposed an attention-based GNN. They report an improved performance compared to the GAT model by introducing a different attention mechanism and including gated residual connections. In

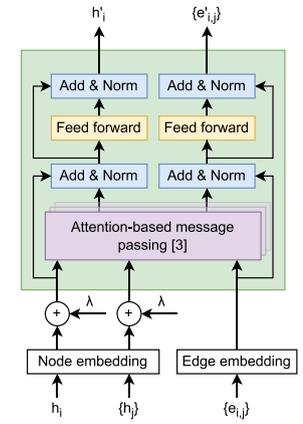
addition, the Graph Transformer (Dwivedi and Bresson, 2020) generalizes the transformer architecture from sequences to graphs. As the Graph Transformer is closely aligned with the transformer encoder architecture, it is expected to integrate well with the overall Graph-to-SFILES transformer approach. Dwivedi and Bresson (2020) state, that the Graph Transformer can be integrated as a black box in various applications due to its simplicity and generality. All architectures utilize



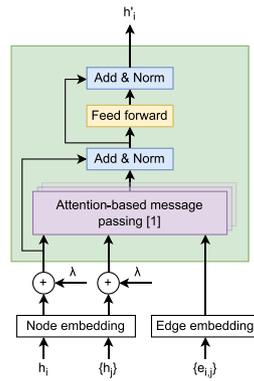
(a) GATv2 (Veličković et al., 2017; Brody et al., 2021)



(b) GraphConv (Shi et al., 2020)



(c) Graph Transformer (Dwivedi and Bresson, 2020)



(d) Combined model

Fig. 4. GNN layers as a drop-in replacement in the graph encoder. Each GNN layer takes the node features  $h$  and the edge features  $e$  as input and returns the updated node feature  $h'$  (and updated edge features  $e'$  for the Graph Transformer). The attention-based message passing is calculated for  $m$  heads in parallel.

the attention mechanism to facilitate the message passing, where attention is the weight of the message between node  $i$  and node  $j$ , denoted as  $\alpha_{i,j}$ .

Here, we describe the main mathematical details of the Graph-to-SFILES encoder. The detailed mathematical expressions of the Combined model are described in Appendix A.1. Firstly, we implement the GATv2 as proposed by Veličković et al. (2017) and modified by Brody et al. (2021) (Fig. 4(a)). In the attention-based message passing block the attention  $\alpha_{i,j} \in \mathbb{R}$  is calculated using

$$\alpha_{i,j} = \text{softmax} \left( \mathbf{a}^T \text{LeakyReLU} \left( \left[ \mathbf{W}_{d_{\text{enc}}} \mathbf{h}_i \parallel \mathbf{W}_{k_{\text{enc}}} \mathbf{h}_j \parallel \mathbf{W}_{e_{\text{enc}}} \mathbf{e}_{i,j} \right] \right) \right)$$

The inputs consisting of node features  $\mathbf{h} \in \mathbb{R}^{d_{\text{enc}}}$  and edge features  $\mathbf{e} \in \mathbb{R}^{d_{\text{enc}}}$  are multiplied by weight matrices  $\mathbf{W} \in \mathbb{R}^{d_{\text{enc}} \times d_{\text{enc}}}$  and then concatenated. After applying the nonlinear function LeakyReLU, the output is multiplied by another weight matrix  $\mathbf{a} \in \mathbb{R}^{3d_{\text{enc}}}$  and normalized using the softmax function. Here,  $d_{\text{enc}}$  is the internal encoder dimension. Note that the first encoder layer uses  $\mathbf{W}_{d_{\text{enc}}}, \mathbf{W}_{k_{\text{enc}}} \in \mathbb{R}^{d_{\text{node}} \times d_{\text{enc}}}$  and  $\mathbf{W}_{e_{\text{enc}}} \in \mathbb{R}^{d_{\text{edge}} \times d_{\text{enc}}}$  to map the node  $d_{\text{node}}$  and edge  $d_{\text{edge}}$  dimension to the encoder dimension. After calculating the attention between the nodes, we determine the updated node features  $\mathbf{h}'$ :

$$\mathbf{h}'_i = \text{ReLU} \left( \sum_{j \in \mathcal{N}'_i} \alpha_{i,j} \mathbf{W}_{v_{\text{enc}}} \mathbf{h}_j \right).$$

For each node  $i$ , the message from node  $j$  in  $\mathcal{N}'_i$  is given as  $\alpha_{i,j} \mathbf{W}_{v_{\text{enc}}} \mathbf{h}_j$  with  $\mathcal{N}'_i$  being the direct neighbors of node  $i$  including itself. In the update step, the messages are summed up, followed by a nonlinearity (ReLU).

Secondly, we implement the GraphConv (Fig. 4(b)). The GraphConv utilizes scaled dot-product attention, where the attention  $\alpha_{i,j}$  is given by

$$\alpha_{i,j} = \text{softmax} \left( \frac{(\mathbf{W}_{d_{\text{enc}}} \mathbf{h}_i)(\mathbf{W}_{k_{\text{enc}}} \mathbf{h}_j + \mathbf{W}_{e_{\text{enc}}} \mathbf{e}_{i,j})^T}{\sqrt{d_{\text{enc}}}} \right).$$

In addition, the GraphConv makes use of a gated residual connection. The gated residual connection should prevent the graph encoder from over-smoothing by injecting the residual node feature  $\mathbf{h}_i$  directly in the updated node feature  $\mathbf{h}'_i$ . Here, a weight ( $\beta = f(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{i,j}) \in [0, 1]$ ) is calculated which determines the importance between the residual and the message passing. Then, the outcome of the gated residual connection is normalized using layer normalization and passed through a final nonlinearity (ReLU). This results in an updated node feature ( $\mathbf{h}'_i$ ).

Thirdly, we implement the Graph Transformer (Fig. 4(c)). The attention used in their model is defined as

$$\alpha_{i,j} = \text{softmax} \left( \frac{(\mathbf{W}_{d_{\text{enc}}} \mathbf{h}_i)(\mathbf{W}_{k_{\text{enc}}} \mathbf{h}_j)^T}{\sqrt{d_{\text{enc}}}} \cdot \mathbf{W}_{e_{\text{enc}}} \mathbf{e}_{i,j} \right).$$

The unique characteristic of the Graph Transformer is the positional encoding. While the message passing step passes messages only from direct neighbors of a node, the positional encoding encodes the complete graph structure in each node feature. Dwivedi and Bresson (2020) use eigenvectors of the graph's Laplacian matrix to create a positional encoding  $\lambda \in \mathbb{R}^{d_{\text{enc}}}$ . The Laplacian positional encoding is added to the node features in the first GNN layer that also includes a linear transformation for node ( $d_{\text{node}} \rightarrow d_{\text{enc}}$ ) and edge embedding ( $d_{\text{edge}} \rightarrow d_{\text{enc}}$ ). Furthermore, the Graph Transformer includes two residual connections after the attention-based message passing with a feed-forward neural network (FFN) between the residual connections.

Fourthly, we propose and implement the Combined model, which combines the dynamic attention mechanism from GATv2 with the positional encoding and residual connections from the Graph Transformer architecture, as shown in Fig. 4(d).

Similar to the attention used in the work of Vaswani et al. (2017), multi-head attention is used for all graph encoders to capture multiple patterns during the learning. Instead of calculating only one attention weight,  $m$  attention weights are calculated in parallel where each attention calculation is called a head. The outcomes of the  $m$  heads are concatenated for the remaining operations. Finally, the updated node features from the last GNN layer, representing the encoded process graph, are passed to the sequence decoder.

### 3.2. Sequence decoder

The decoder generates the CEF in the SFILES 2.0 notation, given the PFD as an encoded process graph. We follow the original implementation of the transformer decoder by Vaswani et al. (2017), as shown in Fig. 3.

The decoder generates an output sequence iteratively. The first token of the output sequence is always a start-of-sequence token (SOS). Then, every forward pass of the decoder adds a new token to the output sequence until the new token is an end-of-sequence token (EOS). This generation technique is called autoregression, meaning the model generates its output step-by-step, based on previously generated tokens and the input sequence. Autoregression allows the model to create novel and coherent sequences.

Before the output tokens are processed by the decoder, they are transformed into continuous vector representations, or embeddings, of dimension  $d_{dec}$  using a linear transformation. The embedding is complemented by positional encoding. The decoder does not contain a function that keeps track of the sequence order, like recurrence or convolutions. To address this, the Graph-to-SFILES model uses pairs of sinusoidal functions (Vaswani et al., 2017). A positional encoding vector is calculated for every input token and added to the embedding, allowing the model to differentiate identical tokens at different positions. The resulting embedding is fed into the decoder.

The decoder contains multiple layers and each layer contains blocks of attention mechanisms and FFNs. First, the decoder analyzes the embedded output  $\mathbf{x} \in \mathbb{R}^{d_{dec}}$  with the scaled dot-product attention mechanism as described by

$$\text{Attention}(\mathbf{x}) = \text{softmax} \left( \frac{(\mathbf{W}_{q_{dec}} \mathbf{x})(\mathbf{W}_{k_{dec}} \mathbf{x})^\top}{\sqrt{d_{dec}}} \right) \mathbf{W}_{v_{dec}} \mathbf{x}.$$

Similar to the attention in the graph encoder (Section 3.1), attention describes the importance of one token for the context of another token. The attention mechanism is enhanced by using  $m$  attention heads that calculate the attention independently. The resulting outputs of each attention head are concatenated as introduced in Section 3.1. Second, we introduce the encoded process graph to the decoder. The encoded graph is multiplied by the query matrix  $\mathbf{W}_{q_{dec}}$  and key matrix  $\mathbf{W}_{k_{dec}}$  in a further attention block. The output of the previous attention block is multiplied by the value matrix  $\mathbf{W}_{v_{dec}}$ . Third, an FFN is applied to transform the output of the self-attention mechanism. Each block has a residual connection where the block's output and the residual are summed and normalized using *LayerNorm*. Finally, the sequence decoder's output is mapped to a probability for the next token to be added to the output sequence.

### 3.3. Decoding strategy

The decoding strategy defines how the next token of the output sequence is selected. The computationally cheapest strategy is greedy search. At every prediction step of the decoder, greedy search selects the token with the highest probability. However, selecting the token with the highest probability likely does not lead to the sequence with

the highest joint probability. In addition, the coherence of the output sequence can be suboptimal, with subsequences that are not related to each other. In contrast to greedy search, calculating all possible sequences and comparing their joint probability is intractable.

To overcome the aforementioned limitations, we use beam search as the decoding strategy. At the first decoding step, beam search keeps track of the  $k$  tokens with the highest probability, where each token represents a beam. In the next decoding step, the model returns a probability distribution for the next token of each beam. Using this distribution, the joined probability of extending the beams is calculated. Then, the  $k$  combinations yielding the highest joined probabilities are selected and used as beams for the next decoding step. This process is repeated until all beams have reached the end-of-sequence token (<EOS>) or a maximum number of tokens. Then, the model returns the beams in order of highest joint probability, yielding top-k SFILES predictions.

### 3.4. Training

The objective function for training the Graph-to-SFILES model is the cross-entropy loss that compares the probability of the predicted next token with the ground truth token. During the training, we use teacher-forcing. Teacher-forcing is a training strategy for supervised learning, c.f., if the correct output sequence is known. At each decoding step, the model output is corrected to force the model to predict the correct output sequence for the given input sequence. The Graph-to-SFILES model is implemented in Python using the PyTorch and PyTorch Geometric libraries.

## 4. Data

We train the proposed Graph-to-SFILES model on a dataset containing 100,000 pairs of PFDs and CEFs which were synthetically generated from a collection of process patterns from literature. Every pattern contains a decentralized control structure. The patterns are combined following engineering heuristics to generate complete chemical process schematics. Each process starts with up to three feed streams which may be preprocessed by a temperature change, pressure change, or mixing. Then, either thermal separation or reaction is selected. This step is either repeated or the process is terminated and the products are conditioned. For further details of the dataset, we refer the interested reader to Hirtreiter et al. (2023). The resulting flowsheets primarily represent oil and gas processes, as these are commonly simulated in Aspen Plus. It includes unit operations such as gas-phase reactors, heat exchangers, compressors, pumps, and distillation columns, along with their associated control structures. Additionally, often multiple valid control options exist. This is reflected in our dataset. For instance, we extend the PFD topology of a distillation column with seven different control structures. An overview of the process patterns used to generate the synthetic data can be found in the Supplementary Information of Hirtreiter et al. (2023). We report dataset statistics in Table 1. While we aim to develop a dataset of flowsheets that resembles oil and gas processes by mimicking their structure and scale, our flowsheets are simplified and do not fully represent an actual industrial process. The diversity and complexity of equipment and control structures in industrial processes are significantly higher. Additionally, since we sample patterns based on a given probability distribution, there is a possibility that incorrect combinations of decentralized control structures may be selected. Furthermore, the current dataset focuses on oil and gas processes. Extending the methodology to other fields, such as bioprocessing, would require developing a new dataset that includes relevant unit operations (e.g., fermenters, liquid-liquid membrane separators) and their corresponding control strategies. However, this is a non-trivial task and comes with a significant workload. Creating a dataset for a different domain would involve: (i) Collecting representative process topologies from sources such as process simulators, literature,

**Table 1**  
Dataset properties.

Statistic	Value
Training samples	1000/10 000/100 000
Validation samples	1000
Test samples	1000
Average number of nodes	52 ± 20
Average number of edges	61 ± 23
Average out-degree	1.2
Vocabulary size	113

**Table 2**  
Encoding scheme used for edge attributes.

Feature dimension	Edge attribute	Encoding value		
		0	1	2
1	Outlet-stream position	Unspecified	Top	Bottom
2	Hex inlet	Unspecified	Side 1	Side 2
3	Hex outlet	Unspecified	Side 1	Side 2

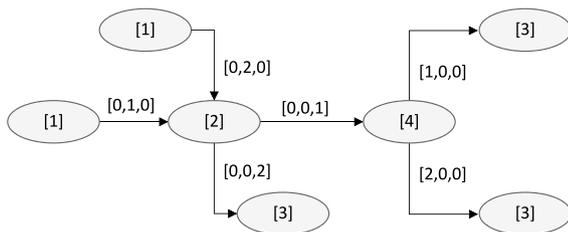


Fig. 5. Graph representation of a PFD, Input to the graph encoder.

patents, or industrial case studies, (ii) defining appropriate control structures for the new domain, as control strategies may differ significantly from those in oil and gas, (iii) extending the SFILES 2.0 notation to include any missing unit operations, and (iv) curating and verifying the dataset to ensure it captures realistic process control principles.

There are two versions of the dataset. First, the controllers are removed from the flowsheet to create the PFD from the CEF. Secondly, the valves were removed in addition to make the schematics more in line with a flowsheet found in simulation software like Aspen. The usage of this dataset will be indicated as without valves.

Pre-processing of the input data is needed to turn the SFILES into the graph format from Section 2.1. Firstly, we utilize the open-source SFILES 2.0 repository (<https://doi.org/10.5281/zenodo.6901932>) to convert the flowsheets from the SFILES 2.0 format to a NetworkX graph. As described in Section 2.1, unit operations, valves, and controllers are represented as nodes and streams and signals as edges.

Secondly, we convert the attributes in the NetworkX graph to node and edge features. The node attributes, specifically unit types, are encoded in a one-dimensional vector using a dictionary. The dictionary stores a relationship between the unit type and an integer value, for instance, (raw) equals 1, (hex) equals 2, etc. The edge attributes are translated into a three-dimensional vector. The first dimension encodes information from which side of a unit a stream is leaving. The second and third dimensions encode to which side of a heat exchanger a stream is connected, according to the scheme shown in Table 2. These two pre-processing steps yield the processed graph shown in Fig. 5.

Similarly to the input data, we pre-process the output data. We convert the output SFILES strings to a numerical vector using the tokenizer by Hirtreiter et al. (2023). The tokenizer breaks the SFILES string into tokens from the SFILES vocabulary. Then the tokens are mapped to a numerical vector which can be processed by the Graph-to-SFILES model. Lastly, the input graph and output SFILES are combined into a PyTorch Geometric data object.

## 5. Results and discussion

In the following, we describe and discuss the performance of the Graph-to-SFILES model when trained on a dataset of synthetic PFD-CEF pairs. In Section 5.1, we report evaluation metrics for the Graph-to-SFILES model and compare the four model architectures. In Section 5.2, we compare the Graph-to-SFILES model and our previous sequence-to-sequence approach (Hirtreiter et al., 2023). In addition, we select an illustrative example from the test dataset and present the model predictions in Section 5.3. Finally, we discuss the results (Section 5.4).

### 5.1. Graph-to-SFILES evaluation

We train the Graph-to-SFILES model using four different graph encoder architectures (Section 3.1) on the 1k, 10k, and 100k dataset from Hirtreiter et al. (2023). All model trainings are performed on a Windows Server 2022 Datacenter with 256 GB RAM and an NVIDIA A100 GPU. We optimize hyperparameters for each architecture using a grid search. In total, we run 288 trainings on the 10k dataset for 7.3 days with an average duration of 36.7 min per training. We select the optimal hyperparameters based on the validation loss. The optimal hyperparameters and a reference to the full grid search are given in Table 5. We observe that the learning rate and the number of decoder layers are the most important hyperparameters as they have the highest correlation with the validation loss. The learning rate is especially important for the training to converge. Trainings that do not converge have a relatively high learning rate of  $1.5e-3$  or  $2e-3$  in common. In addition, the validation loss is generally lower for larger model architectures. For three out of the four models, the optimal architecture has the maximum number of encoder and decoder layers. Furthermore, the effect of the batch size was insignificant. Then, we apply the optimal hyperparameters to train five model samples for the 1k and 10k datasets and three model samples for the 100k dataset, as well as the corresponding datasets without valves. The model samples have a different weight initialization to gain a representative model performance. Each training runs for 100 episodes. Finally, we evaluate all models on an independent test dataset and summarize the results, including the mean and standard deviation of the samples, in Table 3.

We analyze the Graph-to-SFILES model performance using three evaluation metrics. First, the models are evaluated on the CEF accuracy. The CEF accuracy measures the models' capabilities of fully recreating the CEF based on a given PFD, equivalent to the accuracy used by Hirtreiter et al. (2023). Second, we calculate the PFD reconstruction accuracy. Due to the autoregressive Graph-to-SFILES decoder, the model is also tasked with recreating the input PFD as an SFILES string. This means that, besides making an error in predicting a control structure, the model can also make an error in generating the underlying process. Hence, to evaluate whether a model also has difficulties generating the underlying process of a flowsheet, we introduce the PFD reconstruction accuracy metric, which only focuses on the process side of the output. We remove the control structures of the predicted CEF, yielding PFD-like schematics. These newly obtained schematics are then compared to the PFD fed to the model. Thirdly, we calculate the Bilingual Evaluation Understudy (BLEU) score (Papineni et al., 2002). Since the CEF accuracy only considers a CEF correct if the predicted SFILES matches the ground truth SFILES exactly, the CEF accuracy cannot differentiate the degree of error. To overcome this, the BLEU score is implemented. BLEU is a scoring metric commonly used in NLP to evaluate the quality of a machine-translated text. The BLEU score divides the predicted SFILES 2.0 into segments and compares each segment to the ground truth segment. The score is then calculated as a percentage and averaged over the entire dataset. Here, we use a segment length of ten. As explained in Section 3.3, during inference the model utilizes a beam search-based decoding strategy, yielding the top- $k$  predictions sorted by their joined probability. Using these predictions, a top- $k$  score can be calculated for the three

**Table 3**

Performance of the Graph-to-SFILES model on an independent test dataset using different encoder architectures and different training dataset sizes. We report the mean and the standard deviation for multiple model samples. The best performance per dataset and evaluation metric is highlighted in bold font.

Model architecture	Dataset	CEF accuracy		PFD reconstruction		BLEU	
		Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)
Graph-to-SFILES (GATv2 encoder)	1k	0.5 ± 0.3	1.2 ± 0.7	1.0 ± 0.3	2.2 ± 1.1	11.8 ± 1.6	17.3 ± 2.2
	10k	36.0 ± 1.9	64.9 ± 2.0	65.8 ± 2.1	83.1 ± 1.6	62.5 ± 1.0	82.0 ± 1.3
	100k	45.1 ± 0.8	77.4 ± 1.6	79.2 ± 1.0	94.7 ± 0.8	68.6 ± 0.4	88.6 ± 0.6
Graph-to-SFILES (GraphConv encoder)	1k	5.6 ± 1.1	10.9 ± 1.7	9.9 ± 1.5	17.4 ± 2.2	31.5 ± 1.4	41.8 ± 1.8
	10k	39.2 ± 2.2	68.3 ± 2.6	69.7 ± 4.3	86.4 ± 2.9	64.5 ± 1.2	83.9 ± 1.3
	100k	45.9 ± 0.4	78.8 ± 0.6	80.3 ± 1.7	96.7 ± 0.6	69.1 ± 0.4	89.8 ± 0.3
Graph-to-SFILES (Graph Transformer encoder)	1k	10.5 ± 0.7	21.9 ± 1.5	19.1 ± 1.5	34.2 ± 2.0	39.1 ± 0.6	52.1 ± 0.5
	10k	29.0 ± 0.2	63.5 ± 1.2	50.9 ± 2.1	81.9 ± 2.6	60.5 ± 1.2	81.7 ± 0.7
	100k	30.0 ± 3.0	63.0 ± 5.6	46.3 ± 3.6	82.1 ± 6.2	61.1 ± 2.4	81.4 ± 3.6
Graph-to-SFILES (Combined model encoder)	1k	<b>15.2 ± 0.1</b>	<b>28.4 ± 1.7</b>	<b>28.5 ± 1.5</b>	<b>41.8 ± 1.4</b>	<b>43.2 ± 1.1</b>	<b>57.7 ± 0.8</b>
	10k	<b>41.9 ± 0.6</b>	73.2 ± 0.9	74.0 ± 0.2	90.8 ± 1.3	<b>66.7 ± 0.6</b>	87.0 ± 0.6
	100k	46.1 ± 0.7	79.0 ± 0.3	80.5 ± 0.5	96.8 ± 0.5	69.2 ± 0.6	89.8 ± 0.2
Sequence-to-sequence (Hirtreiter et al., 2023)	1k	0.3	0.9	0.3	1.7	10.5	18.7
	10k	37.7	<b>74.8</b>	<b>81.4</b>	<b>91.0</b>	63.3	<b>90.1</b>
	100k	<b>48.6</b>	<b>89.2</b>	<b>99.2</b>	<b>99.5</b>	<b>70.8</b>	<b>96.4</b>

**Table 4**

Evaluation on the datasets where the input PFDs do not contain valves (without valves). We report the mean and the standard deviation for multiple model samples. The best performance per evaluation metric on the 10k dataset is highlighted in bold font.

Model architecture	Dataset	CEF accuracy		PFD reconstruction		BLEU	
		Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)
Graph-to-SFILES (Combined model encoder)	1k	6.7 ± 1.2	17.3 ± 1.3	13.4 ± 1.7	27.2 ± 1.4	38.0 ± 1.2	52.6 ± 0.6
	10k	<b>21.0 ± 0.6</b>	46.9 ± 1.2	37.8 ± 0.8	63.9 ± 0.5	<b>58.9 ± 0.4</b>	78.2 ± 0.4
	100k	24.1 ± 0.3	54.6 ± 0.7	44.8 ± 1.0	69.5 ± 0.4	62.1 ± 0.2	83.4 ± 0.5
Sequence-to-sequence (Hirtreiter et al., 2023)	10k	17.8	<b>48.0</b>	<b>41.5</b>	<b>67.7</b>	55.8	<b>81.6</b>

evaluation metrics. As multiple valid control structures can exist, it is important to evaluate multiple model outputs. This reflects that the engineer can choose among multiple control options. Finally, we convert all predicted SFILES into their canonical form before calculating the evaluation metrics.

First, we compare the graph-to-sequence model architectures proposed in this work. When comparing the CEF accuracy, the results indicate that the Combined model encoder performs best. It reaches the highest CEF accuracy for the 1k (top-1: 15.2%), 10k (top-1: 41.9%), and 100k (top-1: 46.1%) dataset. For the 1k dataset, the performance of the other graph encoders in descending order is as follows: Graph Transformer (top-1: 10.5%), GraphConv (top-1: 5.6%), and GATv2 (top-1: 0.5%). While the GATv2-based model and the GraphConv-based model perform similarly to the Combined model-based model for the 10k and 100k datasets, the GraphTransformer-based model performs significantly worse. We observe similar results when comparing the four graph encoders on the PFD reconstruction accuracy and the BLEU score. The Combined model reaches the highest scores for all datasets, both for the top-1 and top-5 scores. The GATv2 model scores lowest for the 1k dataset, for instance, a top-1 BLEU score of 11.8% while the other architectures score over 30%. Table 3 shows that the GATv2 and the GraphConv reach a similar score to the Combined model for the 10k and 100k datasets. For instance, the GATv2 and GraphConv reach a top-5 PFD reconstruction accuracy for the 100k dataset of 94.7% and 96.7% respectively, less than 2.1% short of the Combined model. We conclude that the Combined model-based architecture is best suited for the given chemical process data. In the following, we further analyze the Graph-to-SFILES model using the Combined model encoder.

The results suggest that the Graph-to-SFILES model can successfully learn to read process graphs and predict the control structure using the SFILES notation. For instance, 46.1% of control structures are correctly predicted when we use 100,000 training samples.

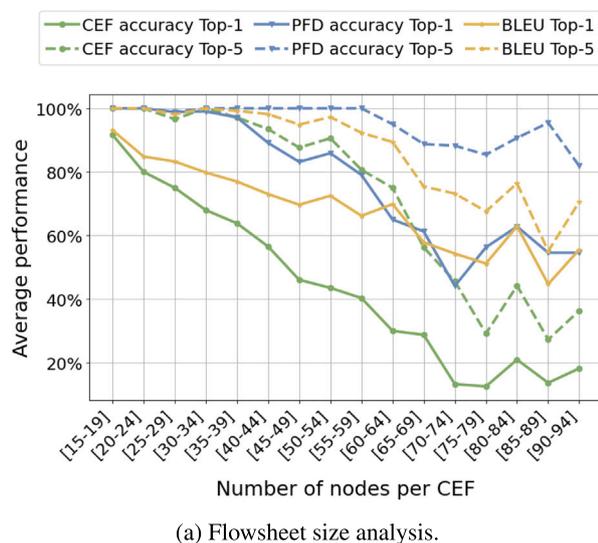
We can observe that the model performance significantly increases with the dataset size. A closer inspection of the table indicates that the marginal benefits of enlarging the dataset decrease with the dataset

size. While the CEF top-1 accuracy increases by 180% from the 1k to the 10k dataset, it increases only by 12% from the 10k to the 100k dataset. We observe a similar trend for the PFD reconstruction accuracy and the BLEU score.

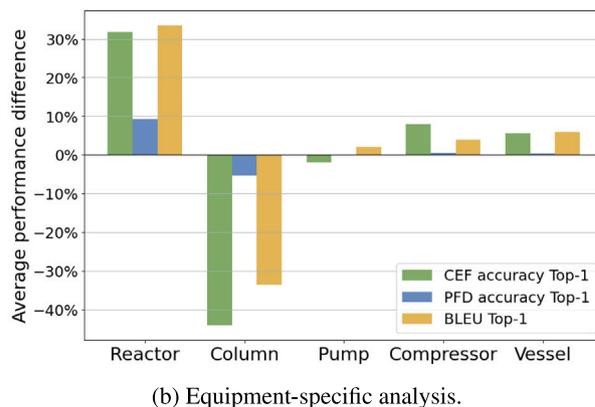
In addition, we compare the CEF and the PFD reconstruction accuracies. The PFD reconstruction is an easier task than the CEF prediction, as the PFD is a subset of the CEF. From the data in Table 3, it is apparent that the model makes mistakes both for the PFD reconstruction and the control structure prediction. The PFD reconstruction accuracy is below 100% for all datasets: The PFD reconstruction accuracy reaches 28.5% (1k dataset), 74% (10k dataset), and 80.5% (100k dataset). The CEF accuracy is always below the PFD reconstruction accuracy, indicating additional mistakes in the control structure prediction.

Another interesting trend is the performance on the smallest, 1k dataset. The BLEU score reaches 43.2% (top-1) and 57.7% (top-5). This shows that if we divide the model output into segments of ten tokens, the Graph-to-SFILES model can predict around half of the output segments correctly. In addition, even with a relatively small dataset, the top-1 CEF accuracy is 15.2%. We expect that the performance especially on the smaller scale dataset, e.g., with 1000 training flowsheets, is relevant. More detailed and industrially-relevant datasets are likely to be small-scale and require data-efficient model architectures.

Finally, we perform a detailed analysis of the Combined model encoder trained on the 100k dataset. Fig. 6(a) shows the average performance of the model for different ranges of flowsheet sizes. We observe that for larger CEFs, all evaluation metrics decrease. For instance, the top-1 CEF accuracy decreases from over 80% for flowsheets with up to 24 nodes to less than 20% for flowsheets with more than 85 nodes. Interestingly, the relative error density increases for larger flowsheets, not just the absolute number of incorrect predictions. This is reflected in the decreasing BLEU score with increasing flowsheet size. A possible explanation is the autoregressive nature of the decoder—once an incorrect token is predicted, it can affect subsequent predictions. Further analysis is needed to confirm this. Besides the flowsheet size, we analyze the impact of the presence of different equipment on the



(a) Flowsheet size analysis.



(b) Equipment-specific analysis.

Fig. 6. Detailed analysis of Graph-to-SFILES (Combined model encoder) performance. Model is trained on 100k dataset. The figure shows (a) the average performance of the model for different ranges of flowsheet sizes and (b) the difference between the average top-1 evaluation metrics for flowsheets where a specific equipment is present or absent.

top-1 performance (Fig. 6(b)). We calculate the average top-1 evaluation metrics for flowsheets where the equipment is present or absent. In Fig. 6(b), we report the difference between these values. For the equipment pump, compressor, and vessel, the difference is below 10% and relatively low. Furthermore, the CEF accuracy and BLEU score increase by more than 30% if a reactor is present while they decrease by more than 30% for a distillation column. Considering that the PFD accuracy only changes by less than 10% in both cases, the performance differences are likely related to the control structure. As mentioned in Section 4, multiple valid control structures can exist for a PFD, resulting in a one-to-many mapping. Especially the distillation column pattern has multiple control options. Therefore, the performance metrics likely underestimate the model performance when distillation columns are present.

### 5.2. Comparison Graph-to-SFILES and sequence-to-sequence model

A comparison between the Graph-to-SFILES model using the Combined model encoder and the sequence-to-sequence model by Hirtreiter et al. (2023) suggests that the model performance depends on the dataset size. On the 1k dataset, we see significantly better performance when using a graph encoder instead of a sequence encoder. Table 3 shows that the CEF accuracy increases from 0.3% to 15.2% (top-1) and from 0.9% to 28.4% (top-5). By contrast, the sequence-to-sequence model reaches higher scores than the Graph-to-SFILES model in all evaluation metrics for the 100k dataset. While the sequence-to-sequence model has a CEF accuracy of 48.6% (top-1) and 89.2% (top-5), the Graph-to-SFILES model scores lower with 46.1% (top-1) and 79% (top-5). Especially the top-1 PFD reconstruction accuracy is significantly higher for the sequence-to-sequence model at 99.2% compared to the Graph-to-SFILES model at 80.5%. On the 10k dataset, we observe a similar performance between the Graph-to-SFILES and the sequence-to-sequence model. Here, the Graph-to-SFILES model reaches a higher top-1 CEF accuracy and BLEU score and the sequence-to-sequence reaches higher scores in the other evaluation metrics. Likewise, both models yield similar results when compared to a dataset where the input PFDs do not contain valves (10k without valves) as shown in Table 4.

Hirtreiter et al. (2023) also trained the sequence-to-sequence model on a small dataset of 250 P&IDs from industrial and academic sources and on a small dataset of 100 synthetic flowsheets. They report accuracies of 0%. We train the Graph-to-SFILES model on the same datasets. First, we use the 250 P&IDs from industrial and academic sources to

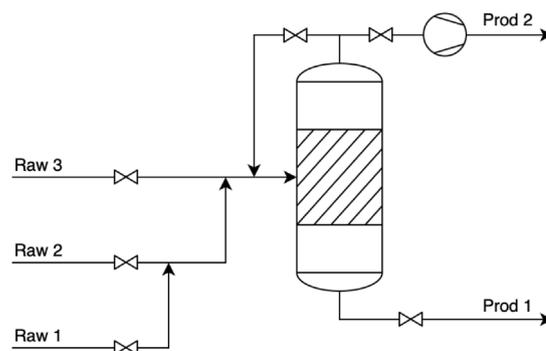


Fig. 7. Illustrative example PFD.

fine-tune the Combined model trained on the 100k dataset. However, the model evaluation showed very low scores for all evaluation metrics. We presume that the distribution of the 250 P&ID graphs from industrial and academic sources is too different from the synthetic dataset used for pre-training. Second, we train the Graph-to-SFILES model on the 100 synthetic flowsheets. The resulting CEF accuracy and PFD reconstruction accuracy are below 1%, suggesting that a larger training dataset is needed to successfully predict control structures.

### 5.3. Illustrative example

To illustrate the Graph-to-SFILES model performance, we predict the CEF for a sample PFD from the independent test dataset. The flowsheet, shown in Fig. 7, contains three raw material streams which are mixed and fed to a reactor. The reactor has two outlets of which one is split to be partly recycled and partly compressed.

We employ the Combined model encoder trained on 100,000 flowsheets. From the three model samples presented in 3, we select the sample with the highest top-1 CEF accuracy. The model predicts the following five SFILES 2.0. Here, the five predictions correspond to the five beams with the highest joint probability during beam search decoding, ordered by descending joint probability. Fig. 8 illustrates the predictions as flowsheets, in which the correctly added controllers are shown in blue, whereas incorrect additions are marked in red.

1. (raw) (C){FC}\_1(v)<\_1(mix)<&|(raw) (C){FC}\_2(v)&<\_2|(mix)<&|(raw) (C){FC}\_3(v)&<\_3|(mix)<1(r)<\_4[(C){TC}\_4] [(C){LC}\_5] [{bout}(v)<\_5(prod)] {tout}(C){PC}\_6(v)<\_6(splt) [(comp) [(C){M}<\_7] (C){PC}\_7(prod)] (C){FC}\_8(v)1<\_8
2. (raw) (C){FC}\_1(v)<\_1(mix)<&|(raw) (C){FC}\_2(v)&<\_2|(C){FT}\_3(mix)<&|(raw) (C){FFC}\_4<\_3(v)&<\_4|(mix)<1(r)<\_5[(C){TC}\_5] [(C){LC}\_6] [{bout}(v)<\_6(prod)] {tout}(C){PC}\_7(v)<\_7(splt) [(comp) [(C){M}<\_8] (C){PC}\_8(prod)] (C){FC}\_9(v)1<\_9
3. (raw) (C){FC}\_1(v)<\_1(mix)<&|(raw) (C){FC}\_2(v)&<\_2|(mix)<&|(raw) (C){FC}\_3(v)&<\_3|(mix)<1(r) [(C){TI}] [(C){LC}\_4] [{bout}(v)<\_4(prod)] {tout}(C){PC}\_5(v)<\_5(splt) [(comp) [(C){M}<\_6] (C){PC}\_6(prod)] (C){FC}\_7(v)1<\_7
4. (raw) (C){FC}\_1(v)<\_1(mix)<&|(raw) (C){FC}\_2(v)&<\_2|(C){FT}\_3(mix)<&|(raw) (C){FFC}\_4<\_3(v)&<\_4|(mix)<1(r) [(C){TI}] [(C){LC}\_5] [{bout}(v)<\_5(prod)] {tout}(C){PC}\_6(v)<\_6(splt) [(comp) [(C){M}<\_7] (C){PC}\_7(prod)] (C){FC}\_8(v)1<\_8
5. (raw) (C){FC}\_1(v)<\_1(mix)<&|(raw) (C){FC}\_2(v)&<\_2|(mix)<&|(raw) (C){FC}\_3(v)&<\_3|(mix)<1(r)<\_4[(C){TC}\_4] [(C){LC}\_5] [{bout}(v)<\_5(prod)] {tout}(C){PC}\_6(v)<\_6(splt) [(C){FC}\_7(v)1<\_7] (comp) [(C){M}<\_8] (C){PC}\_8(prod)

We compare the model predictions with the ground truth CEF. The third prediction (Fig. 8(c)) is equal to the ground truth CEF. Each raw material stream is controlled independently by a flow controller. The reactor has a temperature indicator, a level controller to control the bottom outlet, and a pressure controller to control the top outlet. In addition, the recycle contains a flow controller. Finally, a pressure controller acts on a motor to control the compressor at the top outlet. When looking at the model prediction above, we identify the following trends:

- (i) The syntax of all five generated SFILES is correct.
- (ii) The model correctly regenerates the PFD in all five cases.
- (iii) The model predicts the majority of controllers correctly. In particular, the third prediction is correct and the top 5 predictions have at least 6 out of 8 controllers correctly predicted. This indicates that the model learns to read the process graph and to predict a suitable control structure.
- (iv) The first and fifth predictions are identical in their flowsheet format, but in the SFILES notation they are not. The first prediction is the canonical SFILES which first adds the outlet stream and then closes the recycle. In contrast, the recycle and outlet order is reversed in the fifth prediction. This shows that the model does not always generate canonical, or unique, SFILES representations.
- (v) We observe that the model makes two types of predictions that differ from the ground truth. First, it predicts a temperature controller (TC) instead of an indicator (TI) for the reactor in the first, second, and fifth predictions. A temperature indicator would be sufficient for an adiabatic reactor, while a temperature controller is required for a diabatic reactor. Here, it is not known whether the reactor is adiabatic or diabatic. Second, it predicts a flow ratio controller (FFC) for the inlet Raw 3

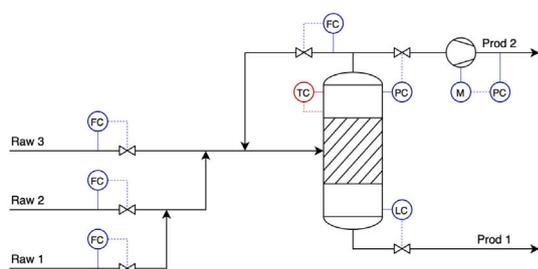
instead of a regular flow controller (FC) in the second and fourth predictions. The flow ratio controller keeps the reactants in the desired proportion instead of controlling the flow of the reactants individually. In both cases, the two control alternatives are valid process configurations and are present in the training data. A possible explanation for the wrong predictions could be limited information in the model input. The PFD data do not contain any process-specific information. Thus, the model has no information to determine the correct control structure other than any potential bias in the dataset.

#### 5.4. Discussion

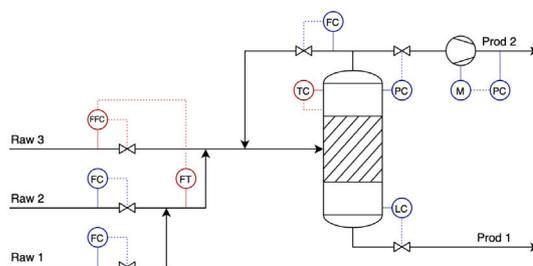
Overall, the results indicate that the Graph-to-SFILES model successfully learns node and edge features in process graphs to generate a CEF. The Graph-to-SFILES model thereby serves as a proof-of-concept for leveraging graph data in control structure prediction. As such, the Graph-to-SFILES model is meant as a support tool for engineers and not to fully automate the control structure design.

A key observation is the difference in performance between the graph encoder and the sequence encoder. We expect that the graph encoder is more data efficient than the sequence transformer but that for larger-scale data, the difference between the encoders diminishes as the sequence encoder has sufficient data to learn the SFILES 2.0 notation. This aligns with our results on the 1k dataset, where the graph encoder outperforms the sequence encoder. However, our results also show that the sequence encoder performs considerably better for the large-scale, 100k dataset. The most likely explanations for the sequence encoder outperforming the graph encoder on the large-scale dataset are: First, the input and output formats of the sequence-to-sequence model are SFILES. Using the same format in input and output might make it easier for the sequence-to-sequence model to preserve information and reconstruct the PFD topology in the output. We see that especially the PFD reconstruction is significantly better for the sequence-based model compared to the graph-based model. Second, GNNs are prone to over-smoothing (Rusch et al., 2023). For an increasing number of layers in the graph encoder, node features become more similar, up to a point where unique features of a node are lost. This limits the depth of GNNs and therefore the learning capacity. Related to this, capturing long-range dependencies is challenging for GNNs (Topping et al., 2021; Rusch et al., 2023). Contrarily, an attention-based sequence encoder assumes that every token attends to every other token, thereby capturing long-range dependencies efficiently. While these points are possible explanations, it is subject to ongoing research as to why the sequence-to-sequence model outperforms the Graph-to-SFILES model on large-scale data.

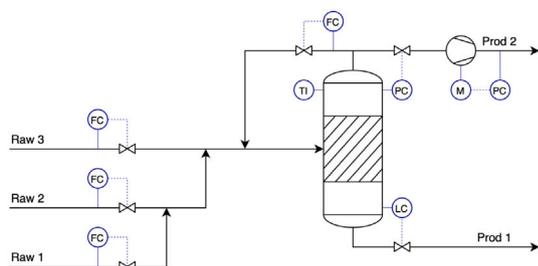
To improve the prediction accuracy of the Graph-to-SFILES model, multiple promising research avenues exist. First, the PFD reconstruction accuracy could be improved as the PFD topology is explicitly known from the model input and should be preserved in the output. We expect that enforcing the PFD topology in the output SFILES will improve the model's performance. For instance, constrained beam search could restrict the predicted unit operations to those present in the input data. Second, multimodal fusion could improve the model performance (Gao et al., 2020). Multimodal fusion aims to integrate data from different modalities. For deep learning, multimodal models have shown superior performance to unimodal models (Xu et al., 2023). Specifically, one could build an architecture that has the same decoder as the Graph-to-SFILES model but a new, multimodal encoder that leverages both SFILES and graph representation. Third, many control structures depend on the field and safety regulations. Including field and safety regulations could be achieved through: (i) domain-specific attributes in node and edge features, (ii) fine-tuning on field-specific datasets, and (iii) enforcing safety constraints via expert systems in a hybrid AI approach.



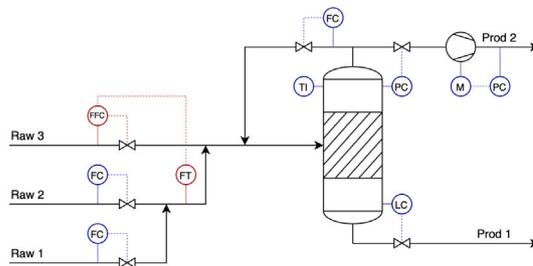
(a) First prediction, incorrect. The model predicts a temperature controller instead of a temperature indicator for the reactor.



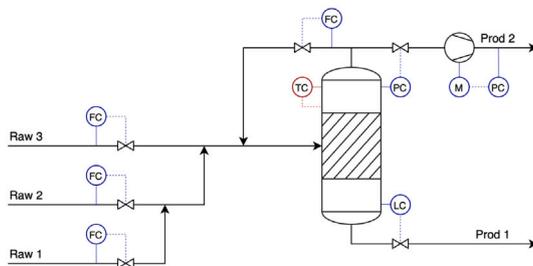
(b) Second prediction, incorrect. Raw 3 should be controlled with a regular flow controller. In addition, the model predicts a temperature controller instead of a temperature indicator for the reactor.



(c) Third prediction, correct.



(d) Fourth prediction, incorrect. Raw 3 should be controlled with a regular flow controller.



(e) Fifth prediction, incorrect. The model predicts a temperature controller instead of a temperature indicator for the reactor.

**Fig. 8.** Control structure prediction for the sample PFD (colored black) where correct predictions are shown in blue while incorrect predictions are shown in red. The five predictions correspond to the five beams with the highest joined probability during beam search decoding, ordered by descending joint probability.

Furthermore, it is important to bear in mind the limitations of the current work. Firstly, the current dataset does not include any equipment, operation, or stream parameters but solely topological information. The proof needs to be made that the Graph-to-SFILES model can learn those parameters and that they improve the model prediction. Secondly, we train our model on a synthetic dataset. In an ideal scenario, we would train our model on curated, industrial process data available at a large scale. Further efforts need to be made to collect data that are close to the final application in industrial practice, e.g., by digitizing industry data (Theisen et al., 2023) or by mining academic data and patents (Schweidtmann, 2024b; Balhorn et al., 2022). Thirdly, our Graph-to-SFILES model is a purely data-driven approach. Data-driven approaches proved to be successful for generative AI tasks if large-scale data are available, for instance for molecular design (Du et al., 2024) or language generation (ChatGPT). However, as described above, a large-scale process dataset is currently missing. Therefore, including physical knowledge in our model could potentially improve its performance. Furthermore, physical knowledge could increase safety, interpretability, and trust in the model. Process modeling and control principles could be integrated into the model as hybrid AI approaches.

In consequence, we believe that overcoming the aforementioned limitations brings the vision of AI-assisted control structure design within reach.

Another important aspect to consider in control structure design is that often multiple valid solutions exist for a given process. Our dataset accounts for this by including multiple control options for the same PFD. However, this leads to a one-to-many mapping in the dataset. Especially if only the topology of the process is considered, selecting a specific control structure can lead to ambiguous results. However, the top-1 CEF accuracy only counts a prediction as correct if the predicted CEF matches the ground-truth CEF exactly. Therefore, there is a theoretical maximum top-1 CEF accuracy that lies well below 100% given the one-to-many mapping. This can be observed as the top-1 CEF accuracy only marginally improves as we increase the dataset from 10,000 to 100,000 samples. Hence, it is important to also calculate the top-k accuracies. We acknowledge that adding further process details such as process dynamics and safety constraints could make the control structure prediction more specific. In general, using a probabilistic model such as the transformer-based Graph-to-SFILES model is advantageous. The Graph-to-SFILES model outputs a probability distribution. This allows to obtain multiple control structure

predictions with their respective probability. Future work could exploit this by analyzing the prediction uncertainty of the model (Ott et al., 2018).

More broadly, a key challenge in evaluating AI-driven control structure design is the lack of standardized metrics. The current evaluation metrics (i.e., CEF accuracy, PFD accuracy, BLEU) do not adequately capture the quality of CEFs. A direct comparison between CEFs is difficult because different control structures are designed for different process objectives, work under different constraints, and have varying levels of centralization. Even if two control structures achieve the same setpoint tracking accuracy, they might differ in robustness, adaptability, explainability, or computational cost. It remains an open research question how the quality of CEFs, not to mention P&IDs, can be quantitatively compared.

In addition, the effectiveness of AI for control structure design compared to expert systems remains an open research question. A meaningful comparison could be conducted on a set of representative processes. For each process, control structures would be generated using (i) the Graph-to-SFILES model and (ii) an expert system, operated by a trained user. The resulting control structures could then be evaluated based on the following criteria: Correctness, computational efficiency, explainability, and ease of use. A potential limitation of this comparison is the dependence on the user's familiarity with the expert system. Since expert systems often require manual rule encoding and extensive training, the quality of results may vary based on the user's expertise. This factor should be considered when interpreting the comparison results.

## 6. Conclusions

This work proposes the Graph-to-SFILES model. The Graph-to-SFILES model automatically extends a process schematic with control structures using a graphical representation of the process. We propose a GNN architecture for the graph encoder and compare it with three architectures from the literature on a synthetic dataset.

The best-performing Graph-to-SFILES model, using a graph encoder with positional encoding, can predict the correct control structure for 46.1% of the synthetic test data. Even in cases where the model did not fully predict the correct schematics, the output tends to be a valid alternative control layout, as demonstrated in an illustrative example. However, it remains an open research question whether the Graph-to-SFILES model can learn from industry-relevant and complex process details, such as stream conditions or composition, provided as node and edge features.

Future work will tackle the remaining limitations of our approach. A first step could be to acquire larger datasets from industrial and academic sources with detailed process information. We expect that including more industrially relevant data in the training can greatly improve the usefulness and acceptance of this model in process engineering. Building on the current work, the Graph-to-SFILES model can also be used for other generative tasks in process design, for instance, the auto-correction of flowsheets (Schulze Balhorn et al., 2024), alternative flowsheet suggestion, or HAZOP report generation.

## CRedit authorship contribution statement

**Lukas Schulze Balhorn:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Kevin Degens:** Software, Methodology, Investigation, Formal analysis. **Artur M. Schweidtmann:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

## 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.

## Appendix

### A.1. Combined model

First, the Combined model calculates a positional encoding  $\lambda_i$  for every node  $i$  from the graph's normalized Laplacian matrix  $L_{\text{norm}} \in \mathbb{R}^{N \times N}$ , where  $N$  is the number of nodes in the graph.

$$L_{\text{norm}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \quad (1)$$

Here,  $D \in \mathbb{R}^{N \times N}$  is the degree matrix,  $L = D - A$  the Laplacian matrix, and  $A \in \mathbb{R}^{N \times N}$  is the adjacency matrix. The  $k$  smallest non-trivial eigenvectors  $\lambda_i \in \mathbb{R}^k$  of  $L_{\text{norm}}$  are added to the original node features  $h_i^0 \in \mathbb{R}^{d_{\text{node}}}$  for each node  $i$ . Both vectors are projected to the encoder embedding dimension  $d_{\text{enc}}$ , with  $W_{\text{node}} \in \mathbb{R}^{d_{\text{enc}} \times d_{\text{node}}}$  and  $W_{\lambda} \in \mathbb{R}^{d_{\text{enc}} \times k}$ , and a bias  $b \in \mathbb{R}^{d_{\text{enc}}}$  is added.

$$h_i = W_{\text{node}} h_i^0 + W_{\lambda} \lambda_i + b_{\text{node}} \quad (2)$$

In addition, the edge feature is projected to the embedding dimension:

$$e_{i,j} = W_{\text{edge}} e_{i,j}^0 + b_{\text{edge}}, \quad (3)$$

with  $W_{\text{edge}} \in \mathbb{R}^{d_{\text{enc}} \times d_{\text{edge}}}$  and  $e_{i,j}^0 \in \mathbb{R}^{d_{\text{edge}}}$  ( $j$  is a neighboring node of node  $i$ ). Note that the Laplacian embedding is only added in the input layer. The attention-based message passing block  $\alpha_{i,j}^m \in \mathbb{R}$  is calculated for  $m$  heads using

$$\alpha_{i,j}^m = \text{softmax} \left( a^m \text{LeakyReLU} \left( \left[ W_{q_{\text{enc}}}^m h_i \parallel W_{k_{\text{enc}}}^m h_j \parallel W_{v_{\text{enc}}}^m e_{i,j} \right] + b^m \right) \right). \quad (4)$$

Here,  $h_i, h_j \in \mathbb{R}^{d_{\text{enc}}}$  and  $e \in \mathbb{R}^{d_{\text{enc}}}$  are the node and edge features of the previous layer, multiplied by weight matrices  $W_{q_{\text{enc}}}^m, W_{k_{\text{enc}}}^m, W_{v_{\text{enc}}}^m \in \mathbb{R}^{d'_{\text{enc}} \times d_{\text{enc}}}$ , concatenated (denoted as  $\parallel$ ), and then summed with a bias  $b^m \in \mathbb{R}^{d'_{\text{enc}}}$ . After applying the nonlinear function LeakyReLU, the output is multiplied by another weight matrix  $a^m \in \mathbb{R}^{3d'_{\text{enc}}}$  and normalized using the softmax function. Here,  $d_{\text{enc}}$  is the internal encoder dimension and  $d'_{\text{enc}} = d_{\text{enc}}/m$  the head-specific dimension. We make use of the PyTorch Geometric implementation of the GATv2 attention mechanism.<sup>2</sup> The node update equation includes (i) message passing from the node's neighbors and (ii) attention head concatenation:

$$\hat{h}_i = \parallel_{m=1}^M \left( \sum_{j \in \mathcal{N}_i} \text{softmax}(\alpha_{i,j}^m) W_{v_{\text{enc}}}^m h_j \right). \quad (5)$$

Finally, the node update is passed through two residual connections with layer normalization and an FFN. Here, the weight matrices have the dimension  $W_1, W_2 \in \mathbb{R}^{d_{\text{enc}} \times d_{\text{enc}}}$ .

$$\hat{\hat{h}}_i = \text{Norm} \left( h_i + \hat{h}_i \right) \quad (6)$$

$$\hat{\hat{h}}_i = W_2 \text{ReLU} \left( W_1 \hat{\hat{h}}_i + b \right), \quad (7)$$

$$h'_i = \text{Norm} \left( \hat{\hat{h}}_i + \hat{\hat{h}}_i \right) \quad (8)$$

### A.2. Hyperparameters

See Table 5.

## Data availability

The data that has been used is confidential.

<sup>2</sup> [https://pytorch-geometric.readthedocs.io/en/latest/generated/torch\\_geometric.nn.conv.GATv2Conv.html](https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GATv2Conv.html).

Table 5

Graph-to-SFILES hyperparameters. The complete grid search of hyperparameters can be found here: <https://doi.org/10.5281/zenodo.14840560>.

Hyperparameter	Values	Grid search results per model architecture			
		GATv2	GraphConv	Graph Transformer	Combined model
Batch size	{64, 128}	128	64	64	64
Learning rate	{5e-4, 1e-3, 1.5e-3, 2e-3}	2e-3	1e-3	1e-3	1e-3
Encoder layers	{2, 4, 6}	4	6	6	6
Decoder layers	{2, 4, 6}	4	6	6	6
Heads $m$	8	–	–	–	–
Embedding $d_{enc}, d_{dec}$	128	–	–	–	–
Dropout	0.1	–	–	–	–
Decoder FFN dimension	2048	–	–	–	–

## References

- Azevedo, A.R., Nabil, T., Loubière, V., Valentinc, B., Privata, R., Neveux, T., Commenge, J.-M., 2024. Representation learning for flowsheets: generating structures for process synthesis. In: Manenti, F., Reklaitis, G.V. (Eds.), Book of Abstract of the 34th European Symposium on Computer Aided Process Engineering.
- Balhorn, L.S., Gao, Q., Goldstein, D., Schweidtmann, A.M., 2022. Flowsheet recognition using deep convolutional neural networks. In: Computer Aided Chemical Engineering. Vol. 49, Elsevier, pp. 1567–1572.
- Beck, D., Haffari, G., Cohn, T., 2018. Graph-to-sequence learning using gated graph neural networks. arXiv preprint arXiv:1806.09835.
- Bramsiepe, C., Krasberg, N., Fleischer, C., Hohmann, L., Kockmann, N., Schembecker, G., 2014. Information technologies for innovative process and plant design. Chem. Ing. Tech. 86 (7), 966–981.
- Brody, S., Alon, U., Yahav, E., 2021. How attentive are graph attention networks?. arXiv preprint arXiv:2105.14491.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. Adv. Neural Inf. Process. Syst. 33, 1877–1901.
- d'Anterrosches, L., 2005. Process Flowsheet Generation & Design through a Group Contribution Approach. [CAPEC], Department of Chemical Engineering, Technical University of Denmark.
- De Cao, N., Kipf, T., 2018. MolGAN: An implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973.
- Dollar, O., Joshi, N., Pfaendner, J., Beck, D.A., 2023. Efficient 3d molecular design with an e(3) invariant transformer vae. J. Phys. Chem. A 127 (37), 7844–7852.
- Du, Y., Jamasb, A.R., Guo, J., Fu, T., Harris, C., Wang, Y., Duan, C., Liò, P., Schwaller, P., Blundell, T.L., 2024. Machine learning-aided generative molecular design. Nat. Mach. Intell. 1–16.
- Dwivedi, V.P., Bresson, X., 2020. A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699.
- Dzhusupova, R., Banotra, R., Bosch, J., Olsson, H.H., 2022. Pattern recognition method for detecting engineering errors on technical drawings. In: 2022 IEEE World AI IoT Congress (AIoT). IEEE, IEEE, pp. 642–648.
- Gao, J., Li, P., Chen, Z., Zhang, J., 2020. A survey on deep learning for multimodal data fusion. Neural Comput. 32 (5), 829–864.
- Hirtreiter, E., Schulze Balhorn, L., Schweidtmann, A.M., 2023. Toward automatic generation of control structures for process flow diagrams with large language models. AIChE J. n/a (n/a), e18259.
- Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.
- Khemani, B., Patil, S., Kotecha, K., Tanwar, S., 2024. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. J. Big Data 11 (1), 18.
- Kipf, T.N., Welling, M., 2016. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.
- Li, J., 2024. Learning hybrid extraction and distillation using phenomena-based string representation. In: Foundations of Computer Aided Process Design. FOCAPD 2024.
- Luyben, M.L., Tyreus, B.D., Luyben, W.L., 1997. Plantwide control design procedure. AIChE J. 43 (12), 3161–3174.
- Mann, V., Sales-Cruz, M., Gani, R., Venkatasubramanian, V., 2024. eSFILES: Intelligent process flowsheet synthesis using process knowledge, symbolic AI, and machine learning. Comput. Chem. Eng. 181, 108505.
- Mao, K., Xiao, X., Xu, T., Rong, Y., Huang, J., Zhao, P., 2021. Molecular graph enhanced transformer for retrosynthesis prediction. Neurocomputing 457, 193–202.
- Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananidou, S., Rong, Y., 2022. Transformer for graphs: An overview from architecture perspective. arXiv preprint arXiv:2202.08455.
- Morari, M., Arkun, Y., Stephanopoulos, G., 1980. Studies in the synthesis of control structures for chemical processes: Part I: Formulation of the problem. Process decomposition and the classification of the control tasks. Analysis of the optimizing control structures. AIChE J. 26 (2), 220–232.
- Morgan, H.L., 1965. The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service. J. Chem. Doc. 5 (2), 107–113.
- Nabil, T., Le Moulec, Y., Le Coz, A., 2019. Machine learning based design of a supercritical CO2 concentrating solar power plant. In: 3rd European Supercritical CO2 Conference September 19–20, 2019, Paris, France.
- Nabil, T., Noaman, M., Morosuk, T., 2023. Data-driven structural synthesis of supercritical CO2 power cycles. Front. Chem. Eng. 5, 1144115.
- Ng, C., Stephanopoulos, G., 1998. Synthesis of control structures for chemical plants. 31, (20), Elsevier, pp. 17–29.
- Oeing, J., Brandt, K., Wiedau, M., Tolksdorf, G., Welscher, W., Kockmann, N., 2023. Graph learning in machine-readable plant topology data. Chem. Ing. Tech. 95 (7), 1049–1060.
- Oeing, J., Welscher, W., Krink, N., Jansen, L., Henke, F., Kockmann, N., 2022. Using artificial intelligence to support the drawing of piping and instrumentation diagrams using DEXPI standard. Digit. Chem. Eng. 4, 100038.
- Ott, M., Auli, M., Grangier, D., Ranzato, M., 2018. Analyzing uncertainty in neural machine translation. In: International Conference on Machine Learning. PMLR, pp. 3956–3965.
- Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. pp. 311–318.
- Popel, M., Tomkova, M., Tomek, J., Kaiser, L., Uszkoreit, J., Bojar, O., Žabokrtský, Z., 2020. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. Nat. Commun. 11 (1).
- Rusch, T.K., Bronstein, M.M., Mishra, S., 2023. A survey on oversmoothing in graph neural networks. arXiv preprint arXiv:2303.10993.
- Schulze Balhorn, L., Caballero, M., Schweidtmann, A.M., 2024. Toward autocorrection of chemical process flowsheets using large language models. In: Computer Aided Chemical Engineering. Vol. 53, Elsevier, pp. 3109–3114.
- Schwaller, P., Laino, T., Bolgar, P., Hunter, C.A., Bekas, C., Lee, A.A., 2019. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. ACS Cent. Sci. 5 (9), 1572–1583.
- Schweidtmann, A.M., 2024a. Generative artificial intelligence in chemical engineering. Nat. Chem. Eng. 1, 193–193.
- Schweidtmann, A.M., 2024b. Mining chemical process information from literature for generative process design: A perspective. In: Foundations of Computer Aided Process Design. FOCAPD 2024.
- Seborg, D.E., Edgar, T.F., Mellichamp, D.A., Doyle III, F.J., 2016. Process Dynamics and Control. John Wiley & Sons.
- Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y., 2020. Masked label prediction: Unified message passing model for semi-supervised classification. arXiv preprint arXiv:2009.03509.
- Song, J.J., Park, S.W., 1990. Intellite 3: A knowledge based expert system for control structure synthesis. Korean J. Chem. Eng. 7, 198–209.
- Song, L., Zhang, Y., Wang, Z., Gildea, D., 2018. A graph-to-sequence model for AMR-to-text generation. arXiv preprint arXiv:1805.02473.
- Stops, L., Leenhouts, R., Gao, Q., Schweidtmann, A.M., 2023. Flowsheet generation through hierarchical reinforcement learning and graph neural networks. AIChE J. 69 (1), e17938.
- Theisen, M.F., Flores, K.N., Schulze Balhorn, L., Schweidtmann, A.M., 2023. Digitization of chemical process flow diagrams using deep convolutional neural networks. Digit. Chem. Eng. 6, 100072.
- Toghræi, M., 2019. Piping and Instrumentation Diagram Development. John Wiley & Sons.
- Topping, J., Di Giovanni, F., Chamberlain, B.P., Dong, X., Bronstein, M.M., 2021. Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522.
- Tu, Z., Coley, C.W., 2023. Permutation invariant graph-to-sequence model for template-free retrosynthesis and reaction prediction.
- Van De Wal, M., De Jager, B., 2001. A review of methods for input/output selection. Automatica 37 (4), 487–510.
- Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.
- Venkatasubramanian, V., 2019. The promise of artificial intelligence in chemical engineering: Is it here, finally? AIChE J. 65 (1).

- Vogel, G., Balhorn, L.S., Schweidtmann, A.M., 2023a. Learning from flowsheets: A generative transformer model for autocompletion of flowsheets. *Comput. Chem. Eng.* 171, 108162.
- Vogel, G., Hirtreiter, E., Balhorn, L.S., Schweidtmann, A.M., 2023b. SFILES 2.0: an extended text-based flowsheet representation. *Optim. Eng.*
- Vogel, G., Sortino, P., Weber, J.M., 2023c. Graph-to-string variational autoencoder for synthetic polymer design. In: *AI for Accelerated Materials Design - NeurIPS 2023 Workshop*.
- Wang, J., Zheng, S., Chen, J., Yang, Y., 2021. Meta learning for low-resource molecular optimization. 61, (4), ACS Publications, pp. 1627–1636.
- Weininger, D., Weininger, A., Weininger, J.L., 1989. SMILES. 2. Algorithm for generation of unique SMILES notation. *J. Chem. Inf. Comput. Sci.* 29 (2), 97–101.
- Williamson, C., 1989. *Computer Aided Process Control Systems Synthesis Using Rule-Based Programming*. University of Canterbury. Chemical Engineering.
- Xu, K., Wu, L., Wang, Z., Feng, Y., Witbrock, M., Sheinin, V., 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.
- Xu, P., Zhu, X., Clifton, D.A., 2023. Multimodal learning with transformers: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 45 (10), 12113–12132.
- You, J., Ying, R., Ren, X., Hamilton, W., Leskovec, J., 2018. GraphRNN: Generating realistic graphs with deep auto-regressive models. In: Dy, J., Krause, A. (Eds.), *Proceedings of the 35th International Conference on Machine Learning*. In: *Proceedings of Machine Learning Research*, vol. 80, PMLR, pp. 5708–5717.