

# Foundation Model for Determining Suitable Process Parameters in Twin-Screw Extrusion

Julia Burr\* and Alex Sarishvili

DOI: 10.1002/cite.70017

 This is an open access article under the terms of the [Creative Commons Attribution](#) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

Extrusion is a complex process, and identifying suitable process parameters to achieve specific product or process properties is often a time-consuming manual task, which hinders automation and requires specialized staff. Machine learning models present a promising solution, but they typically require large amounts of high-variational data for training to achieve satisfactory precision. To address this challenge, we propose the development of a foundation model for co-rotating twin-screw extruders, leveraging extensive simulated data for training. By employing a transformer architecture combined with a masking technique, this model will be capable of suggesting process parameters based on desired outcomes. We will also demonstrate how this model can be effectively fine-tuned for a specific extrusion plant using minimal data.

**Keywords:** Foundation model, Machine learning, Process parameters, Transformer, Twin-screw extrusion

*Received:* May 06, 2025; *revised:* July 03, 2025; *accepted:* July 31, 2025

## 1 Introduction

Extrusion is a complex chemical process utilized for producing various plastic products, food items, and healthcare materials. In this process, polymer granulate is placed into a barrel equipped with a rotating screw. The shear forces generated by the screw melt the polymer, while the screw also transports the molten material to the end of the barrel, where it is forced through a die to achieve its final shape. Heating elements in the barrel maintain temperature to prevent the plastic from cooling and adhering to the walls. Since its introduction in the 1930s, various types of extruders have been developed, including twin-screw extruders with co-rotating or counter-rotating screws, which achieve a higher throughput than single-screw extruders [1]. Depending on different parameters such as intermeshing or the target speed range, co-rotating and counter-rotating screws have different applications [1–3]. We focus on co-rotating screws.

Determining the appropriate process parameters for extrusion is a complex task that often relies on operator experience and trial-and-error, which necessitates specialized staff and limits automation and systematic optimization of product quality and energy consumption. Although physical simulations can provide useful insights, they may not be suitable for identifying optimal process parameters. Simulations can be used iteratively, where experts start with initial guesses for parameters, run simulations, interpret the results, and adjust the parameters accordingly. However, simulation tools can present challenges: complex partial differential equation (PDE)-based simulations require considerable computation time, hindering rapid responses needed for demand side management, while simplified

ID-simulation tools are fast but may lack precision, leading to suboptimal parameters. An overview of different simulation tools was given by Hyvärinen et al. [1].

Data-based models could address both issues, as they typically offer fast inference (application after training) and can utilize measurements from specific extrusion plants for more accurate results. Additionally, these models can directly propose process parameters to achieve desired product properties, eliminating the need for time-consuming iterations. Previous studies have employed data-based models for extruders [4–9]. However, many approaches only model the forward process (from process parameters to process/product properties). Typically, simple models are utilized due to the limited number of sensors on most extruders which results in scarce data [6, 10, 11]. Such models could lack the complexity required to model the extrusion process with its complicated phenomena. To enhance data availability, simulated data can supplement existing databases, generating a broad range of high-variational data for training more complex models. This was demonstrated in previous work by Burr et al. [12]. There, an autoencoder-based model was proposed, which directly considers the goal of finding process parameters, so no time-consuming iterative process is needed. However,

<sup>1,2</sup>Julia Burr  <https://orcid.org/0000-0003-4277-0620> ([julia.burr@itwm.fraunhofer.de](mailto:julia.burr@itwm.fraunhofer.de)), <sup>1</sup>Dr. Alex Sarishvili

<sup>1</sup>Fraunhofer ITWM, Fraunhofer-Platz 1, Kaiserslautern 67663, Germany.

<sup>2</sup>Department of Mathematics, University of Kaiserslautern-Landau, Gottlieb-Daimler-Straße, Kaiserslautern 67663, Germany.

that work considered only one screw configuration and rheology.

Our aim is to create a foundation model (FM) for co-rotating twin-screw extrusion processes. This model will be trained on simulated data encompassing various screw configurations and later fine-tuned with measurements from specific machines. As the overall dynamics will have been established through simulated data, only a few data points will be needed for accurate results, compared to training a model from scratch.

Machine learning (ML) was previously applied to other plastic manufacturing processes. Note that ML results can usually not be transferred between datasets or, in this case, manufacturing processes. Selvaray et al. [13] summarized ML approaches for injection molding. According to them, one of the drawbacks of the studies is that they often only consider one specific setup with one mold and one material. Instead, a FM would be useful, which considers different molds and various materials. Most studies model the forward process, from process parameters to quality of the product, not the reverse, which is our goal. One study by Manjunath et al. [14] did consider the reverse process. However, the database they built their model on was created using simple regression equations, not real or simulated data. It is no surprise that neural networks can approximate regression equations since they are universal approximators. Only few previous studies have considered neural networks to model polymer extrusion using single-screw extruders [8, 9, 15]. The work of Yasith et al. [8] is closest to our setting. They predict the melt pressure using process parameters and a combination of an autoencoder and an additional feed-forward neural network. However, no reverse method for optimization is developed.

To fine-tune the FM to real-world data, we use transfer learning. In the transfer learning setting, sufficient data from a source domain/distribution is available. The goal, however, is to model a target domain/distribution where only few samples are available. One strategy of transfer learning is to train a model on the larger dataset, then use fine-tuning utilizing only the target dataset to fit the model to the target distribution. Fine-tuning has been previously applied to polymer processes. Huang et al. [16] successfully applied transfer learning to an injection molding process where the goal was to predict product properties. The initial training data was simulated, while the fine-tuning dataset consisted of experimental data. Lockner et al. [17] also consider injection molding, but transfer knowledge between polymer classes. Zhang et al. [18] consider transfer learning for online process monitoring in additive manufacturing. In the context of extruders, Ren et al. [19] identified the extruder in additive 3D-printers as a main source of defects due to the variability in printing width. They build a physical model which predicts this variation based on printing speed and acceleration, then transfer knowledge in the form of parameters' covariance structure to new 3D printers with limited historical data. Liang et al. [20] consider transfer learning in

aluminum extrusion for forecasting the energy consumption and detecting anomalies. According to our research, transfer learning has not been applied in the setting of polymer extrusion where the goal is to transfer knowledge from a large ML-based FM to one specific machine.

Choosing the right ML model is essential; it must be complex enough to capture the intricate relationships between process parameters, desired properties, and screw configurations, while also being capable of directly proposing process parameters without manual or automatic iterations. We propose utilizing a transformer model.

Transformers have successfully been applied in language modeling [21, 22]. Using an attention mechanism, they extract connections between words in the context of the sentence. They have also shown promising results in modeling image data [23] and time series data [24, 25]. Our goal is to leverage the transformer's success in language modeling for machine data to create a FM which can suggest process parameters based on the desired outcome in terms of temperature and pressure inside the extruder. We also aim to demonstrate efficient fine-tuning of the FM.

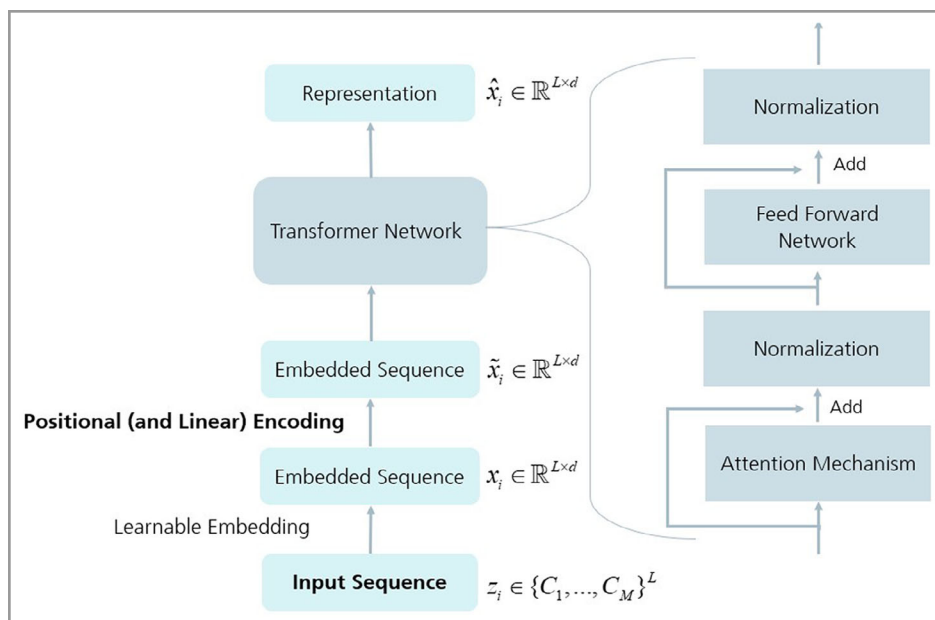
To implement this, we will first convert various extruder sensor signals into discrete classes, with each class representing a specific magnitude range. These classes will serve a similar function to words, allowing us to combine data from different sensors sequentially to form a "sentence." This methodology enables the integration of both time-dependent and time-independent data into a unified structure, facilitating the use of embeddings similar to language modeling.

Next, we will apply the transformer to extract complex relationships between process settings, screw geometry, temperature and pressure sequences in the extruder. For training, we use a scheme similar to BERT [26], where a subset of the words in each sentence is masked, that is, replaced by the word "mask." The transformer model is then trained to predict the masked "words." This training scheme, applied to the extrusion setting, naturally yields a model which can propose settings leading to desired temperature and pressure distributions. During application, the sections of input corresponding to process parameters will be masked and combined with screw geometry and desired temperature and pressure sequences to form sentences fed to the transformer. The transformer will then suggest the most likely process parameters to replace the masked positions.

The article is structured as follows: Sect. 2 discusses the original transformer and our application and modifications. Sect. 3 contains results of the training of the FM. Sect. 4 considers fine-tuning the FM to data from one specific plant. Sect. 5 contains the discussion.

## 2 Transformer Encoder

Transformers were first introduced by Vaswani et al. [22] for natural language processing. We present the basic structure



**Figure 1.** Encoder network. Bold texts indicate aspects we adapted significantly for our specific use case.

of the original transformer encoder and describe in detail how we adapted and employed the transformer in our use case.

## 2.1 Original Encoder Scheme

The original transformer was designed with an encoder and decoder part used for sequence-to-sequence modeling such as translation. If the goal is classification or regression instead, only the encoder part can be used. For detailed explanations, consider the original introduction by Vaswani et al. [22]. We consider discrete input sequences of constant length.

Let  $\{z_i\}_{i=1}^N$  be the set of training data, where  $z_i \in \{C_1, \dots, C_M\}^L$  is a sequence of discrete values of length  $L$ . We want to encode this data using a transformer. We only use the encoder part. Refer to Fig. 1 for a graphical representation of the following steps.

1. Embedding  $z_i \rightarrow x_i \in \mathbb{R}^{L \times d}$ : The embedding is a trainable look-up table. A multidimensional vector of embedding dimension  $d$  is assigned to each discrete value. On the one hand, this enables the following transformer steps since the sequence has a numerical representation now. On the other hand, the embedding itself is a vital part of the transformer. In the embedding space, interdependencies between the discrete values are represented. In language models, this could mean that the representation of the word “blue” is closer to “sky” than to “tree.” The output of this layer is a sequence of vectors. The embedding is learned during training.
2. Positional Encoding  $x_i \rightarrow \tilde{x}_i = x_i + p \in \mathbb{R}^{L \times d}$ : The transformer architecture itself does not consider the position of the values in the sequence. It does not matter

whether 2 values are separated by one or 100 other values. To incorporate the positional information into the model, a sine wave of different frequencies is added to each dimension of the input vectors. This encoding can be viewed as a continuous version of the binary counting scheme.

3. Encoder Blocks  $\tilde{x}_i \rightarrow \hat{x}_i \in \mathbb{R}^{L \times d}$ : Multiple encoder blocks are stacked on top of one another to create the deep network that is the transformer. The output of one encoder block is fed to the next one as input. The initial input is the vectors created by the positional encoding. “Input” in the following refers to the input to one considered encoder block. Let us take a look inside the encoder block. First, the input is fed to the attention mechanism. This creates a representation of the data where each value in the input sequence is considered in the context of all other vectors in the sequence. The output of the attention mechanism is added to the input of the encoder block and a normalization follows. Then, a feed-forward network (FFN) is applied and afterward, the input of the FFN is added to its output. This serves to transition from the contextual information to the desired output, as argued by Yun et al. [27]. It provides more degrees of freedom to the network. The resulting values are normalized, which yields the output of the encoder block.

How do you train such a network? Depending on the task at hand, there are different ways.

We use an unsupervised way of training proposed by Devlin et al. [26]. There, a certain fraction of input values is replaced by “mask,” which is an additional discrete value not contained in the initial set  $\{C_1, \dots, C_M\}$ . The representations of the masked values are fed to a classifier whose goal is to predict/recover the true value.

**Table 1.** Intervals used to assign classes to values computed in the simulation.

	Interval/Class 1	Interval/Class 2	Interval/Class 3	Interval/Class 4
Rotation speed [rpm]	[200, 250)	[250, 300)	[300, 350)	[350, 400)
Throughput [kg h <sup>-1</sup> ]	[10, 15)	[15, 20)	[20, 25)	[25, 30)
Temperature barrels [°C]	[180, 195)	[195, 210)	[210, 225)	[225, 240)
Temperature in extruder [°C]	[131, 174.2)	[174.2, 217.4)	[217.4, 260.6)	[260.6, 303.8)
Pressure in extruder [bar]	[0, 30)	[30, 60)	[60, 90)	[90, 120)

## 2.2 Adaptation of the Original Encoder Scheme

We describe in detail how we adapted the presented encoder scheme to our use case. This entails the available data, the composition of the input sequence, the positional encoding, the transformer architecture, the training scheme, and the evaluation scheme.

### 2.2.1 Available Data

In production, usually one setting is chosen and run until some problem is detected. This means that real-world data collected from extrusion processes usually lack variation, making it hard to learn from. Additionally, to create a model that can handle all kinds of screw geometries, a massive measurement campaign including hundreds of extruders is needed. Instead, we utilize simulated data from the Ludovic<sup>®</sup> tool [28]. This allows the FM to grasp the process dynamics. When applied to a specific machine, the model is fine-tuned with a small amount of measured data, correcting any simulation inaccuracies.

Ludovic<sup>®</sup> is a commercial 1D steady-state simulation software. It solves the Stokes equations, thereby computing thermo-mechanical behavior in twin-screw extrusion. Important assumptions are that, locally, the flow is Newtonian and isothermal. The flow material is considered incompressible and without gravity. There are additional assumptions for different zones, for example, in flow zones, the flow is assumed to be constant along the axial coordinate and the flow along the radial axis is neglected. Burr et al. [12] have shown prior that the simulation successfully captures trends but does not yield exact results due to simplifications.

The material we used in the simulations was high-density polyethylene, which is used, for example, for producing pipes or plastic bottles. In a previous project [4], the material's properties, such as viscosity, thermal conductivity, heat capacity, or density, were measured to represent the material accurately in the simulation software. The Carreau–Yasuda model [4] was used to model viscosity.

We conducted 500 000 simulations. We used a Latin Hypercube Design to determine the settings (rotation speed, throughput, and temperature of barrels) for each simulation, which ensures coverage of the parameter space. The parameter ranges are depicted in Tab. 1. The screw geometries were

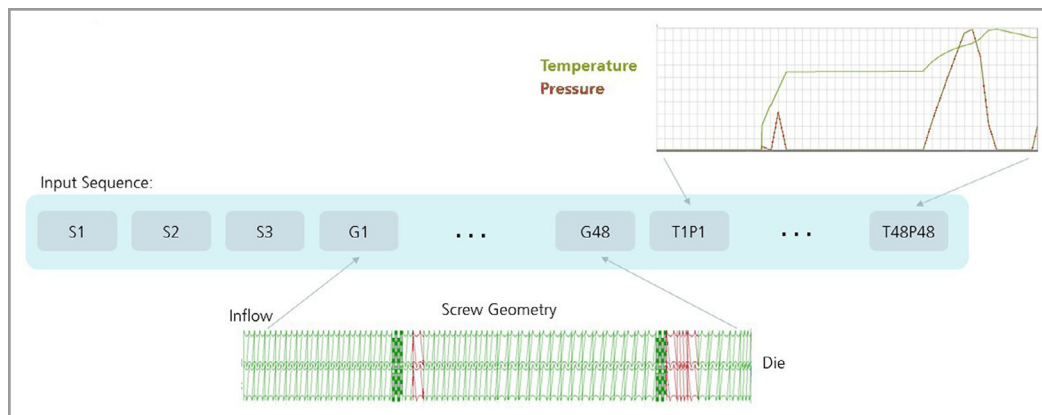
chosen randomly under the following constraints: The screw consists of 48 sections, which are initially set to be forward transport elements. Then, at two random places in the first and last half of the screw, a kneading zone is created. Each kneading zone consists of four elements which are randomly drawn from the set (transport forward, transport backward, kneading forward, kneading backward). We made sure to include each possible kneading zone configuration at least once in the dataset. An abstract representation of one such geometry is depicted in the lower part of Fig. 2. Parameters like the number of disks of kneading elements or the pitch of the transport elements are varied. The first and last few elements of the screw are kept the same. This set up should include many screws used in the industry.

The simulations yielded 500 000 data points consisting of settings, screw geometry, and simulation results. The simulation results include temperature and pressure values at 48 consecutive sections in the extruder.

### 2.2.2 Input Sequence

We transform the simulation's numerical outcomes into classes and consider a classification instead of a regression task. This is beneficial when the decision context is categorical, when robustness and simplicity are desired, or when interpretability is more important than precision. The measurement of melt temperature and pressure in an extruder is affected significantly by measurement uncertainties which result in noisy data. The noise affects the training in a negative way, causing prediction's confidence intervals to be wide. In our previous work [12], we compared the Ludovic<sup>®</sup> simulation results with its real counterpart extruder in a lab. The discrepancies between the simulation and real outcome were significant due to sensor measurement uncertainties, variations in production process cooling system, and variations in material and environment conditions. Regression models are sensitive to small variations in the target variable.

To prevent these effects from negatively influencing our results, we divide the ranges of process parameters, temperature, and pressure in the extruder into four equidistant intervals and assign each measurement to its corresponding interval. The interval bounds are provided in Tab. 1. If for one simulation the rotation speed is 220 rpm, we assign class 1.



**Figure 2.** Assembly of the input sequence to the transformer.  $S_x$  means “setting  $x$ ” (process parameter  $x$ ),  $G_x$  means geometry module in section  $x$ ,  $T_xP_x$  means the temperature and pressure class in section  $x$ .

This discretization yields an additional advantage: We can utilize embeddings, which give the transformer additional degrees of freedom to model the process.

Each input sequence starts with the class for rotation speed, followed by the throughput class and the class assigned to the temperature of the heating elements. We define 48 equidistant sections in the extruder, where each section may contain a forward or backward rotating screw and either kneading or transportation elements, forming four possible modules. This yields a sequence of 48 geometry classes, which is attached to the process parameter sequence.

We then create classes for temperature and pressure values by combining corresponding intervals (e.g., T1P3 indicates temperature in interval 1 and pressure in interval 3). This combination allows us to specify desired temperature and pressure sequences while ensuring that each class is physically feasible, which is another advantage of considering discretized values. Feasible classes are determined by their frequency in the training dataset; for instance, high pressure with low temperature only occurs in specialized cases, which is why we do not consider it here. The sequence is completed with the combined temperature–pressure classes for each extruder section.

In this way, both position-dependent variables (e.g., granulate temperature) and position-independent variables (e.g., screw speed) are combined into one sequence.

### 2.2.3 Positional Encoding

To design the positional encoding, we use a learnable linear encoding for time/position-independent settings, allowing the model to determine a gain and offset for each of the three settings during training. The position-dependent variables (screw geometry, temperature and pressure) can be handled similarly to time-dependent variables in the standard transformer. Therefore, we apply standard positional encoding. Notably, the value added to section  $i$  of the screw geome-

try embedding is the same as that for the temperature and pressure class at section  $i$ , indicating to the model the relationship between these values at different positions in the sequence.

### 2.2.4 Transformer Architecture

We use three encoder blocks with two attention heads each. The embedding dimension  $d$  which is a hyperparameter is chosen to be 10 (refer to Sect. 2.1). The number of hidden layers in the FFN is 5. We have not conducted an extensive hyperparameter study. Instead, we did a manual grid search over the number of heads and layers and the embedding dimension to find a model which provides good results while being as simple as possible. The presented hyperparameters may not be optimal but are sufficient for showing the applicability of the presented approach in the context of extruders.

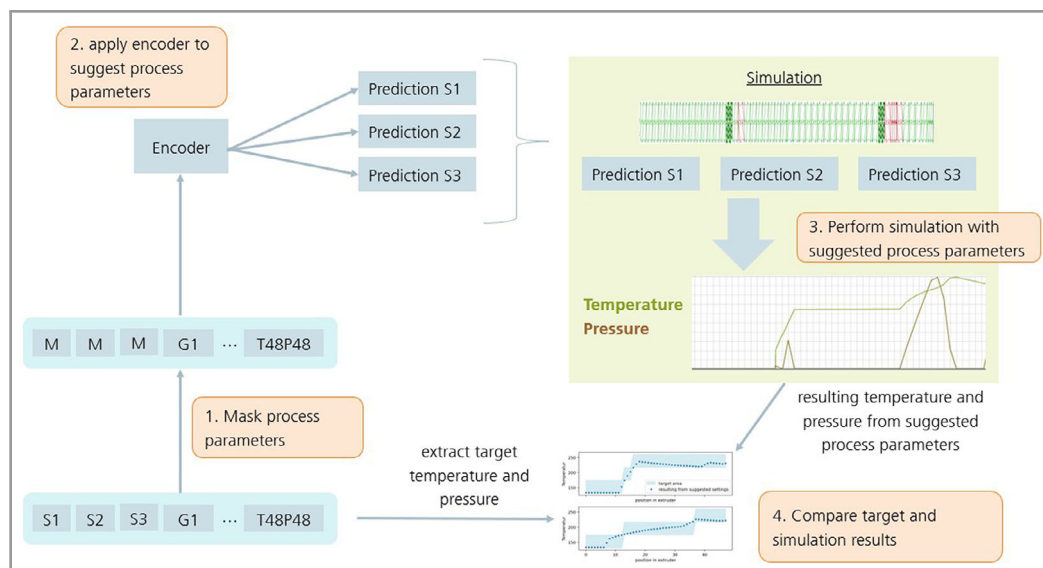
### 2.2.5 Training Scheme

We use a simplification of the approach designed by Devlin et al. [26]. We mask a certain percent of the training data and train the encoder to classify the masked values by attaching a classification head on top of the encoder output. We randomly mask 10 % of the input sequence, which makes  $(3 + 48 \cdot 2) \cdot 0.1 \approx 10$  elements.

We use the AdamW optimizer and a batch size of 64. Due to a significant class imbalance, we use weighted cross-entropy as loss function. We split the dataset into training (80 %) and test dataset (20 %). All presented results are evaluated on the test dataset.

### 2.2.6 Evaluation Scheme

Our goal is to create a model that suggests settings to achieve a desired temperature and pressure sequence in the extruder. When applying the trained model, we mask the three first



**Figure 3.** Evaluation procedure for the suggested process parameters. First, from a test sequence, the three process parameter values are replaced with “mask” (M). Then, the encoder is applied to predict the most likely process parameters to achieve the desired temperature and pressure sequences. We use these suggestions in the simulation. The resulting temperature and pressure sequences are extracted and compared with the target ones.

elements of the input sequence which correspond to the unknown process parameters. We provide the screw geometry, desired temperature, and pressure sequences. The model then recovers these masked settings, proposing those most likely to yield the desired outcome (see Fig. 3).

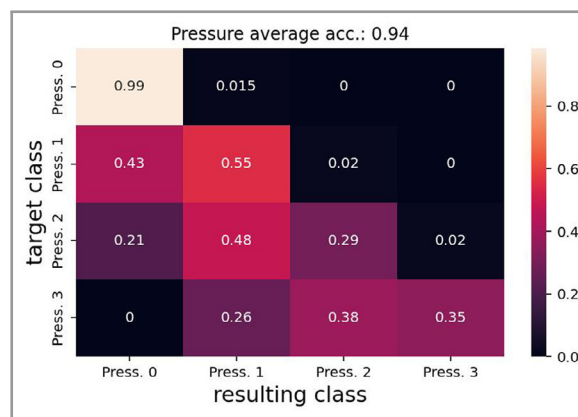
To evaluate the model, we typically mask the settings in a test dataset and compare the proposed settings to the originals for accuracy. However, since different settings can produce the same temperature and pressure sequence, this method may underestimate the model’s performance. Instead, we use the proposed settings to run simulations in Ludovic® and compare the simulation outcomes to the desired results.

### 3 Results: Foundation Model

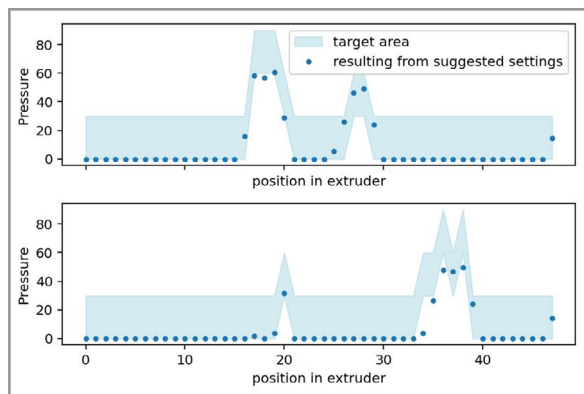
We assess the quality of the suggested settings using Ludovic® simulations to determine if they achieve the desired results. For pressure values, Fig. 4 presents a confusion matrix comparing target and actual pressure classes. For 100 conducted simulations, all 48 elements in the pressure and temperature sequence are considered, and the pressure classes extracted and compared to the target pressure classes. While there is some diagonal dominance, the suggested settings occasionally yield lower pressures. It is important to note that there is a severe class imbalance for pressure. Class 1 occurs in 91.8 % of cases since pressure is always 0 if no kneading element is near the considered position. This may be the cause of the model underestimat-

ing the pressure in some cases. For cases where target and actual pressures differ, the mean deviation is 15 bar, and the median is 11 bar. The total range of pressure values is 129 bar, resulting in a median deviation of approximately 9 %.

Fig. 5 compares two target pressure sequences with those from the suggested parameters. Pressure fluctuations occur along the extruder, spiking with kneading elements or backward transport and dropping to zero otherwise. Although



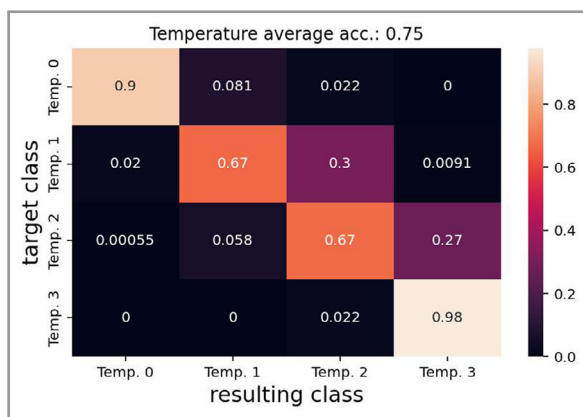
**Figure 4.** Target pressure classes vs. pressure classes resulting from process parameters proposed by the transformer. The numbers should be read in the following way: In 99 % of cases where the target class was 1, the class resulting from the simulation using suggested process parameters was also 1. In 1.5 % of cases with target class 1, the resulting class was 2. The average accuracy is 94 % as stated in the title.



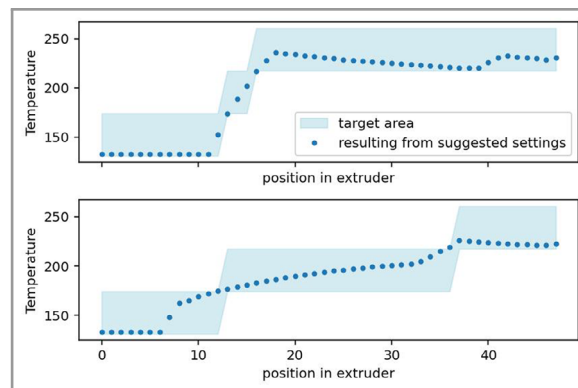
**Figure 5.** Target pressure sequence (light blue) and pressure sequence resulting from suggested process parameters (dark blue). The light blue area's bounds are determined by the bounds of the interval that the target class represents at each of the 48 sections in the extruder. The bounds are shown in Tab. 1.

the actual pressure generally follows the target evolution, this behavior is influenced significantly by screw geometry and thus cannot totally be attributed to the suggested process parameters.

To examine the temperatures, we consider the pressure and temperature class in all 48 sections in each simulation and extract the temperature class for comparison. Fig. 6 shows a confusion matrix for target and actual temperature intervals, with the transformer performing well. When the target interval is 1 or 2, temperatures sometimes fall into neighboring intervals. As illustrated in Fig. 7, the temperatures resulting from suggested process parameters predominantly align with target areas. The correlation coefficient between target and actual temperatures is high at



**Figure 6.** Target temperature classes vs. temperature classes resulting from process parameters proposed by the transformer. Again, the numbers are to be read in the following way: In 90 % of cases where the target class was 1, the class resulting from the simulation using suggested process parameters was also 1. In 0.81 % of cases with target class 1, the resulting class was 2. The average accuracy was 75 % as stated in the title.



**Figure 7.** Target temperature sequence (light blue) and temperature sequence resulting from suggested process parameters (dark blue).

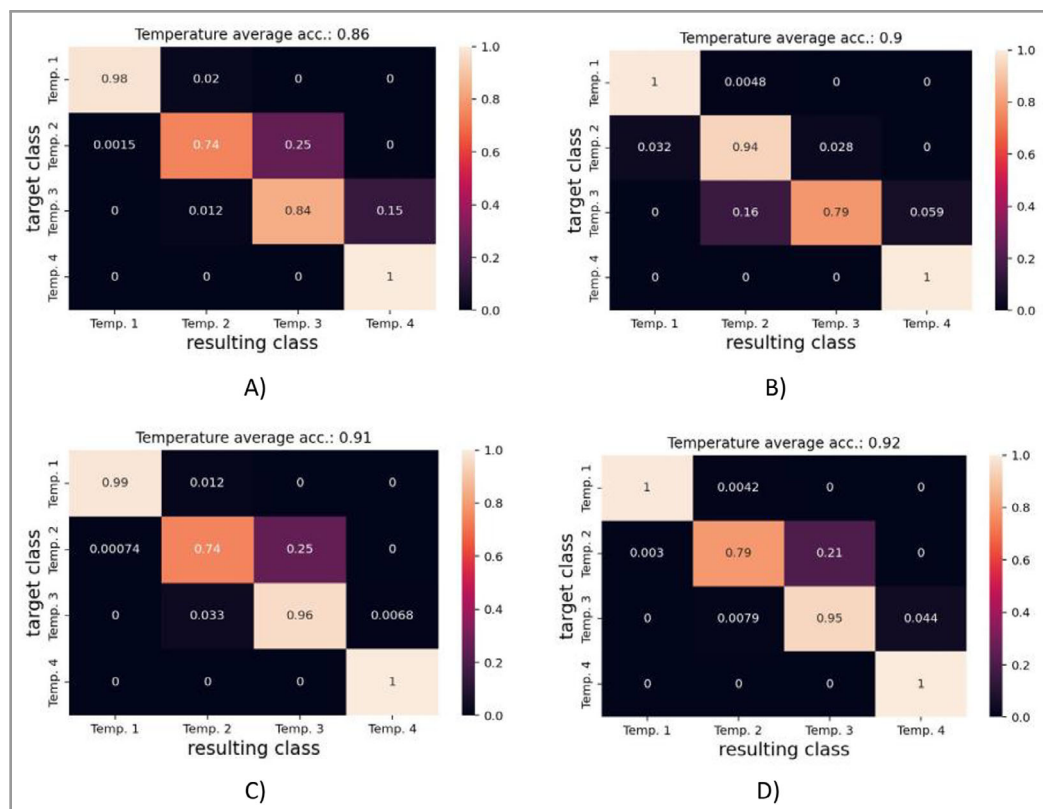
0.88, indicating strong agreement. When considering only such cases where the actual temperature deviates from the target area, the mean deviation is 12 °C, with a median of 9 °C. This corresponds to a median deviation of just 6 % of the temperature range (159 °C).

Overall, these evaluations demonstrate that the transformer model effectively fits the simulation data and suggests appropriate process parameters.

## 4 Fine-Tuning the FM

We aim to evaluate the fine-tuning of the FM to a specific plant. However, we do not have sufficient experimental data from a real plant for a thorough evaluation. Instead, we use simulated data that deviates from the original database in a way that is similar to the deviation of real-world data from simulated data. Measured real-world data deviates from simulated data in two aspects: it considers only one screw geometry and may experience a distribution shift due to simulation inaccuracies and plant specifics like wear. To replicate these differences, we select a screw geometry with a 30 % larger barrel diameter than that used in the initial simulations. We conducted a total of 39 300 simulations using a Latin Hypercube Design to select the process parameters and the same parameter ranges as before (refer to Tab. 1). We will use random subsets for fine-tuning to reflect the data-sparse reality.

We use varying amounts of simulations for fine-tuning. Due to the significant increase in barrel diameter, the pressure distribution differs significantly between the original training and fine-tuning datasets. In the fine-tuning data, only 0.2 % of pressure measurements fall within the first interval, while the rest fall into the second, leading us to exclude pressure from further evaluations. This scenario is unlikely in real applications, as fine-tuning measurements would come from an extruder with a diameter similar to



**Figure 8.** Confusion matrices of temperature results of suggested process parameters. For A), the process parameters were suggested only by the foundation model without fine-tuning. For B), C), and D), 50, 100, and 39 300 data points were used, respectively, for fine-tuning.

the pretraining data. We plan to extend the FM to include extruders of varying diameters in the future.

## 5 Results

To evaluate fine-tuning, the FM is fine-tuned with varying amounts of data. The resulting models suggest settings for specific temperature and pressure sequences in the extruder. We will only show evaluations for temperature due to restrictions on the number of figures and the fact that the pressure values are not as interesting for this fine-tuning dataset, as discussed previously. Nonetheless, the usual combination of pressure and temperature target sequences was provided to the model for the evaluation. We conducted 100 simulations using the process parameters suggested by each fine-tuned model and the previously described screw geometry. The confusion matrices in Fig. 8 display the target temperature in the extruder vs. the resulting temperatures from the suggested process parameters for different amounts of fine-tuning data.

The initial model, despite the barrel diameter difference from training simulations, shows high accuracy. However, the model fine-tuned with just 50 data points demonstrates

a significant improvement, with accuracy increasing further with additional data points, albeit minimally.

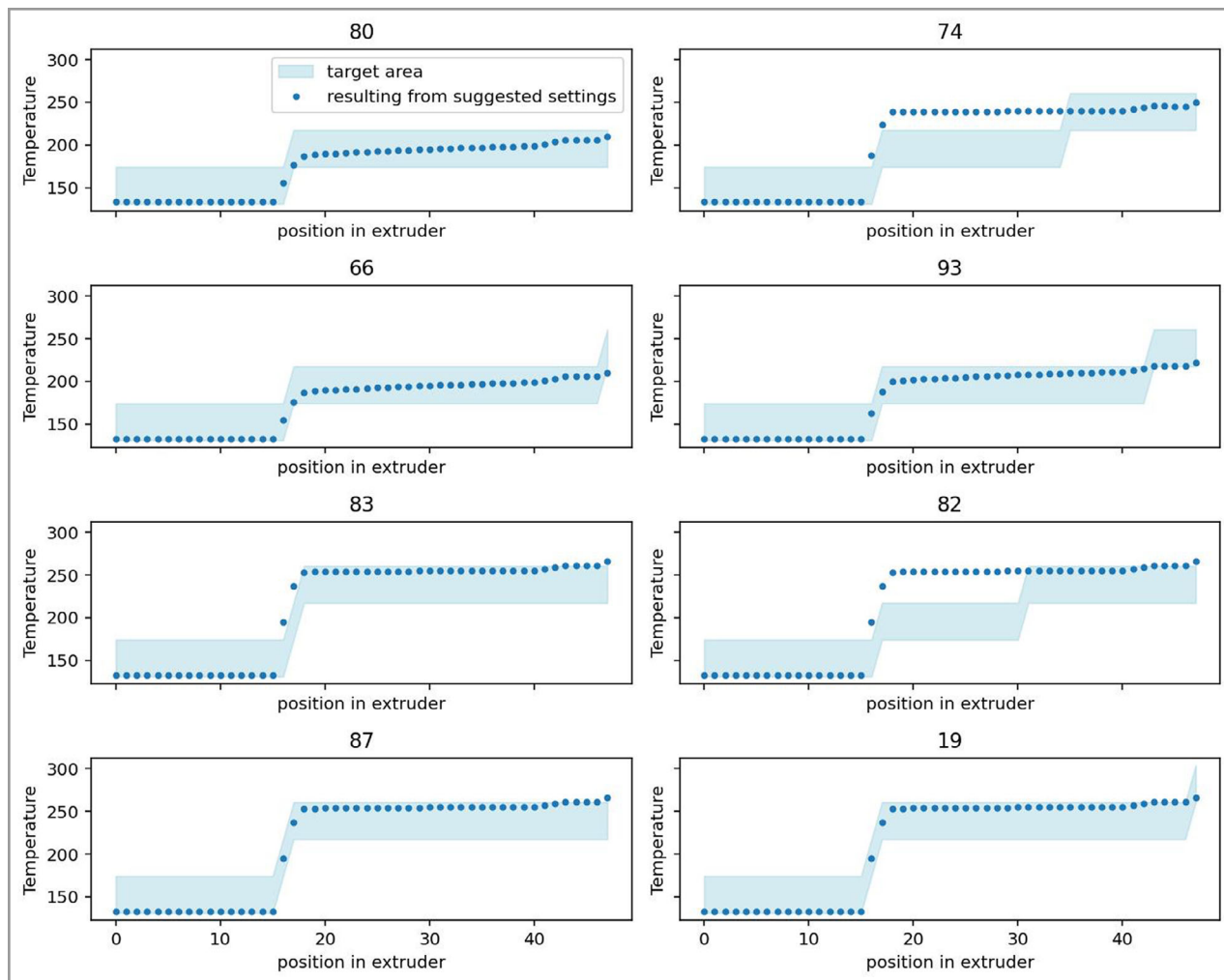
Figs. 9 and 10 compare the target temperature sequences with the achieved temperatures from suggested parameters for the un-tuned FM (Fig. 9) and the model fine-tuned with 50 data points (Fig. 10), revealing considerable enhancement.

In conclusion, fine-tuning the FM with as few as 50 data points significantly improves the performance of suggested process parameters.

## 6 Discussion

In this study, we demonstrated the use of a transformer to develop a FM for extrusion plants with various screw geometries, capable of suggesting process parameters for achieving desired temperature and pressure sequences. We also showcased the model's ability to be fine-tuned to specific extrusion plants not included in the training data, making it suitable for real-world plastic production settings.

Our findings also indicate that while increasing the amount of fine-tuning data improves accuracy, the rate of



**Figure 9.** Target temperatures and temperatures achieved by process parameters suggested by original FM. In some cases, the suggested settings deviate clearly from the desired temperature sequences (74, 82).

improvement diminishes beyond a certain point. This suggests that the model can reach a satisfactory performance level with relatively few data points, making it a practical approach for plants with limited data availability.

Future research should focus on extending this approach to encompass a broader range of extruder diameters and granulate properties, ultimately leading to a more universally applicable model for optimizing extrusion processes. Further, we want to examine the fine-tuning behavior if only a few sensors are available.

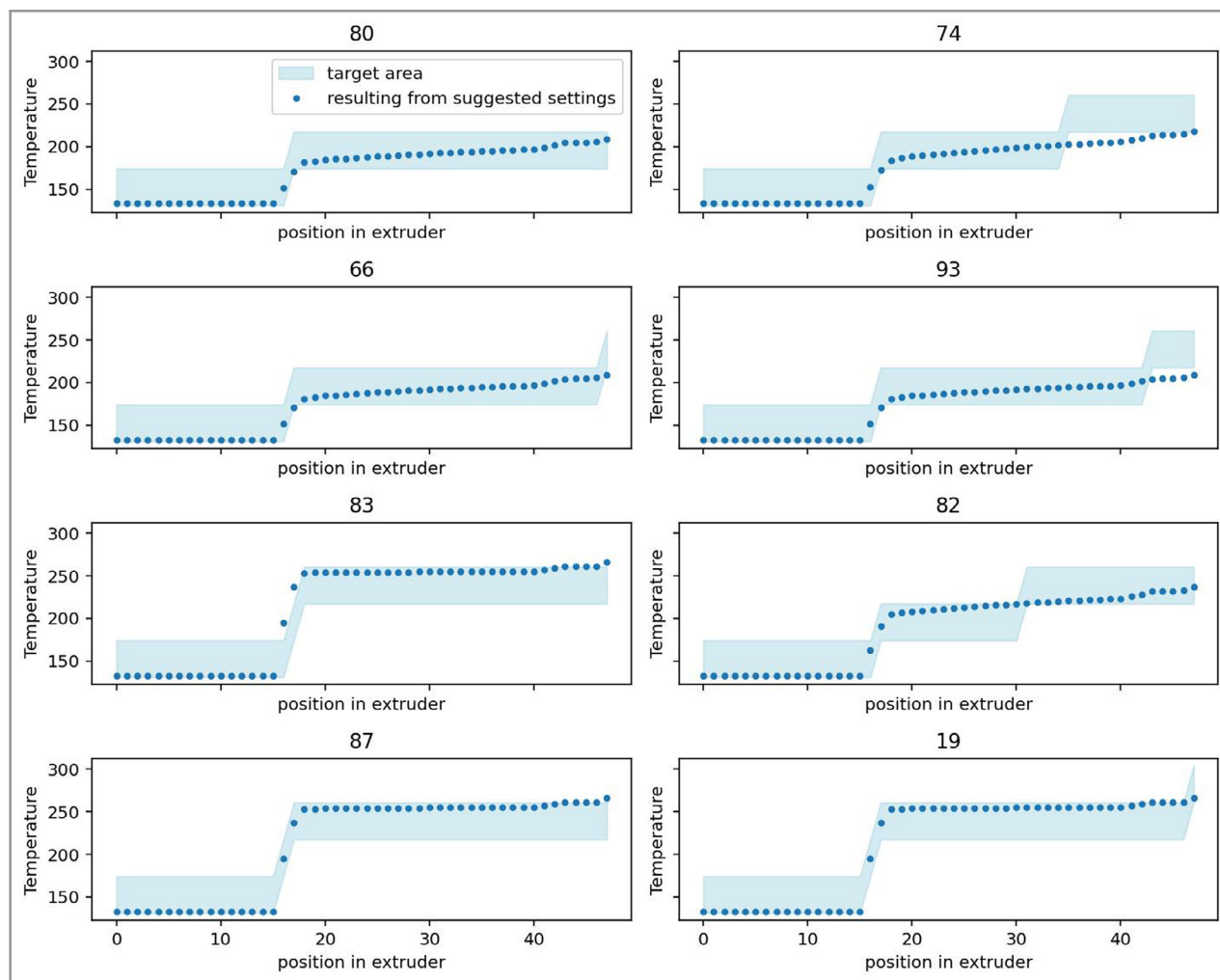
We also want to improve the FM training scheme. FM's outputs should serve as input for various classification and regression tasks, such as predicting energy consumption, motor power, or mean residence time. We attempted to classify these properties based on the FM's output, but the accuracy matched that of a simple model trained from scratch. Future research should focus on developing loss functions better suited for training FMs for this purpose.

## Acknowledgements

Open access funding enabled and organized by Projekt DEAL.

## Symbols used

$C_i$	one token from the set of all tokens which make up the input sequence of the transformer
$d$	embedding dimension
$L$	length of input sequence
$N$	number of sequences in the training set
$x_i$	embedded input sequence
$\tilde{x}_i$	embedded input sequence plus positional encoding
$\hat{x}_i$	encoded input sequence, output of the transformer encoder
$z_i$	input sequence of length $L$



**Figure 10.** Target temperatures and temperatures achieved by process parameters suggested by the model fine-tuned with only 50 data points. The fit especially improved for sequences 74 and 82 compared to the FM before fine-tuning displayed in the previous figure.

## Abbreviations

acc	accuracy
FFN	feed-forward network
FM	foundation model
ML	machine learning
PDE	partial differential equation
Press	pressure
Temp	temperature

## References

- [1] M. Hyvärinen, R. Jabeen, T. Kärki, *Polymers* **2020**, *12* (6), 1306. DOI: <https://doi.org/10.3390/polym12061306>
- [2] C. Martin, *AAPS PharmSciTech* **2016**, *17* (1), 3–19. DOI: <https://doi.org/10.1208/s12249-016-0485-3>
- [3] A. Shah, M. Gupta, *Comparison of the Flow in Co-rotating and Counter-Rotating Twin-Screw Extruders*, *ANTEC-Conf. Proc.*, **2004**.
- [4] A. Sarishvili, D. Just, K. Moser, A. Wirsén, J. Diemert, M. Jirstrand, *Chem. Ing. Tech.* **2021**, *93* (12), 1949–1954. DOI: <https://doi.org/10.1002/cite.202100093>
- [5] D. O. Kazmer, L. Hutson, D. Hazen, in *SPE ANTEC*. **2020**.
- [6] R. Ibañez, F. Casteran, C. Argerich, C. Ghnatios, N. Hascoet, A. Ammar, P. Cassagnau, F. Chinesta, *Fluids* **2020**, *5* (2), 94. DOI: <https://doi.org/10.3390/fluids5020094>
- [7] D. Hilger, N. Hosters, in *8th European Congress on Computational Methods in Applied Sciences and Engineering*, CIMNE **2022**.
- [8] Y. S. Perera, J. Li, A. L. Kelly, C. Abeykoon, *2023 European Control Conf. (ECC)*, IEEE, Piscataway, NJ **2023**.
- [9] J. Deng, K. Li, E. Harkin-Jones, M. Price, M. Fei, A. Kelly, J. Vera-Sorroche, P. Coates, E. Brown, *Trans. Inst. Meas. Control* **2014**, *36* (3), 382–390. DOI: <https://doi.org/10.1177/0142331213502696>

- [10] F. Castéran, K. Delage, N. Hascoët, A. Ammar, F. Chinesta, P. Cassagnau, *Polymers* **2022**, *14* (4), 800. DOI: <https://doi.org/10.3390/polym14040800>
- [11] N. Munir, M. Nugent, D. Whitaker, M. McAfee, *Pharmaceutics* **2021**, *13* (9), 1432. DOI: <https://doi.org/10.3390/pharmaceutics13091432>
- [12] J. Burr, A. Sarishvili, D. Just, N. Katsaouni, K. Moser, *Chem. Ing. Tech.* **2023**, *95*, 1555–1562. DOI: <https://doi.org/10.1002/cite.202200211>
- [13] S. K. Selvaraj, A. Raj, R. Rishikesh Mahadevan, U. Chadha, V. Paramasivam, *Adv. Mater. Sci. Eng.* **2022**, *2022*, 1–28. DOI: <https://doi.org/10.1155/2022/1949061>
- [14] P. G. Manjunath, P. Krishna, *AMR* **2012**, *463-464*, 674–678. DOI: <https://doi.org/10.4028/www.scientific.net/AMR.463-464.674>
- [15] K. Mulrennan, J. Donovan, L. Creedon, I. Rogers, J. G. Lyons, M. McAfee, *Polym. Test.* **2018**, *69*, 462–469. DOI: <https://doi.org/10.1016/j.polymertesting.2018.06.002>
- [16] Y.-M. Huang, W.-R. Jong, S.-C. Chen, *Polymers* **2021**, *13* (22), 3874. DOI: <https://doi.org/10.3390/polym13223874>
- [17] Y. Lockner, C. Hopmann, W. Zhao, *J. Manuf. Process.* **2022**, *73*, 395–408. DOI: <https://doi.org/10.1016/j.jmapro.2021.11.014>
- [18] H. Zhang, Z. Zhao, C. Wang, X. Zhang, X. Chen, *Addit. Manuf.* **2024**, *94*, 104467. DOI: <https://doi.org/10.1016/j.addma.2024.104467>
- [19] J. Ren, A.-T. Wei, Z. Jiang, H. Wang, X. Wang, *IEEE Trans. Automat. Sci. Eng.* **2022**, *19* (3), 2310–2321. DOI: <https://doi.org/10.1109/TASE.2021.3063389>
- [20] P. Liang, H.-D. Yang, W.-S. Chen, S.-Y. Xiao, Z.-Z. Lan, *Int. J. Comput. Integr. Manuf.* **2018**, *31* (4-5), 396–405. DOI: <https://doi.org/10.1080/0951192X.2017.1363410>
- [21] A. Gillioz, J. Casas, E. Mugellini, O. A. Khaled, in *Proc. of the 2020 Federated Conf. on Computer Science and Information Systems*, IEEE **2020**.
- [22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, in *Advances in Neural Information Processing Systems* (Eds: I. Guyon et al.), Curran Associates, Inc **2017**.
- [23] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, M. Shah, *ACM Comput. Surv.* **2022**, *54* (10s), 1–41. DOI: <https://doi.org/10.1145/3505244>
- [24] S. Ahmed, I. E. Nielsen, A. Tripathi, S. Siddiqui, R. P. Ramachandran, G. Rasool, *Circuits Syst. Signal Process.* **2023**, *42* (12), 7433–7466. DOI: <https://doi.org/10.1007/s00034-023-02454-8>
- [25] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, Q. Wen, in *Proc. of the 30th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, ACM, New York, NY **2024**.
- [26] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, in *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, ACL, Minneapolis, Minnesota **2019**.
- [27] C. Yun, S. Bhojanapalli, A. S. Rawat, S. J. Reddi, S. Kumar, *arXiv* **2019**. DOI: <https://doi.org/10.48550/arXiv.1912.10077>
- [28] Sciences Computers Consultants, *Ludovic*, Sciences Computers Consultants, Saint-Etienne **2017**.