DOI: 10.2507/37th.daaam.proceedings.xxx

FROM MANUAL MAPPING TO INTELLIGENT AUTOMATION: REDUCING MACHINE INTEGRATION EFFORT IN BATTERY CELL PRODUCTION

Wilhelm Jaspers, Alexander Lesnjak & Achim Kampker

Working Paper of 37th DAAAM Symposium

This Publication has to be referred as: Jaspers, W[ilhelm]; Lesnjak, A[lexander] & Kampker, A[chim] (2026). From Manual Mapping to Intelligent Automation: Reducing Machine Integration Effort in Battery Cell Production, Proceedings of the 37th DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/37th.daaam.proceedings.xxx

Abstract

Integrating new machines into battery cell manufacturing requires mapping heterogeneous machine parameters to standardised information models. A process that is often performed manually and causes significant engineering effort, time consumption, and cost. This work investigates the central question of how expensive manual mapping is in practice and how this process can be optimised using large language models (LLMs). To quantify the baseline, a structured survey-based method is developed to capture manual mapping effort and associated costs. Building on this baseline, three approaches are compared: manual mapping, mapping with a standard off-the-shelf LLM, and mapping with a domain-adapted LLM integrated into a user interface.

The proposed solution focuses on rejection sampling fine-tuning of a large model using a carefully prepared dataset derived from heterogeneous source formats, synthetic data, and a parameter knowledge base. A complexity-aware data labelling and filtering pipeline is introduced to ensure balanced training and evaluation. Teacher-student training with reasoning-generated ground truth is then used to optimise the student model. The work evaluates accuracy, processing time, and cost savings, demonstrating the high potential of a trained LLM to substantially reduce manual effort in parameter mapping and improve efficiency in machine integration in battery cell manufacturing.

Keywords: AI in Manufacturing; Large Language Models; Parameter Mapping; Reasoning Systems

1. Introduction

The global demand for battery cells is growing rapidly. Meeting this demand confronts battery cell manufacturing with different challenges regarding the optimisation of production processes and associated costs. A significant lever in addressing these challenges lies in the implementation of digital use cases, such as traceability and predictive quality. By providing data-driven transparency across the process chain, such use cases support production operations, facilitate improvements in product quality, and enable a reduction in scrap rates. [1] Comparable AI-driven approaches in industrial process control further illustrate how data-based methods can continuously adapt and optimise process parameters [2].

These use cases rely on data generated on the shop floor. To be usable, this data must be stored and described in a semantically meaningful way in appropriate databases. Since this does not occur automatically and the description of production parameters is highly machine-specific, the mapping of process parameters is largely carried out manually. [3] From an economic perspective, this manual effort constitutes a critical cost factor. Technical integrations are time-consuming and cost-intensive, and their success depends largely on the interplay between thorough preparation and the actual integration process. Moreover, a substantial share of the required effort is difficult to plan, as unknown parameters, missing attributes, or language barriers typically become apparent only during integration. Automating the mapping process therefore represents a promising lever for reducing both integration time and associated costs. [4] A similar pattern can be observed in CAM-based manufacturing, where the reuse of technological parameters for similar parts remains largely template-based or manual and is explicitly motivated by the goal of reducing programming time and increasing productivity [5].

Given the complexity and heterogeneity of the data involved, LLMs are particularly suitable for supporting and partially automating this mapping process. [6] Initial approaches to applying LLMs in industrial domains are promising. However, the technological landscape is evolving rapidly, with an increasing number of competitive models becoming available. Consequently, a key challenge lies in selecting an LLM, an appropriate training method, and a suitable data processing strategy that is not only technologically up to date, but also well aligned with the specific requirements of mapping processes in battery cell production. [7]

Following a brief introduction to the context of mapping in battery cell production and the relevant LLM technologies, the difficulties within the parameter-mapping process are stated and an estimated effort for manual mapping is evaluated. Afterwards, this paper adopts a systematic scientific approach to identify the most suitable model and an appropriate optimisation strategy. As model optimisation requires adequate data, the process begins with the preparation of domain-specific datasets, comprising their selection, cleaning, and structuring. The selected LLM is subsequently optimised using these datasets and implemented in a practical use case on the battery cell production line of the Fraunhofer Research Institution for Battery Cell Production (FFB). Within this use case, the performance of the model and of the optimisation approach is evaluated based on defined metrics, and the automated integration process is compared with the manual integration procedure. The paper concludes with an outlook on potential future work, including further optimisation steps, the incorporation of additional data sources, and the transfer of the approach to other processes in battery cell production.

2. Related Work

The digital integration of machines is a widely discussed topic in the literature. Approaches for plug-and-play functionality and the identification of machine parts appear promising. However, within these approaches, little attention has been paid to achieving a high degree of automation and to precise mapping and semantic enrichment of process parameters. [8,9] The present paper builds on the body of prior work, which investigated the mapping problem primarily from a technical perspective. After identifying LLMs as more suitable for the task than alternative machine-learning approaches, the study focused on selecting an appropriate model and training strategy under severe data constraints. Evaluations designed to reflect the complexity of the mapping task indicated that reasoning-capable models, particularly a model from the Qwen family, were the strongest candidates. In contrast, several other models generated responses that were either insufficiently detailed or hallucinated answers based on a lack of knowledge. [3,4]

However, the available dataset was small and insufficient for extensive training. Training solely on this dataset would therefore not adequately represent the real-world distribution of the mapping problem. These limitations also restricted the range of feasible training methods. To mitigate them, the dataset was augmented with synthetic examples. The augmentation introduced examples in different formats and languages, with the aim of improving data coverage and reducing the influence of the original dataset bias. However, the synthetic examples lacked complete reasoning traces, creating an additional challenge for training methods that require explicit rationale supervision. [6]

To identify the best strategy for optimising the LLM for the automated parameter mapping, it is necessary to have a deeper look and understanding of different training methods and strategies. One method is the Group Relative Policy Optimisation (GRPO), a reinforcement-learning method based on proximal policy optimisation and introduced by DeepSeek. Rather than directly reproducing reference answers, GRPO optimises model responses with respect to a reward function. [10]

GRPO enables training on synthetic data without requiring explicit reference reasoning traces, provided that an appropriate reward function can be defined. Low-Rank Adaptation (LoRA) can be used instead of full-parameter fine-tuning to reduce the number of trainable parameters and mitigate the risk of overfitting. Although GRPO can improve overall model performance, the model did not consistently satisfy the required output format or generate a valid unique identifier (UID). To address this issue, the LoRA adapters can subsequently be refined in a brief supervised fine-tuning (SFT) stage using non-synthetic data with manually created reasoning traces. [11]

Evaluation showed that this additional SFT stage can improve output compliance and overall mapping performance compared with GRPO alone. As a baseline, a standalone SFT model can also be trained without a preceding GRPO stage and with a larger number of training steps. The two-stage approach is only marginally slower than standalone SFT, yet it achieves nearly identical performance while requiring less total training time overall. [6]

These findings motivated further investigation into using SFT with datasets augmented by reasoning traces. Because the examples vary substantially, manually annotating reasoning paths would be time-consuming and could introduce

Working Paper of 37th DAAAM Symposium

inconsistencies. This creates two central design choices: selecting a model for generating reasoning traces and selecting an appropriate training method for using them.

Knowledge Distillation (KD) represents one possible approach. It transfers knowledge from a larger teacher model to a smaller student model, often within the same model family, by training the student to imitate the teacher's outputs. However, conventional KD does not inherently include a mechanism for filtering generated data. As a result, incorrect or unsuitable teacher-generated reasoning traces may be transferred to the student model. [12]

Rejection Sampling Fine-Tuning (RSFT) addresses this limitation by generating multiple candidate reasoning paths for each training example and validating them through software-based checks. Only the validated reasoning paths are retained as training targets during fine-tuning. The generation and training stages can use either the same model or multiple models of comparable scale. While KD aims to reproduce the teacher's outputs, RSFT aims to obtain multiple validated reasoning paths for the same input and to use these as training targets. This difference implies that RSFT preserves a greater diversity of valid solutions. [13]

3. Problem and Evaluation of Effort

3.1. Problem Statement

Machine integration represents a multifaceted challenge that originates prior to the physical delivery of the equipment. It comprises three principal dimensions: the coordination and communication between multiple stakeholders, the technical connection of the machine via transport protocols and interfaces, and its subsequent integration into existing digital systems. In industrial practice, this process is frequently impaired by limited information exchange and by technical specifications that are only partially fulfilled. Additional difficulties arise after delivery, for instance due to unknown parameters, missing attributes, language barriers, unknown abbreviations and more. Figure 1 shows an example of the manual mapping process and its challenges.

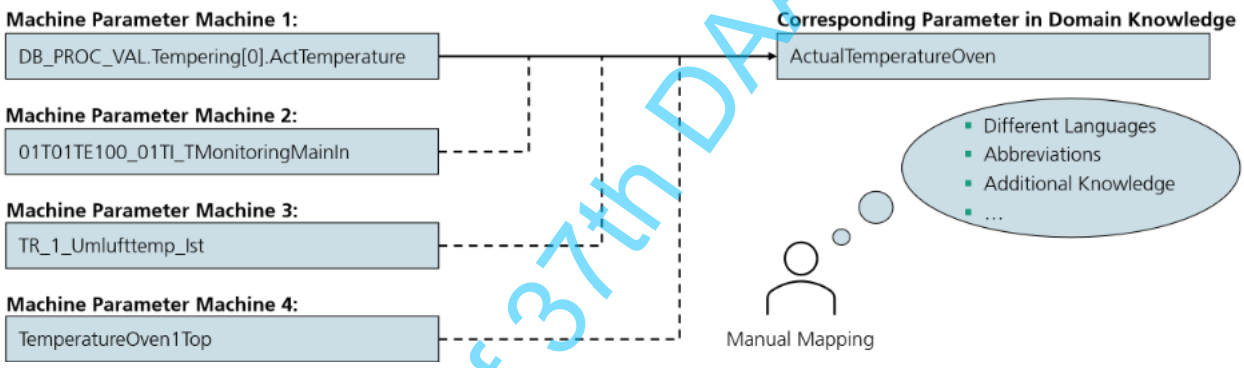

Fig. 1. Challenges of Parameter Mapping in a Production Scenario

Figure 1 shows the exact parameter for the temperature of an oven in battery cell manufacturing. Four different machines describe the same parameter in completely different ways: with abbreviation, in different languages or only understandable with further information. As such mapping issues must be resolved manually by the operating personnel, they generate considerable and largely unpredictable additional workload. This effort is further increased by the need to map machine-side parameters onto the internally defined parameter specification and to establish the final connection of the machine to the digital infrastructure.

3.2. Evaluation of the Estimated Effort

Assessing the effort required for the digital integration of industrial equipment, particularly regarding the mapping of equipment parameters, remains a largely under-explored area. Few systematic evaluations or datasets currently exist. The only published dataset addressing this topic focuses on standardised parameters. It estimates the integration time per parameter to range between 40 and 80 minutes. However, this source provides only an aggregate time value and does not disclose a detailed methodology for the underlying assessment. [14]

For this reason, the present work first examines the distinct phases that constitute the digital integration of industrial machines. This is illustrated using the example of battery cell production at Fraunhofer FFB. In general, a distinction is made between phases with a fixed duration and phases whose duration depends on the number of process parameters involved.

Working Paper of 37th DAAAM Symposium

The integration steps are structured as follows:

1. **Consultation** with the machine manufacturer regarding general data exchange and **system design** (fixed)
2. Definition of relevant **data** points (**fixed**)
3. Definition of relevant **data** points (**per parameter**)
4. **Testing** of the data **interface** and final functional testing (fixed)
5. Implementation of the system in the IT infrastructure (**standard** configuration, per parameter)
6. Implementation of the system in the IT infrastructure (**individual** configuration, per parameter)
7. **Validation** of the transferred **data** (per parameter)
8. **Rework** and retesting (fixed)

To evaluate concrete quantitative values, a survey of experts represents a suitable approach. Since the existing literature offers limited guidance, the number of directly relevant experts is also relatively small. A broadly designed study would likely yield a large volume of results, but at the cost of lower overall quality. Accordingly, participants were selected based on their expertise and existing knowledge of the digital integration of industrial equipment, with a particular focus on parameter mapping. This group of experts was further narrowed down with respect to their specific experience in battery cell production. Participants were asked to state the required time for each integration step in cumulative person-minutes, including the effort of all individuals involved. The results of the survey can be found in Figure 2.

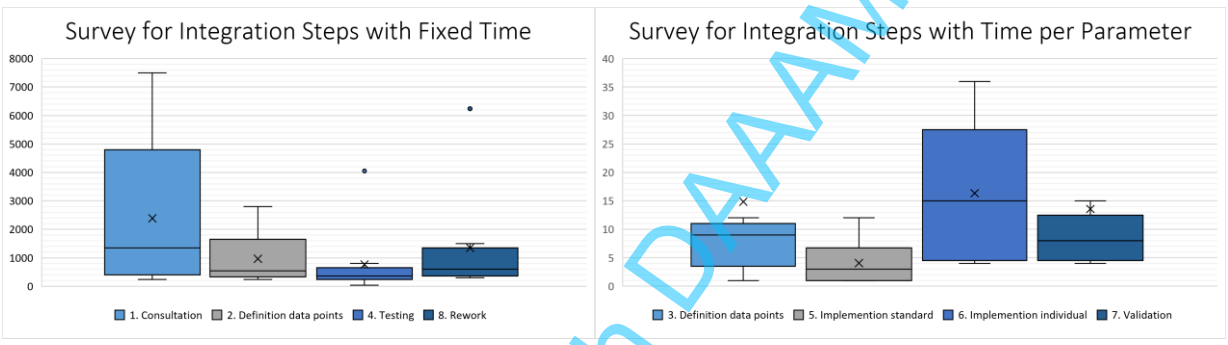

Fig. 2. Results of the Survey on the Effort Required for Digital Integration

Although the group of respondents was carefully selected and access to the survey was limited to experts, the responses show considerable variance, including several outliers. Figure 2 presents the estimated time as a boxplot diagram, separated into fixed time values and time values per parameter. This visualisation highlights individual outliers and allows for a clearer visual classification of the responses. Across all responses, the median required time is lower than the average, primarily due to outliers at the upper end. To better account for these outliers, the median of the individual responses is therefore used as the basis for calculating the required time.

To apply these individual values to a concrete example, an average machine from Fraunhofer FFB's battery cell production is used as a basis. This machine comprises a total of 235 parameters, of which 200 are implemented individually and 35 follow standardised specifications. Standardised specifications include, for example, the machine status, whereas individually implemented parameters encompass all process parameters with custom descriptions. Combining these values with the fixed and parameter-dependent time components results in an overall total time of approximately 165 hours for the digital integration process. The results of the survey as well as the calculations for the use case can be found in Table 5.

4. Fine-Tuning the Large Language Model

4.1. Model Baseline and Training Strategy

Developing an automated parameter-mapping platform requires an AI system capable of assigning a source parameter to the correct target parameter from a large set of possible candidates, despite limited contextual information. The possible target parameters are defined in an internally maintained data repository referred to as the *Parameter Portal*. This target vocabulary is fixed and assumed to be complete, meaning that each input parameter is expected to correspond to one entry in the portal. Leaving a parameter unmapped is therefore not a valid outcome within the training of the model, as it would disrupt the downstream integration process. Consequently, the model must always identify the most suitable target from the provided list for the training. [15] During the integration process, there may be instances where a parameter is not mapped. How to handle this is explained in Chapter 5.

Given the complexity of this task, high mapping accuracy is essential. Building on previous work, reasoning-capable models were selected as the baseline for inference, as their ability to process complex relationships can support more

Working Paper of 37th DAAAM Symposium

reliable mapping decisions. However, accuracy is not the only relevant criterion. Inference time and operational costs are also crucial for the practical deployment of an automated mapping platform. Based on the results of previous experiments, a model from the *Qwen* 3 family, *Qwen3.6-35B-A3B,* model was selected. [6] As a sparse mixture-of-experts architecture, it activates only approximately three billion of its 35 billion total parameters per token. The computational cost per inference step therefore remains comparable to that of a small dense model, while the substantially larger total parameter capacity is retained. This property is particularly relevant for the present task: the increased representational capacity supports the discrimination of semantically closely related target parameters and provides broader domain and terminology knowledge, which stabilises mapping decisions in cases of short, abbreviated, or incomplete parameter designations. At the same time, the sparse activation pattern keeps inference latency and operational expenditure low, since both scale primarily with the number of active rather than total parameters. The additional overhead is largely limited to the memory required to host the model weights, which can be addressed through fine-tuning. Overall, the model offers a favourable trade-off between mapping accuracy, inference time, and operational costs, enabling an economically viable deployment in continuous operation. [16]

To properly prepare this model for parameter mapping, a consistent strategy for structuring the training process is required. The approach is illustrated in Figure 3.

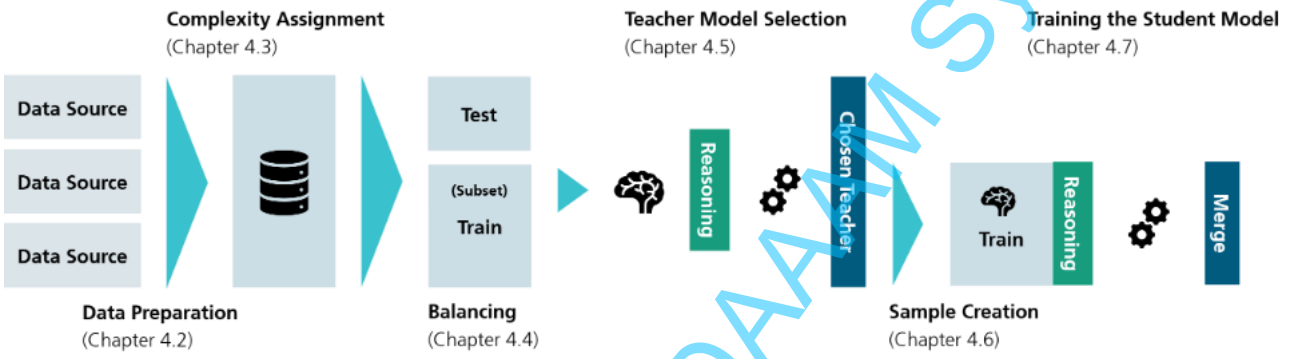

Fig. 3. Process of Fine-Tuning the LLM for Parameter Mapping

As shown in Figure 3, the fine-tuning process follows the RSFT approach. First, different data sources are gathered and the datasets are meaningfully enriched. Following this data preparation step, the complexity of each dataset is assessed to construct the best possible dataset. Through the subsequent balancing step, balanced subsets for training and testing are generated from the large pool of enriched datasets. Part of the training data first serves as the basis for selecting the teacher model, after which the complete datasets for the final training of the student model are created. Since the test data has not been used for training up to this point, it is finally used to validate the process.

4.2. Data Preparation

The available data sources were first consolidated during the data collection phase. The initial dataset consisted of lists of parameters from different machines and contained only a limited and imbalanced set of data points. To increase the size and diversity of the training data, a synthetic dataset previously used in related work was added. [6] All datasets include the correct parameter mappings, which serve as target labels, as well as the corresponding information from the *Parameter Portal* required for the mapping process. During data integration, duplicate and erroneous entries were removed. [15] All datasets were merged and transformed into a uniform format suitable for model training. The resulting dataset was then filtered and evaluated to ensure its suitability for the fine-tuning process.

4.3. Complexity Assignment

Given the imbalance in the available dataset, the complexity of individual mapping tasks must be considered explicitly. Some mappings are easy to identify because they share overlapping vocabulary or clearly comparable terms, while others can only be solved by looking at additional attributes or drawing on a more technical understanding of the production process. Moreover, a candidate list may contain several parameters that appear similar in wording but differ substantially in their semantic meaning. To construct training and test datasets with a balanced distribution of mapping complexity, the complexity of each instance must first be determined. This enables the categorisation of instances into different complexity levels and provides the basis for balancing the dataset for subsequent training and validation. This assessment was conducted using *Qwen 3.6-35B-A3B*. The enriched dataset contains three attributes for each instance: the **parameter**, its **data type**, and the assigned **complexity level**.

The complexity assignment follows a two-step procedure. In the first step, the model receives a prompt containing the evaluation guidelines, the input parameter, and the list of possible target parameters. It independently selects a complexity label and additionally provides a self-assessment of the difficulty of its classification. This stage is executed once for each

Working Paper of 37th DAAAM Symposium

instance. The model may choose between four labels: easy, medium, hard, and impossible. The impossible label serves as a control condition, allowing the model to identify cases that cannot be solved based on the available information rather than forcing an unsuitable classification.

In the second step, the model receives the evaluation guidelines, the input parameter, the correct mapping, the candidate list, and the complete output of the first evaluation, including its reasoning and final decision. It is then asked to reconsider its initial assessment and revise the assigned complexity level if necessary. As the correct mapping is available during this refinement step, the model is restricted to the labels easy, medium, and hard.

4.4. Balancing

Following the complexity assignment, the dataset is divided into training and test subsets. As the data comes from several machines, the instances differ not only in mapping complexity but also in source-specific naming conventions and characteristics. A simple split of the dataset could overrepresent individual data sources or complexity levels in one subset, potentially influencing the evaluation results.

Therefore, the dataset was split at a 3:1 ratio into training, ensuring to keep both the relative contribution of each data source and, within each source, the distribution of easy, medium, and hard instances. This ensures that both subsets remain representative of the overall dataset and enables a more reliable evaluation of the model's generalisation performance.

4.5. Teacher Model Selection

Within the training strategy, a teacher model is required to generate candidate mappings together with their corresponding reasoning traces. Following software-based validation, the accepted outputs are used as supervision for fine-tuning the student model. To identify the most suitable teacher model, four candidates from different deployment settings were evaluated. The models o3 and GPT-5 were accessed through the OpenAI API, whereas *Gemma 4 26B-A4B* and *Qwen 3.6-35B-A3B* were deployed locally. In rejection sampling fine-tuning, a standard practice is that the model acts as its own teacher by generating candidate samples. Following this approach, we used *Qwen 3.6-35B-A3B* as its own teacher, even though it also serves as the student model. [13] To validate whether this self-teacher method is the most suitable for our task, we compared its performance against the other candidate teachers.

The comparison focuses primarily on the downstream mapping performance of the student model trained on teacher-generated data. In addition, the average response time as well as the traceability and consistency of the generated reasoning traces are considered. In this context, traceability refers to the extent to which the reasoning makes the relationship between the input parameter, the available information and the selected target parameter comprehensible.

For the evaluation, a subset of the training dataset is used. Each teacher model generates two complete and correct candidate responses for every data point, consisting of a proposed mapping and an associated reasoning trace. After validation, the resulting outputs are used to create separate teacher-specific fine-tuning datasets. The same student model is then fine-tuned separately on each dataset and evaluated using the not used test dataset.

As the outputs of generative models are not always the same, each test instance is evaluated five times. The results are evaluated using consistency thresholds. Since five repetitions produce only the possible success rates of 0 %, 20 %, 40 %, 60 %, 80 %, and 100 %, an exact 50 % threshold cannot be obtained. The 20 % threshold indicates that a mapping is correct in at least one of five runs. The intermediate threshold is therefore defined as a majority criterion, requiring at least three correct mappings out of five runs, corresponding to 60 %. Finally, the 100 % threshold requires a correct mapping in all five runs. These thresholds allow the evaluation of both the ability to generate a correct result at least once and the consistency of the model across repeated inferences.

The results show a trade-off between generation speed and downstream mapping performance. While the fine-tuned models using OpenAI models as teachers generated responses more quickly, their outputs achieved lower performance under the strict 100% consistency threshold than those trained on data generated by the locally deployed open-source models. The main differences between *Gemma* and *Qwen* were observed at the 20% threshold and in the required response-generation time. On average, *Qwen* required approximately twice as long as *Gemma* to generate a response. Table 1 summarises these performance metrics across the three consistency thresholds for each teacher model.

Fine-tuned	100 %	60 %	≥ 20 %	Avg. correct	Avg. Time
Base	46.0 %	61.0 %	75.0 %	3.05	11.6 s
Gemma 4	59.0 %	71.0 %	82.0 %	3.54	121.2 s
Qwen 3	61.0 %	71.0 %	79.0 %	3.52	250.2 s
o3	48.0 %	69.0 %	83.0 %	3.39	3.4 s
GPT-5	46.0 %	70.0 %	80.0 %	3.31	5.4 s

Table 1. Results for the Evaluation of the Teacher Model

Working Paper of 37th DAAAM Symposium

Based on the evaluation results, *Gemma 4 26B-A4B* was selected as the teacher model. Although the OpenAI models provided shorter response-generation times, their lower performance under the strict consistency criterion reduced their suitability for generating RSFT training data. *Qwen3.6-35B-A3B* achieved competitive results but required substantially more time for response generation. *Gemma* therefore provides the most suitable balance between downstream mapping performance and generation efficiency.

4.6. Sample Creation

In addition to the base information required for inference, reasoning models specifically require a reasoning trace to perform supervised fine-tuning. However, such a trace is not available in the base dataset. Using the selected teacher model, this reasoning trace is generated and added to the dataset as ground truth. The approach follows a generate-and-filter strategy. Many candidate responses are produced, and only the correct ones are retained. In this way, the training dataset is augmented with a reasoning trace consisting exclusively of correct data, a procedure known as rejection sampling fine-tuning.

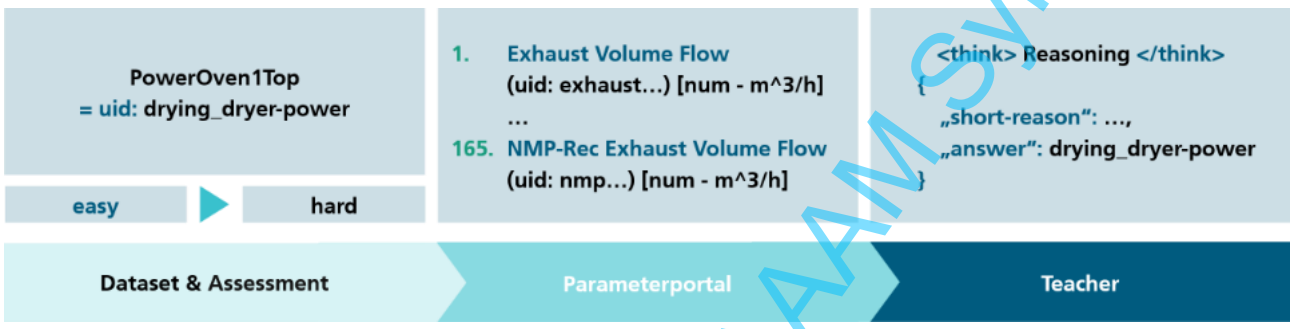

Fig. 4. Resulting Dataset for Training the LLM

In the first step, the model is loaded in its base configuration to preserve diversity in the responses while avoiding substantial hallucinations. The focus is placed on achieving high quality with a low number of generated tokens, thereby ensuring reliability. The token limit is then increased incrementally until 12 correct and distinct responses are available for each data point. The resulting dataset contains the attributes shown in Figure 4.

4.7. Training the Student Model

Building on the complete dataset, SFT is performed on the selected student model, with a small number of training epochs using Quantised LoRA. This approach is chosen to adapt the model efficiently to the mapping task while keeping both the computational cost and the risk of overfitting low. [17] The reasoning traces generated in the previous step serve as the supervision signal, such that the model learns to reproduce the structured reasoning process rather than merely the final target parameter.

The training pipeline was implemented using the *Unsloth* framework, which provides memory- and compute-optimised kernels for efficient fine-tuning of transformer models. *Unsloth* reduces VRAM consumption and roughly doubles training throughput compared with a standard implementation, while additionally supporting substantially longer context lengths. [18] These properties are particularly relevant for the present setting, as the reasoning traces used as the supervision signal, together with the large candidate list of target parameters, result in comparatively long input sequences. The model is loaded in 4-bit quantised form, which further reduces the memory footprint and makes fine-tuning of a 35-billion-parameter mixture-of-experts model feasible.

The adapter rank was set to $r = 16$ with a scaling factor of $lora_alpha = 16$, resulting in an effective scaling ratio of 1.0. This configuration represents a deliberate trade-off. The rank is large enough to capture the task-specific mapping semantics, yet small enough to keep the number of trainable parameters and therefore the risk of forgetting of the model's general reasoning ability low. Adapters are applied to all language layers, whereas the vision layers of the natively multimodal model are excluded from training. This restriction reflects the fact that the parameter-mapping task is purely text-based, so that adapting the visual encoder would increase the number of trainable parameters without any expected benefit. Consistent with the recommendation for mixture-of-experts architectures, the router layer was likewise excluded from adaptation, to preserve the pretrained expert-selection behaviour and to avoid destabilising the load distribution across experts during the comparatively short fine-tuning run.

5. Integration of the LLM into the Mapping Tool

To automate the parameter-mapping process, the existing mapping software was extended with LLM-based functionality. Through the user interface, users can establish a connection to a machine, retrieve its parameter list, and load the available target parameters from the internal *Parameter Portal*. Both datasets are displayed in tabular form,

Working Paper of 37th DAAAM Symposium

providing an overview of the source and target parameters. In the conventional workflow, mappings can be created manually via drag and drop, whereby multiple machine parameters may be assigned to a single *Parameter Portal* entry. The completed mapping can subsequently be exported for use in downstream systems.

The fine-tuned model is deployed in a *Docker* container using *vLLM* and is accessed through a REST interface. An integrated function within the user interface transmits the relevant machine and *Parameter Portal* data to the model and displays the automatically generated mappings in the application (see Figure 5). [19, 20]

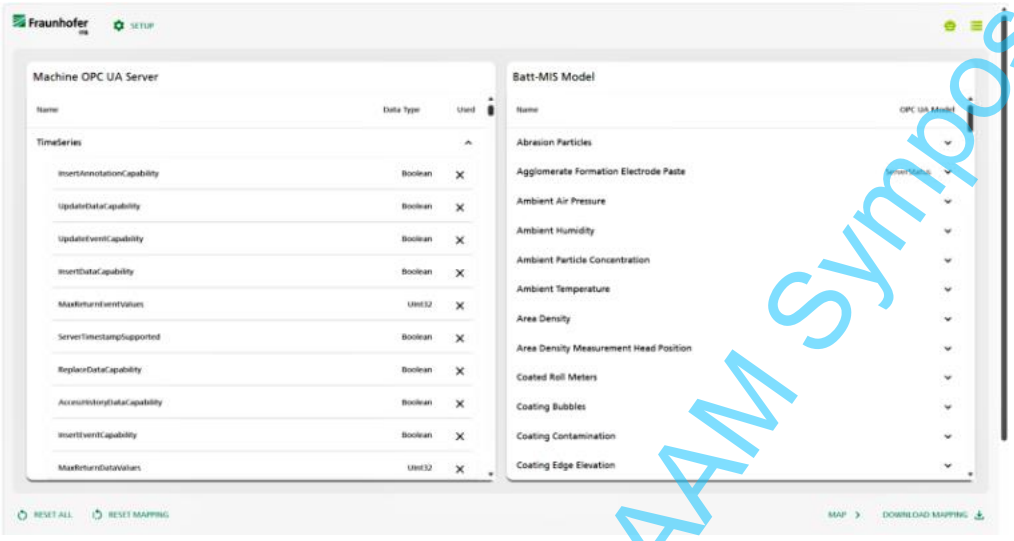

Fig. 5. User Interface to Supervise the Automatic Mapping of Process Parameters

For each proposed mapping, users can access a detailed view containing a concise reasoning trace and additional parameter information. In real-world integration scenarios, the mapping model may be unable to assign a machine parameter clearly, for example because no suitable target parameter exists in the *Parameter Portal* or because the available information is insufficient for a reliable mapping. The user interface therefore supports manual supervision, enabling users to review, modify, or complete such mappings. To preserve the *Parameter Portal* as the single source of truth, requests for the inclusion of previously unavailable parameters can be submitted to its administrator. In the interim, a local variable can be created, assigned an appropriate designation, and manually linked to the corresponding machine parameter via the user interface. Overall, this approach combines automated mapping capabilities with full user control over the final integration result.

6. Results

6.1. Setup of Evaluation Criteria for the LLM

To properly assess the fine-tuning of the LLM for the given use case, as well as the resulting time savings, a structured testing procedure with clearly defined evaluation criteria is required. Performance is assessed with a focus on mapping accuracy, output validity, consistency across repeated generations, and inference speed. A generated mapping is considered correct only if the predicted target parameter exactly matches the ground-truth mapping. To account for the stochastic nature of LLM inference, each test instance is evaluated ten times under each inference configuration.

Criteria	Description
100%	Ten correct mappings across all generations
≥80 %	Eight or more correct responses
≥50 %	Five or more correct responses
<50 %	Up to four correct responses
Existing UID	Proportion of generated target labels corresponding to an existing entry in the *Parameter Portal*
Prompt Time M	Median inference time per generated response

Table 2. Criteria for the Evaluation of the Automated Mapping

Working Paper of 37th DAAAM Symposium

For the evaluation, two sampling configurations are compared. The first configuration uses the recommended sampling parameters of the model and represents the expected behaviour during regular deployment. The second configuration uses a reduced temperature. This more constrained setting reduces variation between generated responses and enables an assessment of the model's mapping performance under more deterministic inference conditions.

Overall, the outcome of the mapping task is evaluated based on consistency and correctness. For each test instance, the number of correct mappings across the ten generations is converted into a consistency rate. Since ten repetitions are performed per instance, possible consistency rates range from 0% to 100% in increments of 10%. To provide a clear and unified overview of results, the summarised criteria in Table 2 are used for evaluation.

In addition to correctness, the validity of the generated responses is also evaluated. By checking whether the predicted UID exists in the *Parameter Portal*, it is possible to assess the extent to which the system hallucinates and generates non-existent UIDs that do not correspond to any predefined parameter. Furthermore, the prompt processing time is recorded, defined as the median response time across the individual generations for a given mapping. As in Section 3.2, the median is chosen over the mean, since individual outliers would otherwise disproportionately raise the average, making it a less representative and less meaningful measure.

6.2. Evaluation of the Fine-Tuning

The evaluation of the results was carried out based on the scenario described in Section 6.1. The base (untrained) model of *Qwen 3.6-35B-A3B* achieved the lowest performance on the mapping task among all evaluated models. *Gemma 4 26B-A4B* served as a reference model and was also selected as the teacher model during training. Training improved the performance by roughly 30 percentage points, while increasing the median time per prompt by two seconds. Despite remaining below the teacher model's performance, the trained model was faster. When evaluated at a low temperature as described earlier, it achieved results approximately 5 percentage points above the teacher model, with a median time per prompt 11 seconds lower, and it performed better in over 50% of cases. The results are presented in Table 3.

Model	100 %	≥80 %	≥50 %	<50 %	Existing UID	Prompt Time M
Base	37.50 %	64.58 %	78.12 %	21.88 %	89.56 %	22s
Trained	67.71 %	75 %	80.21 %	19.79 %	99.79 %	24s
Trained (low temp)	77.60 %	78.12 %	80.73 %	19.27 %	96.35 %	21s
Gemma 4 26B-A4B	72.40 %	77.09 %	79.69 %	20.31 %	100 %	32s

Table 3. Evaluation of the Training for Automated Mapping

Beyond aggregate correctness metrics, it is equally important to assess how well the model handles cases of varying difficulty. To this end, the test instances were grouped into three complexity levels: easy, medium, and hard. This breakdown reveals whether the model performs uniformly across difficulty levels or whether its accuracy declines as mapping complexity increases. The results are reported as the percentage of correctly identified easy, medium, and hard cases across all evaluation runs, as portrayed in Table 4.

Model	Easy	Medium	Hard
Base	31.3 %	45.5 %	38.9 %
Trained	73.5 %	67.3 %	59.3 %
Trained (low temp)	86.8 %	76.4 %	64.8 %
Gemma 4 26B-A4B	79.5 %	69.1 %	64.8 %

Table 4. Comparison of Training Level Results, Broken Down by Difficulty Level

The results in the table highlight the suitability of *Gemma 4 26B-A4B* as the teacher model, as it can correctly distinguish between easier and harder cases during evaluation. In contrast, the base *Qwen 3.6* model handles tasks of varying complexity with inconsistent accuracy. Specifically, easy mappings are correctly identified in only 31.3% of cases, while medium and hard cases achieve 45.5% and 38.9% accuracy, respectively. This suggests that the untrained model is confused by task complexity and tends to hallucinate answers. Training substantially improves performance: the model then correctly identifies 73.5% of easy, 67.3% of medium, and 59.3% of hard tasks. Furthermore, when the trained model is evaluated at a low temperature, performance improves further, surpassing the teacher model by approximately 7 percentage points on both easy and medium tasks, while matching the teacher on hard cases.

In summary, these results demonstrate that the fine-tuned model produces correct mappings in twice as many cases as the base model, while also outperforming its own teacher model in terms of both accuracy and inference speed.

Working Paper of 37th DAAAM Symposium

6.3. Calculation of the Adjusted Effort

Since one of the primary goals of the model is to reduce the effort required for digital integration and a complete parameter mapping, it is essential to reassess this effort considering the model's performance. For this purpose, the data from Section 3.2 serves as a baseline and is compared with the results presented in Section 6.2. Table 5 shows the result of the calculations for the mapping effort comparing manual and automated integration.

In general, all integration steps with fixed time values remain unchanged even with automated mapping. It is primarily the parameter-dependent steps that are affected. Since the model requires approximately 20 seconds to map a single parameter, and manual supervision should still be performed, the estimated time for automatically mapped parameters (see Table 3 - 21s Prompt Time for the trained model) is rounded up to 1 minute.

Use-Case:	35 Standard Parameter 200 Individual Parameter		77% Correct Automated Mapping	
Integration Step	**Base Time Manual (min)**	**Manual Effort (min)**	**Base Time Automated (min)**	**Automated Effort (min)**
1. Consultation (f)	1350	1350	1350	1350
2. Definition (f)	540	540	540	540
3. Definition (p)	9	2115	1	667.4
4. Testing (f)	360	360	360	360
5. Implementation Standard (p)	3	105	1	51.1
6. Implementation Individual (p)	15	3000	1	844
7. Validation (p)	8	1880	1	613.35
8. Rework (f)	600	600	600	600
Sum in min	-	9950	-	5025.85
Sum in h	-	165.8	-	83.8

(f) = fixed time
(p) = time per parameter

Table 5. Effort for Manual and Automated Parameter Mapping

Given that the system correctly maps only 77% of all parameters, the remaining 23% are calculated using the originally evaluated time values. For the example machine introduced in Section 3.2, this results in a newly calculated total time of approximately 84 hours for the complete digital integration process, including full parameter mapping. Compared with manual mapping, this corresponds to a reduction in integration time of 49%. The more complex the machine and the greater the number of parameters involved, the larger the resulting savings in effort.

7. Conclusion and Outlook

7.1. Conclusion

The mapping of machine parameters to existing domain knowledge is currently a labour-intensive, largely manual process. By training an LLM on real-world data and integrating it into a software solution, this mapping process can be automated to a high degree, nearly halving the manual effort required for digital integration. In this paper, a comprehensive survey was first conducted to capture the time required for digital integration, with a particular focus on complete parameter mapping. Building on these findings, a pipeline was developed and implemented to optimise an LLM for the specific use case of automated parameter mapping. A final evaluation of the results confirmed both the feasibility of the proposed approach and the substantial reduction in effort achieved through automated mapping.

Fine-tuning substantially improves the mapping performance of the *Qwen3.6-35B-A3B* model, allowing it to reach a better accuracy level comparable to the much larger reference model, particularly when sampling variability is reduced. This demonstrates that fine-tuning alone can compensate for differences in model size, despite the significant difference in parameter count. Beyond performance, deploying the smaller fine-tuned model further reduces operational costs while offering additional benefits in data privacy, deployment simplicity, and security.

7.2. Outlook

As synthetic data is currently used to compensate for the limited availability of real-world training data, the training process can be repeated and extended as additional validated data become available. A larger and more diverse dataset

Working Paper of 37th DAAAM Symposium

would reduce potential biases introduced by synthetic examples and may further improve the model's generalisation performance.

The proposed training pipeline is also transferable to future model generations. As LLM capabilities continue to evolve rapidly, newer models can be fine-tuned using the same multi-stage approach. In particular, the generation of reasoning paths currently requires large context windows and high output-token budgets. Future models may be able to generate more accurate, concise, and efficient reasoning traces, which could improve both training efficiency and mapping performance. The iterative generation, validation, and use of reasoning paths may therefore be repeated as stronger models and improved data become available.

Independently of the selected training strategy, the results demonstrate that the model can improve substantially through task-specific fine-tuning. *Qwen3.6-35B-A3B* also provides capabilities beyond text generation, such as tool calling. In future work, the fine-tuned mapping model could be integrated into an agentic architecture alongside smaller specialised models. A routing mechanism could distribute subtasks among these models and pre-filter potential mapping candidates before the main mapping step. Such an approach could reduce context requirements, decrease computational cost, and further improve overall system performance.

In addition to further improving the training of the LLM, the mapping process itself offers potential for future enhancement. As more parameter mappings are performed, domain knowledge can be progressively formalised and incorporated, resulting in a more comprehensive reference dataset that benefits future parameter mapping tasks. Furthermore, this approach could be extended to other industries and application domains within digital integration.

8. Acknowledgements

Funding reference: The project *"FoFeBat – Research Fab Battery Cells"* is funded by the German Federal Ministry of Education and Research. Grant number 03XP0256.

9. References

[1] Krauß, J.; Fitzner, A.; Mohring, L.; Dahmen, C.; Wunderlich, P. & Ehteshami-Flammer, N. (2024). Potentials of digitalization in battery cell manufacturing, 10.24406/publica-2955

[2] Masnec, L[ea] & Stapic, Z[latko] (2025). System Dynamics and Digital Twins: A Bibliometric Insight into Real-Time Data Integration, Proceedings of the 36th DAAAM International Symposium, pp. 0180-0185, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-47-1, ISSN 1726-9679, Vienna, Austria DOI: 10.2507/36th.daaam.proceedings.024

[3] Gönnheimer, P.; et al. (2023). Interoperable system for automated extraction and identification of machine control data in brownfield production, Manufacturing Letters, Volume 35, pp 915- 925

[4] Jaspers, W.; Schmetz, A.; Roth, D.; Becker, B. & Kampker, A. (2025). Identification of machine parameters in battery cell production: a comparison of different methods and approaches for mapping parameters. Procedia CIRP, Vol. 134, pp. 157-162

[5] Peterka, J[ozef] & Hipp, J[ozef] (2024). Comparison of Creating Technological Operations Based on Similarity in CAM Software, Proceedings of the 35th DAAAM International Symposium, pp. 0026-0032, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-44-0, ISSN 1726-9679, Vienna, Austria DOI: 10.2507/35th.daaam.proceedings.004

[6] Jaspers, W.; Lesnjak, A. & Kampker, A. (2026). Automating the parameter mapping in battery cell production: optimizing an LLM on complex metadata from a productive plant, Proceedings of the Procedia CIRP Conference on Intelligent Computation in Manufacturing

[7] Gomaa, W. & Fahmy, A. (2013). A survey of text similarity approaches. International Journal of Computer Applications, Vol. 68, No. 13, pp. 13-18

[8] Bennulf, M.; Danielsson, F. & Svensson, B. (2019). Identification of resources and parts in a Plug and Produce system using OPC UA. Procedia Manufacturing, Vol. 38, pp. 858-865

[9] Dürkop, L. & Jasperneite, J. (2017). „Plug & Produce" als Anwendungsfall von Industrie 4.0 [„Plug & Produce" as a Use Case of Industrie 4.0]. In B. Vogel-Heuser, T. Bauernhansl & M. ten Hompel (Hrsg.), Handbuch Industrie 4.0: Bd. 2. Automatisierung, pp 59-71, Springer Berlin Heidelberg

[10] Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y. & Guo, D. (2024). DeepSeekMath: pushing the limits of mathematical reasoning in open language models, arXiv, Available from: https://doi.org/10.48550/arXiv.2402.03300 Accessed: 2026-10-09

[11] Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L. & Chen, W. (2021). LoRA: low-rank adaptation of large language models, arXiv, Available from: https://doi.org/10.48550/arXiv.2106.09685 Accessed: 2026-10-09

[12] Hinton, G.; Vinyals, O. & Dean, J. (2015). Distilling the knowledge in a neural network, arXiv, Available from: https://doi.org/10.48550/arXiv.1503.02531 Accessed: 2026-10-09

[13] Yuan, Z.; Yuan, H.; Li, C.; Dong, G.; Lu, K.; Tan, C.; Zhou, C. & Zhou, J. (2023). Scaling relationship on learning mathematical reasoning with large language models, arXiv, Available from: https://doi.org/10.48550/arXiv.2308.01825 Accessed: 2026-10-09

Working Paper of 37th DAAAM Symposium

[14] Görisch, G. (2018). umati – universal machine tool interface OPC UA companion specification for machine tools and manufacturing systems to external communication partners, Available from: https://opcfoundation.org/markets-collaboration/umati/ Accessed: 2026-10-09
[15] Pouls, K. & Hülsmann, T.-H. (2025). Weaving a net of knowledge: exploring graph-based approaches for the applied modeling of battery cell production data and information. Procedia CIRP, Vol. 134, pp. 283-288
[16] Qwen Team. (2026). Qwen3.6-35B-A3B: agentic coding power, now open to all, Qwen Blog, Available from: https://qwen.ai/blog?id=qwen3.6-35b-a3b Accessed: 2026-10-09
[17] Dettmers, T.; Pagnoni, A.; Holtzman, A. & Zettlemoyer, L. (2023). QLoRA: efficient finetuning of quantized LLMs, arXiv, Available from: https://doi.org/10.48550/arXiv.2305.14314 Accessed: 2026-10-09
[18] Unsloth. (2026). Fine-tuning Qwen3 with Unsloth, Unsloth Blog, Available from: https://unsloth.ai/blog/qwen3 Accessed: 2026-10-09
[19] Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H. & Stoica, I. (2023). vLLM: easy, fast, and cheap LLM serving with PagedAttention, arXiv, Available from: https://doi.org/10.48550/arXiv.2309.06180 Accessed: 2026-10-09
[20] Docker Inc. (2026). Docker Engine, Docker Docs, Available from: https://docs.docker.com/ Accessed: 2026-10-09

Working Paper of 37th DAAAM Symposium