Fine-Tuned Thoughts: Leveraging Chain-of-Thought Reasoning for Industrial Asset Health Monitoring

Shuxin Lin[1] **Dhaval Patel**[1] **Christodoulos Constantinides**[2]
[1]IBM Research [2]IBM
{shuxin.lin@, pateldha@us., christodoulos.constantinides@}ibm.com

Abstract

Small Language Models (SLMs) are becoming increasingly popular in specialized fields, such as industrial applications, due to their efficiency, lower computational requirements, and ability to be fine-tuned for domain-specific tasks, enabling accurate and cost-effective solutions. However, performing complex reasoning using SLMs in specialized fields such as Industry 4.0 remains challenging. In this paper, we propose a knowledge distillation framework for industrial asset health, which transfers reasoning capabilities via Chain-of-Thought (CoT) distillation from Large Language Models (LLMs) to smaller, more efficient models (SLMs). We discuss the advantages and the process of distilling LLMs using multi-choice question answering (MCQA) prompts to enhance reasoning and refine decision-making. We also perform in-context learning to verify the quality of the generated knowledge and benchmark the performance of fine-tuned SLMs with generated knowledge against widely used LLMs. The results show that the fine-tuned SLMs with CoT reasoning outperform the base models by a significant margin, narrowing the gap to their LLM counterparts. Our code is open-sourced at: https://github.com/IBM/FailureSensorIQ.

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional proficiency in both generic and specialized domains due to their extensive pre-training on vast amounts of text data from diverse sources, that enables strong contextual understanding and reasoning. Small Language Models (SLMs), on the other hand, while perform well in common NLP tasks (Wang et al., 2024) such as text classification, sentiment analysis, their limited parameter capacity ($\leq$8B) constrains their ability to store extensive knowledge and perform complex reasoning, making them less effective for specialized domains without substantial modifications.

Recent studies on Knowledge Distillation from industries and academics have shown that Small Language Models (SLMs) hold great potential in specialized domains, such as math reasoning (Shridhar et al., 2023). The reduced computational requirements of SLMs allow for faster inference and deployment on resource-constrained devices. Also, SLMs can be fine-tuned, hosted and operated locally on computing machines, minimizing the need for sensitive user data and domain information to be exposed or leaked. The benefits of SLMs bring opportunities to the manufacturing industries, including maintenance and monitoring, process optimization, and quality control.

Failure Mode	**Sensor/Parameter Reading**				
	Power	*Speed*	*Pressure*	*Vibr.*	*Temp.*
Bearing wear		✓	✓		✓
Gear Defect			✓	✓	
Unbalance	✓				✓
Shaft Misalignment	✓	✓		✓	
Overheating			✓		✓

Table 1: Expert Knowledge: Failure Faults $\leftrightarrow$ Sensors/Parameters: ✓indicates that parameter or sensor change if failure occurs

In this paper, we focus on the adoption of small language models in Industrial Asset Health applications, which involve monitoring the health of assets using sensor data. Typically, Internet of Things (IoT) devices collect data from a variety of sensors, including those that measure temperature, power, and pressure. The sensor data is then analyzed to predict potential failures, such as "overheating", and to recommend proactive maintenance before breakdowns occur. To improve failure detection, **Failure Modes and Effects Analysis (FMEA)** used in reliability engineering is commonly applied to both sensor data and failure modes (see Table 1). This method establishes connections between an

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 20687–20700

Question Category	Example Question
Asset to Sensors	What are the sensors that could be useful in monitoring the condition of an asset?
Failure Mode to Class	Given a failure description, which failure mode class does it belong to?
Failure Mode to Sensor	To prevent an occurrence of a failure, what are the sensors that can be used to detect it early?
Sensor to Failure Mode	When anomalies are detected in a sensor reading, what failure modes can be the root cause?

Table 2: Example FMEA Questions by Category in Asset Health Monitoring Application

asset's potential failures and the monitored sensors that can signal these failures when anomalies are detected. Such analysis is a core component of industrial asset health monitoring.

Can a Large Language Model (LLM) act as a potential knowledge generator (see Table 1), offering insights into the relationships between failure modes and sensor parameters? An LLM-based workflow could enhance decision-making by providing contextual understanding and reasoning for FMEA-related questions, as outlined in Table 2.

1.1 SLM Challenges for Asset Health Monitoring

Scarcity of high-quality, labeled data. The general guidelines of FMEA in industrial asset maintenance was published by ISO Standards[1] that cover only 10 assets. For emerging technologies or newer machinery that hasn't yet gone through extensive operational lifecycles, there may be limited data or documents on failure modes. This scarcity makes it difficult to perform a thorough FMEA and predict potential failures accurately, as historical failure data simply doesn't recorded yet.

Genericness of LLM response to a specific domain. LLMs are trained on vast, general datasets that cover a wide range of topics. A domain-specific context (e.g., FMEA) is usually underrepresented in the training data and the model may generate more generic answers, lacking the deep domain expertise when reasoning.

Complexity and heterogeneity of industrial entities. Industrial systems often consist of highly complex and heterogeneous assets, each with unique sensors and failure modes. LLMs are expecting to tell the difference between *Compressor fouled* and *Compressor stalled*. Distilling expert knowledge about such varied and intricate concepts into a smaller model is difficult.

1.2 Contributions

To tackle all the challenges above, we propose a knowledge distillation framework for industrial asset health monitoring applications. The paper will cover contributions listed as followed:

1. We present a novel distillation framework designed to semi-automatically transfer Chain-of-Thought reasoning on multi-choice question answering tasks from LLMs to SLMs.

2. We introduce a novel KnowledgeGraph-inspired method to generate synthetic instructions including pseudo label for industrial domain completely without seed documents.

3. We perform a thorough qualitative evaluation of in-context learning and fine-tuning using domain knowledge generated by the framework, concluding that the student models achieve substantial performance improvements, ranging from 11% to 23%, depending on the base models.

2 Related Work

Chain-of-Thought (CoT) prompting has significantly improved interactions with large language models (LLMs), leading to better results across various datasets, such as MathQA (Wei et al., 2023). This success has inspired research focused on utilizing CoT traces from larger models to distill information, knowledge, and reasoning (Mitra et al., 2023) (Zelikman et al., 2022). The core premise of this research is that information for a target domain can be either pre-existing in the form of question-answer pairs or generated using an additional LLM.

Aligning an LLM to a specific skill has recently emerged as another area of focus. Knowledge generation from a teacher LLM typically begins by leveraging documents or seed instructions (Wang et al., 2022), (Sudalairaj et al., 2024). More recently, the Magpie-based approach has gained interest, allowing LLMs to generate alignment data in an instruction-free manner (Xu et al., 2024). However, extracting domain-specific knowledge from LLMs remains a challenge. Various attempts have been made to generate domain-specific aligned

[1] https://www.iso.org/standard/71194.html

models, such as MediTron-70B for the medical domain (Chen et al., 2023) and EntiGraph CPT for long-passage question answering on articles (Yang et al., 2024), among others. These approaches typically rely on large-scale corpora with billions of tokens or begin with a few million tokens to generate additional synthetic data using a teacher model. However, they largely overlook the Industry 4.0 domain. One primary reason for this may be the lack of a qualitatively constructed validation dataset such as other domains PubMedQA (Jin et al., 2019)) and chemical safety (e.g., ChemSafety (Zhao et al., 2024)) as examples.

3 Methodology

We present our proposed Knowledge Distillation framework in Figure 1, which transfers FMEA knowledge from a teacher model to a student model. The process for generating synthetic multiple-choice question answering (MCQA) data using Chain of Thought (CoT) prompting is described step-by-step in the following sections. Notably, the generation process is seed-free, meaning it does not rely on an initial dataset.

3.1 KG-based Instruction Generation

In manufacturing, Knowledge Graphs (KGs) are commonly used to organize relationships and derive domain knowledge (PCA Reference Data and Services, 2025). Each edge in a knowledge graph can be represented as a collection of relational triplets (s, o, r), where s denotes the subject entity, o represents the object entity, and r defines the relation between these entities. Within the context of FMEA, three critical relations, as identified by domain experts, are:

- *mountedOn*: indicates that a physical **sensor** is mounted on an industrial **asset** for the purpose of monitoring or tracking.
- *experiencedBy*: indicates that a **failure mode** is experienced by an industrial **asset**.
- *detectedBy*: indicates that a **failure mode** can be detected by a **sensor** associated with the industrial asset.

When the subject (s) or object (o) of a triplet $T(s, o, r)$ is omitted, the remaining element becomes a seed for generating LLM instructions. Table 3 illustrates an example of how such a seed can be transformed into a question. To facilitate the generation of these questions, we have designed a variety of handcrafted seed templates for each node type. In total, we have 23 distinct seed templates (See Appendix 11), which covers all four question categories, as shown in Table 2.

triplet (T)	(____, Wind Turbine, mountedOn)
question (Q)	Which sensor is mounted on Wind Turbine for monitoring performance of the asset?

Table 3: Mapping: a seed to a natural language question

3.2 Options Generation

Generating the correct option and multiple distractors is crucial for the effectiveness of a question (Q). Since the data used is entirely synthetic, we rely on teacher LLMs to generate potential answer choices for a given Q. Let U represent the universal set of available options in our study, where the content is determined by the type of node, which is omitted at the time of generating Q. We then prompt the LLM to rank candidate options based on their **correctness criteria** and extract the top K options from the universal set U (as specified in the prompt Appendix Table 12). For each of these K options, we treat each as a correct answer and generate a distinct problem with a slightly rephrased instruction. This approach not only introduces diversity in the correct answers but also reduces the risk of bias by preventing reliance on a single, potentially erroneous, option.

To generate distractors, we retain the bottom $2K$ options (i.e., less relevant) from the generated response which are those with the least correctness and use them as candidates for the distractors. To further avoid patterns, we randomize the positions of the answers. In summary we generated K instances of original question Q with option.

The use of correctness criteria is our contribution. For example, for a question in Table 3, "the sensors that can be installed on asset" is an example of criteria in order to obtain the relevant set of K options from U. We have defined criteria for each question category in Table 2. We set K = 5.

3.3 Pseudo Ground Truth Labelling

After constructing the prompt with the instruction, question q, and the options denoted as *A, B, C, D, E, ...*, the LLM is prompted to generate an answer. The labeling process involves selecting the best

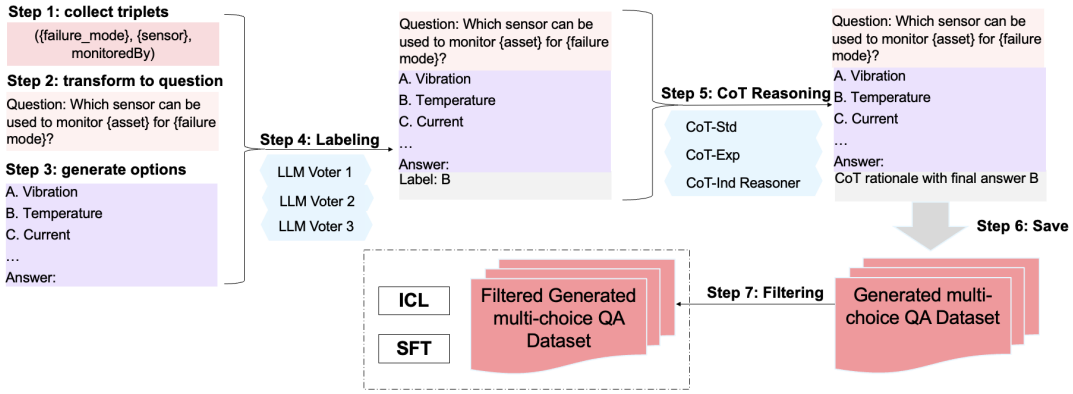

Figure 1: Proposed Knowledge Distillation Framework: Workflow leading to Fine-Tune

option from the available choices. We implement a **majority voting** mechanism using three LLMs (Mixtral Large, Llama-3.1 405b, and ChatGPT) to produce the final answers.

For each multiple-choice question generated, we assign a label if there is clear consensus among the voters. If two voters agree on a particular answer, we evaluate the LLM generated confidence scores and assign a label only if both confidence scores exceed 90. We have used a "self-guess" prompt to interact with LLM for generating final answer (Appendix - Table 13).

3.4 Rationale Generation

To distill knowledge from Large Language Models (LLMs), we implement Chain of Thought (CoT) based prompting. CoT prompting aids reasoning in multiple-choice question answering by breaking down the process into step-by-step logical steps as well as taking into account the available options. This approach reduces reliance on shallow heuristics and enhances accuracy when tackling complex problems. We selected three variations of trigger statements, as shown in Table 4. The question, along with its options, is used with the CoT trigger to generate an LLM-based answer and its rational. If the LLM-generated answer matches the pseudo-label, a rationale is subsequently applied.

CoT Stype	Trigger Statement
Standard	Let me think step by step
Inductive	Let me think step by step as a reliability engineer
Expert	Let's use step by step inductive reasoning, given the domain specific nature of the question

Table 4: Chain-Of-Thought Trigger Statements

For **CoT Expert**, the approach mimics expert-level thinking by structuring reasoning around engineering principles and best practices in FMEA. This method aids the model in prioritizing key factors such as failure modes, causes, effects, and detection methods. On the other hand, **CoT Inductive** allows LLMs to derive conclusions by identifying patterns and relationships within the text. This is particularly useful for handling unfamiliar scenarios or edge cases (e.g., unknown sensors or failure modes), where expert knowledge alone may not suffice. Research suggests that these two CoT variations could potentially outperform **CoT Standard** in certain situations (Liévin et al., 2023).

3.5 Quality Filtering

We apply several heuristics from (Xu et al., 2024) to select high-quality generation for down-streaming fine-tuning. Here are the proposed metrics and the empirical thresholds:

- Input and Output Length: the total number of characters combining input and output. We filter those generations exceeding max context length of LLMs.
- Minimum Neighbor Distance: The embedding distance to the nearest neighbor. Filter the lowest 5% of generations based on scores.
- Input Difficulty: LLM-as-a-Judge to determine the difficulty of question on 5 scales (very easy, easy, medium, hard and very hard). Remove very easy generations.
- Output Quality: LLM-as-a-Judge to determine the quality of output on 5 scales (very poor, poor, average, good and excellent). Remove very poor and poor generations.

Although these quality control methods effectively filter out clearly flawed generations, it remains necessary to assess the quality of the generated data using the teacher model, as discussed in Section 4.2.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of the FMEA-related QA data generated by `FailureSensorIQ` (Appendix: Table 14). `FailureSensorIQ` is a new dataset we introduced to the community for testing LLM's ability to reason about sensor and failure mode relation. Our goal is to enhance the student model's performance by leveraging knowledge distilled from the teacher model. In other words, we aim to improve SLMs so they can achieve reasoning capabilities comparable to LLMs in the scope of FMEA.

4.1 Generated Data Statistics

We use **Mistral Large** as the teacher model for CoT reasoning generation and rely on **Mistral Large**, **Llama-3.1-405B-Instruct**, and **GPT-4** as the models for majority voting. The choice of teacher models follows prior research findings (Constantinides et al., 2025). We have collected 54 candidates for assets, 66 for sensors, and 59 for failure modes from ISO standards (iso, 2018; ISO, 2016). For each seed question, we generate 5 variations (with $K = 5$), each featuring different rephrased questions and correct options. By applying three different **Chain of Thought (CoT)** prompts for reasoning and filtering, the final number of generations is $6.2k$, $6.1k$, and $6.2k$ for **CoT-Expert**, **CoT-Inductive**, and **CoT-Standard**, respectively. We denote the generated datasets as $D_{\text{gen}}^{\text{CoT-Std}}$, $D_{\text{gen}}^{\text{CoT-Exp}}$, and $D_{\text{gen}}^{\text{CoT-Ind}}$. The distribution of assets in the generated data is uneven, with the percentage of each asset ranging from 1% to 5%.

4.2 Generated Data Factuality Analysis

To assess the factual consistency of the teacher model-generated data, we conducted an evaluation using an extended version of **FActScore** (Min et al., 2023), a recent metric that measures the percentage of atomic facts from LLM generations supported by a trusted source, e.g. Wikipedia, Arxiv. We sampled 700 examples from $D_{\text{gen}}^{\text{CoT-Exp}}$, ensuring even distribution across 54 industrial asset types. We run FActScore script with Llama-4-Maverick.

The results in Table 5 indicate that the teacher-generated data achieves a FActScore of 70.8%, which is slightly higher than the 69.8% achieved by the FailureSensorIQ benchmark, which means nearly 70% of facts of our teacher LLM generations are backed by trusted source. Additionally, the responding rate remains high at 89.6%, reinforcing the usability of the generated samples, as the model provides substantive responses rather than abstaining.

	Responding (%)	**FActScore** (%)
FailureSensorIQ	94.7	69.8
Generated data	89.6	70.8

Table 5: FActScore: Factuality Comparison Between FailureSensorIQ and Teacher Generated Data

4.3 Benchmark Dataset

We select the **FailureSensorIQ**[2] dataset (Constantinides et al., 2025) derived and curated from ISO Standards which contains 2667 multi-choice single-true questions with ground truths. The dataset is designed to access the ability to reason and understand the relations between sensor/parameter and failures/faults for assets in Industry 4.0. This dataset covers 10 assets and Table 6 lists some distributional information of the dataset.

Distribution Type	**Distribution Value**
Asset Distribution	Electric Motor (234), Steam Turbine (171), Aero Gas Turbine (336), Industrial Gas Turbine (240), Pump (152), Compressor (220), Reciprocating IC Engine (336), Electric Generator (234), Fan (200), Power Transformer (544)
Option Distribution	Option A: 752, Option B: 729, Option C: 491, Option D: 408, Option E: 208
Distribution of Questions	2-options: 487, 3-options: 266, 4-options: 389, 5-options: 1525

Table 6: Distribution Types and Their Values for FailureSensorIQ MCQA dataset

4.4 Evaluation Metrics

Accuracy is the most commonly used metric for tests on MCQA tasks. However recent study (Li et al., 2024) demonstrates a collection of metrics

[2] https://huggingface.co/datasets/ibm-research/assetopsbench

that comprehensively examine the performance of LLMs on MCQA tasks based on response pattern. The proposed metrics are the percentage of LLM responses patterns as followed with denotation: (1) $P_{\textbf{single-correct}}$ - single correct selection (accuracy); (2) $P_{\textbf{single-wrong}}$ - single wrong selection; (3) $P_{\textbf{invalid}}$ - invalid selection (none of answers in the response); (4) $P_{\textbf{mul-correct}}$ - multiple selections with the correct one; (5) $P_{\textbf{mul-wrong}}$ - multiple selections w/o the correct one.

4.5 In-Context Learning

In-context learning (ICL) is a technique that allows Large Language Models (LLMs) to adapt to new tasks during inference by providing a prompt containing task examples. Recently, with the increasing context length of LLMs ($\geq$ 128K), researchers have explored the impact of many-shot learning versus fine-tuning the model (Agarwal et al., 2024). In this experiment, we evaluate the effects of incorporating examples from D_{gen} into the prompt when performing inference on the benchmark dataset **FailureSensorIQ**. We experimented with three options: (1) **Zero-shot learning**, which tests LLMs with no external knowledge during inference; (2) **Few-shot learning**, using 5 expert-curated examples. These examples are carefully handpicked to demonstrate the FMEA task, including reasoning processes. The quality of the examples is evaluated by domain experts; (3) **Many-shot learning** using examples from $D_{\text{gen}}^{\text{CoT-Exp}}$.

In the case of many-shot learning, we use the **all-MiniLM-L6-v2** model (Reimers and Gurevych, 2020) to compute the embeddings (vector representations) of each question in the synthetic data. During inference, we also convert each query from the benchmark dataset into an embedding using the same embedding model. We utilize cosine similarity to select the top N relevant generations from $D_{\text{gen}}^{\text{CoT-Exp}}$ to use as context for the query. Here, $N \in \{1, 5, 10, 20\}$. It is important to note that the out-of-the-box **all-MiniLM-L6-v2** is specifically optimized for speed and memory efficiency.

We randomly sample 500 questions from the 2667-question **FailureSensorIQ** dataset. The number of correct inferences made by LLMs from three open-source LLM families : Llama, Mistral, and Granite, are shown in Table 7. The results demonstrate that performance generally improves when transitioning from zero-shot to few-shot learning. **Llama-3.1-70B-Instruct** shows a substantial increase, reaching 303 correct inferences with 5 curated examples, while **Mistral-Large-Instruct** achieves 320 correct inferences, maintaining strong performance across various few-shot setups. Larger models tend to consistently outperform smaller models within the same family, primarily due to their larger parameter space. Larger language models also tend to have longer context lengths, which enables them to consume more context and knowledge from additional examples, thereby boosting performance. However, the performance gains begin to plateau when the number of generated examples exceeds 20. Interestingly, larger model tend to perform with less samples whereas the smaller model tend to perform better with more examples.

The performance improvement from zero-shot to curated few-shot learning is more significant than the improvement from curated few-shot to generation-based few-shot. This is because the model effectively "learns" the task when adding curated few-shot examples, whereas the transition from curated few-shot to generation-based few-shot does not introduce a substantial learning step. Additionally, many-shot learning using $N = 5$ generally outperforms curated samples based example, clearly demonstrating the advantage of in-context learning, where samples are dynamically selected based on the input query.

In conclusion, the generated data from our proposed framework provides a noticeable improvement in contextual understanding during the inference. However, due to model saturation and the noise introduced by potentially low-quality generations, in-context learning may not fully capitalize on the distilled knowledge from the teacher model.

4.6 Model Fine Tuning

With Chain of Thought (CoT) knowledge distillation, the goal of fine-tuning the student model is not only to produce accurate predictions but also to internalize the reasoning behind those predictions in asset health monitoring. Our experimental setup uses **QLoRA** with 4-bit precision for model fine-tuning. The model leverages *FlashAttention2* for efficient attention computation and supports mixed precision training with *bf16* and *tf32*. The maximum sequence length is set to 2048 tokens, and packing is enabled to optimize memory usage. The training runs for 1 epoch, with a batch size of 8 and gradient accumulation over 2 steps. The learning rate is set to 2.0×10^{-4}, with a constant learning rate scheduler and a warmup ratio of 0.1 to gradually ramp up the learning rate. Training is

LLM	Zero-Shot	Few-Shot	Many-Shot (generation-based)			
		5 curated	$N = 5$	$N = 10$	$N = 20$	$N = 50$
Llama-3.1-70B-Instruct	249	303	**316**	304	313	310
Llama-3.1-405B-Instruct	251	313	**317**	315	316	**317**
Mistral-Large-Instruct	248	298	**320**	315	310	315
Llama-3.2-90B-Vision-Instruct	249	300	317	304	**318**	312
Ministral-8B-Instruct	218	245	262	275	275	**278**
Llama-3.1-8B-Instruct	220	277	288	292	**294**	287
Granite-3.1-8B-Instruct	187	196	206	206	209	**210**

Table 7: Correctness of LLM inference on 500 FailureSensorIQ questions. This comparison examines the zero-shot baseline against few-shot/many-shot learning, using expert-curated examples vs. examples from $D_{\text{gen}}^{\text{CoT-Exp}}$.

Model	Experiment Settings		Evaluation Scores				
(baselines)	Synth. Data	Prompting	$P_{\text{single-correct}}$	P_{invalid}	$P_{\text{mul-correct}}$	$P_{\text{single-wrong}}$	$P_{\text{mul-wrong}}$
Llama-3.1-405B-Instruct	N/A	CoT Std	0.5126	0.0019	0.1691	0.258	0.0585
Llama-3.1-8B-Instruct	N/A	direct	0.4012	0.012	0.1991	0.3048	0.0829
Mistral-Large-Instruct	N/A	direct	0.5009	0.0244	0.186	0.2295	0.0592
Ministral-8B-Instruct	N/A	direct	0.264	0	0.4113	0.1894	0.1354
Granite-3.1-8B Instruct	N/A	direct	0.2411	0.0015	0.4046	0.1916	0.1612
FT on Llama-3.1-8B-Instruct	$D_{\text{gen}}^{\text{CoT-Std}}$	direct	0.5111	0.0071	0.1095	0.3116	0.0607
	$D_{\text{gen}}^{\text{CoT-Exp}}$	direct	0.4698	0.0049	0.0979	0.375	0.0525
	$D_{\text{gen}}^{\text{CoT-Ind}}$	CoT Ind	0.4387	0.0165	0.1365	0.3168	0.0915
FT on Ministral-8B-Instruct	$D_{\text{gen}}^{\text{CoT-Std}}$	direct	0.4402	0	0.144	0.3573	0.0585
	$D_{\text{gen}}^{\text{CoT-Exp}}$	direct	0.4623	0.0004	0.1301	0.3495	0.0577
	$D_{\text{gen}}^{\text{CoT-Ind}}$	direct	0.4938	0	0.1537	0.2913	0.0611
FT on Granite-3.1-8B-Instruct	$D_{\text{gen}}^{\text{CoT-Std}}$	CoT Std	0.3813	0.0427	0.1552	0.3142	0.1065
	$D_{\text{gen}}^{\text{CoT-Exp}}$	direct	0.4083	0	0.2583	0.2163	0.117
	$D_{\text{gen}}^{\text{CoT-Ind}}$	CoT Std	0.4062	0.006	0.1407	0.3952	0.0519

Table 8: Evaluation scores of base models and fine-tuned models on FailureSensorIQ dataset. Column Synth. Data represents the generated dataset used for fine-tuning, and Column Prompting shows the best prompting technique with the highest accuracy. "N/A" indicates that the experiment does not involve fine-tuning.

conducted on 2 NVIDIA A100 80GB GPUs. We provide additional discussion on the selection of fine-tuning specifications in the Appendix A.1.

The three student models focused on in this experiment are all 8B small language models: **Llama 3.1 8B Instruct**, **Mistral 8B Instruct**, and **Granite 3.1 8B Instruct**. We compare the performance of these student models after knowledge distillation with the baseline models, as shown in Table 8.

4.6.1 Baselines

Llama-3.1-405B-Instruct model achieves the highest $P_{\text{single-correct}}$ (**0.51**) among the baselines. **Mistral-Large-Instruct** model also performs well with $P_{\text{single-correct}} = 0.50$, but it has a higher $P_{\text{invalid}} = 0.024$, suggesting it generates more invalid responses. Both **Mistral-8B-Instruct** and **Granite-3.1-8B-Instruct** perform the worst among the baselines, with much higher $P_{\text{mul-correct}}$ values, indicating they often predict multiple correct answers rather than providing a single precise answer.

4.6.2 Impact of Fine-Tuning

In general, fine-tuning on $D_{\text{gen}}^{\text{CoT-Std}}$, $D_{\text{CoT-Exp}}^{\text{CoT-Std}}$, and $D_{\text{gen}}^{\text{CoT-Ind}}$ leads to notable changes in performance across all three student models, with 0.11, 0.23, 0.16 $P_{\text{single-correct}}$ gain respectively. Fine-tuning Llama-3.1-8B-Instruct on $D_{\text{gen}}^{\text{CoT-Std}}$ shows a compa-

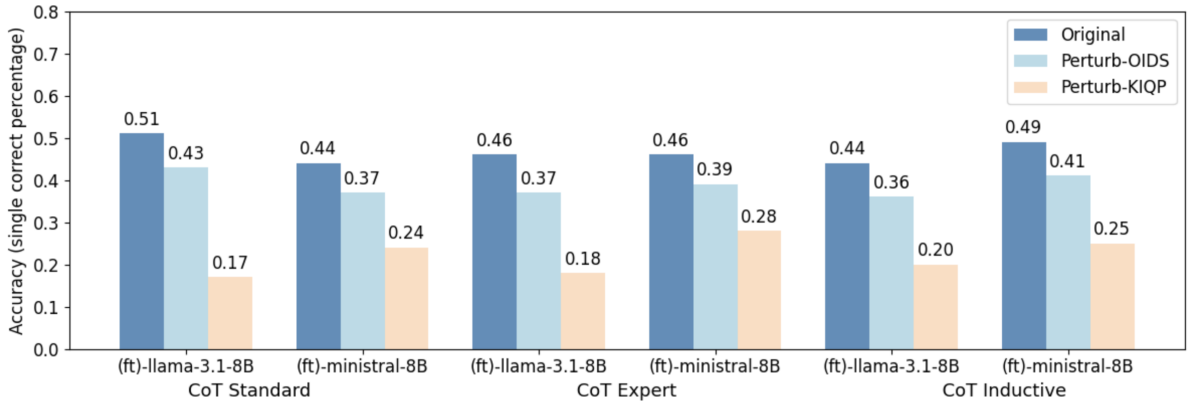

Figure 2: Real knowledge capacity measurement with two knowledge-invariant perturbations: Option ID Shifting (Perturb-OIDS), and Knowledge-Invariant Question Paraphrasing (Perturb-KIQP).

rable $P_{\text{single-correct}} = 0.51$ to Llama-3.1-405B, indicating the SLM, when fine-tuned, achieves a level of performance that is close to that of its larger counterpart. In terms of the choice of CoT style during distilling large models, there is not clear winner among the three. We could argue that CoT-Expert has a slight edge since it has a low change to generate invalid responses, and maintain higher scores in both $P_{\text{single-correct}}$ and $P_{\text{mul-correct}}$.

4.6.3 Impact of CoT Fine-Tuning

During inference, we apply prompt engineering to each queries including direct prompting, and three CoT variations listed in Table 4. Then we record the best prompting style with highest $P_{\text{single-correct}}$ in Column *Prompting* in Table 4. We notice that direct prompting is the best one in most experiment settings. This proves the direct prompting is effective enough after student models learn the CoT-style reasoning via knowledge distillation.

Observations on Failure Types. P_{invalid} remains low across most models, except for the baseline Mistral-Large-Instruct and fine-tuned Llama-3.1-8B-Instruct (CoT-Inductive). Fine-tuning generally reduces $P_{\text{mul-correct}}$, meaning models are more confident in selecting a single correct answer instead of multiple. $P_{\text{single-wrong}}$ increases for fine-tuned models, suggesting fine-tuning makes models more decisive but also slightly increases the risk of choosing incorrect answers.

4.7 Ablation Study

We conducted two ablation studies that isolates the impact of individual components in the framework. Specifically, we evaluate:

1. **Without rationale** (direct answer only): In this setting, we fine-tuned the student model using only the final answers from the teacher without any accompanying chain-of-thought (CoT) rationales.

2. **Incorrect pseudo-labeling** (mismatched answer-rationale pairs): Here, we deliberately introduced noise into the pseudo-labeling process by pairing rationales with incorrect final answers where the rationale and answer do not align.

We compare the $P_{\text{single-correct}}$ of the two experiments with the baseline (Table 9). All fine-tuning is conducted with QLoRA and the rationale generation is based on CoT Standard. The ablation study results reveal the dramatic performance degradation under incorrect pseudo-labeling, with drops ranging from 13.3% to a severe 21.4%. This suggests that students learn not just the reasoning patterns but also the consistency between thinking process and outcome. Interestingly, the impact of removing rationales varies significantly across models: while Llama-3.1-8B shows a moderate 5.2% performance drop without rationales, Ministral-8B actually improves slightly. This heterogeneous response suggests the capacity to effectively utilize CoT reasoning during fine-tuning is model-dependent. For practical applications, these findings suggest practitioners should first evaluate model performance with direct answer fine-tuning before implementing more computationally expensive CoT reasoning approaches.

4.8 Perturbation Study

One significant challenge in evaluating the performance of Large Language Models (LLMs) on

Model	Baseline	Without Rationale	Incorrect Pseudo-labeling
Llama-3.1-8B	0.5111	0.4593	0.3780
Ministral-8B	0.4402	0.5182	0.2265
Granite-3.1-8B	0.3813	0.3296	0.1429

Table 9: $P_{\text{single-correct}}$ of Ablation Studies vs. Baseline

multiple-choice question answering (QA) benchmarks is determining how accurately the scores reflect the model's true reasoning ability and knowledge capacity. To address this issue, a recent evaluation framework, PertEval (Li et al., 2024), introduces a suite of tests that apply various knowledge-invariant perturbations to benchmarks. We utilize two of these perturbations to assess the performance of our fine-tuned student models:

1. **Option ID Shifting (OIDS)**: This technique substitutes the original option IDs (A/B/C/D/E) with new identifiers (P/Q/R/S/T). OIDS explores potential LLM selection biases in question answering, a phenomenon observed in certain models. By applying this perturbation to the dataset, we can assess whether the choice of option IDs influences the model's performance.

2. **Knowledge-Invariant Question Paraphrasing (KIQP)**: The questions in the original FailureSensorIQ dataset are concise and straightforward. We apply paraphrasing using Llama-3.0-70B to reword these questions while preserving their intended meaning.

These two perturbations allow us to evaluate our models on both format and content levels. In Figure 2, we compare the $P_{\text{single-correct}}$ scores of the fine-tuned Llama-3.1-8B-Instruct and Ministral-8B-Instruct models across three datasets: the original FailureSensorIQ, Perturb-OIDS, and Perturb-KIQP. We observe a significant performance drop after applying the perturbations. Specifically, fine-tuned Llama-3.1-8B-Instruct shows greater instability under perturbations, with an average drop of 0.19 per perturbation ($P_{\text{before perturb}} - P_{\text{after perturb}}$), compared to a 0.14 drop for Ministral-8B-Instruct. The KIQP perturbation leads to a more substantial decline in reasoning ability than the OIDS perturbation, indicating that LLMs often rely on memorized patterns and heuristics rather than a deep understanding of the context.

We find that the KIQP perturbation not only alters the questions but also affects the structure of the MCQA task itself, as it causes the options to appear before the question. Additionally, when applying the base model for the same set of experiments, the accuracy of the Perturb-KIQP dataset was notably lower (close to 0.14 for Llama-3.1-8B-Instruct). In summary, even with fine-tuning, small language models (SLMs) improved their grasp on key concepts (e.g., failure modes, sensors). The performance drop across the three types of CoT fine-tuning remains similar, underscoring the challenges LLMs face when the task context is altered.

5 Conclusion

We present an innovative knowledge distillation framework designed for asset health monitoring tasks. Our framework generates high-quality synthetic data using LLMs without relying on initial knowledge documents for assets. By fine-tuning small language models (SLMs) with this domain-specific data, the models exhibit reduced hallucination, improved reasoning accuracy, and enhanced consistency in responses. CoT distillation on multi-choice QA tasks further strengthens the SLM's contextual understanding of industrial entities. The low cost of SLM QLoRA fine-tuning (less than 1 hour per experiment, under 4GB adapter for 8B models) makes it a practical solution for adapting models to industry tasks while maintaining scalability and efficiency. Despite these advantages, challenges remain, particularly in handling perturbations, suggesting that future work could focus on perturbation-aware training or incorporating more diverse perturbation scenarios into the synthetic data generation process.

Limitations

Our approach leverages a larger teacher model to generate rationales and question-answer pairs for fine-tuning a smaller student model. Given the domain-specific nature of the task, ensuring the factual accuracy and faithfulness of the generated content is essential. However, large-scale human validation is infeasible, and existing automated methods for verifying scientific truthfulness remain limited in both reliability and domain coverage. Consequently, the student model may inherit subtle inaccuracies from the teacher model, particularly in cases involving less-documented or highly specialized knowledge.

Additionally, this work focuses on three prototypical and high-frequency Failure Mode and Effect Analysis (FMEA) relations : mountedOn, experiencedBy, and detectedBy, as an initial proof of concept to demonstrate the viability of CoT-based distillation in this context. While our framework is modular and readily extensible to accommodate more complex relationships, the current limited relational coverage may affect generalizability to the full FMEA relational space.

Future work should focus on developing more robust, domain-sensitive evaluation techniques for low-resource and high-precision scientific applications, as well as expanding the framework to encompass a broader range of FMEA relationships to enhance the model's comprehensive understanding of complex industrial systems.

References

2016. Iso 14224:2016 - petroleum, petrochemical and natural gas industries — collection and exchange of reliability and maintenance data for equipment. Last reviewed and confirmed in 2022; remains current.

2018. Condition monitoring and diagnostics of machines — general guidelines. Geneva, Switzerland. International Organization for Standardization (ISO). This publication was last reviewed and confirmed in 2023. Therefore, this version remains current.

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. 2024. Many-shot in-context learning. *Preprint*, arXiv:2404.11018.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. 2023. Meditron-70b: Scaling medical pretraining for large language models. *Preprint*, arXiv:2311.16079.

Christodoulos Constantinides, Dhaval Patel, Shuxin Lin, Claudio Guerrero, Sunil Dagajirao Patil, and Jayant Kalagnanam. 2025. Failuresensoriq: A multi-choice qa dataset for understanding sensor relationships and failure modes. *Preprint*, arXiv:2506.03278.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Jiatong Li, Renjun Hu, Kunzhe Huang, Yan Zhuang, Qi Liu, Mengxiao Zhu, Xing Shi, and Wei Lin. 2024. Perteval: Unveiling real knowledge capacity of llms with knowledge-invariant perturbations. *Preprint*, arXiv:2405.19740.

Valentin Liévin, Christoffer Egeberg Hother, Andreas Geert Motzfeldt, and Ole Winther. 2023. Can large language models reason about medical questions? *Preprint*, arXiv:2207.08143.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *Preprint*, arXiv:2305.14251.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason. *Preprint*, arXiv:2311.11045.

PCA Reference Data and Services. 2025. Pca rds system. Accessed: 2025-05-19.

Nils Reimers and Iryna Gurevych. 2020. all-minilm-l6-v2: A highly efficient transformer model for sentence embedding. `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`. Accessed: 2025-05-19.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073, Toronto, Canada. Association for Computational Linguistics.

Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D. Cox, and Akash Srivastava. 2024. Lab: Large-scale alignment for chatbots. *Preprint*, arXiv:2403.01081.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. 2024. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *Preprint*, arXiv:2411.03350.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *ArXiv*, abs/2406.08464.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candès, and Tatsunori Hashimoto. 2024. Synthetic continued pretraining. *Preprint*, arXiv:2409.07431.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Preprint*, arXiv:2203.14465.

Haochen Zhao, Xiangru Tang, Ziran Yang, Xiao Han, Xuanzhi Feng, Yueqing Fan, Senhao Cheng, Di Jin, Yilun Zhao, Arman Cohan, and Mark Gerstein. 2024. Chemsafetybench: Benchmarking llm safety on chemistry domain. *Preprint*, arXiv:2411.16736.

A Appendix

A.1 Fine Tuning: Full FT vs. LoRA vs. QLoRA

In this section, we compare the performance of three fine-tuning approaches: Full Fine-Tuning (Full FT), Low-Rank Adaptation (LoRA), and Quantized LoRA (QLoRA), on the tasks described in Section 4.6. We further discuss the rationale behind ultimately selecting QLoRA.

Our results in Table 10 show that Full FT frequently degrades performance relative to LoRA, largely due to a high proportion of invalid or incomplete responses. This degradation likely arises from the model overfitting to surface-level patterns in the training data while failing to preserve the intended reasoning behaviors.

Both LoRA and QLoRA preserve response quality, producing substantially more valid and reasoned outputs, with comparable performance on FailureSensorIQ benchmark tasks. Between the two, LoRA and QLoRA achieve similar accuracy, but QLoRA provides real-world advantages, including a lower memory footprint, faster training time, and more efficient inference.

Table 10: Performance Comparison Across Different Fine-tuning Methods

LLM	**Prompting Technique**	**Full FT:** $P_{\textbf{single-correct}}$	**Full FT:** $P_{\textbf{invalid}}$	**LoRA:** $P_{\textbf{single-correct}}$	**QLoRA:** $P_{\textbf{single-correct}}$
Llama-3.1-8B	CoT standard	0.1005	0.4229	0.4796	0.5111
	CoT expert	0.1080	0.4421	0.4743	0.4698
	CoT inductive	0.0829	0.4631	0.4747	0.4387
Ministral-8B	CoT standard	0.0937	0.5692	0.4214	0.4402
	CoT expert	0.0960	0.5816	0.5088	0.4623
	CoT inductive	0.0956	0.5681	0.4627	0.4938
Granite-3.1-8B	CoT standard	0.2343	0.0907	0.4143	0.3813
	CoT expert	0.2373	0.0900	0.4526	0.4083
	CoT inductive	0.2295	0.0885	0.4128	0.4062

Seed Templates for Sensor and Failure Mode Inquiry	
Category	**Example Templates**
Asset to Sensor	Which sensor could be installed on this asset {asset_class}?
	Is there a sensor that can be mounted on this asset {asset_class}?
	Can you identify a sensor that could work with this asset {asset_class}?
	Which sensor is recommended to track performance and identify anomalies for this asset {asset_class}?
Asset to Failure Mode	Which is the most common failure mode associated with the asset {asset_class}?
	Which failure mode should be monitored for the asset {asset_class}?
	Which failure mode can occur in the asset {asset_class} during operation?
	Which is the failure scenario that the asset {asset_class} might encounter?
	Which failure mode is most likely to occur with the asset {asset_class}?
Sensor to Failure Mode	In the context of {asset_class}, which failure mode is most relevant when {relevant_sensor} shows abnormal readings?
	Which is the most relevant failure mode for {asset_class} if {relevant_sensor} exhibits abnormal readings?
	Which failure mode should be considered for {asset_class} when abnormal readings are detected by {relevant_sensor}?
	When {relevant_sensor} in {asset_class} displays abnormal readings, which failure mode is the most applicable?
	For {asset_class}, what is the key failure mode when {relevant_sensor} has abnormal readings?
	What is the most likely failure mode for {asset_class} when {relevant_sensor} indicates abnormal behavior?
Failure Mode to Sensor	Which sensor can be used to monitor asset {asset_class} for failure mode {relevant_failure_mode}?
	What sensor is suitable for monitoring {asset_class} to detect {relevant_failure_mode}?
	What sensor can be utilized to monitor {asset_class} for signs of {relevant_failure_mode}?
	Which sensor is best suited to monitor {asset_class} for the occurrence of {relevant_failure_mode}?
	In an {asset_class}, which sensor is designed to track {relevant_failure_mode}?
	In the context of {asset_class}, which sensor can help in identifying {relevant_failure_mode}?
	Which sensor would you recommend for monitoring {asset_class} to detect {relevant_failure_mode}?
	Which sensor can effectively monitor {asset_class} for potential {relevant_failure_mode}?

Table 11: Template for Question Generation

Question template	Divide the following choices into two groups. First group is {relevance criteria}. Second group is {irrelevance criteria}. Here are a list of choices: {choices}. Output the first group in the first line. Output the second group in the second line. Format of the output should be: First group: ["choice1", "choice2", "choice3", ...] Second group: ["choice4", "choice5", "choice6", ...]
Relevance criteria	failure modes that are the most common failure modes associated with {asset class} sorted by relevancy
Irrelevance criteria	the failure modes that are most unlikely to occur with {asset class} sorted by unlikelihood
Choices	fail to start, failure to stop, ..., bearing wear, unbalance, ...

Table 12: Example of question template that groups and ranks the options in options generation process.

```
Here is the question:
Question: {question}
{options}

Please provide your best guess for the answer to the following
question and include a confidence score between 0 to 100, an
explanation, and a rationale for your answer in the following
JSON format:
“‘json
{
"answer": "Your answer here",
"explanation": "Your explanation here",
"confidence_score": "Your score here",
"rationale": "Your answer here",
}
“‘
```

Table 13: Self-guess prompting to extract confidence score and rationale from the response

Failure Mode to Sensor	For electric motor, if a failure event rotor windings fault occurs, which sensor out of the choices is the most relevant sensor regarding the occurrence of the failure event? A. partial discharge B. resistance C. oil debris D. current E. voltage Answer: D
Sensor to Failure Mode	Which failure mode is most relevant for steam turbine if there are abnormal readings from coast down time? A. unequal expansion B. misalignment C. bearing damage D. unbalance E. damaged labyrinth Answer: C

Table 14: Examples of FailureSensorIQ: a multi-choice question-answering dataset for failure mode and sensor relations