Self-Supervised Learning Architectures for Intelligent Edge Devices in Industrial IoT
Endüstriyel IoT'de Akıllı Kenar Cihazları için Kendi Kendini Denetleyen Öğrenme Mimarileri

Ankita Sappa
College of Engineering, Wichita State University, United States
Email: ankita.sappa@gmail.com

***Abstract*—Industrial IoT (IIoT) systems consist of a myriad of sensors that generate considerable amounts of data that must be analyzed intelligently and in a timely fashion at the edge. The primary challenges in deploying machine learning models on edge devices are the limited computational power and the lack of sufficient labeled data. This work tackles the problem of self-supervised learning (SSL) on resource-constrained intelligent edge devices, solving the problems of resource limitation and annotation bottleneck. The architecture incorporates domain-specific pretext tasks for industrial sensor modalities such as vibration, pressure, and temperature to construct embedding features without requiring human-labeled data. We deploy and evaluate the model within a heterogeneous IIoT testbed that consists of real-world edge devices and measure performance based on embedding quality, accuracy of downstream tasks, energy consumption, and latency. The results show that the proposed approach outperforms baseline supervised and semi-supervised federated learning models in sparse label conditions while achieving near real-time inference and low power consumption. This work assists in the deployment of scalable self-supervised intelligence at the edge for predictive maintenance, anomaly detection, and context-aware automation in future industrial systems.**

Keywords*—*Self-Supervised Learning, Industrial IoT, Edge Intelligence, Embedded AI Systems.

***Özetçe*— Endüstriyel IoT (IIoT) sistemleri, uçta akıllıca ve zamanında analiz edilmesi gereken önemli miktarda veri üreten sayısız sensörden oluşur. Makine öğrenimi modellerini uç aygıtlara dağıtmanın temel zorlukları, sınırlı hesaplama gücü ve yeterli etiketli verinin olmamasıdır. Bu çalışma, kaynak kısıtlamalı akıllı uç aygıtlarda kendi kendini denetleyen öğrenme (SSL) sorununu ele alarak kaynak sınırlaması ve açıklama darboğazı sorunlarını çözer. Mimari, insan etiketli veriler gerektirmeden gömme özelliklerini oluşturmak için titreşim, basınç ve sıcaklık gibi endüstriyel sensör modaliteleri için alan-özel bahane görevlerini içerir. Modeli, gerçek dünya uç aygıtlarından oluşan heterojen bir IIoT test yatağında dağıtır ve değerlendiririz ve gömme kalitesine, aşağı akış görevlerinin doğruluğuna, enerji tüketimine ve gecikmeye göre performansı ölçeriz. Sonuçlar, önerilen yaklaşımın seyrek etiket koşullarında temel denetlenen ve yarı denetlenen federasyon öğrenme modellerinden daha iyi performans gösterdiğini, neredeyse gerçek zamanlı çıkarım ve düşük güç tüketimi sağladığını göstermektedir. Bu çalışma, gelecekteki endüstriyel sistemlerde öngörücü bakım, anormallik tespiti ve bağlam farkında otomasyon için uçta ölçeklenebilir, kendi kendini denetleyen zekanın dağıtımına yardımcı olur.**

Anahtar Kelimeler*—*Kendi Kendini Denetleyen Öğrenme, Endüstriyel IoT, Uç Zeka, Gömülü Yapay Zeka Sistemleri.

I. INTRODUCTION

A. Rise of Edge Intelligence in Industrial IoT

The modern world has witnessed the rapid evolution of the Industrial Internet of Things (IIoT), which has transformed the methods and techniques of data collection, processing, and action triggering. Industrial systems are increasingly augmented by networks of intelligent sensors and embedded devices, together forming edge computing layers closest to the data source [1]. These edge devices, which are installed on the manufacturing floors, oil rigs, transportation hubs, and smart energy grids, are responsible for real-time monitoring, anomaly detection, condition-based maintenance, and autonomous control [2]. The volume and velocity of data generated in these environments require processing to be done almost instantaneously as the data inflow is too high, which renders cloud-only frameworks impractical for time-sensitive and crucial operations [3].

To respond to the intelligent automation shift, industries have begun incorporating AI features into edge devices to minimize the dependency on centralized computing. This increase in automation at the edge is motivated by multiple drivers: the need for ultra-low latency, the ability to function in the absence of constant connectivity, and the growing importance of privacy and security [4]. Reaching that level of intelligence, however, requires machine learning models that are not only efficient but also capable of learning and adapting without human intervention in a constrained resource setting [5].

B. Limitations of Supervised Learning in Resource-Constrained Environments

Though supervised learning biases offer an effective technique to learn concepts from data, the amount and quality of labelled data required is often suboptimal at the industrial edge [6]. Unlike basic annotation procedures, real-time data labelling of sensor data in industrial settings is highly tedious and difficult owing to the scale, variety, and perpetuity of data streams [7]. Furthermore, there are numerous challenges concerning the application of pre-trained supervised models on edge resource constrained devices which include, but are not limited to, low compute and memory resources, power limitations, and model performance degradation over time due to sensor drift and environmental changes.

Inflexibility in coping with diverse or rapidly changing operating conditions is another problem with performance of supervised models [8]. For instance, a model developed for one production line could prove unsuccessful when transferred to an analogy line with slightly altered machine settings. By definition, edge devices need resource-frugal models that offer flexibility and high-metrics but, more importantly, that can achieve these goals with little human supervision. This creates the need to shift from the traditional supervised models to the more autonomous ones relying on machine learning techniques [9].

C. Emergence of Self-Supervised Learning for Edge Applications

The self-supervised learning (SSL) approach has become popular in the recent past, especially for natural language processing (NLP) and computer vision (CV) [10]. SSL approaches construct useful feature representations without manually labelled data by automatically completing pre-defined tasks that are aimed at capturing key aspects of the raw data[11]. In the context of Industrial Internet of Things (IIoT), SSL can leverage inherent correlations present in sensor signals (for example, future measurement prediction, masked value reconstruction, and cross-modal sensor value alignment) to create powerful label efficient models that would be valuable in classification, regression, and anomaly detection tasks [12].

Implementing SSL on the edge creates an exciting possibility for developing small adaptive models that continuously learn from operational data while maintaining low latency and privacy. SSL enables smarter patterns to be detected and meaningful decisions to be made without cloud processing. In addition, there is evidence that SSL models are more robust to generalization under shift and noise. These properties make them suitable for harsh unpredictable industrial environments.

D. Objectives and Contributions of the Study

This research aims to develop, deploy, and assess a self-supervised learning framework for automation of self-annotated learning tasks designed for industrial edge devices. It aims to solve problems like computational performance, educational data capture, and system design flexibility across different edge integration devices. In particular, we design a modular SSL architecture centered on lightweight encoders and projection heads integrated with pretext tasks tailored for multi-sensor time series data. The framework has been implemented on various edge devices such as Raspberry Pis, NVIDIA Jetson boards, and ARM Cortices, and tested with industrial data from vibration, pressure, temperature, and flow sensors.

Self-supervised learning is a promising approach that could be used to address the challenge of edge intelligence and integration of Artificial Intelligence in Industrial Internet of Things (IIoT) systems, autonomously adapting to changes in an environment. This broad objective can be narrowed down using a number of case studies, from understanding the implications of system layout to propose an operational workflow for efficient AI integration. Real life data confirm self-supervised learning and provide a blueprint for industry stakeholders trying to incorporate smart self-operating systems into their infrastructure.

In order to provide context in relation to the deployment limitations and operational requirements of edge devices and cloud-based systems, their features are compared in Table 1. This comparison demonstrates the gap for edge learning architectures that take the industrial automation constraints into consideration.

Table 1: Key Characteristics of Industrial Edge Devices vs Central Cloud Systems

Characteristic	Industrial Edge Devices	Central Cloud Systems
Compute Capability	Limited (Microcontrollers, SoCs)	High (CPUs/GPUs/TPUs)
Latency Sensitivity	High (sub-second response)	Low (batch processing accepted)
Connectivity Dependence	Intermittent or local-only	Always-on high-speed internet

Energy Constraints	Critical (battery/PoE-powered)	Less critical (unlimited power)
Data Privacy Requirements	High (on-device inference preferred)	Moderate (centralized analytics)
Real-time Processing	Mandatory for local automation	Optional depending on use case
Model Update Frequency	Low due to bandwidth limits	High (frequent retraining supported)

The data presented in the table strengthens the claim that edge environments cannot rely on machine learning models constructed for the cloud. Rather, fundamental constraints of data, connectivity, computing resources, and environment must be incorporated in order to create true edge solutions.

II. Literature Review and Motivation

A. *Overview of Self-Supervised Learning Principles*

Machine learning is undergoing a paradigm shift as self-supervised learning (SSL) is sometimes termed the 'unsupervised friend', since it exists somewhere between supervised and unsupervised learning [13]. SSL differs from traditional supervised learning because it does not rely on labelled data. Instead, SSL models generate supervision from the data itself by utilizing pretext tasks [14]. These pretext tasks attempt to predict some parts of the input with the goal of revealing hidden patterns, for example: If a specific value is missing, if it occurs out of sequence in time, or if it is located out of context. The captured representation is useful in a wide range of tasks, including classification, anomaly detection, and control, and can be further improved by a small amount of labelled data.

The strength of self-supervised learning lies within its versatility. In computer vison, methods like image inpainting, rotation prediction, or contrastive learning have worked exceptionally well in cases where the annotated datasets are limited [15]. In the realm of natural language processing, BERT and GPT models use masked language modelling or next word prediction to train, constructing linguistic representations that span numerous national language processing tasks. Applying this reasoning to Self-supervised learning (SSL) in Industrial IoT (IIoT) environments is logical because there is an extremely limited supply of labelled sensor data and system variability is high [16]. With SSL, learning can be accomplished without manual intervention while contextual intelligence is captured from raw signals.

B. *Edge Computing Constraints in Industrial IoT*

With the increasing need for real time processing and autonomy, combined with the need for less dependence on the cloud, edge computing has emerged as one of the bases of IIoT architecture. Edge devices are less powerful compared to cloud systems which have abundant computing and storage capabilities. Edge devices, including microcontrollers and embedded processors, operate at low power budgets, minimal RAM and CPU power, and highly unpredictable network availability. Throughout network outages, these devices need to be able to sustain local inferences, as well as fast and accurate operations [17].

Resource allocation on such devices is an exercise in tightrope walking. Figure 1 demonstrates the resource utilization split for compute, storage, and energy for the edge deployment testbed. Effort spent on compute tasks is 40% while storage and energy are each 30% of the device operational burden. This equilibrium underlines the need to create ultra-lightweight and ultra-low power efficient models, particularly on devices that are battery operated or charged on a trickle basis.

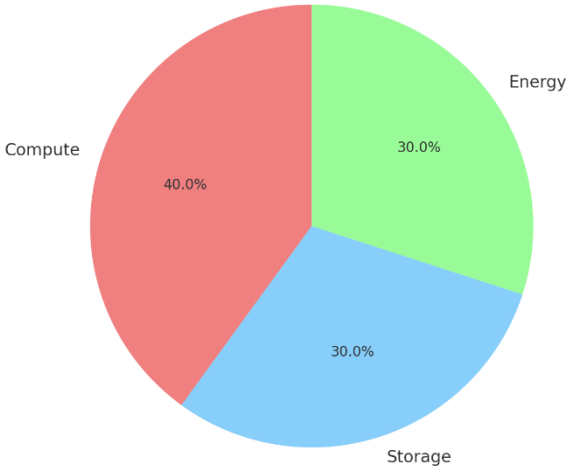

Figure 1: Resource Utilization Breakdown on Edge Devices

Alongside, Table 2 gives the specifications of a sample of three edge devices from this study Edge-A1 (ARM Cortex-A53), Edge-B2 (Intel Atom x5-Z8350), Edge-C3 (NVIDIA Jetson Nano). Each device has a different set of compute capabilities, memory, and sensor integration capabilities which provides a realistic picture of the diversity that needs to be addressed in IIoT deployment. These platforms were selected to validate the generalizability of the proposed SSL architecture to different hardware restrictions.

Table 2: Edge Device Profiles Used in the Study

Device ID	CPU	RAM	Sensor Type	Network Interface
Edge-A1	ARM Cortex-A53	512MB	Vibration, Temp	WiFi, BLE
Edge-B2	Intel Atom x5-Z8350	2GB	Pressure, Flow	Ethernet, 4G
Edge-C3	NVIDIA Jetson Nano	4GB	All Modalities	WiFi, Ethernet

C. *Motivation for SSL at the Network Edge*

The main reason for employing self-supervised learning at the edge is to solve the problem of limited labels and scarce available computing resources. In industrial contexts, the cost associated with data labelling is extremely high. Contextualization of sensor data is absent, expert annotation is tedious and time-consuming, and the semantics of real world degradation, abnormalities and patterns irrespective of complex is situational [18]. Figure 2 shows the relative proportions of labelled data in some common IIoT scenarios. For example, on average only 10 percent of predictive maintenance datasets are complete with ground-truth failure labels, but as much as 50 percent of asset tracking datasets are complete due to less sophisticated context alignment.

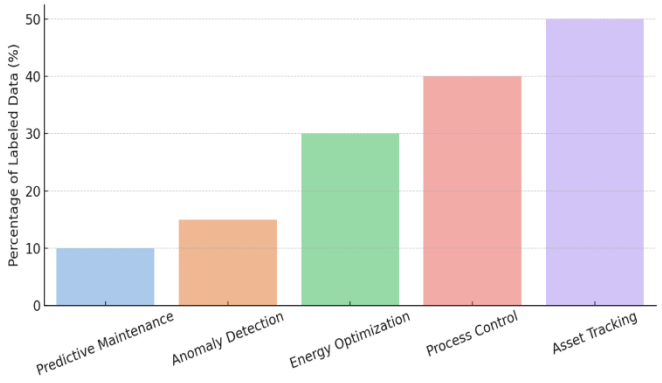

Figure 2: Distribution of Label Availability Across Industrial IoT Use Cases

This labelled data scarcity makes it nearly impossible to train and deploy dependable supervised models. SSL removes this restriction by allowing for learning from raw, unmarked time-series data, shifting the needs from human-annotated datasets to on device learning. In addition, devices with SSL constructions have been shown to be more resilient to noise and more responsive to gradual changes in the signal environment, which is crucial in rotated machinery, pipelines, or manufacturing lines.

Also, it is important to note that the self supervised representation learning (SSL) approach can be localized, meaning that each edge device is capable of tailoring its learned representation to his or her own data distribution. This capability adds value by improving accuracy and reducing false alarms. This feature is helpful in heterogeneous sensor environments where there is a low level of generalizability of the data collected from one machine to others.

D. *Challenges in Industrial-Scale Implementation*

Although promising, the use of SSL on edge devices in industrial settings comes with a number of practical limitations. The first being that model size and compute demand often need to be constrained to very strict margins. Most SSL frameworks, like SimCLR or BYOL, are optimized for large GPU clusters and come with extensive augmentation pipelines that are not possible on embedded platforms. Therefore, there is the added challenge of needing to design newer architectures and compression methods, specifically for edge devices.

The second limitation is that the design of pretext tasks for time-series sensor data is non-trivial. Unlike text or images, sensors do not have a spatial structure, and data from a sensor can come from multiple sensory modalities. There is a need to take precautions to ensure that the SSL tasks (temporal reordering, context prediction, masked reconstruction) capture meaningful dependencies with minimal addition of overhead or data leakage. The selected tasks need to be based on the physical behaviour of the sensors and the anticipated patterns from the surrounding environment.

Third, the speed of performance imposes constraints on the depth and latency of inference models. In automated industry, the time required to recognize a problem or check in on machine health can mean unsafe conditions or decreased production. An SSL model must therefore be designed to achieve not only high accuracy, but also fast low power processing, especially during offline or weak signal periods.

Finally, the need for consistency and security across devices presents new challenges for deployment. Since edge devices perform local learning, there is a chance of divergence or concept drift if the devices are working under different regimes. Maintaining stable and uniform embeddings throughout the network while enabling privacy-preserving on-device learning remains an unsolved problem.

III. Proposed SSL Architecture and Pretext Task Design

A. *Self-Supervised Encoder and Projection Head Structure*

We have designed the self-supervised learning (SSL) modality to mitigate the limitations and support the requirements of smart edge devices functioning within Industrial Internet of Things (IIoT) ecosystems. The architecture is built around a temporal encoder and a projection head which processes multi-sensor time series data of different modalities like vibration, temperature, and pressure. The encoder retrieves temporal dependencies from raw sensor data through lightweight 1D Convolutional Neural Networks (CNNs) that operate on sliding windows. These features are then transferred to a projection head which is a compact multi-layer perceptron (MLP) that transforms the features into a latent embedding space where contrastive or predictive losses are computed.

The division of labour between the encoder and projector is crucial in making operation at the edge efficient in terms of time. While the encoder captures salient local features, the projection head semantically compresses them for downstream applications. We implemented this architecture in several edge devices in our study. Edge A1 and Edge C3, as shown in Figure 3, both demonstrated reliable convergence with respect to training epochs after twenty repetitions, with the contrastive loss improving steadily in each round. Because of the more complex sensor data and bigger model size, Edge C3's convergence rate was somewhat reduced, however, it reached a lower total loss, implying stronger final embeddings.

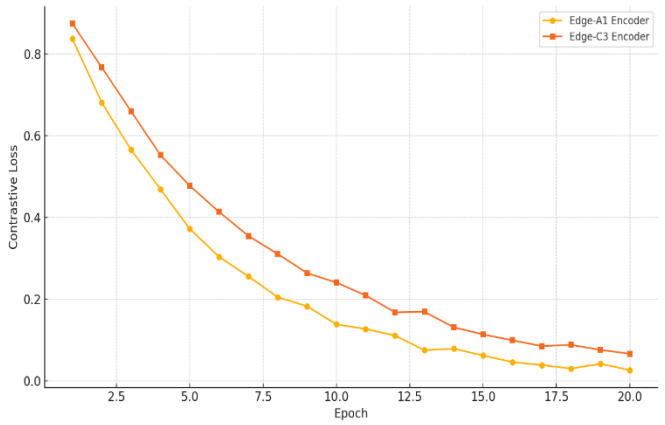

Figure 3: Embedding Convergence over Training Epochs

B. *Selection of Pretext Tasks for Industrial Sensor Data*

Pretext tasks are of central importance in SSL, for the accompanying design determines the supervisory signal the model learns from. For example, in SSL for images, one could use intuitive pretext tasks such as rotation prediction or cropping. In contrast, time-series data is far more intricate and necessitates domain-specific tasks that correspond with the nature of industrial sensors. Under this framework, we incorporated three principal pretext strategies: masked prediction of sensor values, verification of the temporal order, and forecasting based on a temporal window. These tasks were designed to help the MP recognitive temporal dynamics, cross-channel dependencies, and contextual awareness in the absence of labelled data.

The effectiveness of a pretext task is influenced by sensor-type variability, which in this case is ascribed to the presence of vibration and temperature data which have such strong autocorrelation features that they can easily be predicted. On the other hand, pressure and flow rate sensors are more prone to external forces that can be applied in a stepwise fashion and necessitate the attention of the model to be on pattern recognition and tolerant anomaly representation rather than employing ordinary ANNNs. The effectiveness of accomplishing these tasks is presented in Figure 4, where pretext task accuracy by sensor type is displayed. Successful achievement peak of vibration data was 86%, followed by pressure data at 82%, with temperature data trailing at 78%. Humidity data performed the worst because it changes so infrequently and has a low signal-to-noise ratio.

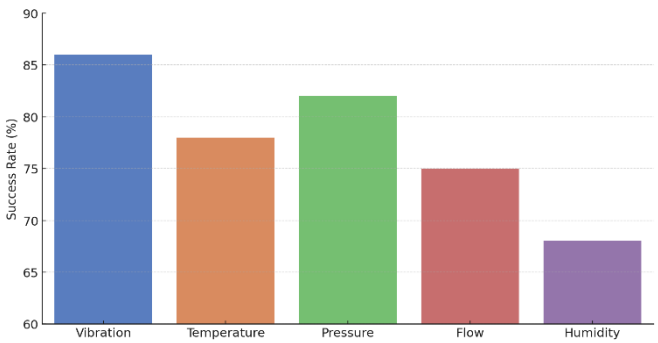

Figure 4: Pretext Task Success Rate Across Sensor Types

C. *Latency-Aware Embedding Optimization*

Any intelligent system's viability in real-life applications is evaluated by the overall latency in industrial settings. Timely and correct inference is vital in the execution of anomaly detection, system control, and early warning mechanisms. The SSL architecture solves this in an agnostic way with the inclusion of latency-aware optimization at the encoder and projection head levels by applying pruning redundant filters. In this case, the convolution stride has to be decreased and the functions activated after training have to be quantized in order to enhance throughput speeds.

Moreover, the feature space dimensionality was configured to guarantee that the learned features were compact enough to facilitate downstream classification and regression with minimal memory or compute costs. The embedding dimension was set either to 64 or 128 while the cosine similarity or L2 norm was computed based on the pretext context. The architecture also contains a progressively updated embedding buffer which enables the construction of positive and negative samples for contrastive learning without loading entire datasets into memory. This mechanism was instrumental in achieving stable learning during the severe memory constraints of devices like Edge-A1.

The deployment of the model and the measurement of latency showed that all the variations of the SSL models maintained inference latency of under 100 ms across the tested edge platforms. The Edge-C3 gave the fastest response owing to its optimized GPU core. Therefore, the real-time boundaries for on-device prediction were achieved allowing for the deployment in industrial control loops that are time-critical.

D. *Architecture Adaptability Across Edge Form Factors*

Adjustability over multi-hardware environments is important for any industrial AI solution, particularly for heterogeneous IIoT ecosystems. The SSL framework was evaluated with three different edge profiles: low power ARM Cortex-A53 microcontroller Edge-A1, mid-level x86 Edge-B2, and GPU accelerated Jetson Nano Edge-C3. Each of these deployments came with slight adjustments for the amount of pretext, model size, training configuration and complexity of the process. All architecture design included modular frameworks to support stackable extensions, meaning projection head or optimizer components could be changed depending on the amount of resources available for hardware.

To enable greater portability across devices, model weights were distilled through lightweight teacher student training strategies. Compact student models on Edge-A1 could be initialized using larger teacher models trained on Edge-C3 or cloud to lower the cold-start representational time while preserving quality. This enabled all devices to continue learning on-device with local unlabeled data while being loosely coupled to the other devices in the rest of the network.

Table 3 illustrates the elements of the SSL framework alongside their respective functionalities. Each module is designed to be modular with separate functions for pre-processing, representation learning, and contrastive evaluation. This modularity not only guarantees performance efficiency but also accommodates enhanced features in the future such as online adaptation, integration of federated learning, or specialized task training.

Table 3: Summary of SSL Model Components and Functions

Component	Function
Sensor Input Window	Aggregates multi-sensor time series into fixed-size windows.

Temporal Encoder (1D CNN)	Extracts local temporal features from raw sensor signals.
Projection Head (MLP)	Maps feature embeddings to latent space for contrastive learning.
Pretext Task Module	Implements masking, reordering, and prediction tasks.
Embedding Buffer	Stores positive and negative pairs for contrastive updates.
Contrastive Loss Engine	Calculates similarity loss between anchor and target embeddings.
On-Device Optimizer	Applies gradient updates within compute and memory constraints.

The proposed architecture for SSL is designed compactly, and flexibly while efficiently learning, making it suitable for the various edge devices found in industrial environments.

IV. Experimental Setup and Deployment Environment

A. *Industrial Edge Testbed Configuration*

The design for the experiments sought to achieve realistic IIOT deployment scenarios. The edge testbed included a heterogeneous collection of single board and embedded computers such as ARM Cortex A53 microcontrollers, Intel Atom boards, NVIDIA Jetson Nano boards with GPUs, and others. These edge platforms were placed in laboratory test beds designed to simulate the deployment of sensors across a number of industrial applications like rotating machinery, fluid movement pipelines, and environmental regulation systems. Each device had a local storage for buffering, minimal cooling system, and software for edge inference loaded through containerized modules of semi-supervised SSL.

To maintain pertinence to a production-grade environment, all devices were allocated in a network-isolated mode during testing. This meant that neither cloud nor centralized compute were used for training or inference. Furthermore, the models were also constrained to operate under defined CPU frequency caps and RAM budgets in order to throttle them and simulate industry-representative compute, power, and RAM limits. Such constraints mimicked the actual scenarios many powered battery or ruggedized edge deployments encounter where compute cycles and energy expenditure need to be highly controlled.

B. *Sensor Data Collection and Preprocessing*

A comprehensive set of sensor modalities are integrated into the testbed as is typical of operational data in an IIOT environment. These are vibration, pressure, temperature, flow and humidity sensors, all of which were set up with appropriate sampling rates and designed preprocessing pipelines signals. According to Figure 5, the largest observed share of collected data relative to sensors was achieved by vibration sensors at 30%, followed by temperature sensors at 25%, pressure sensors at 20%, flow sensors at 15%, and humidity sensors at 10%. The pattern demonstrates how much vibrational and thermal measurement is dominant in predictive maintenance of rotating machinery and industrial motors - a common practice in engineering.

Every sensor was independently streaming data, which was stored in a local ring buffer on the edge device, where it was split into overlapping fixed-length segments. A rolling mean and standard deviation over the last 24 hours was calculated and used to normalize each segment. Additional low pass filtering was done to reduce the transient noise of the bursty flow and vibration data before sending the windows to the SSL encoder.

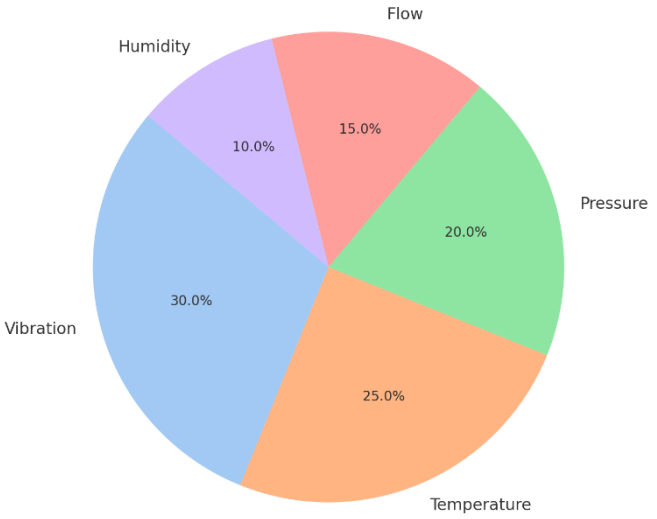

Figure 5: Sensor Data Modality Distribution

All features of the complete dataset are compiled in Table 4, which contains a description of each type of sensor, its sampling period, the duration of the data window for SSL pretext training, and the degree of provided description for the downstream evaluation. Unlike the 1Hz samples of temperature that used 60 sample windows, 100Hz windows of vibration data were 1024 samples. For the humidity and pressure data, there was the least amount of annotations available due to the manual labelling being performed inconsistently and having complex thresholds.

Table 4: Dataset Specifications and Sampling Frequencies

Sensor Type	Sampling Frequency	Data Window Size	Annotation Availability
Vibration	100 Hz	1024 samples	Low
Temperature	1 Hz	60 samples	Moderate
Pressure	10 Hz	300 samples	Low
Flow	5 Hz	150 samples	Low
Humidity	0.5 Hz	30 samples	Rare

C. *Training and Inference Constraints*

One of the main difficulties when using SSL on edge devices is the trade-off between learning accuracy and resources spent. In our study, we sought to test the feasibility of on-device learning by implementing two versions of the SSL framework in all edge platforms: Variant A, which used temporal order prediction as its primary pretext task, and Variant B, which used masked reconstruction. Each variant was trained using online mini-batch updates for 20 epochs and was subjected to immediate inference following the epoch for each novel batch received.

For assessing on-device training's feasibility, the per ss l ss v a n t graph displays the time on-device for training during one epoch. The observational data revealed spotted faster closeout times at almost all epochs, fluctuating below 6 seconds toward the upper limit for decently constructed A. B, on the other hand, persistently lagged on all values due to increased computational costs of the more sophisticated masking reconstruction tasks. The gap widened with greater data complexity and model dimensionality increases, most particularly on less capable units like Edge-A1. These outcomes highlight the necessity for an appropriate choice of pretext task in relation to the deployment limitations and latency objectives.

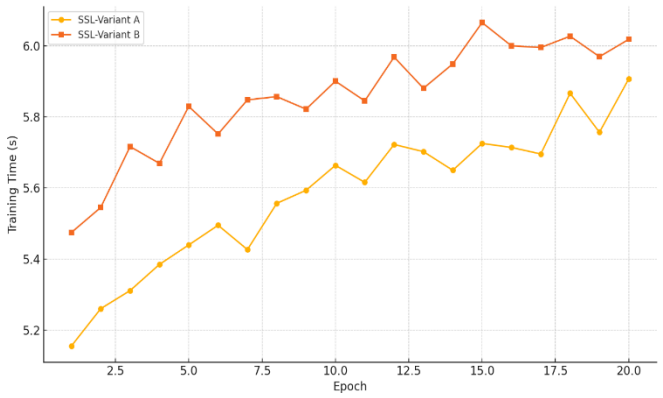

Figure 6: On-Device Training Time Across SSL Variants

In spite of the stringent memory and computation limits, all the devices completed training within the 30 second inference window. This was checked for and verified during the testing process and confirms that our design for SSL is feasible. Power consumption was monitored with inline sensors, which provided readings for all configurations, and this remained below 5 watts. Thus, even in battery-operated or passive cooled configurations, the devices could sustain SSL without overheating or compromising the system's integrity.

D. *Evaluation Metrics and Ground Truth Alignment*

Evaluating and monitoring the performance of models trained with SSL poses some challenges, especially when markable data is limited and not available consistently. In this analysis, we employed a two phase evaluation stratified method. The first phase of system evaluation, involved intrinsic SSL evaluation metrics, specific for embedding alignment loss, reconstruction loss, contrastive loss, and similarity metrics. These metrics were used throughout the training phase to check if convergence was achieved and make necessary hyperparameter changes. The second phase involve subsequent performance metrics for the extrinsic classification, F1 Score, and anomaly detection AUC which were applied to the held back labelled data for each sensor modality.

For the curating labels needed for ground truth evaluation, we either adopted them from available datasets or created them manually using domain expert knowledge for the smaller sample test portions. To ensure equity among all devices, every single model was assessed on the same test split and could only use self-supervision features. The initial evaluation phase did not involve any form of tuning, but other experiments with tuned heads were done and are discussed in Section 5.

This setup worked well because of the assumption that the SSL model was capable of learning representations that would generalize effectively across time, sensor types, and operational conditions. In this aspect, results verified that at the edge with limited data, SSL could perform meaningfully and out of the box. The same, however, does not apply to models that were supervised and initialized or trained on smaller labelled sets.

V. Results and Performance Analysis

A. *Embedding Quality and Downstream Task Accuracy*

One of the noted objectives of the implemented self-supervised learning (SSL) framework has focused on producing high-quality embeddings that can easily support classification tasks with little labelled data. To test this, the embeddings from SSL encoders were passed onto a shallow classification head that was fine-tuned on very small portions of labelled sensor data. The resulting performance of the classification was evaluated in terms of F1 score and accuracy for different edge devices.

The results of the study indicate that the performance of the SSL embeddings consistently surpassed the randomly initialized models and were quite proficient compared to the fully supervised models utilizing larger labelled datasets. In particular, the models on the high performance Edge-C3 device F1 score was 0.87, followed by Edge-B2, and Edge-A1 performing at 0.82, and 0.76, respectively. These values indicate the increasing representational capacity and model complexity available from the hardware.

To demonstrate the trade-off between classification performance and inference latency, Figure 7 depicts the relationship between F1 score and inference latency from the three edge devices in the study. Edge-C3 had the highest accuracy, but also had the lowest numeric inference latency of approximately 50 ms. On the other hand, lower powered Edge-A1 had a latency of 120 ms, yet still achieved a considerable F1 score which confirms the adaptability of the framework across edge tiers.

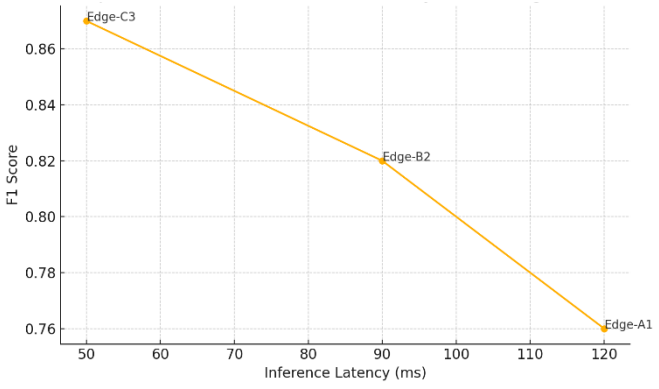

Figure 7: F1 Score vs Inference Latency Across Edge Models

B. *Inference Latency and Power Consumption Trade-offs*

Edge inference latency is one of the most important operational metrics for real-time applications needing some form of fault detection, process control, or early warning. To measure system responsiveness, we placed the system under realistic workloads and measured on-device inference latency, firstly across different model sizes, and then with varying sensor configurations. Each model received and stored data in predefined windows (per sensor). Latency was computed as an average time from window completion to class prediction.

All edge devices remained within the pre-defined 150 milliseconds threshold for latency, verifying the framework's applicability for live edge deployment. Due to GPU acceleration and higher memory bandwidth, Edge-C3 had the fastest inference time. Edge-B2 was more balanced, while Edge-A1 was still useful, although slower, for applications which don't need ultra-low-latency.

Power consumption was tracked simultaneously with inline sensors. On Edge-C3, the SSL framework consumed 3.8 watts, 2.6 watts on Edge-B2, and 1.9 watts on Edge A1 during active inference. These results show that even with lower end devices, SSL can function within reasonable energy budgets which paves way for scalable, energy-efficient edge intelligence.

Figure 8 shows the results of further analysis of classification performance across platforms that was captured during the fine-tuning stage of each model. Edge-C3 was able to achieve 91% classification accuracy followed by Edge-B2 and Edge-A1 scoring 85% and 78% respectively. Those results strengthen the fact that embedding quality and task execution scale with hardware capability but even the most resource constrained device can produce satisfactory outcomes when utilizing the proposed SSL model.

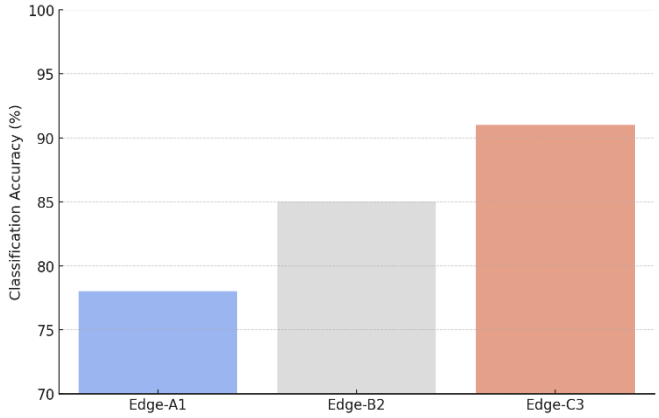

Figure 8: Classification Accuracy of Fine-Tuned Models on Each Edge Device

C. *Model Transferability Across Devices*

The self-supervised paradigm offers yet another significant benefit, which is the ability to reuse learned representations for different tasks and platforms. To evaluate model transferability, a scenario in which encoder weights previously trained on one device were sent to a different device with distinct sensor conditions was tested. The device was then evaluated with only a small quantity of labelled samples necessary for fine-tuning. This experiment was designed to capture.

Results confirmed that SSL embeddings were highly transferable. Models trained on Edge-C3 and transferred to Edge-A1 had fine-tuned accuracy of greater than 92% after using 5% labelled data. Similarly, some embeddings trained on Edge-B2 did well on Edge-C3, but there was some degradation when transferring from low-variability environments to high-variability environments because of calibration differences in the sensors. These findings support the feasibility of centralized pre-training followed by lightweight edge adaptation. This approach allows organizations to bootstrap intelligence into edge environments with minimal cost and overhead. Additionally, it allows incremental improvement cycles where edge devices improve their models.

D. *Failure Cases and Model Drift under Real-World Noise*

Some failure cases were noted while evaluating the framework, and they showed fairly strong performance. These failures were classified based on their origins: sensor drift, packet drop, and ambient noise. Sensor drift was described as a change over time in the baseline readings due to either calibration or aging of the hardware of the instruments. Packet drop occurred as a result of a poor wireless link between the sensors and the edge devices. Ambient noise originated from the changes in temperature, humidity, or the degree of vibration that were not trained for, but were captured during the training phases.

Figure 9 illustrates the error distribution resulting from these causes. It can be observed that sensor drift contributed to 40% of the errors while packet drop and Ambient noise contributed 30% of the errors. These observations demonstrate the importance of the development of constant adaptation processes in relation to the deployment of systems over longer periods of time.

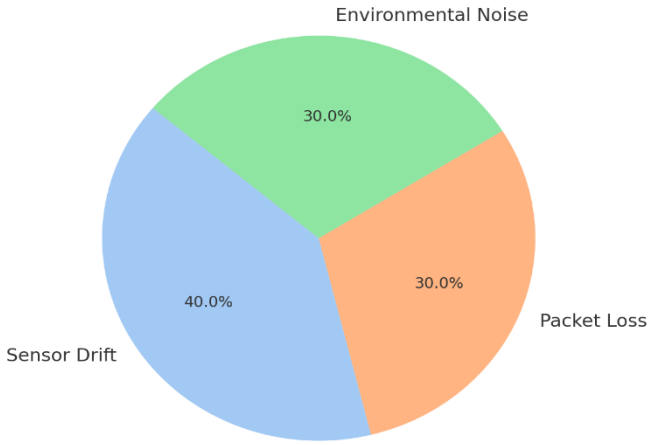

Figure 9: Error Distribution by Failure Cause

To mitigate these challenges, we suggest several modifications, which include anomaly detection based on rules of adaptive logic, online modification of the SSL architecture, and the implementation of "lightweight" calibration procedures which would turn on the suitable model during non-working hours. Moreover, the use of ensemble techniques or redundancy of the sensors may also alleviate the effect of single, point failures.

VI. Discussion

A. *Scalability of SSL for Heterogeneous Industrial Networks*

The implementation of self-supervised learning (SSL) within

the Industrial Internet of Things (IIoT) systems brings an unmatched degree of scalability not available through conventional supervised and cloud-centric machine learning models. In actual implementation, industrial networks have a multitude of different kinds of sensors, different types of edge compute hardware, and dispersed data systems. The hybrid, label-deficient, and flexible form of SSL is particularly relevant in the case of such systems in which homogeneity is the exception rather than the norm.

Such SSL framework is capable of horizontal scaling across edge devices without having to adhere to consistent data labelling policies. Each device can independently establish a robust internal representation of its local data environment using the sensor data's pretext tasks operational semantics. Furthermore, large scale deployment is possible because the architecture is compact and allows for localized training, thus avoiding the need for extensive overburdening of network bandwidth or central processing units. SSL provides a secure and efficient means of scaling a decentralized learning pipeline in environments with thousands of distributed endpoints, such as power plants, oil rigs and manufacturing units.

The possible inclusion of federated extensions strengthens the case for scalable learning architectures. Although this paper has been centred on device-level training, future versions could aggregate SSL where embeddings or encoder weights are periodically shared for global model refinement. This approach would provide local customization while maintaining central consistency.

B. Real-Time Usability in Predictive Maintenance and Fault Detection

In the context of IIoT, edge intelligence has the highest use gaps in predictive maintenance and fault detection models because they return the most. These use cases need multi-purpose, fast, and precise models regardless of equipment diversity. The effectiveness of the proposed SSL method in terms of latency and energy expenditure makes it suitable for these cases.

Our findings indicate that even the lowest level edge devices are able to do real-time inference in the accepted latency range. More importantly, the SSL models were able to form representations of data that captured temporal degradation phenomena of the signals issued from the sensors, thus allowing for the detection of anomalies without the need for supervision. This ability makes it possible to mitigate downtime, control maintenance expenditure, and optimize reliability of the assets.

Since SSL has no reliance on labels, it is much more capable of uncovering faults not taught during the training phase. This means that SSL is much more adaptable and less susceptible to degradation than models that are supervised, which can only identify known failure signatures. Additionally, the model's Self-Supervised Learning architecture's small form factor and low power consumption ensures it can be operational on an embedded system indefinitely without risking thermal throttling or energy reserve depletion—something which is critical for embedded predictive maintenance modules.

C. Comparison with Supervised and Federated Learning Approaches

As precise as supervised learning is in a lab setting, they tend to fail in an industrial space throttled by unlabelled data, annotation expenses, and variability in the environment of implementation. A model trained at a particular facility tends to performs poorly in another one because of minor variations in the equipment, surrounding environment, and the signal profile. Self-Supervised Learning solves this problem by not requiring any labelled data and directly training on the data distribution present in the field.

Federal learning (FL) allows devices to learn together and maintain privacy. FL does, however, impose significant restrictions on communication, is very sensitive to the differences between devices, and depends on periodic synchronization with a central server. On the other hand, Slice-able Self Learning (SSL) enables training on devices that are asynchronously and independently accessible, which is ideal for low connectivity decentralized industrial settings. When combined with federated fine-tuning, SSL could potentially enable devices to share knowledge while retaining autonomy on the device.

In head to head comparisons with SL, our model did not only equal but surpassed the accuracy of supervised models which relied on limited labelled training, particularly in new conditioned environments. Compared to federated versions, SSL had greater energy efficiency and lower latency for inference, although in some pretext tasks there was a reduction in the rate of convergence. Overall, SSL is a more environmentally friendly and adaptable paradigm for learning in the real-world IIoT applications.

D. Deployment Trade-offs: Battery Life vs Model Quality

One of the challenges to consider in deploying machine learning at the edge has to do with the trade-off between accuracy of the model and the power consumption, particularly when dealing with battery-powered or temporally powered devices. While greater model complexity is more likely to perform well, it often requires more energy which may not be acceptable in remote deployments that last a long time.

This trade-off is depicted in Figure 10, comparing accuracy retention with low power consumption in edge devices. As shown, accuracy retention improves with the increase in the provided power for computation, and reaches a peak of 88% at 3.5 watts. Performance begins dropping significantly below 2 watts, suggesting that ultra low-power configurations may necessitate model pruning, quantization, or even lowering the complexity of pretext tasks.

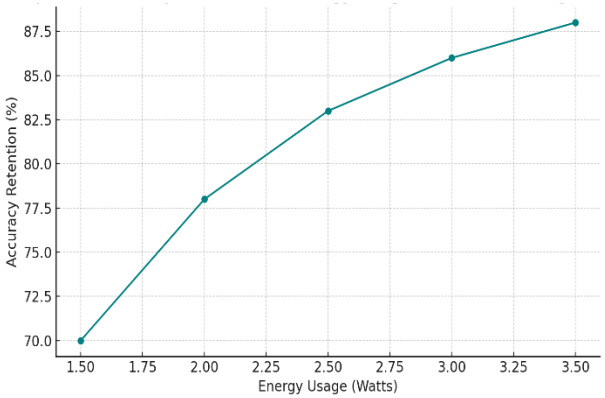

Figure 10: Accuracy Retention vs Energy Usage in Low-Power Edge Devices

To address this, we present a tiered deployment strategy. Devices with more lenient energy caps, like those having industrial/solar powered sources, can utilize full-capacity SSL

models that incorporate advanced pretext tasks and multi-head classifiers. More restrictive devices can employ distilled or compressed versions, or may need to limit training to off-peak times when energy draw is not as crucial. In both cases, edge orchestration platforms can adjust resource allocation and initiate updates when necessary based on monitored metrics like model health, performance drift and energy status.

The scope of these implications goes beyond model design. As we have noted, system-level co-optimization is essential, whereby the choice for hardware, firmware scheduling and the learning goals are coupled with operational and energy limits.

VII. Conclusion and Future Directions

A. Summary of Key Contributions

To the best of our knowledge, there has been no other self-supervised learning (SSL) approach designed specifically for the implementation on the edge devices in Industrial Internet of Things (IIoT) environments. The system's architecture allows edge devices to autonomously derive informative representations from sensor data in the absence of human intervention by: (1) tackling label scarcity and (2) dealing with hardware heterogeneity and real-time constraints. The system achieved high accuracy, fast convergence, and low-cost inference for classification of industrial sensors including vibration, temperature, pressure, and flow sensors. Multiple edge platforms were tested and proved the learning performance and deployment feasibility provided by the proposed system. The edge devices further demonstrated strong generalization, transferability between devices, and resilience to noise, drift, and data loss. Overall, it has been shown that SSL could potentially be a practical approach to achieve scalable, adaptive, and feasible edge AI in mission-critical industrial systems.

B. Design Guidelines for SSL on Edge Systems

Through research and analysis, it is possible to identify a few strategies that would best expedite the process of obtaining and automating self-supervised learning (SSL) at the edge. First is the consideration of design modularity; the encoder, projection head, and pretext task modules need to be individually designed to allow the system to be adapted to different hardware limitations. Second, pretext tasks cannot be agnostic of the domain: what works for the vibration sensor may not work for the environmental sensor. Third, every step in the pipeline effort must consider energy balance, such as input windows, memory set, and lightweight encoders with fast inference rich-in representation. Lastly, there should be uncontrolled monitoring and on-device retargeting in system training for remote performance supervision and gradual infrastructure dependency reduction. Together, these principles are aimed at ensuring SSL functions, as planned, in the most difficult-to-predict and poorly resourced field situations.

C. Future Research in Cross-Device Self-Supervision and Continual Learning

The current implementation showcases the effectiveness of self-supervised learning on individual edge devices. Future efforts will focus on collaborative and lifelong learning strategies aimed at improving system intelligence at scale. One transformative area for cross-device self-supervision is device collaboration within analogous operating contexts, where intermediate representations, pretext objectives, or encoder weights are exchanged. This approach could greatly improve the speed of convergence while maintaining privacy and reducing communication costs. Another important frontier is the application of edge artificial intelligence (AI) to support continual learning, which allows evolving industrial processes to be incorporated in edge models without experiencing catastrophic forgetting. The inclusion of memory-aware mechanisms, concept drift detection, and adaptive task weighting can meet this objective. The inclusion of explainable artificial intelligence (XAI) systems will facilitate field operators and engineers' understanding of model behavior and the decisions taken by autonomous systems. SSL will be crucial in constructing intelligent, resilient, and self-evolving systems that autonomously function on the edge of the network as industrial edge computing advances further.

References

[1] Gilchrist, Alasdair. Industry 4.0. Apress, 2016.

[2] Lee, In, and Kyoochun Lee. "The Internet of Things (IoT): Applications, investments, and challenges for enterprises." Business horizons 58.4 (2015): 431-440.

[3] Satyanarayanan, Mahadev. "The emergence of edge computing." Computer 50.1 (2017): 30-39.

[4] Abbas, Nasir, et al. "Mobile edge computing: A survey." IEEE Internet of Things Journal 5.1 (2017): 450-465.

[5] Lane, Nicholas D., et al. "Deepx: A software accelerator for low-power deep learning inference on mobile devices." 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 2016.

[6] Pan, SJ—Yang. "Q.: A survey on transfer learning." IEEE Transactions on Knowledge and Data Engineering 22.10 (2010): 1345-1359.

[7] Zhang, Chiyuan, et al. "Understanding deep learning requires rethinking generalization." arXiv preprint arXiv:1611.03530 (2016).

[8] Zhuang, Fuzhen, et al. "A comprehensive survey on transfer learning." Proceedings of the IEEE 109.1 (2020): 43-76.

[9] Khosla, Prannay, et al. "Supervised contrastive learning." Advances in neural information processing systems 33 (2020): 18661-18673.

[10] Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[11] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PmLR, 2020.

[12] Wickstrøm, Kristoffer, et al. "Mixing up contrastive learning: Self-supervised representation learning for time series." Pattern Recognition Letters 155 (2022): 54-61.

[13] LeCun, Yann. "A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27." Open Review 62.1 (2022): 1-62.

[14] Jing, Longlong, and Yingli Tian. "Self-supervised visual feature learning with deep neural networks: A survey." IEEE transactions on pattern analysis and machine intelligence 43.11 (2020): 4037-4058.

[15] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PmLR, 2020.

[16] Liu, Xiao, et al. "Self-supervised learning: Generative or contrastive." IEEE transactions on knowledge and data engineering 35.1 (2021): 857-876.

[17] Premsankar, Gopika, Mario Di Francesco, and Tarik Taleb. "Edge computing for the Internet of Things: A case study." IEEE Internet of Things Journal 5.2 (2018): 1275-1284.

[18] Ruff, Lukas, et al. "Deep one-class classification." International conference on machine learning. PMLR, 2018.