# A Scalable Multi-Task Model for Virtual Sensors

**Leon Götz**[1 2] **Lars Frederik Peiss**[1†] **Erik Sauer**[1†] **Andreas Udo Sass**[1]
**Thorsten Bagdonat**[1] **Stephan Günnemann**[2] **Leo Schwinn**[2 3]

[1]Volkswagen AG [2]Technical University of Munich [3]Helmholtz AI
[†]Work done during an internship at Volkswagen AG

Correspondence to: `leon.goetz@volkswagen.de`

## Abstract

Virtual sensors replace expensive physical sensors in critical applications through machine learning by predicting target signals from available measurements. Existing virtual sensor approaches require application-specific models with hand-selected inputs for each sensor, cannot leverage task synergies, and lack consistent benchmarks. While emerging time series foundation models offer general-purpose, pretrained solutions in other domains, they are computationally expensive and limited to predicting their input signals, making them incompatible with virtual sensors. We introduce the first multi-task model for virtual sensors addressing both limitations. Our unified model can simultaneously predict diverse virtual sensors exploiting synergies while maintaining computational efficiency. It learns relevant input signals for each virtual sensor, eliminating expert knowledge requirements while adding explainability. In our large-scale evaluation on three standard benchmarks and an application-specific dataset with over 18 billion samples, our architecture reduces computation time by up to $415\times$ and memory requirements by $951\times$, while maintaining or even improving predictive quality compared to unified baselines. Compared to existing isolated models for a single virtual sensor, our unified approach generates superior predictions at similar inference speed while scaling gracefully to hundreds of virtual sensors with nearly constant parameter count, enabling practical deployment in large-scale sensor networks.

## 1 Introduction

Many important applications in areas such as autonomous driving, environmental monitoring, and healthcare rely on high-quality sensors. Unfortunately, these sensors can be expensive, difficult to maintain, or unsuitable for large-scale deployment. Virtual sensors address this limitation by computing target signals from available other measurements through statistical or learning-based methods [1]. They can replace physical sensors, saving costs, act as redundancy in safety-critical applications, allow for non-invasive measurements, or compute quantities that are not measurable with physical sensors, such as a battery's state of health [2].

While some deep learning-based approaches exist for virtual sensor simulation, they exhibit four key limitations. First, they rely on highly application-specific models that are trained, deployed, and maintained

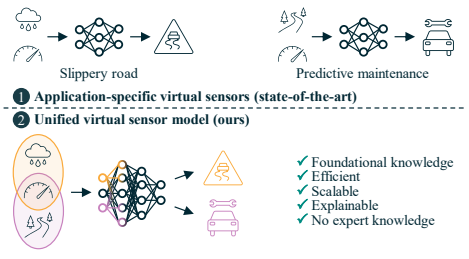



Figure 1: ① Recent works train, maintain, and deploy isolated application-specific models for every virtual sensor. ② We unify multiple virtual sensors into a single model, exploiting synergies and achieving superior scalability. It learns relevant input signals and selectively predicts virtual sensors, increasing efficiency and explainability without expert knowledge.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

separately for each virtual sensor, with input signals typically being hand-selected by domain experts [3]. As a result, current approaches do not scale to large sensor networks. Second, these isolated models cannot leverage synergies among different virtual sensor tasks, requiring one individual model for every simulated sensor. Third, there is little consistency in how virtual sensors are evaluated, with few works using standard benchmark datasets [4]. Lastly, existing methods tend to rely on relatively simple deep learning models [5], potentially limiting their performance.
Recently, foundation models have shown impressive performance in time series forecasting, outperforming specialized approaches [6]. These unified models exploit synergies among different tasks and datasets. However, large foundation models impose considerable computation requirements [7], limiting their application in sensor networks with thousands of signals [8]. Additionally, current time series foundation models only forecast the input signals and cannot predict new signals, making them unsuitable for virtual sensor tasks.
We address these complementary gaps through a unified model for virtual sensors. By training a single model across multiple virtual sensor tasks, we overcome the scalability and synergy limitations of existing virtual sensor approaches. Simultaneously, we design our architecture to be computationally efficient, addressing the deployment constraints that prevent current foundation models from being applied to large-scale sensor networks with thousands of signals. Our key contributions are:

**Large-scale benchmark** We provide the first large-scale analysis of unified virtual sensor models on three common time series forecasting benchmarks and an application-specific automotive dataset with over $17\,500\,\mathrm{km}$ of driving and 18 B time series samples using over $48\,500$ H100 GPU hours.

**Unified model for virtual sensors** Our unified model can predict multiple virtual sensors. By exploiting synergies among them and leveraging foundational time series knowledge, it scales gracefully to large sensor networks while even improving predictive quality in some cases. Our approach learns the set of relevant input signals for each virtual sensor, thereby adding explainability and eliminating the need for expert knowledge in virtual sensor design. Due to its flexible sensor selection mechanism, it achieves high efficiency, making it suitable for on-device computing.

**Results** Our evaluation on three standard benchmarks and an application-specific automotive dataset reveals substantial computational savings while maintaining or improving performance. Our model reduces computation time up to $415\times$ and minimizes memory requirements by $951\times$ compared to standard multi-task approaches. Compared to isolated models, it generates superior predictions and scales to hundreds of virtual sensors with an almost constant number of trainable parameters.

## 2 Related work

In this work, we design a scalable multi-task model for virtual sensor applications, aligning two previously disjoint literature branches for application-specific virtual sensors and foundation models for time series forecasting.

**Virtual sensors** Virtual sensors reduce costs, improve reliability, and estimate quantities that are difficult to measure directly. In automotive applications, they estimate dynamic parameters such as the side slip angle improving safety [3, 9], environmentally relevant nitrogen oxide emissions [10], battery temperatures [11], and battery state of health [2]. Further applications include determining ore mass flow from conveyor belt currents [5] in the mining domain, estimating moisture in food production [12], discriminating gases in chemical processes [13], and non-invasive measurements in healthcare [14] to detect disease from a patient's breath [4]. While early virtual sensors used handcrafted models [1], current work predominantly learns simple models from data [5]. Nevertheless, inputs are generally selected by experts and an application-specific model is trained, deployed, and maintained for each task. These isolated models limit scalability and cannot exploit synergies among virtual sensors. Moreover, application-driven evaluations lack consistent benchmarks.

**Time series foundation models** Foundation models forecast univariate [6, 15, 16, 17, 18, 19, 20] or multivariate time series [21, 22, 23, 24, 25], optionally using covariates [26, 27, 28, 29, 30], and can outperform specialized forecasters [6]. However, their computational cost limits deployment in sensor networks and on edge devices [7]. Moreover, they require a target signal's past context and therefore cannot infer new signals solely from other measurements, as virtual sensors require. Building on the assumption of self-contained time series [31], some methods restrict cross-signal interactions to reduce model complexity, often estimating importance through proxies such as correlation [32, 33, 34, 35, 36], but their unstructured sparsity provides no speedup on standard hardware.

We propose a unified and scalable model for multiple virtual sensors within a sensor network that leverages cross-task synergies while maintaining computational efficiency. It learns the set of relevant input signals, minimizing expert knowledge requirements and enhancing explainability. We provide a more detailed positioning of our work in appendix A.

## 3 A scalable multi-task model for virtual sensors

We design a unified model exploiting synergies among multiple virtual sensors. After introducing our basic model architecture, we propose a novel mechanism to selectively predict new signals that are not in the input set. We then design a mechanism to learn the set of relevant input signals per virtual sensor, minimizing expert knowledge, enhancing explainability, and improving efficiency. Finally, we introduce efficient training methods.

Let $\mathcal{Z} = \{z_i\}_{i=1}^{M}$ be a family of univariate real-valued time series (input signals) indexed by $i=1,\ldots,M$ and $\mathcal{Z}' = \{z'_j\}_{j=1}^{N}$ be another family of univariate real-valued time series (virtual sensors) indexed by $j=1,\ldots,N$, where $\mathcal{Z}$ and $\mathcal{Z}'$ are typically disjoint but can also be overlapping, or identical. Let a neural network $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\Phi}_L \circ \boldsymbol{\Phi}_{L-1} \circ \cdots \circ \boldsymbol{\Phi}_1(\mathbf{x})$ with parameters $\boldsymbol{\theta}$ consist of $L$ layers denoted as $\boldsymbol{\Phi}_l$, where each layer takes the output of the previous layer as input. Based on a model $\mathbf{f}_{\boldsymbol{\theta}}$, we predict $N$ virtual sensors $\mathcal{Z}'$ from $M$ input time series $\mathcal{Z}$. We assume that the neural network's input $\mathbf{x} \in \mathbb{R}^{s\times d}$ consists of $s$ tokens with dimension $d$. Here, the input tokens are generated by a tokenizer $\mathbf{g}$ out of input data $\mathcal{Z}$.

### 3.1 Base architecture

Based on the competitive results in Das et al. [15], we use a causal decoder-only transformer as the base architecture for our model $\mathbf{f}_{\boldsymbol{\theta}}$. We tokenize our input time series $z_i \in \mathcal{Z}$ individually. To this end, we first normalize each series to have zero mean and unit standard deviation and extract non-overlapping patches of length $p$ in a second step [37]. Subsequently, we embed each patch into the token dimension $d$ using a multi-layer perceptron $\mathbb{R}^p \to \mathbb{R}^d$ [15]. Finally, our tokenizer $\mathbf{g}$ concatenates all tokens into a single token sequence $\mathbf{x} \in \mathbb{R}^{s\times d}$ with time-based ordering as in figure 2. We adopt a similar approach for de-embedding predicted tokens into time-series representations. In the transformer, we employ a block-diagonal causal mask to impose temporal ordering without constraining interactions among time series.

### 3.2 Selective prediction of virtual sensors

We embed temporal and variate information into our tokens using sin-cos [38] and learned position embedding, respectively. While current time series literature that forecasts only input signals omits variate information [21, 30], embedding it explicitly is crucial for our proposed mechanism to predict virtual sensors. The temporal position embedding is shared among variates, while the variate embedding is shared orthogonally across time. We learn additive variate embedding vectors $\mathcal{V} = \{v_i\}_{i=1}^{M}$ for input signals $z_i \in \mathcal{Z}$ and $\mathcal{V}' = \{v'_j\}_{j=1}^{N}$ for virtual sensors $z'_j \in \mathcal{Z}'$, with $v_i, v'_j \in R^d$.

We design a novel mechanism to forecast new signals (virtual sensors) autoregressively in a selective way. To predict the $j$-th virtual sensor from the input signals $\mathcal{Z}$, we insert a zero-vector $\vec{0}$ as an initially empty prototype token into our token sequence $\mathbf{x}$ (see figure 2). Subsequently, the additive time and variate embedding $v'_j$ allows the neural network to learn the mapping of relevant target information into the prototype token. We train our model $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$ to forecast the next $p$ time steps of our virtual sensor $j$ into the future. For the following autoregressive steps, we leverage our model's previous prediction as new input for virtual sensor signals $z'_j$. This way, our model starts autoregressive generation from an empty prototype token in the first step and only needs to predict residuals from increasing context in the following. For input signals $\mathcal{Z}$, we use ground truth measurements in all autoregressive steps. This setup mimics practical deployment scenarios where ground truth measurements from physical input sensors are available, but virtual sensor outputs must be predicted autoregressively. To forecast multiple virtual sensors in a single step, we simply add multiple prototype tokens with their respective position embeddings to the network's input. Our approach enables flexible selection of which virtual sensors to predict in each generation step, ranging from a single sensor to all sensors simultaneously. This capability addresses two practical requirements in sensor networks. First, different signals are often needed at different frequencies; for example,

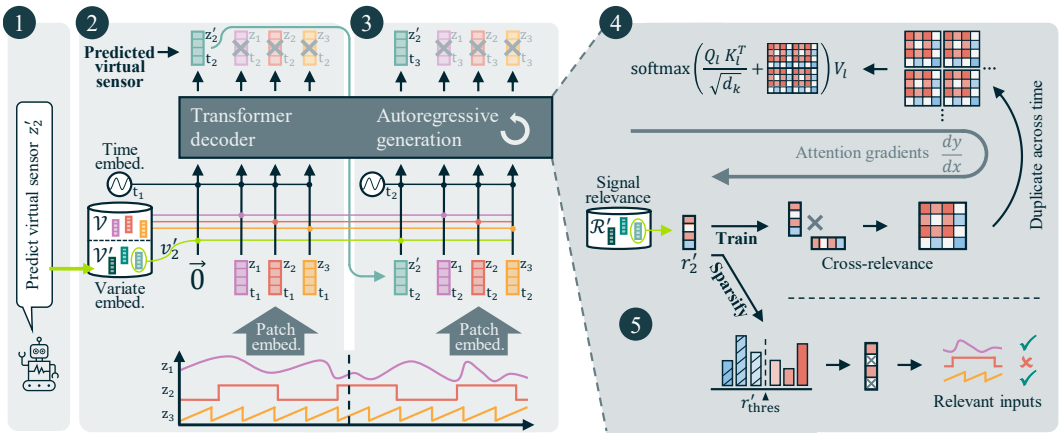



Figure 2: We present a unified model that can predict a user-selected target virtual sensor ($z_2'$, green) out of multiple possible virtual sensors. ① We first define which virtual sensor to forecast (here sensor $z_2'$). ② Next, we introduce an initially empty prototype token $\vec{0}$ and add the corresponding variate embedding $v_2' \in \mathcal{V}'$ to guide the neural network to predict the specified virtual sensor, based on available sensor signals ($z_1$, pink; $z_2$, red; $z_3$, orange). We divide input signals $z_1, z_2, z_3$ into patches, embed them into tokens, and add time and variate embedding $\mathcal{V}$ to them. Starting from empty prototypes in cycle $t_1$, ③ we forecast virtual sensors autoregressively using the model's previous prediction $z_2'$ as current input in step $t_2$. ④ Within the transformer, our architecture learns relevant input signals for each virtual sensor from attention gradients through trainable signal relevance vectors $\mathcal{R}'$, adding explainability. After selecting the corresponding signal relevance vector $r_2'$, we compute the cross-relevance for signal-to-signal communication, duplicate our mask across time to learn time-independent signal importance, and apply it as attention bias. ⑤ By later sparsifying signal relevance vectors $\mathcal{R}'$ based on a threshold $r'_{\text{thres}}$, we structurally prune irrelevant input signals to achieve considerable efficiency gains.

a vehicle's engine temperature requires higher update rates than ambient temperature due to faster control cycles. Second, by forecasting into the future, our model compensates for its own inference latency and enables predictive functionalities such as early fault detection [1].

### 3.3 Learning relevant input signals

In classical virtual sensor literature, the set of input signals $\mathcal{Z}$ is chosen by experts. However, in large sensor networks, this can be challenging and prone to error. To address this, we end-to-end learn the set of relevant input signals for each virtual sensor, minimizing expert knowledge and enhancing explainability and efficiency. To this end, we introduce signal relevance vectors $\mathcal{R}' = \{r_j'\}_{j=1}^{N}$ with $r_j' \in \mathbb{R}^{M+N}$ for every virtual sensor $j$. Each vector $r_j'$ encodes the relevance of all available sensors, both input signals and other virtual sensors, for predicting virtual sensor $j$, enabling hierarchical dependencies among virtual sensors. When predicting a virtual sensor, we select the corresponding relevance vector $r_j'$ and compute the cross-relevance for the communication of signal pairs using the outer product $(r_j' \cdot r_j'^{T}) \in \mathbb{R}^{(M+N)\times(M+N)}$. We then duplicate our relevance matrix across time (see figure 2), as we aim to learn global time-independent signal relevance. Finally, we apply this matrix $B$ in the attention score computation [38] of every transformer layer $l$ as static bias in equation (1). At the beginning of training, we initialize our relevance vectors $\mathcal{R}'$ with ones, as zero vectors would prevent gradient flow, utilizing the translation invariance of the softmax function.

$$\text{Attention}(Q, K, V)_l = \text{softmax}\left(\frac{Q_l K_l^T}{\sqrt{d_k}} + B\right) V_l \tag{1}$$

While current work determines signal importance using proxy measures such as correlation at the time series level [32], we directly learn relevance by backpropagating gradients from attention score computation to our relevance vectors $\mathcal{R}'$. As we share our mask across transformer layers, signal relevance is determined based on simple features in the first layers, but can leverage more complex representations in deeper layers. Note that our relevance mechanism is not designed to learn complex inter-signal dynamics, which are modeled by the transformer, but to identify the general set of input signals relevant to a given virtual sensor.

**Efficient processing via sparsification** To increase the efficiency of our unified model and enable applications on edge devices, we sparsify our relevance vectors $r'_j \in \mathcal{R}'$ at late training stages and inference time. To this end we mask unimportant signals in $r'_j$ with $-\infty$ based on a threshold[1] $r'_{\text{thres}}$, which controls the sparsity of the attention map and which signals are used for forecasting. Our masking is equivalent to completely disregarding unimportant signals as the outer product operation and duplication across time generate fully masked rows and columns in our attention bias $B$ in a structured way. This enables us to learn individual input signal sets $\{\mathcal{Z}_j \subseteq \mathcal{Z}\}_{j=1}^{N}$ for every virtual sensor. Together with our virtual sensor selection mechanism in section 3.2, we achieve maximum efficiency by querying our model only for currently requested virtual sensors and using only relevant input signals for these sensors. This reduces the number of tokens $s$ to be processed to a minimum. When requesting multiple virtual sensors simultaneously, we simply average the corresponding attention biases $B$ and union the individual input signal sets.

### 3.4 Efficient training strategies

We optimize the parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ of our model $\mathbf{f}_{\boldsymbol{\theta}}$ to minimize the mean squared error only of virtual sensor forecasts. Predicted input signals are not included in loss computation (see figure 2). Training all virtual sensors $z'_j \in \mathcal{Z}'$ simultaneously in every training step, however, is impractical as all signal relevance vectors $r'_j \in \mathcal{R}'$ would be updated with identical gradients, resulting in identical input sets $\mathcal{Z}_j \subseteq \mathcal{Z}$ for all sensors[2]. Training a single virtual sensor at a time, instead, is inefficient, prolonging the training process due to less informative gradients. As a consequence, we select a random subset of $N_{\text{train}} < N$ virtual sensors in every training iteration. Similar to dropout [39], this reduces the correlation among relevance vectors.

**Teacher forcing** Our model generates virtual sensor predictions autoregressively as described in section 3.2. Standard training via backpropagation through time [40] would incur high memory and computational costs due to long recurrent gradient paths and multiple forward passes per training step. Thus, we employ teacher forcing [41], removing the recurrent path in figure 2, and use ground truth data for virtual sensors during training. This enables simultaneous training on multiple context lengths with a single model iteration, yielding short gradient paths and fast training cycles. However, teacher forcing limits the model's ability to learn error propagation strategies. To mitigate this, we explore hybrid training schedules combining teacher forcing and backpropagation through time in section 5.6.

## 4 Experiments

We systematically train and evaluate our unified model on $16$ virtual sensor tasks and $4$ datasets, demonstrating performance and efficiency gains of our scalable architecture. In appendix B, we provide further experimental details.

**Datasets** We use Traffic, Electricity, and Solar as standard time series forecasting benchmarks [8]. For application-specific evaluation in a realistic deployment scenario we use an automotive CAN bus dataset. The CAN bus (controller area network bus) enables communication between electronic control units in vehicles [42]. Our dataset contains $1713$ variates including continuous sensor measurements, categorical states, and event-based signals. It features substantially more complex signal interactions compared to the standard time series datasets, e.g., the categorical driving mode (eco, sport, comfort, ...) influences pedal curves, stability, power delivery, efficiency, and generally driving behavior. Our CAN bus dataset is recorded with an Audi e-tron electric vehicle covering $287$ hours of driving and over $17\,500\,\text{km}$ on German roads. It is synchronized to $100\,\text{ms}$ granularity and over $1000\times$ larger than the other datasets with approximately $18\,\text{B}$ time series samples.

**Virtual sensors** For the Traffic, Electricity, and Solar datasets, we randomly select our set $\mathcal{Z}'$ of $N = 16$ virtual sensors, as all variates present the same quantity, measured at different locations. In the CAN bus dataset, every signal has a distinct purpose. Here, we hand-pick $N = 16$ virtual sensors with clear intuitive interpretations, including electrical powertrain voltages, currents, and temperatures, coolant intake and outflow temperatures and flow-rates, torques and velocities.

[1] Relevance is learned as soft, differentiable scores before hard thresholding removes already-downweighted signals without destabilizing optimization.

[2] Adding random noise mitigates identical gradients but cannot ensure the correct mapping of input signals to virtual sensors.

**Models** Throughout our experiments we train and evaluate models with $L = 4$ transformer layers, $4$ heads, and token dimension $d = 512$. All our models and baselines have similar capacity with $7.4\,\text{M}$ trainable parameters for Traffic, $7.2\,\text{M}$ for Electricity, $7.1\,\text{M}$ for Solar, and $7.8\,\text{M}$ for the CAN bus dataset, due to different numbers of learned variate embedding vectors $\mathcal{V}$. The signal relevance vectors $\mathcal{R}'$ account for only $13.8\,\text{k}$, $5.1\,\text{k}$, $2.2\,\text{k}$, and $27.4\,\text{k}$ parameters, respectively. We utilize patches with length $p = 32$ to divide time series into tokens. For evaluation, we perform $6$ autoregressive generation cycles. For LSTM-based literature models, we likewise employ $4$ layers with a hidden dimension of $512$, yielding strong baselines with a larger number of parameters.

# 5 Results

The overarching goal of our experiments is to investigate the feasibility of a unified virtual sensor model. First, we demonstrate efficiency and superior scaling properties (sec. 5.1). In particular, we investigate efficiency gains through focusing only on relevant input signals (sec. 5.2) and through flexibly predicting a subset of virtual sensors (sec. 5.3). We then explore the scaling of our unified model to hundreds of virtual sensors (sec. 5.4) and highlight explainability benefits of our architecture (sec. 5.5). Finally, we investigate efficient training techniques (sec. 5.6) including transfer learning (app. C.9).

## 5.1 Main experiments

The application-specific nature of virtual sensor methods has resulted in a lack of consistent benchmarks. Meanwhile, time series foundation models cannot predict new, unseen signals from other measurements, limiting their applicability to virtual sensor tasks (see appendix A for a detailed comparison). We address both gaps through two contributions. First, we establish a baseline on standardized benchmarks. Here, we train an individual model for every sensor as done in prior work [3]. Second, we introduce a unified baseline that predicts all virtual sensors from available input signals at every step and systematically introduce and ablate novel components that enable the training and deployment of a large-scale multi-task model: (i) variate embeddings, (ii) learned relevance masking, (iii) sparsification, and (iv) sensor selection.

**Individual models** Following state-of-the-art approaches from literature, we train $16$ isolated models to predict a single virtual sensor each. Prior works predominantly rely on standard recurrent architectures [3, 10, 11] and we thus implement LSTMs [43] as a baseline. For an architecture-independent evaluation, we additionally train individual models with our transformer architecture. Our unified approach outperforms all literature baselines and is superior in $3$ out of $4$ settings in our atomic comparison in tables 1 and 9, demonstrating its effectiveness. Remarkably, our unified model for $16$ virtual sensors achieves comparable average inference speed and, in some cases, even exceeds the efficiency of isolated models trained for a single virtual sensor. Moreover, training our models using backpropagation through time, analogous to LSTM-based virtual sensors, further reduces the MSE to $0.083$ on the CAN bus dataset in table 3, improving considerably over the LSTM baseline. Relying on foundational time series understanding, our multi-task model scales superior compared to training, maintaining, and deploying $16$ isolated models with in total $16\times$ the number of parameters, which we further investigate in section 5.4.

Table 1: Comparison of a simple **unified model** predicting all 16 virtual sensors, where we subsequently introduce our proposed mechanisms, with 16 **individual models** predicting a single virtual sensor each. We list MSE, inference time, dynamic CUDA peak memory, and the number of trainable parameters for Traffic and CAN bus datasets.

| Architecture | Traffic | | | | CAN bus | | | |
|---|---|---|---|---|---|---|---|---|
| | MSE | Time | Mem. | Param. | MSE | Time | Mem. | Param. |
| **Individual models** | | | | | | | | |
| LSTM (literature) | 0.316 | 0.33 ms | 0.0010 GB | 145.9 M | 0.124 | 0.35 ms | 0.0011 GB | 174.7 M |
| Transformer (ours) | 0.285 | 16.62 ms | 0.90 GB | 118.4 M | 0.318 | 58.05 ms | 3.42 GB | 124.8 M |
| **Unified model** | 0.556 | 16.70 ms | 0.90 GB | 7.0 M | 0.727 | 58.19 ms | 3.46 GB | 7.0 M |
| + Variate embedding $\mathcal{V}, \mathcal{V}'$ | 0.275 | 16.76 ms | 0.90 GB | 7.4 M | 0.119 | 58.23 ms | 3.46 GB | 7.8 M |
| + Signal relevance $\mathcal{R}'$<br>+ $N_{\text{train}} = 4$ | 0.272 | 16.85 ms | 0.90 GB | 7.4 M | 0.136 | 58.41 ms | 3.46 GB | 7.8 M |
| + Sparse input sets $\mathcal{Z}_j$<br>+ Sensor selection | 0.282 | 0.87 ms | 0.0352 GB | 7.4 M | 0.121 | 0.14 ms | 0.0036 GB | 7.8 M |

**Unified models** We systematically introduce our novel mechanisms in a simple unified baseline. **(i)** Our results in tables 1 and 9 demonstrate that variate embeddings are crucial for predicting virtual sensors. This is contrary to current time series forecasting literature that can only predict input signals [21, 30]. **(ii)** Further, introducing signal relevance vectors and training only a subset of virtual sensors per iteration $N_{\text{train}}/N = 4/16$ adds explainability and does not significantly affect predictive quality or efficiency. **(iii)** Additionally, we sparsify our learned signal relevance to determine individual input signal subsets $\mathcal{Z}_j \subseteq \mathcal{Z}$ for each virtual sensor. **(iv)** Together with our sensor selection mechanism, where we flexibly query our model to predict only currently required virtual sensors, we achieve high efficiency gains. Remarkably, we improve inference time by $415\times$ and reduce memory requirements by $951\times$ without affecting MSE on the CAN bus dataset. This enables our unified model to run on edge devices in large sensor networks. In appendix C.1 we present results for Electricity and Solar.

### 5.2 Different MSE efficiency trade-offs for variable test-time computation

To distinguish between important input signals $\mathcal{Z}_j \subseteq \mathcal{Z}$ and unimportant ones for each virtual sensor, we threshold our signal relevance vectors $r'_j \in \mathcal{R}'$ with $r'_{\text{thres}}$. Varying this threshold, we achieve different trade-offs between predictive quality and efficiency. Sparsification towards the end of training allows the model to adapt to reduced input sets, while sparsification only during inference might be more flexible. To compare both approaches, we train models with different sparsity thresholds.

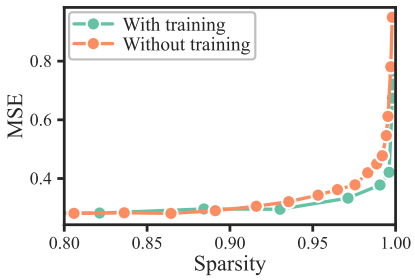



Figure 3: Varying our signal importance threshold $r'_{\text{thres}}$ during training or inference generates smooth trade-offs between input set sparsity and MSE on the Traffic dataset.

Our results in figure 3 show that we can achieve high input sparsity without affecting predictive quality as most input signals are unimportant for individual virtual sensor tasks. Towards very high sparsity, MSE increases as we now also remove informative signals from the input sets. The approach is highly flexible in practice, as sparsification can be applied solely at inference time with minimal performance degradation. On the Traffic dataset, models trained with sparse inputs achieve only marginal improvements over inference-time sparsification. Moreover, on CAN bus data, both sparsification approaches exhibit similar performance (see appendix C.2). This allows inference-time sparsification to flexibly adapt to varying computational budgets for each prediction, without requiring the loading of different model checkpoints. Only the set of input signals is dynamically adjusted. This is particularly relevant in real-world applications where computational resources may be limited during critical situations, or where faster inference is required to ensure timely decision-making.

### 5.3 Sensor selection maximizes efficiency without affecting predictive quality

In practical applications, virtual sensors are often required at varying frequencies. For example, engine-related measurements typically demand higher sampling rates than ambient signals in vehicles. Moreover, different virtual sensors generally rely on distinct sets of relevant input signals. To predict multiple virtual sensors simultaneously, we union their respective input signal sets, reducing sparsity. In the limiting case of equally sized, disjoint input sets, sparsity decreases linearly with the number of predicted virtual sensors. To address this, we introduce a sensor selection mechanism that enables the model to predict only the virtual sensors currently required, ensuring maximum efficiency when a specific signal is requested. In the following, we demonstrate the effectiveness of our approach by predicting an increasing number of virtual sensors simultaneously. By selectively predicting subsets of virtual sensors – ranging from all sensors to individual sensors – per autoregressive step, we increase sparsity from $58.4\,\%$ to $81.9\,\%$ on the Traffic dataset (see figure 4). This translates to an additional $4.8\times$ inference acceleration and $4.6\times$ memory reduction. For CAN bus data efficiency gains are even higher with $7.9\times$ and $13.4\times$, respectively (see appendix C.3). As more virtual sensors are predicted simultaneously, sparsity decreases sub-linearly since related sensors often share common input signals. The number of simultaneously predicted signals does not affect predictive quality in our experiments.

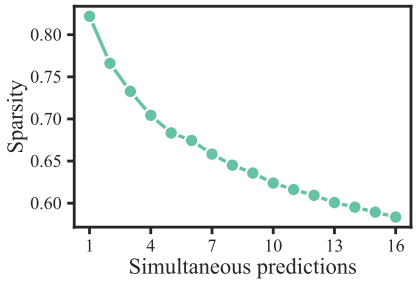



Figure 4: Sparsity increases as fewer virtual sensors are predicted simultaneously through our sensor selection mechanism. Results are shown for the Traffic dataset.

### 5.4 Scaling to hundreds of virtual sensors

Classical approaches train a single model for each virtual sensor. As the number of virtual sensors grows, this causes a linear increase in training and deployment effort and model parameters, making it impractical at scale. In contrast, our unified model maintains an almost constant number of trainable parameters when scaling to large sensor networks. However, modeling capacity might be a limitation for our fixed-size neural network. To investigate this, we train models to predict increasing numbers of virtual sensors $N_{\text{Traffic}} \in \{4, 8, 16, ..., 512\}, N_{\text{CAN}} \in \{4, 8, 12, 16\}$ and evaluate them on a constant set of $4$ core sensors.
As the number of learned virtual sensors increases, predictive quality of our models remains stable with a standard deviation of $0.028$ on Traffic and $0.008$ on the CAN bus dataset. This demonstrates that our multi-task model possesses sufficient capacity to learn hundreds of virtual sensors. We argue that our architecture scales gracefully by leveraging shared knowledge across virtual sensors.

### 5.5 Explainability of virtual sensors

Beyond substantial efficiency improvements, learning the relevant input signal sets for each virtual sensor enhances explainability, which is particularly important in safety-critical applications. Furthermore, it allows for robustness analysis to identify which virtual sensors remain functional under conditions of incomplete input availability. Here, we investigate the learned signal relevance in detail.

**Qualitative assessment** We utilize expert knowledge to analyze the learned input signal sets for the individual virtual sensors. For this purpose, we use the CAN bus dataset, as every virtual sensor can be associated with an interpretable physical quantity. We find that our relevance vectors learn meaningful selections. Virtual sensors for temperatures mainly rely on other temperatures together with coolant flow rates, while virtual sensors for torques utilize engine speeds, velocities, and acceleration signals. Sensors for electrical voltages, in contrast, rely on other voltages and electrical current signals as we visualize in appendix C.5. This facilitates the benefit of learning individual input sets for each virtual sensor. Overall, the input sets align well with domain expert assessments, confirming their plausibility.

**Quantitative assessment** We learn input signal relevance through attention gradients. Here, we compare our approach to choosing input signals randomly, utilizing signal correlations between inputs and virtual sensors as a proxy metric, and using SHAP [44] as a post-hoc relevance attribution method. For a fixed sparsity, our approach substantially outperforms random, correlation-, and SHAP-based input signal sets as our results in table 2 show. We argue that learning relevant signals directly through attention gradients exploits both semantically simple features and complex ones in deeper neural network layers. In contrast, correlation as a proxy only relies on simple time series features, which may represent a considerable limitation for complex sensor networks, such as the CAN bus. Further, end-to-end learning relevance throughout the training process improves over post-hoc attribution via SHAP.

Table 2: Comparison of different methods to determine signal relevance on the Traffic and the CAN bus dataset. **Best** in bold.

| Signal relevance | Traffic | | CAN bus | |
|---|---|---|---|---|
| | Sparsity | MSE | Sparsity | MSE |
| Random | 0.820 | 0.319 | 0.977 | 0.601 |
| Correlation | 0.820 | 0.492 | 0.978 | 0.267 |
| SHAP | 0.820 | 0.311 | 0.978 | 0.209 |
| Learned | 0.821 | **0.282** | 0.978 | **0.121** |

**Synthetic data** Our model learns to identify input signals based on their variate embedding, to attribute importance scores to them through signal relevance vectors, and to compute the right virtual sensor according to our sensor selection mechanism. Here, we validate the seamless interplay of these three mechanisms, leveraging synthetic data. To this end, we generate a dataset with $M = 1000$ random, uncorrelated signals. We train a model to replicate $N = 2$ input signals $z_{100}, z_{200}$ as virtual sensors $z'_1, z'_2$, respectively. Throughout the learning process, our model subsequently identifies both input signals (see figure 5). It attributes at least $4.22\times$ higher relevance to them compared to all other $998$ signals. Finally, it selects the correct signal as virtual sensor output. This experiment demonstrates the seamless interaction of identification, relevance, and selection mechanisms in our architecture.

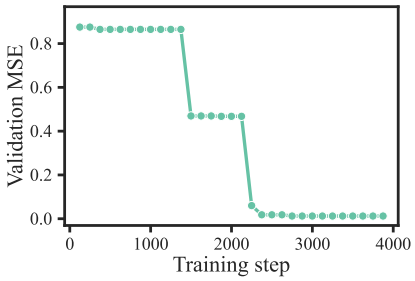



Figure 5: Our architecture subsequently identifies 2 virtual sensors among 1000 signals in our synthetic random dataset during training, reducing the validation loss in a stepwise manner.

### 5.6 Efficient training strategies

**Training virtual sensors simultaneously** Learning multiple virtual sensors per training iteration improves training efficiency. However, training all virtual sensors in every cycle ultimately results in identical signal relevance vectors, as they receive identical gradients and therefore identical input signal sets for all virtual sensors. We explore this trade-off in detail by training models with $N_{\text{train}} \in \{1, 2, 3, 4, 6, 8, 10, 13, 16\}$ and measure the average similarity of the sensors' input signal sets $\mathcal{Z}_j \subseteq \mathcal{Z}$. Training a single virtual sensor at a time results in most specialized input sets that share $12.7\,\%$ of the signals on average as in figure 6. However, training generally requires more iterations due to less informative gradients. In contrast, learning multiple virtual sensors simultaneously results in more similar input signal sets, but makes training more efficient. In practice, we find $N_{\text{train}} = 4$ to be a good tradeoff between training time and input set separation. We present CAN bus dataset results in appendix C.4 and additional analysis in appendix C.6.

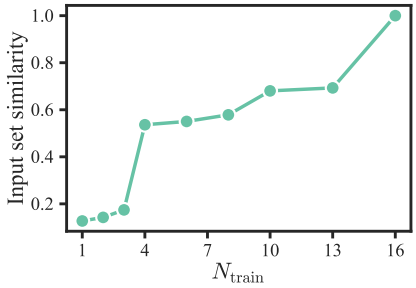


Figure 6: Learning different numbers of virtual sensors $N_{\text{train}}$ in a single training iteration influences the similarity of their input signal sets on the Traffic dataset.

**Teacher forcing** Our model generates virtual sensor predictions autoregressively during inference. For training, however, we utilize teacher forcing as a more efficient strategy. Compared to backpropagation through time, teacher forcing requires only a single model forward pass, substantially reducing memory usage and training time. However, this approach prevents the model from learning to correct for accumulated errors during inference. We compare both training strategies in the following.

Table 3: Comparison of teacher forcing, back propagation through time (BPTT), and a combination of both for training our model on the CAN bus dataset. We list MSE, time per training iteration, and dynamic CUDA peak memory. **Best** in bold.

| Training strategy | MSE | Time | Mem. |
|---|---|---|---|
| Teacher forcing | 0.136 | **0.19 s** | **16.6 GB** |
| BPTT | ✗ | 1.11 s | 72.0 GB |
| BPTT fine tuning | **0.083** | 1.10 s | 72.0 GB |

Even with extensive hyperparameter tuning, training with backpropagation through time diverges. We argue that this is due to long recurrent gradient paths. Conversely, training with teacher forcing converges smoothly, while being $5.8\times$ faster and consuming $4.3\times$ less memory, as shown in table 3. Throughout our paper, the low inference MSE indicates that models trained with teacher forcing successfully predict initial tokens from empty prototypes, as well as subsequent tokens, and do not suffer from extensive error accumulation in autoregressive forecasting. We do not find a significant difference when comparing the predictive loss of initial tokens to that of subsequent ones. We provide further evidence for the effectiveness of teacher forcing in appendix C.8. Additionally, we experiment with backpropagation through time as a fine-tuning step starting from a teacher-forcing checkpoint. This hybrid schedule allows us to maintain efficient training for most of the process while explicitly introducing error propagation toward the end of training. Table 3 shows that this approach further improves predictive performance.

**Transfer learning** We explore transfer learning to introduce a new virtual sensor into an already trained model with limited data to explore the generalization ability of our unified approach. Our results in appendix C.9 show that transfer learning achieves the same MSE as training from scratch, while only using $10\,\%$ of training data and $3.6\,\%$ of computation.

## 6 Conclusion

In this work, we present the first unified model for virtual sensors. Our architecture simultaneously predicts variable sensor subsets, exploiting task synergies while focusing only on relevant input signals to ensure both explainability and efficiency. In our large-scale evaluation on standard and application-specific datasets with billions of samples, we demonstrate substantial efficiency gains, flexible test-time computation trade-offs, and scalability to hundreds of virtual sensors. This enables practical deployment in large sensor networks and on resource-constrained devices. We hope our multi-task model unlocks new virtual sensor applications and contributes to more environmentally sustainable time series modeling.

**Limitations** In our work, we do not conduct hyperparameter search for all possible parameters due to the high cost of training large-scale models. We expect even better results with optimized settings.

## Disclaimer

The results, opinions, and conclusions expressed in this publication are not necessarily those of Volkswagen Aktiengesellschaft.

## Funding and competing interests

No funding or competing interests beyond those already disclosed through the authors' affiliations supported this work.

## References

[1] Pedro Albertos and Graham C Goodwin. Virtual sensors for control applications. In *Annual Reviews in Control*, 2002.

[2] Friedrich von Bülow, Joshua Mentz, and Tobias Meisen. State of health forecasting of lithium-ion batteries applicable in real-world operational conditions. In *Journal of Energy Storage*, 2021.

[3] Zygimantas Ziaukas, Alexander Busch, and Mark Wielitzka. Estimation of vehicle side-slip angle at varying road friction coefficients using a recurrent artificial neural network. In *IEEE Conference on Control Technology and Applications*, 2021.

[4] Jiawang Hu, Hao Qian, Sanyang Han, Ping Zhang, and Yuan Lu. Light-activated virtual sensor array with machine learning for non-invasive diagnosis of coronary heart disease. In *Nano-Micro Letters*, 2024.

[5] Francisco José Dos Santos Diniz, Saulo N. Matos, Jó Ueyama, Leandro S. Marcolino, Eduardo José Da Silva Luz, and Gustavo Pessin. Uncertainty quantification in measuring ore mass flow rate with data-driven soft sensors. In *IEEE International Instrumentation and Measurement Technology Conference*, 2025.

[6] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. In *Transactions on Machine Learning Research*, 2024.

[7] Leon Götz, Marcel Kollovieh, Stephan Günnemann, and Leo Schwinn. Efficient time series processing for transformers and state-space models through token merging. In *International Conference on Machine Learning*, 2025.

[8] Rakshitha Wathsadini Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[9] Abinav Kalyanasundaram, Karthikeyan Chandra Sekaran, Philipp Stäuber, Michael Lange, Wolfgang Utschick, and Michael Botsch. Uncertainty-aware hybrid machine learning in virtual sensors for vehicle sideslip angle estimation. In *IEEE Intelligent Vehicles Symposium*, 2025.

[10] Ivan Arsie, Andrea Cricchio, Matteo De Cesare, Francesco Lazzarini, Cesare Pianese, and Marco Sorrentino. Neural network models for virtual sensing of nox emissions in automotive diesel engines with least square-based adaptation. In *Control Engineering Practice*, 2017.

[11] Safieh Bamati, Hicham Chaoui, and Hamid Gualous. Enhancing battery thermal management with virtual temperature sensor using hybrid cnn-lstm. In *IEEE Transactions on Transportation Electrification*, 2024.

[12] Liuping Wang, Charlie Chessari, and Elizabeth Karpiel. Inferential control of product quality attributes - application to food cooking extrusion process. In *Journal of Process Control*, 2001.

[13] Zafer Ankara, Thomas Kammerer, Andreas Gramm, and Andreas Schütze. Low power virtual sensor array based on a micromachined gas sensor for fast discrimination between h2, co and relative humidity. In *Sensors and Actuators B: Chemical*, 2004.

[14] Filippo Attivissimo, Vito Ivano D'Alessandro, Luisa De Palma, Anna Maria Lucia Lanzolla, and Attilio Di Nisio. Non-invasive blood pressure sensing via machine learning. In *Sensors*, 2023.

[15] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv:2310.10688*, 2023.

[16] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv:2310.08278*, 2023.

[17] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

[18] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In *International Conference on Machine Learning*, 2024.

[19] Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. In *International Conference on Machine Learning*, 2025.

[20] Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. *arXiv:2505.23719*, 2025.

[21] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. *arXiv:2402.02592*, 2024.

[22] Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. Units: A unified multi-task time series model. In *Advances in Neural Information Processing Systems*, 2024.

[23] Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability. *arXiv:2407.07874*, 2024.

[24] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In *Advances in Neural Information Processing Systems*, 2024.

[25] Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era. *2605.20119*, 2026.

[26] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv:2304.08424*, 2023.

[27] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *Advances in Neural Information Processing Systems*, 2024.

[28] Andreas Auer, Raghul Parthipan, Pedro Mercado, Abdul Fatir Ansari, Lorenzo Stella, Bernie Wang, Michael Bohlke-Schneider, and Syama Sundar Rangapuram. Zero-shot time series forecasting with covariates via in-context learning. *arXiv:2506.03128*, 2025.

[29] Prathamesh Patil, Amit Varshney, Manoj Cherukumalli, Harsh Deshpande, Leonard Eun, Dushyant Sahoo, and Naren Chittar. MORPHEUS: A foundation model for multivariate time series forecasting. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025.

[30] Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv:2510.15821*, 2025.

[31] Yu Chen, Nathalia Céspedes, and Payam Barnaghi. A closer look at transformers for time series forecasting: Understanding why they work and where they struggle. In *International Conference on Machine Learning*, 2025.

[32] Seunghan Lee, Taeyoung Park, and Kibok Lee. Dataset-driven channel masks in transformers for multivariate time series. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2026.

[33] Jialin Chen, Jan Eric Lenssen, Aosong Feng, Weihua Hu, Matthias Fey, Leandros Tassiulas, Jure Leskovec, and Rex Ying. From similarity to superiority: Channel clustering for time series forecasting. In *Advances in Neural Information Processing Systems*, 2024.

[34] Qinshuo Liu, Yanwen Fang, Pengtao Jiang, and Guodong Li. Dgcformer: Deep graph clustering transformer for multivariate time series forecasting. *arXiv:2405.08440*, 2024.

[35] Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. Duet: Dual clustering enhanced multivariate time series forecasting. In *Association for Computing Machinery SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

[36] Yifan Hu, Guibin Zhang, Peiyuan Liu, Disen Lan, Naiqi Li, Dawei Cheng, Tao Dai, Shu-Tao Xia, and Shirui Pan. Timefilter: Patch-specific spatial-temporal graph filtration for time series forecasting. In *International Conference on Machine Learning*, 2025.

[37] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[39] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, 2014.

[40] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, Institute for Cognitive Science, University of California, San Diego, 1985.

[41] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. In *Neural computation*, 1989.

[42] ISO 11898. Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication, 1993.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural Computation*, 1997.

[44] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017.

[45] Antonio Leanza, Giulio Reina, and José-Luis Blanco-Claraco. A factor-graph-based approach to vehicle sideslip angle estimation. In *Sensors*, 2021.

[46] Alexander Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan-Rhys Griffiths, Alexandre Maravel, Jianye Hao, Jun Wang, Jan Peters, and Haitham Bou Ammar. Hebo: Pushing the limits of sample-efficient hyperparameter optimisation. In *Journal of Artificial Intelligence Research*, 2022.

[47] Diederik P. Kingma and Jimmy L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

# A Positioning of this work

Our overarching goal is to make virtual sensors scalable to large sensor networks. This positions our work between two previously disjoint literature branches. Classical virtual sensor approaches solve the relevant application problem, but usually train an isolated model with expert-selected inputs for each target sensor. Time series foundation models, in contrast, demonstrate the benefit of shared time series knowledge, but forecast target signals from their own past context. Our work brings the shared-model principle of time series foundation models to the virtual sensor setting, where target signals must be inferred from other measurements rather than extrapolated from their own histories.

**Virtual sensor literature** Classical virtual sensors typically train an isolated and application-specific model with expert-selected inputs per virtual sensor, limiting scalability. We, in contrast, use one shared backbone for multiple virtual sensors, achieving superior scalability to hundreds of sensors and better prediction quality at comparable inference efficiency. Our approach learns relevant input signal sets, adding explainability without expert selection. Our unified model enables practical virtual sensor deployment in large-scale sensor networks, drastically reducing the training, deployment, and maintenance effort compared to hundreds of isolated models.

**Time series foundation model literature** Time series foundation models demonstrate the benefit of shared time series knowledge across tasks. However, current foundation models rely on in-context learning to forecast future values of signals for which the past context is provided. Virtual sensors solve a different problem: the target signal may be absent from the input context and must be inferred solely from other measurements. Therefore, in-context forecasting alone cannot solve this task, making current time series foundation models incompatible with virtual sensors. We demonstrate that they collapse to zero-prediction in the virtual sensor setting, while our model remains predictive (see below). Our work adapts the unification idea of foundation models to learning multiple virtual sensors in a single model.

**Zero-shot generalization** Unlike time series foundation models, we do not claim generalization to unseen target sensors, as their past context is absent from the input, hindering zero-shot generalization. Instead, we evaluate the practical case of adding a new virtual sensor via transfer learning. This reaches the performance of training from scratch with only a small fraction of the data and computation in appendix C.9.

**Multi-task model** We use "multi-task" in the virtual sensing sense that each target virtual sensor is a distinct prediction task with its own output identity and relevant input subset, while all tasks share one backbone. Our unified approach can predict multiple virtual sensors, in contrast to the literature that only focuses on predicting a single output or task. We do not claim that our model handles different task types such as classification, anomaly detection, and imputation.

**Architectural novelty** Our backbone architecture deliberately builds on established decoder-only Transformer and patch-tokenization components to isolate our virtual sensor contributions from backbone effects. The novelty of our work is the combination of mechanisms required to make a single model usable for multiple virtual sensors: (i) target-specific prototype tokens and variate embeddings for predicting absent target signals, (ii) sensor selection for flexibly requesting subsets of virtual sensors, (iii) target-specific relevance vectors that learn input signal sets without expert selection, and (iv) structured sparsification that translates learned relevance into runtime and memory savings.

**Baseline models** LSTMs are common in application-specific virtual sensor literature, making them a relevant practical baseline. We also include transformer baselines with the same backbone as ours to separate backbone effects from the contributions of unification, target-specific prototype tokens and variate embeddings, sensor selection, relevance learning, and structured sparsification. This atomic comparison shows that our performance improvements are not merely due to the transformer architecture.

Here, we additionally compare with time series foundation models in two ways, as shown in table 4. Please note that they solve standard forecasting tasks and are not directly applicable to virtual sensors as discussed above. (i) In our *virtual sensor* setting, where the target signal's own past is unavailable, all time series foundation models collapse to zero-prediction as expected. This demonstrates that current time series foundation models are incompatible with virtual sensors. In contrast, our model performs well. (ii) In the *standard forecasting* setting, the foundation models additionally receive the target signal's past context. This is not the virtual sensor setting and gives them an unfair advantage over our model and we report it as a reference. Here, all time series foundation models perform poorly on Solar and generate reasonable forecasts on the other datasets. Even in this setting, our approach achieves superior forecasting performance.
Beyond predictive quality, these foundation models lack our efficiency mechanisms for selecting requested target sensors, learning target-specific input sets, and structurally pruning inputs and therefore cannot realize the corresponding runtime and memory savings.
Combining these time series foundation models with our proposed mechanisms (prototype tokens, variate embeddings, sensor selection, relevance vectors, and structured sparsification) to transform them from standard forecasting to virtual sensors is theoretically possible but would result in entirely new architectures.

**Datasets** For our evaluation, we use standard time series forecasting datasets with many variates to stress-test our main contributions. Our model has to learn relevant input subsets from a large pool of possible physical sensors, and the output selection mechanism is essential because multiple virtual sensor outputs can be requested at different rates. Traffic, Electricity, and Solar allow us to construct controlled benchmarks with many candidate input signals and multiple virtual sensor outputs, while the CAN bus dataset provides a realistic automotive setting with physically meaningful battery, engine, temperature, torque, current, voltage, and coolant-related virtual sensors. In contrast, common virtual sensor datasets are already small and expert-curated, such as the sideslip-angle dataset [45] with 13 input signals and a single predefined target. Such datasets are generally useful for single-task virtual sensor models but not for demonstrating the scalability of our multi-task approach. We nevertheless apply our model to sideslip-angle estimation as a standard virtual sensor task, improving the literature RMSE of $0.57^\circ$ by $5\,\%$ down to $0.54^\circ$, showing that our approach extends to conventional single-target virtual sensors.

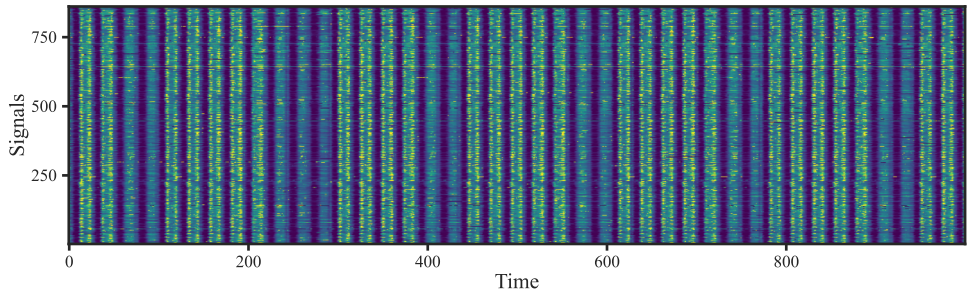
Table 4: MSE comparison of our unified virtual sensor model with time series foundation models in the virtual sensor setting and the standard forecasting setting.

| Dataset | Virtual sensor | | | | | Standard forecasting | | |
|---|---|---|---|---|---|---|---|---|
| | Ours | Chronos 2 | TimesFM 2.5 | Toto 2 | Zero-prediction | Chronos 2 | TimesFM 2.5 | Toto 2 |
| CAN bus | **0.121** | 0.779 | 0.779 | 0.779 | 0.779 | 0.368 | 0.304 | 0.326 |
| Traffic | **0.282** | 1.121 | 1.121 | 1.121 | 1.121 | 0.584 | 0.634 | 0.693 |
| Electricity | **0.173** | 0.887 | 0.887 | 0.887 | 0.887 | 0.221 | 0.339 | 0.295 |
| Solar | **0.128** | 0.967 | 0.967 | 0.967 | 0.967 | 1.497 | 1.873 | 1.595 |

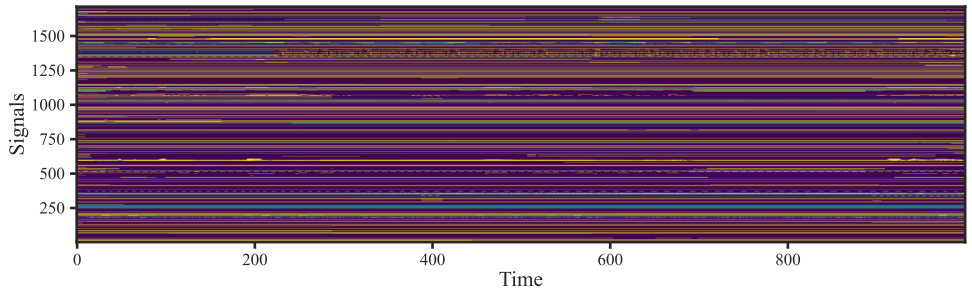
# B Experiments

In this section, we provide additional information about our experimental settings and resources.

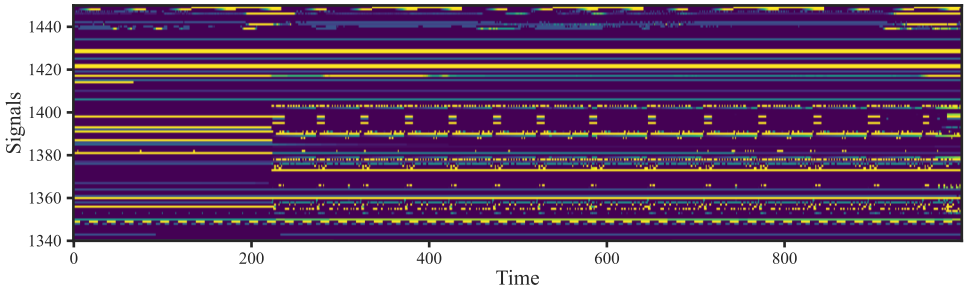
**Datasets** Throughout our experiments, we utilize Traffic, Electricity, and Solar as standard benchmark datasets and a CAN bus dataset for evaluating our architecture in more complex and realistic deployment scenarios. Traffic consists of 862 sensors measuring the road occupancy in the San Francisco Bay Area every hour and contains roughly 15 M time series samples. Electricity records the hourly power demand of 321 households and consists of approximately 8 M time series samples. Solar measures power production of 137 photovoltaic plants in 10 minute intervals with roughly 7 M time series samples. Our large-scale CAN bus dataset records 1713 time series within a modern electric vehicle. It consists of continuous sensor measurements, categorical states, and event-based signals, in 100 ms granularity. Here, each signal has a distinct and interpretable purpose.
While time series in the Traffic dataset follow roughly a daily and weekly periodic structure, e.g., due to work traffic, signal interactions in our CAN bus dataset are substantially more complex and diverse. We visualize these different characteristics in figure 7. Nevertheless, the Traffic dataset still exhibits substantial local heterogeneity across different roads (see figure 15).




(a) Traffic




(b) CAN bus




(c) CAN bus (zoomed)

Figure 7: Signals on the Traffic and the CAN bus dataset show different temporal characteristics.

**Virtual sensors** We chose our $N = 16$ virtual sensors $\mathcal{Z}'$ randomly for the standard time series datasets, including variate IDs $\{74, 88, 113, 173, 280, 311, 312, 352, 355, 478, 539, 646, 650, 715, 775, 776\}$ for Traffic, $\{3, 15, 17, 19, 69, 115, 116, 152, 154, 210, 224, 226, 267, 278, 280, 316\}$ for Electricity, and $\{10, 19, 20, 26, 30, 42, 62, 68, 69, 73, 90, 96, 101, 126, 131, 134\}$ for Solar, as all signals present the same quantity. For the CAN bus dataset with distinct signals, we hand-pick interesting virtual sensor tasks with clear intuitive interpretations according to table 5.

Table 5: Virtual sensors for the CAN bus dataset, which is recorded with an Audi e-tron electric vehicle.

| Quantity | $j$ |
|---|---|
| **Battery** | |
| Battery voltage | 1 |
| Battery current | 2 |
| Battery temperature | 3 |
| Battery coolant outflow temperature | 4 |
| Inverter temperature | 5 |
| **Engine** | |
| Engine 1 torque | 6 |
| Engine 1 voltage | 7 |
| Engine 1 current | 8 |
| Engine 1 power loss | 9 |
| Engine 1 temperature | 10 |
| Engine 1 rotor temperature | 11 |
| Engine 1 coolant intake temperature | 12 |
| Engine 1 coolant flow rate | 13 |
| **Vehicle** | |
| Wheel speed (rear, left) | 14 |
| Chassis ground clearance (rear, left) | 15 |
| Air condition power | 16 |

**Hyperparameters** In preliminary experiments, we find promising hyperparameters which we list in table 6. We keep these architectural hyperparameters constant throughout our work to ensure a fair and isolated comparison of baselines with models with our proposed mechanisms. For all reported results and their underlying models, we conduct an extensive hyperparameter search over optimizer-related parameters, including the learning rate, using HEBO [46]. This way we train competitive models that are robust to varying training dynamics.

Table 6: Hyperparameters for training our models. We denote sets as $\{\cdots\}$ and continuous search spaces as $[\cdots]$.

| Hyperparameter | Value |
|---|---|
| **Architectural** | |
| Patch length | $p = 32$ |
| Transformer layers | $L = 4$ |
| Attention heads | 4 |
| Token dimension | $d = 512$ |
| Multi-layer perceptron hidden dimension | 512 |
| Activation | ReLU |
| **Optimizer-related** | |
| Seed | 2025 |
| Optimizer | Adam [47] |
| Learning rate | $[10^{-6}, 10^{-4}]$ |
| Learning rate warm-up iterations | $\{0, 2, 4\}$ |
| Learning rate warm-up start factor | $[0.3, 0.9]$ |
| Learning rate decay step size | $\{0, 1\}$ |
| Learning rate decay gamma | $[0.96, 0.99]$ |
| Dropout | $[0.0, 0.2]$ |
| Batch size | 32 |
| Epochs | 80 |
| Early stopping patience | 9 |
| Loss | MSE |

**Training stability** For our main experiments, we utilize teacher forcing as the standard training strategy and observe stable training dynamics. During our extensive hyperparameter search, we train models with different learning rate and dropout values, which converge to similar MSE, and we find very low sensitivity to these hyperparameters. Specifically, we find MSE standard deviations of $2.5\,\%$ among hyperparameter settings and low correlations of learning rate and MSE of $0.000\,15$ and $0.077$ for dropout. This highlights the robustness and generalization of a single standard training method. In section 5.6 we additionally experiment with multi-stage training strategies, further improving MSE.

**Reproducibility of measurements** Regarding **predictive quality**, we report MSE confidence intervals for our most common experimental settings via test-set bootstrapping here, as repeating all experiments with multiple seeds is computationally infeasible (see computational effort below). Table 7 shows tight confidence intervals supporting the significance of our results.
For our main experiments, we report efficiency gains in end-to-end **inference time** and memory, as these real-world measurements are of most practical interest. For runtime profiling, we use the same Nvidia A6000 GPU with 2 warm-up and 2 measurement runs per batch to achieve inference time standard deviations $< 2\,\%$. Regarding **memory** allocation, we report dynamic CUDA peak memory, which is a deterministic quantity. Static memory offsets from model parameters are irrelevant in our evaluations as we explore models of similar size throughout our paper.
For more in-depth evaluations, we additionally utilize **sparsity** as a hardware-independent efficiency measure. We explore relations between inference time, memory, and sparsity in appendix C.7.

Table 7: Bootstrapped test-set MSE confidence intervals.

| Dataset | Mean | $95\,\%$ | $99\,\%$ |
|---|---|---|---|
| CAN bus | $0.136$ | $[0.122, 0.151]$ | $[0.118, 0.156]$ |
| Traffic | $0.272$ | $[0.256, 0.289]$ | $[0.251, 0.295]$ |
| Electricity | $0.174$ | $[0.169, 0.182]$ | $[0.167, 0.184]$ |
| Solar | $0.125$ | $[0.121, 0.129]$ | $[0.120, 0.130]$ |

**Computational effort** We utilize over $48\,500$ compute hours on Nvidia H100 GPUs to train our unified models. To improve training efficiency, we employ 16-bit mixed-precision strategies with `bfloat16` data types. We estimate the computational effort to reproduce our experiments in table 8. Please note that we reuse previously trained models in many of our experiments.

Table 8: Computational effort to reproduce our experiments.

| Experiment | GPU hours | Models trained |
|---|---|---|
| Main experiment | $11\,384$ | $1272$ |
| Main experiment – individual baselines | $9392$ | $1078$ |
| Main experiment – unified baselines | $2044$ | $492$ |
| Different MSE efficiency trade-offs for variable test-time computation | $4242$ | $160$ |
| Scaling to hundreds of virtual sensors | $8776$ | $660$ |
| Explainability of virtual sensors – Quantitative assessment | $786$ | $42$ |
| Explainability of virtual sensors – Synthetic data | $1$ | $1$ |
| Efficient training strategies – Training virtual sensors simultaneously | $7920$ | $560$ |
| Efficient training strategies – Teacher forcing | $3964$ | $36$ |

# C Results

Here, we show additional experiments and results.

## C.1 Main experiments

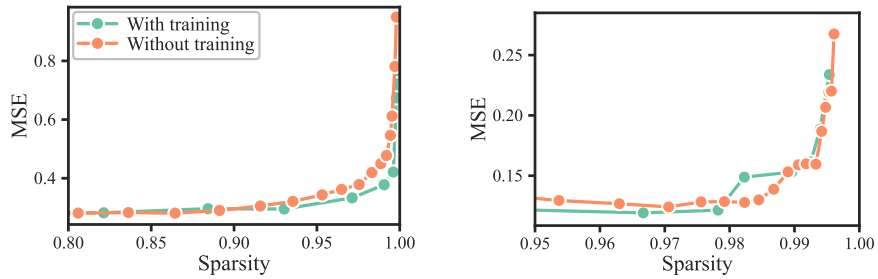
In table 9, we show additional results of our main experiments on the Electricity and the Solar dataset. On these datasets, our unified approach scales to multiple virtual sensors while substantially reducing computation and memory requirements without affecting predictive quality.

Table 9: Comparison of a simple **unified model** predicting all 16 virtual sensors, where we subsequently introduce our proposed mechanisms, with 16 **individual models** predicting a single virtual sensor each. We list MSE, inference time, dynamic CUDA peak memory, and the number of trainable parameters for Electricity and Solar datasets.

| Architecture | Electricity | | | | Solar | | | |
|---|---|---|---|---|---|---|---|---|
| | MSE | Time | Mem. | Param. | MSE | Time | Mem. | Param. |
| **Individual models** | | | | | | | | |
| LSTM (literature) | 0.219 | 0.31 ms | 0.0009 GB | 128.3 M | 0.242 | 0.30 ms | 0.0008 GB | 121.9 M |
| Transformer (ours) | 0.145 | 2.59 ms | 0.13 GB | 115.2 M | 0.128 | 0.69 ms | 0.029 GB | 113.6 M |
| **Unified model** | 0.521 | 2.63 ms | 0.13 GB | 7.0 M | 0.149 | 0.70 ms | 0.029 GB | 7.0 M |
| + Variate embedding $\mathcal{V}, \mathcal{V}'$ | 0.170 | 2.64 ms | 0.13 GB | 7.2 M | 0.126 | 0.71 ms | 0.029 GB | 7.1 M |
| + Signal relevance $\mathcal{R}'$<br>+ $N_{\text{train}} = 4$ | 0.174 | 2.65 ms | 0.13 GB | 7.2 M | 0.125 | 0.72 ms | 0.029 GB | 7.1 M |
| + Sparse input sets $\mathcal{Z}_j$<br>+ Sensor selection | 0.173 | 0.31 ms | 0.0089 GB | 7.2 M | 0.128 | 0.14 ms | 0.0036 GB | 7.1 M |

## C.2 Different MSE efficiency trade-offs for variable test-time computation

In figure 8 we provide complete results of our investigations of different MSE efficiency trade-offs on the Traffic and the CAN bus dataset.

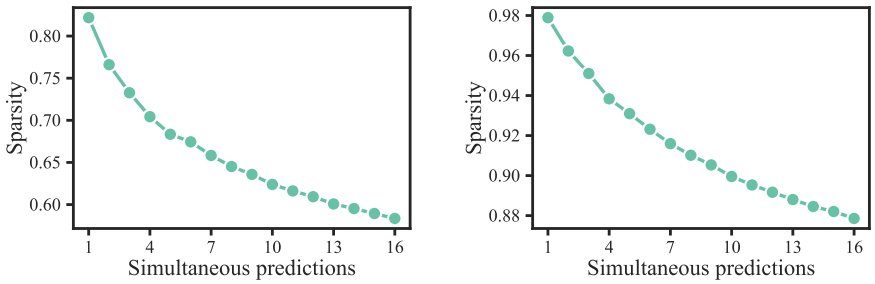



(a) Traffic (b) CAN bus

Figure 8: Varying our signal importance threshold $r'_{\text{thres}}$ during training or inference generates smooth trade-offs between input set sparsity and MSE on the Traffic and the CAN bus dataset.

### C.3 Sensor selection maximizes efficiency without affecting predictive quality

Here, we demonstrate efficiency gains through our sensor selection mechanism on the Traffic and the CAN bus dataset in figure 9.

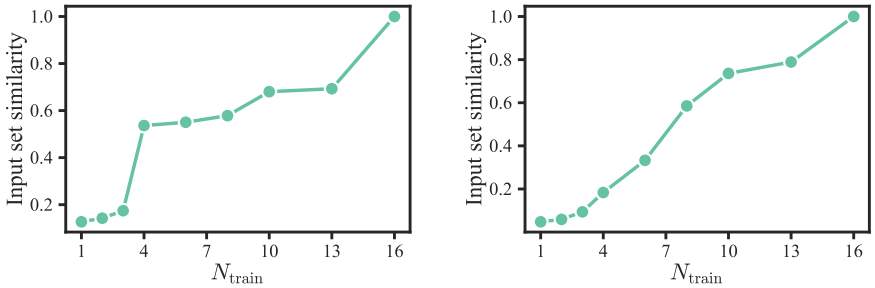



(a) Traffic (b) CAN bus

Figure 9: Sparsity increases as fewer virtual sensors are predicted simultaneously through our sensor selection mechanism. Results are shown for the Traffic and the CAN bus dataset.

### C.4 Efficient training strategies – Training virtual sensors simultaneously

Figure 10 shows complete results for training multiple virtual sensors simultaneously on the Traffic and the CAN bus dataset.

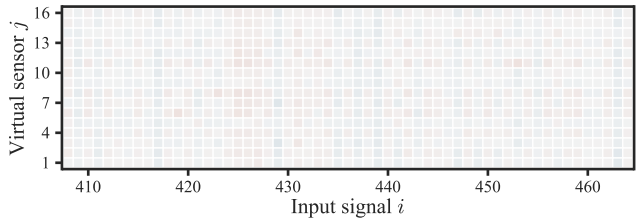



(a) Traffic (b) CAN bus

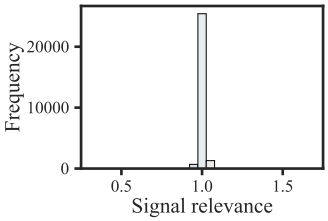
Figure 10: Learning different numbers of virtual sensors $N_{\text{train}}$ in a single training iteration influences the similarity of their input signal sets on the Traffic and the CAN bus dataset.

## C.5 Learning relevant input signals

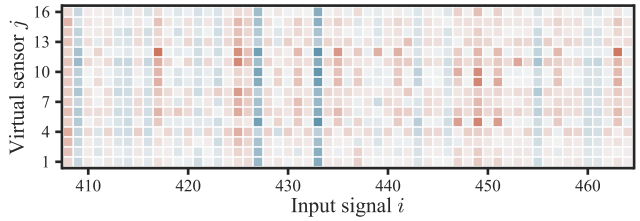
Our architecture learns individual input signal sets for each virtual sensor as described in sections 3.3 and 5.2. Here, we provide further insights into the training process. Throughout training, our signal relevance vectors $\mathcal{R}'$ learn signal importance from attention gradients, becoming more diverse as training proceeds in figures 11a to 11f. Next, we distinguish between important and unimportant signals using a threshold $r'_{\text{thres}}$ to sparsify our signal relevance vectors in figures 11g and 11h, effectively disregarding irrelevant input signals. This finally results in individual input signal sets $\mathcal{Z}_j \subseteq \mathcal{Z}$ for every virtual sensor in figure 12. On the CAN bus dataset, where each signal has a distinct and interpretable purpose, this offers novel explainability of the virtual sensors. Assessment from domain experts further validates meaningful signal selections.



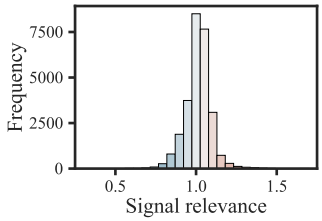

(a) Signal relevance vectors $r'_j \in \mathcal{R}'$, early training



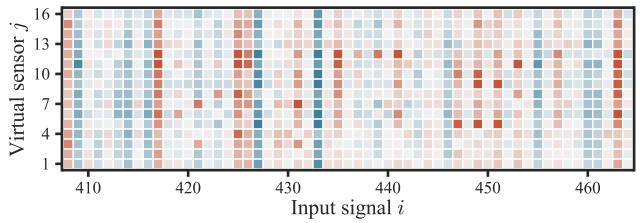

(b) Relevance distribution, early



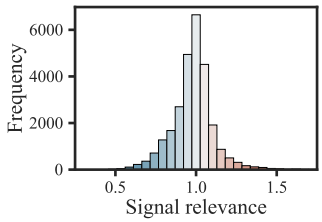

(c) Signal relevance vectors $r'_j \in \mathcal{R}'$, mid training



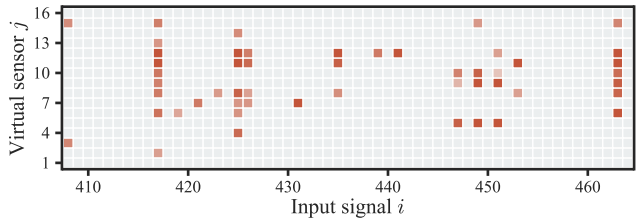

(d) Relevance distribution, mid



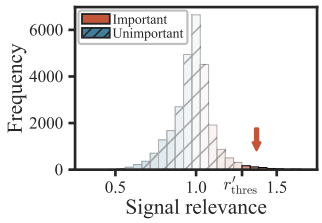

(e) Signal relevance vectors $r'_j \in \mathcal{R}'$, late training




(f) Relevance distribution, late




(g) Signal relevance vectors $r'_j \in \mathcal{R}'$, sparse




(h) Relevance distribution, sparse

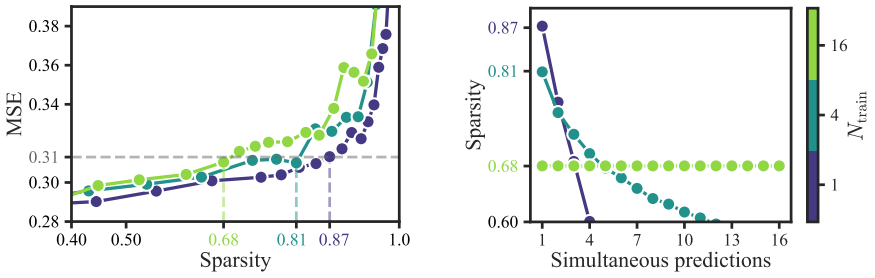
Figure 11: **(a-f)** Signal relevance vectors evolve during training on the CAN bus dataset. **(g,h)** We sparsify signal relevance using a threshold $r'_{\text{thres}}$ to exclude irrelevant signals, keeping only important ones (arrow).




Figure 12: Individual input signal sets $\mathcal{Z}_j \subseteq \mathcal{Z}$ for each virtual sensor $j$ for the CAN bus dataset, learned by our model. For clarity, we list only 4 input signals, while virtual sensors use 38 on average. Further, heating and cooling demand signals relate to the vehicle's interior but also to drivetrain components such as the battery and engine. (See table 5 for non-abbreviated virtual sensor names.)

## C.6 Interactions

There are complex interactions between training efficiency, inference efficiency, and prediction quality, which we explore here in detail. Increasing $N_{\text{train}}$ to learn multiple virtual sensors at a time improves training efficiency. It also leads to more similar input sets $\mathcal{Z}_j \subseteq \mathcal{Z}$ for the virtual sensors (see figure 6). This has two major effects during inference. First, for a fixed prediction quality, the sensor's input sets need to be larger and less sparse as they are less specialized (see figure 13a). This reduces peak efficiency when predicting virtual sensors individually figure 13b. Second, when forecasting multiple virtual sensors at a time, more similar input sets lead to graceful scaling as in figure 13b. Here, synergies among input sets benefit efficiency.
For the Traffic dataset, peak inference efficiency for individual virtual sensors is $3.8\times$ higher for runtime and $5.2\times$ for memory compared to constant scaling behavior for multiple virtual sensors. Generally, we suggest training multiple virtual sensors simultaneously for multi-sensor inference use cases, while learning single virtual sensors at a time for peak inference efficiency of individual virtual sensors. Note that quantitative efficiency gains are strongly dependent on the choice of virtual sensors and the dataset.



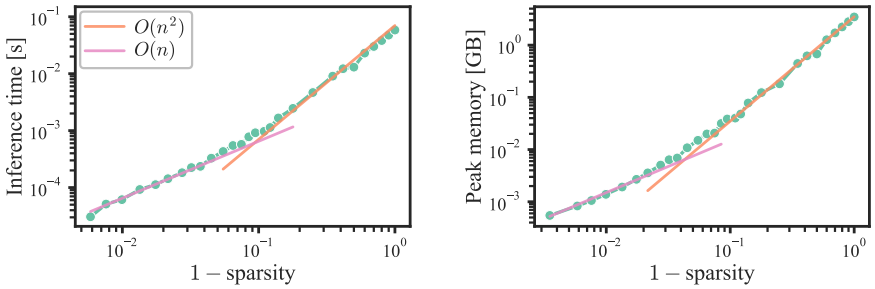

(a) Prediction quality and sparsity

(b) Prediction of multiple virtual sensors

Figure 13: Learning different numbers of virtual sensors $N_{\text{train}}$ in every training iteration influences **(a)** tradeoffs between predictive quality and sparsity and **(b)** inference efficiency when predicting multiple virtual sensors simultaneously on the Traffic dataset.

## C.7 Real-world efficiency gains through sparsity

In our main experiments, we report efficiency gains in inference time and memory, which are of high practical relevance. For more in-depth analysis, we also report sparsity. Here, we explore relations between sparsity, runtime, and memory allocation in detail.
Our architecture employs sparsity in a structured way by reducing the transformer's input tokens $s$ to a minimum. It focuses only on relevant input signals for a requested set of virtual sensors as described in sections 3.2, 3.3, 5.2 and 5.3. We profile our architecture with varying sparsity values in the following. Remarkably, our structured sparsity directly yields real-world savings in computation time and memory over several orders of magnitude, as our results in figure 14 show. For low sparsity values, the attention operation with quadratic complexity is most dominant, while the transformer's multi-layer perceptron with linear complexity is the limiting operation in high-sparsity regimes.



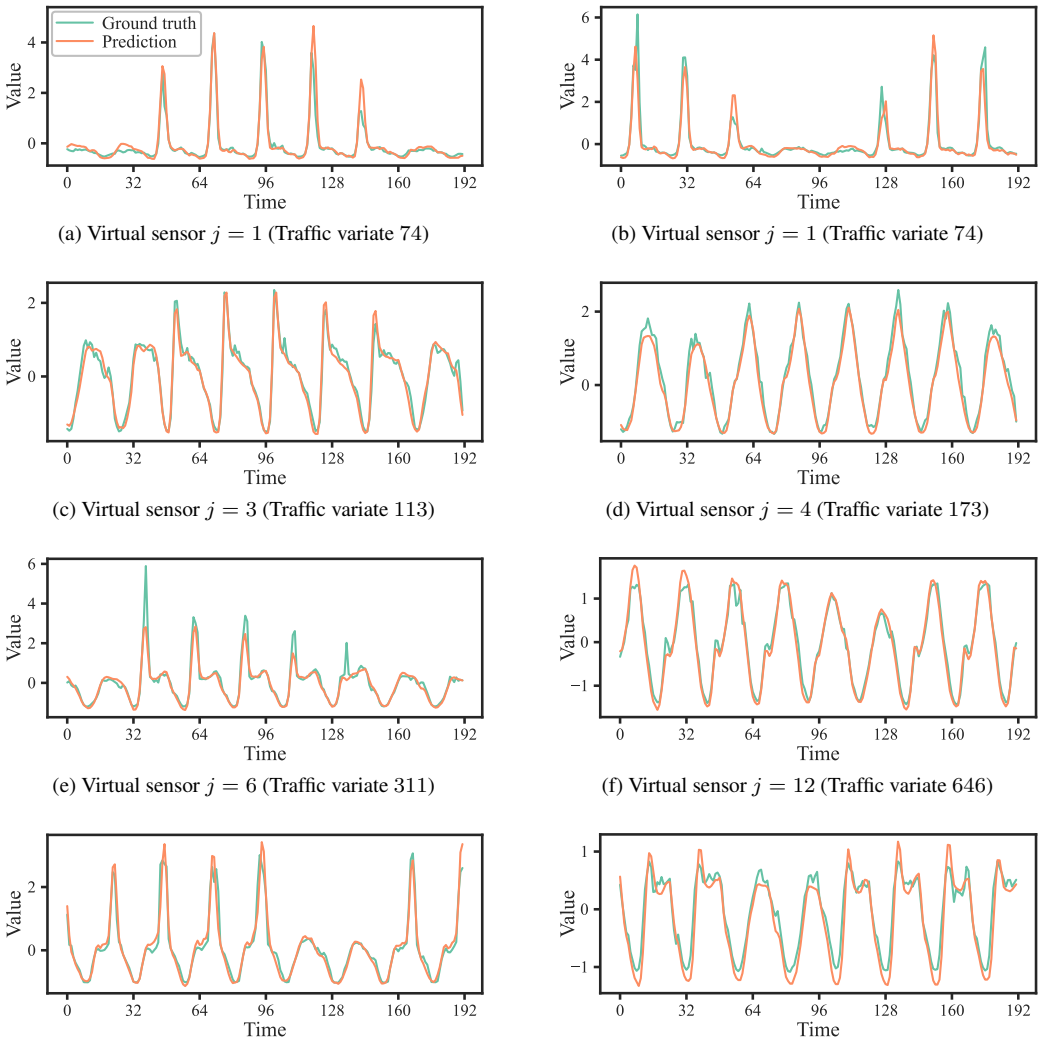

(a) Inference time

(b) Memory

Figure 14: Structured sparsity induces savings in inference time and dynamic CUDA peak memory. Note that we visualize $1-$ sparsity on logarithmic axes. Therefore, quadratic and linear complexities transform into linear functions with slopes of 2 and 1, respectively.

### C.8 Visualization of virtual sensor predictions

In this section, we illustrate predictions of our unified virtual sensor model on the Traffic dataset. In contrast to standard time series forecasting literature [37], our architecture does not have a fixed context horizon. Instead, it forecasts virtual sensors $p = 32$ time steps from other measurement signals, starting from empty prototype tokens. In the following autoregressive cycles, our model also utilizes its past virtual sensor predictions. Over time, past context grows, mimicking practical deployment scenarios of virtual sensors.
Our visualizations in figure 15 demonstrate that our architecture successfully predicts initial virtual sensor values from empty prototypes. Subsequent autoregressive cycles do not show error accumulation and predicted patches connect smoothly. Even over long-horizon rollouts exceeding 1000 hours of traffic flow, the predictive loss remains stable in figure 16. Finally, our sensor selection mechanism successfully switches between multiple virtual sensors with different temporal characteristics, measuring traffic flow at different locations in San Francisco. All this visual evidence aligns with our previous results, demonstrating the flexibility of our approach.




(a) Virtual sensor $j = 1$ (Traffic variate 74)

(b) Virtual sensor $j = 1$ (Traffic variate 74)

(c) Virtual sensor $j = 3$ (Traffic variate 113)

(d) Virtual sensor $j = 4$ (Traffic variate 173)

(e) Virtual sensor $j = 6$ (Traffic variate 311)

(f) Virtual sensor $j = 12$ (Traffic variate 646)

(g) Virtual sensor $j = 14$ (Traffic variate 715)

(h) Virtual sensor $j = 15$ (Traffic variate 775)

Figure 15: Virtual sensor predictions on the Traffic dataset. Our sensor selection mechanism successfully switches between virtual sensors with different temporal patterns.

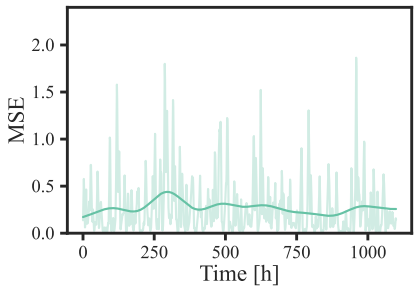



Figure 16: MSE remains stable for very long virtual sensor operation on the Traffic dataset.

### C.9 Transfer learning

In practical applications, data availability is often limited, and new virtual sensors may be required over the lifetime of a model. Time series foundation models offer appealing properties, including zero-shot generalization to unseen datasets and transfer learning. While comparable zero-shot generalization is infeasible for virtual sensors[3] and our scope is to learn multiple sensors within a single network, we demonstrate that transfer learning allows new virtual sensors to be efficiently introduced into an already trained model using only limited data.
To investigate the transfer learning capabilities of our architecture, we first train a model on the CAN bus dataset with $N = 15$ virtual sensors, excluding the wheel speed sensor ($j = 14$). In a second step, we introduce the wheel speed virtual sensor via transfer learning using only $10\,\%$ of the original training data. Figure 17 shows that this approach enables the efficient integration of a new virtual sensor into an existing model under constrained data availability. Transfer learning achieves the same predictive performance as training from scratch with full data while requiring only $3.6\,\%$ of the training iterations. Moreover, compared to training from scratch under limited data, transfer learning yields models with improved predictive accuracy. These results indicate that transfer learning exploits synergies and previously acquired system knowledge to efficiently integrate new virtual sensors.

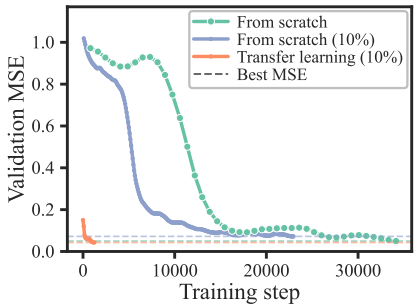



Figure 17: Comparison of training from scratch with full and limited ($10\,\%$) data to transfer learning for introducing a new virtual sensor on the CAN bus dataset under constrained data availability. Note that validation MSE is reported at the end of each epoch – this is why initial transfer learning loss is already low.

[3]Zero-shot generalization in time series foundation models relies on in-context learning, i.e., identifying patterns in the input series and extrapolating them as predictions. Virtual sensors, however, correspond to new target signals that are not present in the input context, precluding in-context learning.

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We design a multi-task virtual sensor architecture that is scalable and efficient. We demonstrate efficiency in sections 5.1 to 5.3, scaling in sections 5.1 and 5.4, and explainability benefits in section 5.5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in section 6 in a separate limitations paragraph.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: Our results are mainly empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide full experimental details in section 4 and appendix B including dataset, architectural, training, and hyperparameter settings to reproduce our results. We further describe all details of our novel architecture in section 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We conduct our experiments also on three standard open access datasets.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide full experimental details in section 4 and appendix B including dataset, architectural, training, and hyperparameter settings to reproduce our results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report MSE confidence intervals for our most common experimental settings via test-set bootstrapping in table 7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report required computation for each experiment in table 8 and the paragraph above.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We believe the design of very efficient neural network architectures directly impacts society and environment in a positive way (see section 6).

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper does not pose such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited all prior work and comply with the licenses of all datasets used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.