Gumbel-Based Active Sparse Mobile Crowd Sensing with Time Series Transformer

Victor Li
Computer Science Department
Emory University
victor.li@emory.edu

Carson Lam
Computer Science Department
Emory University
carson.lam@emory.edu

Ting Li
Computer Science Department
Emory University
ting.li@emory.edu

***Abstract*—Mobile Crowd Sensing (MCS) has revolutionized the way we collect environmental and urban data by leveraging a tremendous number of distributed sensors equipped on wearable devices like smartphones, smart watches, and so on. Gathering continuous data from these sensors allows us to generate a complete spatiotemporal field for the desired measurements. However, obtaining continuous sensing data at all locations and times is often infeasible due to resource limits, cost constraints, and practicality. To address this problem, we proposed a Learned Gumbel-Based Active Sparse MCS framework utilizing an Encoder-Decoder Time Series Transformer that reconstructs the full spatiotemporal field for a given time cycle and a small subset of available sensors. Notably, an active sensor selection layer is included, which leverages the Gumbel distribution to create noise from learned weights, helping to dynamically select the most effective sensors for the next sensing cycle. We verified the effectiveness of our proposed model by comparing it to random selection, attention-scoring, and statistical methods. Our Gumbel-Based selector achieves higher performance in both first- and second-order norm metrics.**

***Index Terms*—Mobile Crowd Sensing, Active Learning, Time-Series Reconstruction, Deep Learning**

I. Introduction

Mobile Crowd Sensing (MCS) has revolutionized the way we collect spatiotemporal data through a large number of distributed mobile sensors, shown in Figure 1. It has been widely used as a pioneer in monitoring air quality, traffic conditions, and much more. In MCS, there are four major components: task requester, platform, a large group of tasks, and participants. Our focus is on the platform, which recruits participants for certain tasks requested by the task requester. Since the foundation of MCS is to have as many available participants as possible and leverage the power of the crowd, one of the many goals is to optimize the balance between participant hiring costs and accuracy.

Because continuous sensing entails substantial costs and logistical impracticalities (e.g., financial limitations, power availability, and data transmission bandwidth) that lead to inactive sensors in Figure 1, Sparse MCS offers a markedly more viable option for these monitoring applications. Sparse MCS gathers data from a small subset of available sensors and predicts the remaining spatiotemporal field. By leveraging observations from nearby or similarly sampled regions, Sparse MCS enhances data efficiency through these inferences on the missing data.

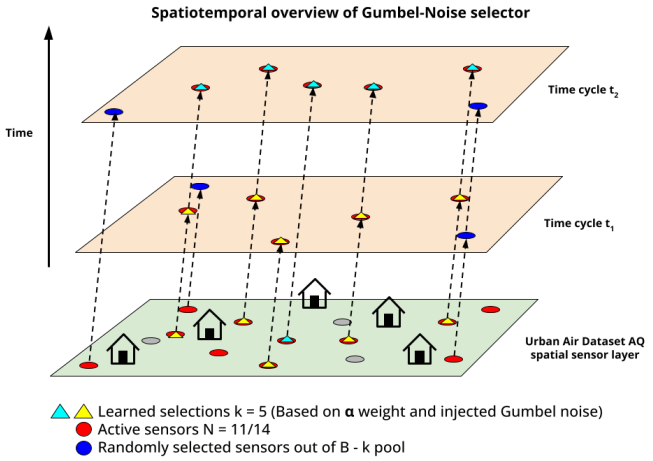

Fig. 1. **Illustration of Active Sensor Selection**. At each time cycle, the framework selects the top-k most informative sensors. This strategy reduces sensing costs and resource demands while still enabling accurate reconstruction of the spatiotemporal field.

Today, deep learning through transformer-based foundation models—such as TabPFN [6], Chronos [2], and TimesFM [4]—has been the go-to method of data reconstruction for Sparse MCS. Although these methods can achieve high accuracy, they depend heavily on very large and often uninterrupted time-series datasets for pre-training. Circling back to the practicality constraints, gathering such complete datasets is rarely feasible. As a result, Active Sparse MCS (AS-MCS) frameworks have been proposed that actively choose more influential and higher-quality sensors during the training process. However, because of the constraints of gradient-based backpropagation used in deep learning, many approaches involve purely statistical selections.

To address this problem, we proposed a novel AS-MCS framework leveraging an Encoder-Decoder Time Series Transformer [13] to reconstruct the full spatiotemporal field with an active sensor selection layer in between. For this selection layer, we propose a Learned Gumbel-Based selector that samples Gumbel-Noise from trainable weights to merge with

encoder embeddings for active sensor selection. From the pool of available sensors in each cycle, this layer picks the top-*k* most influential to reuse in the next. To train this layer, we also introduce a selector loss term, defined by the reconstruction error of the sensing map using only the top-*k* sensor readings. Even though existing works [11] have adopted the transformer model to guide the inference of missing data, they relied on fixed spatiotemporal embeddings and prior knowledge (e.g., geographic POIs, weather). Our framework, combined with our selector layer, introduces a fully trainable model for AS-MCS that dynamically optimizes sensor selection during training without requiring auxiliary domain knowledge.

Lastly, we evaluate the reconstruction loss of a single target variable for our Learned Gumbel-Based AS-MCS model and benchmark it against 3 other selectors within our AS-MCS frameworks with different selector mechanisms: (1) Random Sampling, (2) Attention-Score Sparsemax, and (3) Statistical Gumbel Noise with embeddings.

To summarize, our work has the following contributions:

- We introduced a framework for implementing selector layers for AS-MCS, built upon the Time Series Transformer. This should be the first work that includes an additional selector layer in the traditional Encoder-Decoder Transformer model.
- We integrated a selector loss term that allows for a fully trained selector layer in the Learned Gumbel-Based AS-MCS model.
- We conducted an empirical study and demonstrated that a fully trained selector layer outperforms random sampling, attention scoring, and statistical selection.

II. Related Works

We now expand on the aforementioned foundational models and training frameworks, as well as introduce works related to AS-MCS.

A. Chronos

Chronos [2] is a pretrained probabilistic time series framework that also utilizes transformer-based modeling by tokenizing a stream of continuous signals into discrete sequences. Trained based on a large-scale combination of both real-world and synthetic data obtained through Gaussian (stochastic) processes, Chronos was seen to have achieved stronger zero-shot predictive performance than similar models such as ForecastPFN [5]. This demonstrated the power of foundational models in interpolating temporal patterns without the need for task-specific fine-tuning. However, despite excellent performance in univariate and fully observed time series data during both training and inference, full sensor coverage is often impractical under MCS environments. Our work addressed this by actively choosing higher-quality sensor data, resulting in more accurate spatiotemporal reconstruction under strict sensing budgets.

B. TimesFM

TimesFM [4] introduced another model capable of time series modeling through a patched decoder-only attention architecture, where inputs and predictions are broken into patches. This structure allows the model to perform exceptionally well in long-sequence forecasting at a notably small scale of 200M parameters, compared to the 20M to 710M range of Chronos [2]. Similarly, TimesFM motivated the use of data-hungry pretrained models in forecasting for various domains, which often require training data that consists of large-scale real-world time series such as *Google Trends* and *Wikipedia* page visits combined with synthetic data. Although TimesFM's attention-based architecture bears resemblance to our model, our introduction of a selector layer reduced the reliance on large and continuously available datasets such as trends, traffic, weather, etc.

C. TabPFN-TS

TabPFN-TS [6] extended the established general tabular foundation model TabPFN-v2 to zero-shot forecasting applications by treating time series as tabular data intended for the prior-data fitted network (PFN) framework. TabPFN-TS leverages feature engineering for calendar-based and seasonal features, which run on cyclic models and Fourier Transformational algorithms, respectively. Furthermore, despite a compact size of 11M parameters and minimal pretraining data, TabPFN-TS has been shown to outperform other leading models on public scoring metrics. Like TabPFN-TS, we aimed to build a framework for generalizing to any domain on minimal training data. However, instead of enriching the input through manual calendar and seasonal feature engineering, we forced the model to learn which sensors are most important through a dedicated selector layer.

D. Sparsemax

Sparsemax [7] is a longstanding alternative to Softmax activation that produces sparse probability distributions by projecting outputs onto the boundary of the probability simplex. Sparsemax's capability to assign zero probability to irrelevant elements yields more interpretable and selective attention mechanisms and has led to applications in signal filtering and computational resource management. In our work, we adapted Sparsemax as a selector baseline for AS-MCS. Specifically, our Attention-Score Sparsemax selector uses encoder attention scores to rank sensors, enforcing sparsity by deterministically selecting the top-*k* sensors for the next sensing cycle. Shown in our results later, this deterministic method falls short when budgets are tight and sensor selection becomes critical.

III. Problem Formulation

Let N be the number of sensors that can collect data along W time cycles, and $\mathbf{X}'_{w,i}$ be the ground truth of the sensed data at time cycle w and sensor i. Because of resource limitations and practicality constraints, we can only gather data from a small subset of sensors $\mathbf{B}$, where $\text{len}(\mathbf{B}) \ll N$. The sensed data is then denoted as $\mathbf{X}_{w,s}$ for sensor $s \in \mathbf{B}$. Using our

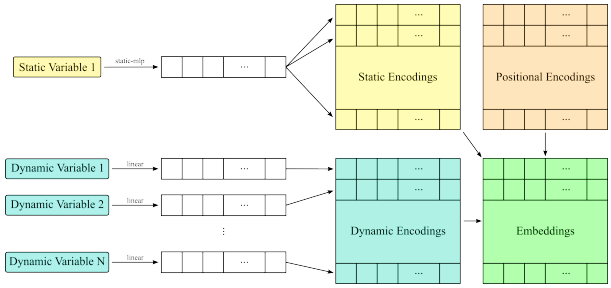

Fig. 2. **Overview of the encoder preprocessing pipeline.** At each time step, multivariate sensor readings are first linearly projected into the model dimension. Static spatial coordinates are processed separately through a two-layer *MLP* with *ReLU* activation, then broadcast across σ and added to the dynamic embeddings. Finally, fully learned positional encodings are incorporated.

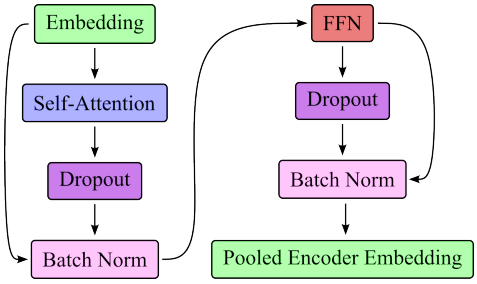

Fig. 3. **Architecture of a single encoder layer.** Each layer applies multi-head self-attention, followed by batch normalization in place of layer normalization. The output is then passed through a feed-forward network and normalized again before being forwarded to subsequent layers or pooled across the temporal dimension in the final layer.

data $\mathbf{X}$ collected across $\mathbf{B}$ sensors, our goal is to generate predictions for a single target variable $\widetilde{\mathbf{y}}_{w,r}$ for sensors $r \notin \mathbf{B}$ that minimize the norm between itself and the ground truth of that target variable $\mathbf{y}'_{w,r}$. Any norms can be used, and we use first-order (MAE) and second-order (MSE) losses in the later simulations. Hence, our problem can be formulated below as:

$$\begin{aligned} &\textbf{\textit{Min.}} \ ||\widetilde{\mathbf{y}}_{w,r} - \mathbf{y}'_{w,r}|| \\ &\text{s.t.} \quad w \in W, \ r \notin \mathbf{B} \end{aligned} \tag{1}$$

IV. Methodology

Our framework for AS-MCS involves three main components: (A) encoder, (B) selector layer, and (C) decoder. From complete historical data with N sensors, we first separated the static positional coordinates from the length-l, m-variable multivariate time series into two datasets $\Sigma \in \Re^{N\times 2}$ and $\mathbf{X}' \in \Re^{N\times l\times m}$, respectively. To support active learning and mimic the intervals at which sensors transmit their readings, we assume that our input data comes in sensing cycles of a sequence length σ. This implies that when training on historical data, we partitioned long multivariate time series into σ-long chunks denoted $\mathbf{X}'_w$. To simulate sparsity, we defined a subset of sensors $\mathbf{B}$: initially sampled at random, and thereafter determined by the model using the prior cycle's selections. To ensure selection diversity and avoid repeatedly sampling the same sensors, we also introduced a parameter $k < \text{len}(\mathbf{B})$ and used the model's selection layer to populate a set of sensors $\mathbf{B}_{selected} \subseteq \mathbf{B}$. The remaining $\text{len}(\mathbf{B}) - k$ sensors are selected at random.

Therefore, for each cycle w the model receives sensed data $\mathbf{X}_{w,s}$, Σ_s for $s \in \mathbf{B}$, and Σ_r for $r \notin \mathbf{B}$. Passing $\mathbf{X}_{w,s}$ and Σ_s through the encoder, the model creates contextualized embeddings for each sensor. These embeddings and other encoder outputs are then passed through the selector layer to select k sensors for the next cycle. Lastly, the decoder then uses the encoder embeddings and Σ_r to predict sequences $\widetilde{\mathbf{y}}_{w,r}$ for the target variable.

A. Encoder

Our encoder design closely follows the original Transformer [10] architecture introduced by Google in 2017, but with a few changes added in the Multivariate Times Series Transformer [13]. In the data pre-processing pipeline shown in Figure 2, we first linearly projected the variable vector at each time stamp to our model dimension d, essentially expanding our multivariate data $\mathbf{X}_{w,s} \in \Re^{\sigma\times m}$ into $\Re^{\sigma\times d}$. For the static coordinates, we used a Multi-Layer Perceptron (*static-MLP*) with a *ReLU* activation stacked between two linear layers to transform them into vectors of $\Re^d$. Broadcasted across σ, these static coordinate encodings were added to $\mathbf{X}_{w,s}$. Lastly, fully learned positional encodings, as opposed to classic sinusoidal encodings, were added to the embeddings, ensuring temporal order is preserved before passing the embeddings into the encoder layers.

Following the original Transformer [10] design, each encoder layer leverages self-attention for contextualization across sensors. However, instead of the original layer normalization, we implemented batch normalization, which is more effective for time series [13]. Thus, a single encoder layer sequentially follows Figure 3. Adding more layers, each batch-normalized output is treated as the next layer's input, with the output of the last layer being pooled across the sequence length.

B. Selector Layer

Depending on the selector layer, our framework uses various encoder outputs as inputs to select the indices of k sensors for the next cycle.

1) **Random Sampling:** Requires only a list of available sensor indices. The layer randomly samples k indices from this list.
2) **Attention-Score Sparsemax:** Uses the attention scores from the final encoder layer as input. Applying the sparsemax function produces a sparse probability vector, from which k sensor indices are sampled.
3) **Statistical Gumbel Selector:** Takes pooled encoder embeddings as input. After averaging across d_{model}, scores are computed for each sensor. Gumbel noise is added to these scores, and the top-k values are selected as sensor indices.
4) **Learned Gumbel-Based Selector:** Also takes pooled encoder embeddings as input, but instead of using them directly for selection. They are used to train a learnable

weight matrix $\alpha \in \Re^{N \times k}$. Each weight α_{ij} encodes the probability of selecting the i-th sensor for the j-th position. Sensor indices are sampled from these learned probabilities. The training procedure is described in Section VI.

C. Decoder

The decoder design also follows the original Transformer [10] decoder design, leveraging cross-attention shown in Figure 4. For the query, the static coordinates Σ_s of unsensed sensors $s \notin \mathbf{B}$ are processed through *static_MLP* and added together with the learned positional encoding. Using the pooled encoder embeddings as the key and value, the model cross-attends and contextualizes the query locations with the available sensor data. Once again, batch normalization is used instead of layer normalization. The outputs of the last layer are then viewed as a large matrix with each vector corresponding to a sensor and timestamp. Lastly, an output layer projects each vector to a singular interpolated value and concatenates them across the sequence length σ.

V. Learned Gumbel-Based Selector Layer

In contrast to the Attention-Score Sparsemax and Statistical Gumbel Selectors, our Learned Gumbel-Based approach does not directly use the encoder embeddings for sensor selection. Instead, it uses these embeddings to train a learnable weight matrix α. This matrix α not only defines the selection probabilities for each sensor but also a mask on the embeddings. Multiplying the mask with the pooled encoder embeddings, we created masked embeddings that feed into the decoder to reconstruct the target variable for sensors not selected.

A. Mask

The mask $\mathbf{U}$ is created from α [9] following the method shown in Equation 2: sample and add Gumbel-noise $\mathbf{G} \in \Re^{N*k}$, divide by a temperature parameter β, and then apply softmax. As β approaches 0, the softmax—and thus the mask—becomes increasingly discrete. During training, we exponentially decrease β from 1 down to 0 across epochs.

$$\mathbf{U}_{nk} = \frac{\exp((\log\ \alpha_{nk} + \mathbf{G}_{nk})/\beta)}{\sum_{j=1}^{N} \exp((\log\ \alpha_{jk} + \mathbf{G}_{jk})/\beta)} \tag{2}$$

We choose the Gumbel distribution over alternatives such as Gaussian noise because it is inherently tied to categorical sampling and the modeling of maxima, making it well-suited for top-k sensor selection tasks [9]. This property allows the model to converge toward sharper, near one-hot selections as β decreases.

B. Selector Loss

Using the masked embeddings and the decoder, we reconstructed the target variable sequences $\widetilde{\mathbf{y}}_p$ for sensors $p \in \mathbf{B} \setminus \mathbf{B}_{selected}$ using $\mathbf{X}_q$ for sensors $q \in \mathbf{B}_{selected}$. A selector loss term $\mathcal{L}_{sel}$ is then computed by evaluating the given loss $\mathcal{L}(\widetilde{\mathbf{y}}_p, \mathbf{y}'_p)$ on these values. We then combine $\mathcal{L}_{sel}$ with the reconstruction loss $\mathcal{L}_{rec}$ using a balancing parameter λ as shown below in Equation 3. Prioritizing selector training first, we let a balancing term λ decrease exponentially from 1 to 0.

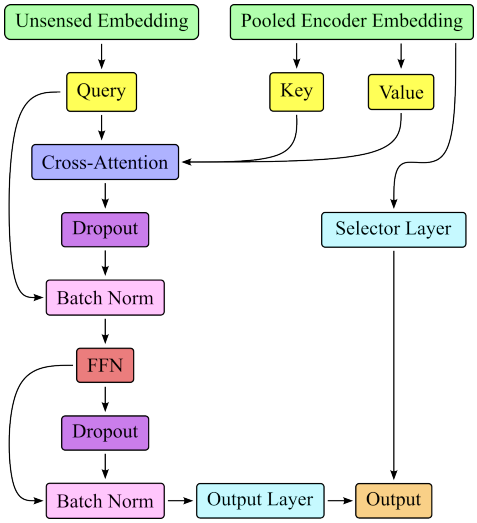

Fig. 4. **Architecture of a single decoder layer.** Queries are formed from the static coordinates of unsensed sensors, transformed through a *static MLP* and combined with learned positional encodings. These queries cross-attend to the pooled encoder embeddings. Batch normalization replaces the standard layer normalization. Lastly, the decoder outputs are projected to interpolated values.

$$\mathcal{L} = \lambda \mathcal{L}_{sel} + (1 - \lambda)\mathcal{L}_{rec} \tag{3}$$

VI. Experiments

A. Data

In our study, we employed the Urban Air [14] and SensorScope's St-Bernard [3] datasets. The Urban Air dataset includes measurements from 437 weather stations across China, each recording hourly air quality data over the span of one year. In contrast, the St-Bernard dataset consists of 31 temperature and humidity sensors located within a small region, providing much denser readings at two-minute intervals over two months. For both datasets, we chose a sequence length of 24 and created samples from long series using a sliding window. Additionally, we excluded sensors that are completely inactive during a cycle and linearly interpolated partially missing sensor readings.

B. Training

We partitioned the data into training and validation sets chronologically, training on earlier samples and evaluating on subsequent, more recent observations. This enables the model to build a historical understanding of the sensors while preventing data leakage and look-ahead bias. For each of the 4 selection layers, we trained under first-order (MAE) and second-order (MSE) losses at a learning rate of *1e-3* and *3e-4*. Figure 5 plots the validation loss on the Urban Air dataset over 200 epochs for our Learned Gumbel-Based Selector. Training on the St-Bernard dataset produces comparable validation loss curves, which we omit here for brevity.

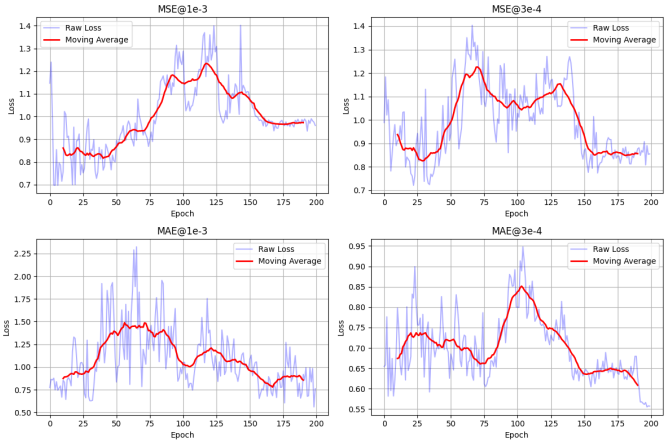

Fig. 5. Evaluation loss plots for Learned Gumbel-Based Selector model trained across 200 epochs using MSE and MAE at learning rates of *1e-3* and *3e-4*.

In Figure 6, we reported the final validation losses for first-order (MAE) and second-order (MSE) norms of each model trained across the two datasets. We saw only marginal loss reductions when comparing Attention-Score Sparsemax and Statistical Gumbel selectors to the Random selector. More notably, our Learned Gumbel-Based selector achieves more significantly reduced losses relative to the other approaches, with the exception of *MSE@3e-4*.

VII. Discussion

Utilizing the random selector layer as a baseline and the other selectors as comparisons, we can see that our Learned Gumbel-Based Selector consistently outperforms the other selector layers on the U-Air dataset in Figure 6.

However, improvements in performance are less significant for the St-Bernard datasets, where both the number and distribution of sensors are smaller. Interestingly, the random selector even surpasses the Learned Gumbel-Based Selector under the *MSE@3e-4* setting. This high performance can be attributed to the additional complexity introduced by the selector layers, which increases the risk of overfitting. Moreover, since the St-Bernard sensors were standardized and deployed simultaneously, random sampling can surpass complex selection layers that attempt to exploit negligible differences in sensor quality or history.

Looking at Figure 5, the evaluation loss of our Learned Gumbel-Based Selector typically begins at a low value, rises during the early stages, and then gradually decreases to convergence around 200 epochs. This pattern arises because the weights α are randomly initialized, making the initial selection process effectively random. Consequently, the early losses align closely with the final evaluation loss of the random selector baseline.

An important factor in the effectiveness of the Learned Gumbel-Based Selector is its ability to avoid redundant sensor selections and reduce overfitting through appropriate tuning of the parameter λ. Unlike methods that rely on pooling

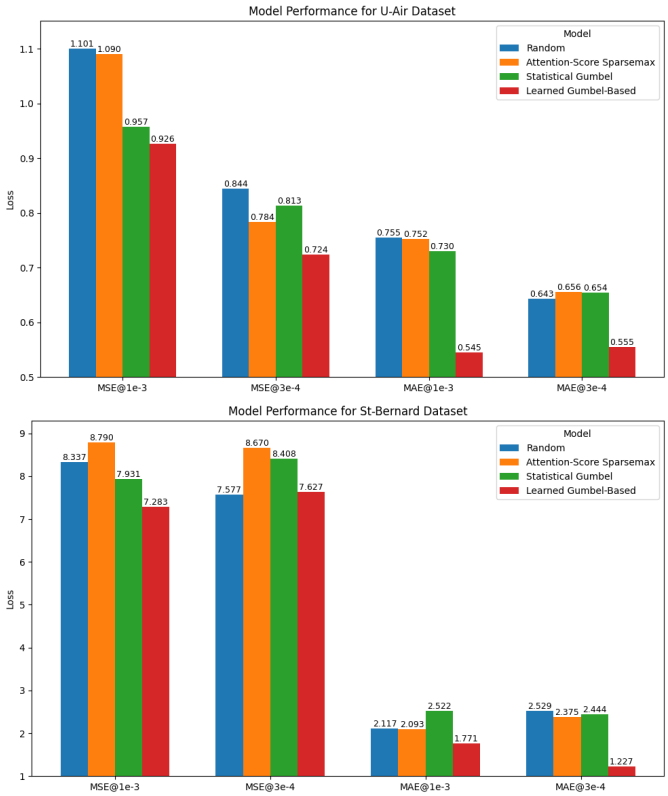

Fig. 6. Final evaluation losses at 200 epochs for the Random, Attention-Score Sparsemax, Statistical Gumbel, and Learned Gumbel-Based Selectors across both the U-Air and St-Bernard datasets.

encoder embeddings into scores, the selector incorporates distinct trainable weights α, which capture the historical importance of sensors and enable stronger generalization. This design allows the model to draw simultaneously on the global context provided by encoder embeddings and on sensor-specific reliability encoded in the learned weights.

The benefits of this approach are evident in models trained with a learning rate of *1e-3* under an MSE loss. As shown in Figure 7, the sets of sensors chosen by the Learned Gumbel-Based Selector and by naïve random sampling appear quite similar in distribution, with both methods avoiding repeated clustering in the same regions. However, despite this superficial resemblance, their reconstruction performances are markedly different: the random sampling model yields an MSE nearly three times higher than that of the Learned Gumbel-Based Selector. Since the selection mechanism is the only difference between the two models, this substantial performance gap underscores that the Learned Gumbel-Based Selector not only enforces diversity in sensor choices but also learns to identify and prioritize the most informative sensors, thereby achieving a significant reduction in reconstruction error.

VIII. Future Work

Although our method yields substantial preliminary improvements over existing approaches, we plan to perform an

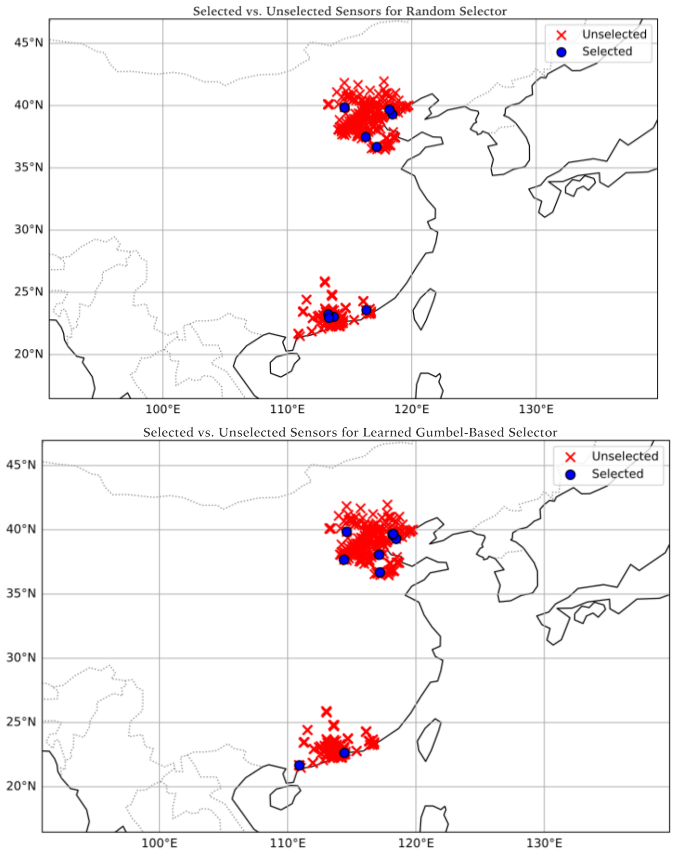

Fig. 7. Comparison of sensor selection patterns between the Random Selector (top) and the Learned Gumbel-Based Selector (bottom). While both approaches produce visually similar and diverse spatial distributions of selected sensors, their reconstruction performance differs substantially, with the Random Selector yielding an MSE nearly three times higher than the Learned Gumbel-Based Selector.

ablation study by varying the balancing term λ and the temperature β. Additionally, several promising research directions can further enhance the applicability and accuracy of AS-MCS frameworks.

A. Patch-Based Selection

Our current method processes each sensing cycle in its entirety, which can become computationally expensive for long, densely sampled cycles. To leverage all available data without downsampling, future work could adopt a PatchTST [8] inspired framework where each patch of a sensing cycle independently selects a subset of sensors, enabling the model to learn which sensor patterns are most informative based on the time period. This approach would not only enable the model to capture intra-day patterns, such as morning, afternoon, and night, but also substantially reduce its size and computational load when handling longer cycles of densely sampled data.

B. Multi-Modal Sensor Fusion

While many existing MCS frameworks, including ours, are limited to single-modality data, recent advances in multi-modal sensor fusion have demonstrated that deep learning–based inference methods can integrate diverse modalities to produce more accurate and robust predictions [1]. These approaches typically operate under three main data fusion paradigms, employing models such as MVAE [12] to adapt to different modality combinations. A promising extension of our framework would be to expand the multivariate dimension m in the time series representation $X \in \Re^{N \times l \times m}$ to incorporate visual and spatial modalities such as camera and LiDAR data, enabling richer representations for comprehensive environmental monitoring.

Acknowledgment

This work was supported by Emory University's Summer Oxford Research Scholars program. We also thank Emory's HyPER C3 Cluster for providing compute nodes.

References

[1] "A comparative review on multi-modal sensors fusion based on deep learning". In: *Signal Processing* 213 (2023), p. 109165.

[2] Abdul Fatir Ansari et al. *Chronos: Learning the Language of Time Series*. 2024.

[3] Guillermo Barrenetxea. *Sensorscope Data*. Zenodo. Apr. 2019. DOI: 10.5281/zenodo.2654726. URL: https://doi.org/10.5281/zenodo.2654726.

[4] Abhimanyu Das et al. *A decoder-only foundation model for time-series forecasting*. 2024.

[5] Samuel Dooley et al. *ForecastPFN: Synthetically-Trained Zero-Shot Forecasting*. 2023.

[6] Shi Bin Hoo et al. *From Tables to Time: How TabPFN-v2 Outperforms Specialized Time Series Forecasting Models*. 2025.

[7] André F. T. Martins and Ramón Fernandez Astudillo. *From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification*. 2016.

[8] Yuqi Nie et al. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2023.

[9] T. Strypsteen and A. Bertrand. "End-to-End Learnable EEG Channel Selection for Deep Neural Networks with Gumbel-Softmax". In: *Journal of Neural Engineering* ().

[10] Ashish Vaswani et al. *Attention Is All You Need*. 2023.

[11] En Wang et al. "Spatiotemporal Transformer for Data Inference and Long Prediction in Sparse Mobile Crowd-Sensing". In: *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. 2023.

[12] Mike Wu and Noah Goodman. *Multimodal Generative Models for Scalable Weakly-Supervised Learning*. 2018.

[13] George Zerveas et al. "A Transformer-based Framework for Multivariate Time Series Representation Learning". In: Association for Computing Machinery, 2021. ISBN: 9781450383325.

[14] Yu Zheng et al. "Forecasting Fine-Grained Air Quality Based on Big Data". In: *Proceedings of the 21st SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2015)*. 2015.