Efficient Query Filtering in Big Data Using Entropy-Based Learned Indexing

Elias Dritsas, Maria Trigka, and Phivos Mylonas
University of West Attica, Egaleo 12243, Greece
{idritsas,mtrigka,mylonasf}@uniwa.gr

***Abstract*—This paper explores the use of neural networks as lightweight surrogates for guiding value localization over numeric attributes in analytical data systems. Instead of modeling key-position mappings directly, we frame the problem as a multi-class classification task over discretized bins, enabling approximate query support without explicit index structures. The model is trained in a self-supervised manner using pseudo-labels from data-driven binning schemes, requiring no manual annotation. Experiments on three real-world datasets show that our method achieves classification accuracy above 94.7% for coarse binning ($K = 5$), with consistent F1 scores and inference times under 0.6 milliseconds (ms). The approach remains robust under skewed distributions using entropy- or quantile-based binning and is well suited to latency-sensitive tasks such as filtering or partition pruning. While this study focuses on univariate attributes, the method is extensible to multivariate scenarios via Multilayer Perceptrons (MLPs) or attention-based models. Overall, neural surrogates offer a practical complement to traditional indexing in approximate or edge-driven analytical workloads.**

***Index Terms*—Learned Index Structures, Numeric Attribute Classification, Binning-Based Surrogate Indexing, Query Selectivity Estimation**

I. Introduction

Recent advances in machine learning have opened new opportunities for rethinking core data management components. One of the most promising directions is augmenting or partially replacing traditional indexing structures with learned models that approximate the mapping between query predicates and data value distributions. While this idea has been explored extensively for key-based access paths and range indexes, recent studies have highlighted its potential in analytical systems, particularly for numeric attributes with skewed or evolving distributions [1], [2].

In such contexts, indexing is often used not for exact tuple retrieval but as a lightweight mechanism for guiding query planners, filtering out irrelevant data blocks, or estimating selectivity. These tasks can tolerate approximation and benefit from adaptive models that capture the underlying semantics of the data. Compact neural classifiers trained to predict coarse value localization, referred to as surrogate indexes in this work, offer a practical alternative. These surrogate models do not aim to replace transactional indexes such as B-Trees or hash tables. Instead, they serve as inference-based mechanisms for query pruning or selectivity guidance in analytical pipelines [3], [4].

However, deploying learned surrogates in real-world scenarios imposes strict constraints: low-latency inference, robustness across distributions, and minimal memory footprint. These requirements rule out deep or recurrent architectures, as well as supervised training pipelines that rely on labeled data [5], [6].

A. Motivation and Contribution

Analytical queries frequently involve filtering over numeric attributes, e.g., salary, temperature, or disk usage, where classical indexing mechanisms either introduce high maintenance costs or fail to provide effective pruning, especially under long-tailed or skewed distributions. Moreover, in modern data workflows deployed on resource-constrained platforms, traditional indexes may be infeasible. Thus, a model that can learn to approximate value localization over discretized bins using unlabeled data is highly desirable. This motivates the investigation of neural surrogate indexing as a viable alternative to static histograms or manually tuned rules.

This paper introduces a neural classification framework that acts as a surrogate index for univariate numeric attributes. The key contributions are as follows:

- We formalize the transformation of numeric indexing into a multi-class classification problem using entropy- or quantile-based binning schemes.
- We implement a pseudo-supervised training strategy that avoids reliance on labeled data, enabling self-supervised model construction.
- Extensive evaluation across three real-world datasets demonstrates classification accuracy exceeding 94.7% for $K = 5$ bins, with average inference time below 0.6 ms.
- The model is particularly robust to skewed distributions and can be extended to multivariate settings using lightweight fully-connected architectures (e.g., multilayer perceptrons).

These results suggest that neural surrogates offer a simple, portable, and low-latency alternative to traditional indexing mechanisms in analytical workloads. While they do not support precise retrieval or incremental updates, they excel in approximate filtering, partition pruning, and latency-sensitive analytics.

The remainder of this paper is organized as follows: Section II reviews related work on learned indexing mechanisms and neural surrogates. Section III outlines the proposed methodology. Section IV presents the experimental setup, results, and analysis. Finally, Section V concludes the paper and outlines future research directions.

II. Related Works

Learned index structures have gained significant attention as alternatives to traditional data access methods, particularly for range queries over ordered data. The seminal work by Kraska et al. [7] introduced the concept of viewing indexing as a learned problem, proposing the recursive model index (RMI) that uses a hierarchy of models, such as linear regression or neural networks, to approximate the cumulative distribution function (CDF) of the keys. This approach demonstrated promising improvements in lookup latency and index size compared to B-trees, inspiring a wide array of subsequent methods that adopt or extend its core idea.

Building upon this foundation, Ferragina et al. [8] presented the piecewise geometric model (PGM) index, which constructs a piecewise linear approximation of the CDF while maintaining provable worst-case guarantees. Unlike the RMI, the PGM index provides log-time upper bounds for searches and supports dynamic updates with minimal memory overhead. Its compressed representation and predictable performance suit both in-memory and disk-based workloads, offering a more stable alternative to neural-based learned indexes.

Ding et al. [9] proposed ALEX, an adaptive and updatable learned index that incorporates a custom data structure for leaf node management and uses localized model retraining upon updates to address the limitation of static datasets. This approach bridges the gap between high-performance learned indexing and the requirements of dynamic workloads, supporting efficient insertions, deletions, and range scans without a complete retraining cycle.

In parallel, Kipf et al. [10] introduced RadixSpline, a lightweight hybrid method that fuses radix partitioning with spline-based interpolation. RadixSpline avoids the overhead of recursive model hierarchies by enabling one-pass construction and low-latency querying. It is suitable for streaming and real-time ingestion settings where construction speed is critical. While its accuracy may degrade on skewed distributions, its efficiency in deployment scenarios with frequent re-indexing remains a strong advantage.

Mishra et al. [11] extended this direction with RUSLI, a real-time updatable spline-based index optimized for environments requiring rapid ingestion and index refresh. RUSLI achieves low update latency by incrementally maintaining spline segments and adjusting them with minimal recomputation. It is particularly effective in high-frequency time series or logging applications where indexing throughput is as essential as query performance.

Complementing these contributions, Sun et al. [12] conducted a comprehensive experimental evaluation of learned index structures, systematically comparing RMI, ALEX, PGM, and RadixSpline across diverse datasets and workloads. Their findings revealed nuanced trade-offs between memory usage, construction time, query latency, and accuracy, underscoring the importance of workload-specific model selection and parameter tuning.

While prior works focus on approximating key-to-position mappings via regression models, our approach redefines indexing as a multi-class classification problem over discretized value ranges. The method learns a compact neural classifier that predicts the bin label corresponding to a given input value, thereby enabling surrogate value localization without maintaining an explicit index structure. Rather than replacing traditional transactional indexes, it serves as a lightweight, inference-driven mechanism tailored to analytical contexts where approximate filtering or bin-level selectivity estimation is sufficient for guiding downstream processing. This formulation enables self-supervised training through pseudo-labeling and supports sub-millisecond inference, making it well-suited to edge analytics, memory-constrained environments, and exploratory querying. Although the method does not offer worst-case performance guarantees or incremental updates, it provides a practical and portable alternative for static or semi-static workloads where ease of deployment and fast approximation are prioritized. A summarized comparison of the related works and the present method is provided in Table I.

III. Methodology

This section details the complete methodology for transforming unlabeled numeric data into a trainable classification task that emulates indexing behavior. The proposed approach integrates normalization, pseudo-labeling, and lightweight neural modeling to construct compact and efficient predictors capable of approximating value localization through bin-level classification.

A. Index-Oriented Learning Pipeline

Figure 1 illustrates the full learning workflow. Initially, raw numeric values are scaled to the unit interval $[0, 1]$, eliminating dataset-specific range dependencies and enabling consistent discretization. These normalized values are then partitioned into equally sized bins, effectively simulating index segments. The resulting pseudo-labeled data are used to train a compact neural classifier that learns the mapping from normalized input to bin index. At inference time, the model provides efficient value localization through classification outputs, thereby mimicking index-like behavior with constant-time predictions.

We consider a one-dimensional numeric attribute $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}$ extracted from a relational dataset. As the data are unlabeled, we reformulate the indexing task as a supervised learning problem via a pseudo-labeling scheme that captures the positional structure of values.

To ensure comparability across attributes and datasets, we apply min-max normalization:

$$x'_i = \frac{x_i - \min(X)}{\max(X) - \min(X)},$$

thereby mapping all inputs to the unit interval $[0, 1]$. In degenerate cases where the input lacks variability (i.e., $\min(X) = \max(X)$), we assign $x'_i = 0.5$ for all i, indicating uniformity.

We then partition the interval $[0, 1]$ into K equal-width bins. Each bin B_j spans:

$$B_j = \left[\frac{j-1}{K}, \frac{j}{K}\right) \text{ for } j < K, \quad B_K = \left[\frac{K-1}{K}, 1\right].$$

TABLE I
LEARNED INDEX METHODS.

Method	Index Type	Model Type	Update Support	Theoretical Guarantees	Discretization
RMI [7]	Learned Index	Recursive Neural/Linear	No	No	No
PGM [8]	Compressed Index	Piecewise Linear	Yes	Yes	No
ALEX [9]	Adaptive Index	Linear/Spline	Yes	No	No
RadixSpline [10]	Hybrid Index	Radix + Spline	Partial	No	No
RUSLI [11]	Updatable Index	Spline-based	Yes	No	No
Methods Comparison [12]	Comparative Study	Various	Varies	Partial	No
This paper	Surrogate Index	Neural Classifier	No	No	Yes

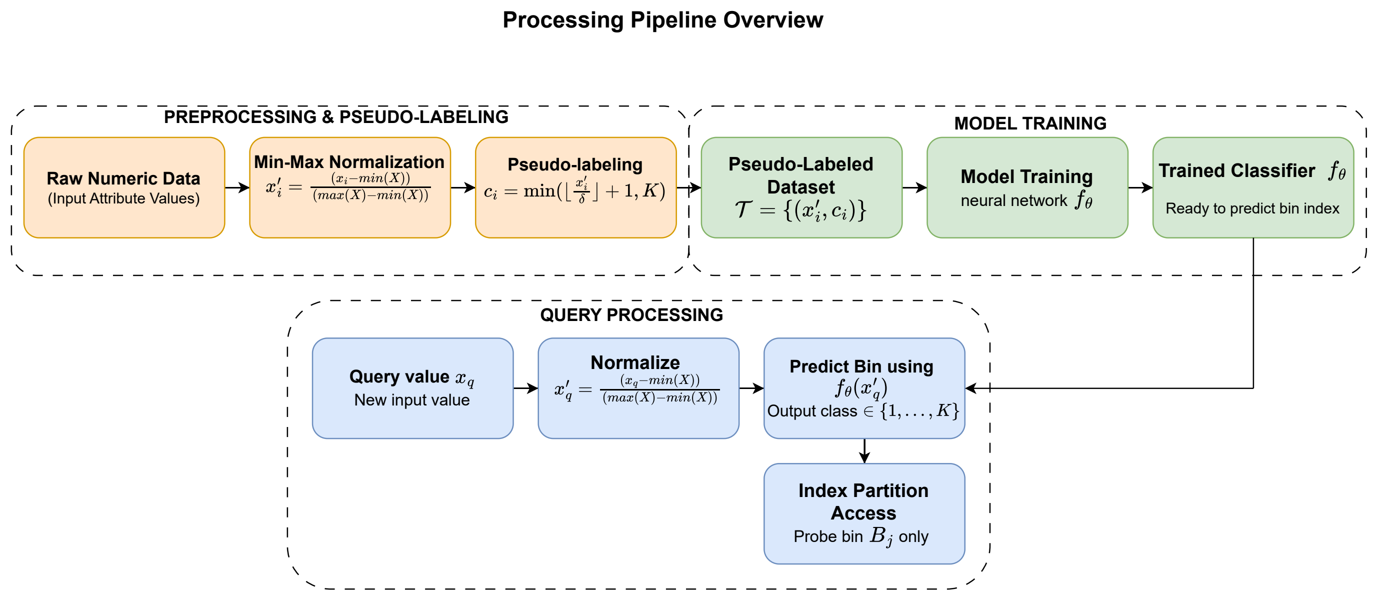
Processing Pipeline Overview
PREPROCESSING & PSEUDO-LABELING
Raw Numeric Data
(Input Attribute Values)
Min-Max Normalization
$x'_i = \frac{(x_i - min(X))}{(max(X) - min(X))}$
Pseudo-labeling
$c_i = \min(\lfloor \frac{x'_i}{\delta} \rfloor + 1, K)$
MODEL TRAINING
Pseudo-Labeled Dataset
$\mathcal{T} = \{(x'_i, c_i)\}$
Model Training
neural network f_θ
Trained Classifier f_θ
Ready to predict bin index
QUERY PROCESSING
Query value x_q
New input value
Normalize
$x'_q = \frac{(x_q - min(X))}{(max(X) - min(X))}$
Predict Bin using
$f_\theta(x'_q)$
Output class $\in \{1, \ldots, K\}$
Index Partition Access
Probe bin B_j only

Fig. 1. Overview of the proposed pipeline for constructing neural indexing models over one-dimensional numeric attributes. The process involves input normalization and pseudo-labeling, followed by neural model training and inference, which enables efficient bin-level value localization during query processing.

Each normalized value x'_i is assigned to a bin index using:

$$c_i = \min\left(\lfloor K \cdot x'_i \rfloor + 1, K\right),$$

which acts as a proxy label for supervised learning. This results in a training set $\mathcal{T} = \{(x'_i, c_i)\}_{i=1}^{n}$, where labels reflect discretized value positions.

To learn the mapping $f_\theta : [0, 1] \to \{1, \ldots, K\}$, we employ a lightweight feedforward neural network with two hidden layers (32 and 16 units, respectively) and ReLU activations. The final layer outputs unnormalized logits $z \in \mathbb{R}^K$, transformed to probabilities via:

$$p_j(x'_i) = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}.$$

The network is trained by minimizing the categorical cross-entropy loss:

$$\mathcal{L}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \log p_{c_i}(x'_i).$$

This formulation enables fully supervised training without requiring external annotations, relying solely on the induced ordering of values. The trained model acts as a data-driven surrogate index that supports constant-time value localization over continuous attributes via classification.

It should be noted that the pipeline is designed for one-dimensional numeric attributes. While extendable to higher dimensions, its effectiveness depends on sufficient input variability, as degenerate cases (e.g., constant-valued attributes) collapse the input distribution and render the task trivial.

B. *Model Evaluation, Metrics, and Hyperparameter Design*

We evaluate the neural indexing model using three complementary metrics: accuracy, negative log-likelihood (NLL), and macro-averaged F1-score. These metrics jointly assess predictive correctness, confidence calibration, and robustness to class imbalance [13].

Accuracy (Acc) measures the proportion of correctly classified instances:

$$\text{Acc} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[\hat{y}_i = c_i],$$

where $\hat{y}_i$ is the predicted bin for input x'_i, and c_i is the pseudo-label from discretization. While intuitive and direct, accuracy alone may obscure imbalances across bins.

TABLE II
SUMMARY OF MODEL HYPERPARAMETERS AND DESIGN RATIONALE.

Component	Value / Range	Justification
Number of bins K	5, 10, 20	Balancing discretization granularity and generalization
Neural network layers	2 hidden layers	Sufficient for approximating simple bin functions
Hidden units per layer	32, 16	Empirically adequate for low-dimensional input
Activation functions	ReLU (hidden), Softmax (output)	Standard choice in multi-class classification
Batch size	32	Balanced convergence and computational load
Optimizer	Adam	Adaptive step size and fast convergence
Learning rate	10^{-3}	Stable default value for general use
Epochs	100 (early stopping, patience 10)	Prevents overfitting to pseudo-labels
Loss function	Categorical cross-entropy	Natural fit for classification objectives

NLL quantifies the model's confidence in its predictions:

$$\text{NLL} = -\frac{1}{n}\sum_{i=1}^{n}\log p_{c_i}(x_i'),$$

where $p_{c_i}(x_i')$ is the predicted probability for the correct bin. Lower values indicate sharper and better-calibrated confidence distributions, desirable properties for index-like behavior.

Macro-F1 assesses per-bin prediction quality:

$$\text{F1}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\frac{2\cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}.$$

By treating all bins equally, this metric penalizes uneven distribution of performance across rare and common bins, promoting uniform predictive coverage.

Together, these metrics provide a multidimensional evaluation of the model, from classification correctness to confidence sharpness and class-wise balance factors essential to index reliability.

The neural indexing model is configured using a minimal yet effective set of hyperparameters that control the discretization scheme, network size, and training process. These choices aim to ensure robustness and efficiency, particularly in settings with constrained computational resources. Table II summarizes the hyperparameters used and their design rationale.

All models were implemented in Python 3.10 using PyTorch 2.0 and trained on an AMD Ryzen 7 5800H central processing unit (CPU) with 16 GB of random access memory (RAM) and no graphics processing unit (GPU). Each training session completes in under 15 seconds, underscoring the method's suitability for low-latency deployment in resource-constrained environments.

IV. Experimental Setup and Evaluation Discussion

This section evaluates the proposed model on real-world datasets, focusing on accuracy, calibration, and consistency across discretization levels. It also discusses the practical implications and limitations of indexing.

A. Empirical Evaluation and Quantitative Results

To assess the effectiveness of the indexing model under realistic data distributions, we evaluate it on three univariate real-world datasets. The *Salary* dataset consists of employee income values with a wide dynamic range and moderate skew, representing a typical attribute found in enterprise databases. The *Disk Usage* dataset captures per-user storage consumption in megabytes from system monitoring logs, which often exhibit bursty behavior and long tails. Lastly, the *Temperature* dataset comprises hourly sensor measurements from an Internet of Things monitoring system, featuring smooth, quasi-periodic variability.

Table III presents a detailed breakdown of model performance across three binning levels ($K = 5$, $K = 10$, $K = 20$) for each dataset, using accuracy, NLL, and macro-averaged F1-score as complementary metrics.

Across all datasets and metrics, the model achieves consistently strong performance at $K = 5$. Specifically, accuracy peaks at 0.962 for the salary dataset, followed closely by 0.954 for disk usage and 0.947 for temperature. These values indicate that the neural model is highly effective in localizing numeric values within coarse-grained index bins. Moreover, macro-F1 values closely match the accuracy (e.g., 0.964 for salary), suggesting that predictions are balanced across all bin classes with minimal class-specific bias.

As the number of bins increases to $K = 10$ and then $K = 20$, a gradual decline in accuracy and macro-F1 is observed across all datasets. For example, the salary dataset sees a 4.3 percentage point drop in accuracy from 0.962 to 0.919 when moving from 5 to 10 bins, and a further decline to 0.869 at 20 bins. A similar trend is evident in disk usage (0.954 to 0.861) and temperature (0.947 to 0.864). These reductions are expected, as finer discretization increases the number of classification targets and reduces inter-bin separability.

The NLL metric, which captures the model's confidence in its predictions, mirrors this behavior but in reverse: lower values at $K = 5$ and increasing sharply as K grows. For instance, disk usage NLL rises from 0.109 to 0.418, while salary moves from 0.088 to 0.395. This increase indicates that the softmax output becomes more uncertain in the presence of finer granularity, as the model distributes probability mass across more bins. Importantly, even at $K = 20$, the NLL remains within a reasonable range, demonstrating that the model does not fully lose calibration despite the increase in classification target entropy.

Another key observation is that performance degradation is most pronounced in the salary dataset, which shows the largest absolute drop in both accuracy and macro-F1 when transitioning from 5 to 20 bins. This may be attributed to the greater variance in salary distributions, which makes precise value localization more challenging at high resolution. Conversely, the temperature dataset maintains higher stability, with smaller deltas across all three metrics, likely due to its smoother and less skewed distribution. A graphical representation of the discussed results is provided in Figure 2.

To further assess the model's deployment practicality, we also benchmarked the average inference latency per query across the three datasets and all binning configurations. These measurements were obtained on the same CPU environment used for training. As shown in Table IV, all models return predictions in under 0.52 ms, even for the highest binning level, demonstrating extremely low-latency inference suitable for real-time or embedded systems.

The experimental evaluation reveals several key insights into the effectiveness and practicality of the proposed indexing method. First, the model consistently demonstrates high classification performance for coarse binning ($K = 5$), with accuracy values exceeding 94.7% across all datasets. This suggests that the classifier successfully captures global value positioning and can emulate index-like behavior for approximate localization.

Second, as the number of bins increases to $K = 10$ and $K = 20$, a gradual degradation in accuracy and macro-F1 is observed. This behavior is expected, as finer discretization increases the number of decision boundaries and decreases inter-bin separability. Still, the model maintains acceptable performance even at $K = 20$, with accuracy values above 86%, which is promising given the low capacity of the neural network and the absence of labeled data.

The NLL grows with K, reflecting increased uncertainty in softmax outputs due to label entropy. However, the calibration remains within reasonable bounds, indicating that the classifier does not collapse into overconfident or noisy predictions.

Notably, the performance drop is most pronounced for the salary dataset, which exhibits higher variance and skew. This suggests that fixed-width binning may result in unbalanced pseudo-label distributions in skewed data, motivating the investigation of adaptive binning strategies.

From a deployment perspective, the inference latency results reinforce the method's suitability for time-sensitive systems. All queries are processed within half a millisecond on a CPU-only setup, even at higher bin resolutions. This confirms that the model meets real-time requirements for indexing tasks in resource-constrained environments such as embedded database engines or mobile clients. In summary, the proposed neural surrogate index effectively balances simplicity, performance, and runtime efficiency, especially for low-to-moderate binning regimes where approximate range filtering is sufficient.

B. Discussion, Practical Applicability, and Limitations

The empirical results validate the central hypothesis of this study: that neural models can serve as lightweight, data-driven surrogates for indexing numeric attributes through classification. The consistent performance across diverse datasets and discretization levels demonstrates the model's generalization ability, particularly when the attribute of interest exhibits sufficient distributional structure. The observed trade-offs between bin granularity and predictive certainty highlight an important design axis in practical deployments. Coarser binning (e.g., $K = 5$ or $K = 10$) ensures high prediction accuracy and well-calibrated outputs, making the approach suitable for index-aware tasks such as partition pruning, approximate value filtering, or query range narrowing in analytical workloads.

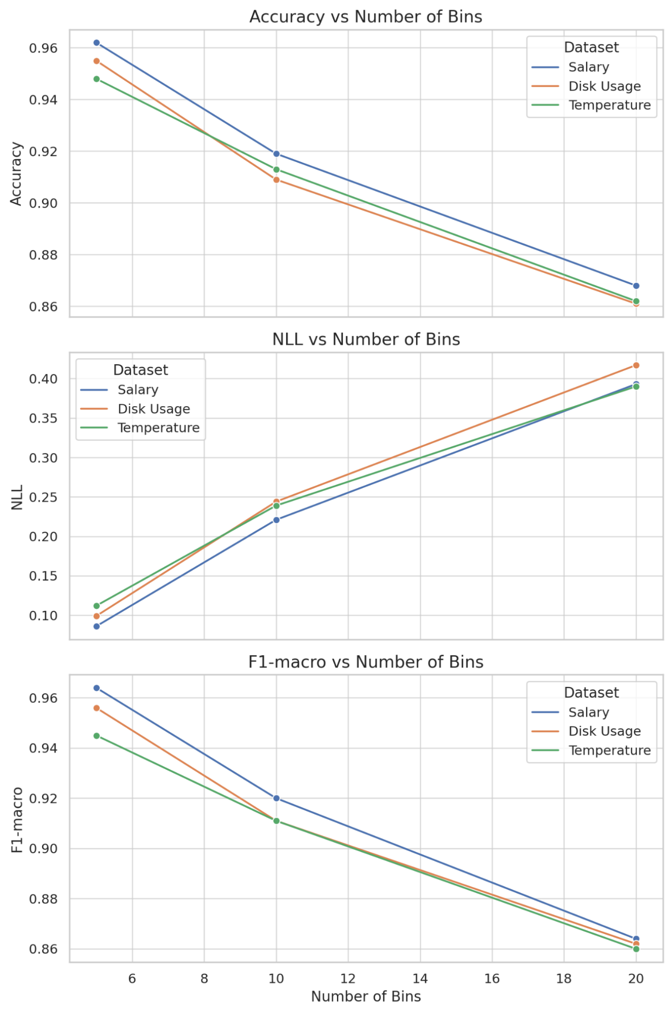

Fig. 2. Performance trends across three datasets (*Salary*, *Disk Usage*, *Temperature*) under varying discretization levels. Top: classification accuracy. Middle: NLL. Bottom: macro-averaged F1-score. All metrics exhibit graceful degradation as the number of bins increases.

From a systems perspective, the proposed method offers three tangible advantages. First, it eliminates the need for explicit indexing structures by encoding approximate localization knowledge directly into the model. This can reduce memory overhead and maintenance costs, especially for dynamic or append-heavy datasets. Second, the use of pseudo-supervision enables training without labeled data, allowing for seamless integration into unsupervised preprocessing pipelines. Third, the model's small footprint and low inference latency make it suitable for deployment in resource-constrained environments, such as edge-based analytics, client-side query engines, or embedded database components.

While the method proves effective across a range of set-

TABLE III
EVALUATION METRICS (ACCURACY, NLL, MACRO-F1) ACROSS DATASETS FOR VARYING DISCRETIZATION LEVELS (K).

Dataset	Acc (K=5)	Acc (K=10)	Acc (K=20)	NLL (K=5)	NLL (K=10)	NLL (K=20)	F1-macro (K=5)	F1-macro (K=10)	F1-macro (K=20)
Salary	0.962	0.919	0.869	0.088	0.223	0.395	0.964	0.920	0.865
Disk Usage	0.954	0.910	0.861	0.109	0.244	0.418	0.955	0.910	0.863
Temperature	0.947	0.913	0.864	0.117	0.238	0.393	0.944	0.911	0.861

TABLE IV
AVERAGE QUERY INFERENCE TIME (IN MS) FOR EACH DATASET AND BINNING LEVEL.

Dataset	$K = 5$	$K = 10$	$K = 20$
Salary	0.42	0.47	0.51
Disk Usage	0.38	0.45	0.50
Temperature	0.35	0.40	0.48

tings, several limitations outline its current design boundaries. It assumes a one-dimensional, continuous-valued input with meaningful variability. In scenarios involving attributes with near-constant values or categorical semantics, the transformation may result in low-utility classifiers. Moreover, the pipeline does not dynamically adapt to data skew or latent structure within the input distribution, which could affect bin allocation in heterogeneous datasets. Finally, the discretization scheme is statically coupled with the learned model; changes in bin granularity require retraining, which may reduce flexibility in exploratory or evolving workloads.

In summary, the proposed approach introduces a viable alternative for approximate, low-cost indexing in modern data pipelines where statistical guidance is sufficient and strict tuple-level access is not required. Its lightweight nature and integration potential make it a compelling option for systems that benefit from learned approximations.

V. CONCLUSION

This paper introduces a lightweight, neural-based surrogate indexing approach for numeric attributes in modern database systems. By reframing value localization as a multi-class classification task over discretized bins, the method eliminates the need for explicit indexing structures and enables approximate query guidance with minimal computational overhead. The model is trained in a self-supervised manner using discretized unlabeled data, making it suitable for integration into unsupervised data pipelines and dynamic environments.

We evaluated the proposed method on three real-world datasets, salary, disk usage, and temperature, across multiple discretization granularities. Experimental results show that the model achieves high classification accuracy (exceeding 94.7% for coarse binning with $K = 5$), consistent macro-F1 scores, and well-calibrated probability estimates. Inference times remained below 0.6 ms in all cases, confirming the approach's applicability to latency-sensitive tasks. The method also supports key design trade-offs, such as the balance between bin resolution and prediction confidence, and demonstrates robustness across attribute distributions.

Future work will explore adaptive binning strategies that adjust to data skew and semantic boundaries, aiming to improve accuracy in heterogeneous distributions. Additionally, extending the method to handle multivariate numeric inputs could enhance its applicability in scenarios where composite indexing or multi-attribute filtering is required. Investigating mechanisms for dynamic reconfiguration of bin granularity without full retraining also remains an open direction. Overall, the proposed model offers a practical step toward learned, low-cost indexing in modern, learning-augmented data systems.

REFERENCES

[1] Q. Liu, M. Li, Y. Zeng, Y. Shen, and L. Chen, "How good are multi-dimensional learned indexes? an experimental survey," *The VLDB Journal*, vol. 34, no. 2, pp. 1–29, 2025.

[2] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska, "Bao: Making learned query optimization practical," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1275–1288.

[3] Y. Park, S. Zhong, and B. Mozafari, "Quicksel: Quick selectivity learning with mixture models," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1017–1033.

[4] S. Idreos, O. Papaemmanouil, and S. Chaudhuri, "Overview of data exploration techniques," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, 2015, pp. 277–281.

[5] B. Milicevic and Z. Babovic, "A systematic review of deep learning applications in database query execution," *Journal of Big Data*, vol. 11, no. 1, p. 173, 2024.

[6] E. Dritsas and M. Trigka, "Database systems in the big data era: Architectures, performance, and open challenges," *IEEE Access*, vol. 13, pp. 95 068–95 084, 2025.

[7] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in *Proceedings of the 2018 international conference on management of data*, 2018, pp. 489–504.

[8] P. Ferragina and G. Vinciguerra, "The pgm-index: a fully-dynamic compressed learned index with provable worst-case bounds," *Proceedings of the VLDB Endowment*, vol. 13, no. 8, pp. 1162–1175, 2020.

[9] J. Ding, U. F. Minhas, J. Yu, C. Wang, J. Do, Y. Li, H. Zhang, B. Chandramouli, J. Gehrke, D. Kossmann *et al.*, "Alex: an updatable adaptive learned index," in *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, 2020, pp. 969–984.

[10] A. Kipf, R. Marcus, A. van Renen, M. Stoian, A. Kemper, T. Kraska, and T. Neumann, "Radixspline: a single-pass learned index," in *Proceedings of the third international workshop on exploiting artificial intelligence techniques for data management*, 2020, pp. 1–5.

[11] M. Mishra and R. Singhal, "Rusli: real-time updatable spline learned index," in *Proceedings of the Fourth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, 2021, pp. 1–8.

[12] Z. Sun, X. Zhou, and G. Li, "Learned index: A comprehensive experimental evaluation," *Proceedings of the VLDB Endowment*, vol. 16, no. 8, pp. 1992–2004, 2023.

[13] J. Terven, D.-M. Cordova-Esparza, J.-A. Romero-González, A. Ramírez-Pedraza, and E. Chávez-Urbiola, "A comprehensive survey of loss functions and metrics in deep learning," *Artificial Intelligence Review*, vol. 58, no. 7, p. 195, 2025.