BlinkViz: Fast and Scalable Approximate Visualization on Very Large Datasets using Neural-Enhanced Mixed Sum-Product Networks

Yimeng Qiao[*]
Fudan University
Shanghai, China
ymqiao20@fudan.edu.cn

Yinan Jing[*][†]
Fudan University
Shanghai, China
jingyn@fudan.edu.cn

Hanbing Zhang[*]
Fudan University
Shanghai, China
hbzhang17@fudan.edu.cn

Zhenying He
Fudan University
Shanghai, China
zhenying@fudan.edu.cn

Kai Zhang
Fudan University
Shanghai, China
zhangk@fudan.edu.cn

X. Sean Wang
Fudan University
Shanghai, China
xywangCS@fudan.edu.cn

ABSTRACT

Web-based online interactive visual analytics enjoys popularity in recent years. Traditionally, visualizations are produced directly from querying the underlying data. However, for a very large dataset, this way is so time-consuming that it cannot meet the low-latency requirements of interactive visual analytics. In this paper, we propose a learning-based visualization approach called BlinkViz, which uses a learned model to produce approximate visualizations by leveraging mixed sum-product networks to learn the distribution of the original data. In such a way, it makes visualization faster and more scalable by decoupling visualization and data. In addition, to improve the accuracy of approximate visualizations, we propose an enhanced model by incorporating a neural network with residual structures, which can refine prediction results, especially for visual requests with low selectivity. Extensive experiments show that BlinkViz is extremely fast even on a large dataset with hundreds of millions of data records (over 30GB), responding in sub-seconds (from 2ms to less than 500ms for different requests) while keeping a low error rate. Furthermore, our approach remains scalable on latency and memory footprint size regardless of data size.

CCS CONCEPTS

• **Human-centered computing** → *Visualization*; **Visualization toolkits**; • **Information systems** → **Database web servers**.

KEYWORDS

visualization, sum-product networks, neural networks, approximation

[*]All authors contributed equally to this research.
[†]Corresponding author.

ACM Reference Format:
Yimeng Qiao, Yinan Jing, Hanbing Zhang, Zhenying He, Kai Zhang, and X. Sean Wang. 2023. BlinkViz: Fast and Scalable Approximate Visualization on Very Large Datasets using Neural-Enhanced Mixed Sum-Product Networks. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3543507.3583411

1 INTRODUCTION

The continuous appealing of web-based online interactive visual analytics tools such as Tableau Online [2] and Apache Superset [1] makes more and more companies elaborate strategies to generate actionable insights. By leveraging these analytics tools, data scientists monitor large datasets from various sources, enabling visualizing and analyzing on the fly. For example, Tableau Online [2] provides business intelligence web application, making users analyze and collaborate from anywhere. However, under such circumstances, it is vital to produce visualizations in interactive timescales. Previous research [20] has shown that once the time to generate a visualization exceeds 500 milliseconds, high latency would significantly hinder the user's performance and decision-making behavior. Traditionally, visualizations are produced directly from querying the underlying data, which is stored on premise or cloud servers. Obviously, the larger the data size, the longer the visualization takes. Specifically, for a very large dataset, this way is so prohibitively time-consuming that it cannot meet the low-latency requirements of interactive visual analytics. For example, it takes more than 20 seconds to execute a simple aggregation query on a Flights dataset with 500 million records, which is unacceptable.

To reduce the time cost of visualizations on large datasets, there are some approaches [4, 10, 17, 29, 38] using samples instead of the original data to produce approximate visualizations by trading accuracy for faster response to visualization requests. However, when the data is large, these sampling-based approximate visualization methods still suffer from a long latency because more samples are needed to achieve acceptable accuracy. For instance, in the experiments of IFocus [17], when the number of groups in a visualization exceeds 20, at least 20% sampling rate is required to return relatively reliable visualizations. Therefore, traditional visualization methods are tightly coupled to the underlying data or samples.

In this paper, we shift the paradigm of visualization as shown in Figure 1. Traditional online visualization techniques send requests to the data server, perform query on it and return results, which is accompanied by large latency. Instead of producing visualizations from the underlying data or samples on the data server, we propose a learning-based visualization approach called *BlinkViz*, which uses a learned model to substitute data samples to create approximate visualizations. The core of BlinkViz comprises several parts. One of the main parts is the Mixed Sum-Product Network (MSPN) [14], an unsupervised data-driven model that can be used to learn the original data distribution. MSPN can be used to answer approximately visual requests. However, a single MSPN cannot guarantee a high accuracy for approximate visualizations, especially for those visual requests with low selectivity[1]. Hence, to improve the accuracy of approximate visualizations, we incorporate a neural network with residual structures into the core of BlinkViz and combine it with multiple MSPNs. The neural network is a supervised query-driven model which can be used to refine prediction results, especially for low-selectivity visual requests.

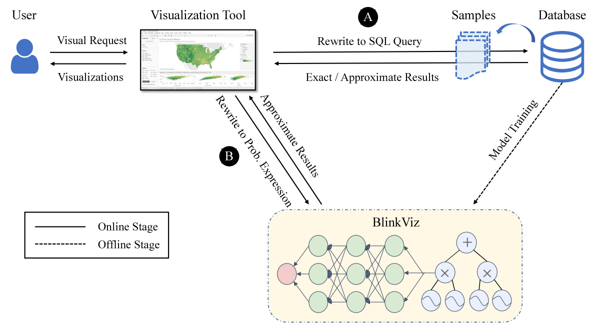


Figure 1: Framework of BlinkViz

By decoupling visualization and data and using a learned model as a middleware to answer visualization requests, BlinkViz has three advantages compared to traditional visualization methods. First is *fast*, as BlinkViz leverages model inference to answer visualization requests instead of query processing behind traditional methods. Models are generally much smaller than large data, so BlinkViz is extremely fast, even on a large dataset with hundreds of millions of data records (over 30GB), responding in a sub-second. Second is *scalable*. As we all know, the main time cost of traditional visualization methods is the query processing time. In general, query processing time increases along with the size of the data. When the volume of data is large, the time overhead of visualization will inevitably be large. In contrast, BlinkViz relies only on the model at the core, not on the large volume of the data or samples. Therefore, BlinkViz remains scalable on latency and memory footprint size regardless of data size. Third is *lightweight*. BlinkViz is lightweight due to its small memory footprint. Also, the model can be kept small no matter how large the data is. This advantage makes it possible to deploy BlinkViz on mobile devices to provide visualization services without relying on the data server, especially in collaborative mobile office or offline use scenarios.

[1]The selectivity is the fraction of records in a table that is chosen by the predicate. It is a number between 0 and 1.

In summary, we make the following contributions:

- We shift the paradigm of data visualization and propose a learning-based visualization approach BlinkViz, which uses a learned model to answer visualization requests instead of data or samples. By decoupling visualizations and data, BlinkViz can be extremely fast even on a very large dataset and more scalable compared to traditional visualization methods.
- We propose a neural-enhanced model that integrates multiple mixed sum-product networks with a neural network to improve the accuracy of the approximate visualizations produced by BlinkViz, especially for low-selectivity visualization requests.
- We conduct extensive experiments on both a single-table dataset and a multi-table dataset to demonstrate the effectiveness and scalability of BlinkViz. For a large dataset with hundreds of millions of data records (over 30GB), BlinkViz is able to respond in a sub-second while keeping a low error rate. Furthermore, our approach remains scalable on latency and memory footprint size regardless of data size.

2 PRELIMINARIES

2.1 Visual Request Template

We define a simple visual request template similar to DeepEye [22] as shown in Figure 2. Different from visual languages like Vega-lite [31], a visual request is supposed to contain common specifications to describe '*what*' visualizations should be generated rather than '*how*' to make it. Thus, we pay more attention to data manipulation instead of graphical description. A visual request includes two parts: *visualization type* and *data query*.

```
VISUALIZE   [ line | bar | pie ]
SELECT      X, AGGR(Y)
FROM        D
WHERE       [ where clause ]
GROUP BY    X
```

Figure 2: Visual Request Template

- **Visualization type.** The **VISUALIZE** clause specifies the visualization type. In principle, BlinkViz is able to generate complex visualizations like high-dimensional charts. For simplicity, this paper focuses on three popular two-dimensional visualization types - bar charts, line charts, and pie charts.
- **Data query.** Behind each visual request is a typical SQL SPJA[2] (Select-Project-Join-Aggregation) analytical query, as shown in Figure 2. In a two-dimensional chart, the X-axis is usually the categorical group-by attribute, and the Y-axis is some aggregated value of a specific column to facilitate data analysis. BlinkViz supports common aggregation queries, including COUNT, AVG, SUM, etc.

[2]SPJA queries are those that consist of any combinations of select, project and join operators followed by an aggregation operator.

Once a visual request is posed, we need to rewrite it as a SQL query and execute it on a relational database for traditional visualization methods, as shown in Figure 1. Instead, BlinkViz will rewrite the request to a probabilistic expression and conduct the model inference to answer the request.

2.2 Approximate Visualization

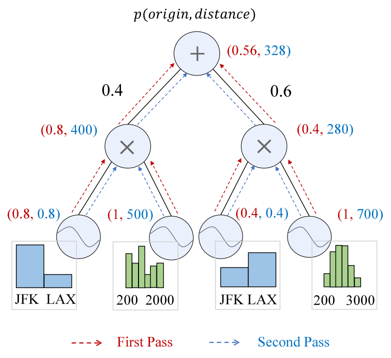
Since obtaining exact visualizations on large datasets is too time-consuming, this paper focuses on approximate visualization methods. Approximate methods can achieve faster response by sacrificing the accuracy of visualizations to some extent. Traditional approximate methods are usually sampling-based. In this paper, we investigate a learning-based approximate visualization approach.

2.3 Mixed Sum-Product Networks

In recent years, the deep generative models have been applied in database learning, such as deep autoregressive models [37], Normalizing Flow [34] and VAE [33], to build density estimators of data. However, these models above are *intractable* when answering complex queries like marginal probability and expectations, so they perform approximate inference with no guarantees. Inspired by DeepDB [14], we turn to *tractable probabilistic models* and choose to learn the data distribution via Mixed Sum-Product Networks [23], which enables exact inference for complex queries in polynomial time.

origin	distance (km)
JFK	267
LAX	236
JFK	1597
JFK	1182
⋮	⋮
⋮	⋮
LAX	954
JFK	1028
JFK	264
LAX	2454
LAX	1536

(a) Example Table



(b) Trained MSPN

Figure 3: Tailored Flights table with corresponding MSPN

Mixed Sum-Product Networks (MSPNs) is a variant of Sum-Product Networks (SPNs) [28]. SPN is a probabilistic model that learns the joint probability distribution of the variables in a dataset. Conventional SPNs learning algorithm [11] recursively splits the dataset into clusters of rows or columns. Different row clusters are combined with a sum node, while column clusters are combined with a product node to construct a tree finally. MSPNs extend SPNs to fit hybrid domains by adopting piecewise polynomial leaf distributions like histograms. As illustrated in Figure 3, the example MSPN learns the joint probabilistic distribution of attributes *origin* and *distance* in a tailored Flights table. In the beginning, the whole table is split into two groups, accounting for 40% and 60% of the total data, respectively, and existing as left and right children of the root sum node. The left group is dominated by flights from JFK Airport, while the right group contains most flights departing from LAX Airport. Then, since there is weak correlation between attributes *origin* and *distance*, both groups are split into two leaves connected by a product node. In the end, the leaf nodes represent the distribution of corresponding attributes via histograms.

Now consider an example query to compute the average travel distance of flights departing from JFK Airport. In the inference stage, this query could be transformed into computing a conditional expectation $\mathbb{E}(distance|origin = JFK)$, which is equal to $\mathbb{E}(distance \cdot 1_{origin=JFK})/P(origin = JFK)$, and these two parts can be computed by MSPN by two passes as shown in Figure 3(b). In the first pass (red line), the probability of $origin = JFK$ is inferred bottom-up. In the leaves of the left cluster, the probabilities of that are 0.8 and 1, respectively. Then the probabilities propagate along the tree, and the value of the left product node becomes 0.8. Similarly, the probability of the right product node is 0.4. Hence, we have $P(origin = JFK) = 0.8 \times 0.4 + 0.4 \times 0.6 = 0.56$. In the same way, we can reach the answer of $\mathbb{E}(distance \cdot 1_{origin=JFK}) = 328$ by the second pass (blue line). Finally, we have the answer to the query $\mathbb{E}(distance|origin = JFK) = 328/0.56 = 585.7$, which means the average distance of flights departing from JFK Airport is 585.7 kilometers. Following a similar manner as [14], all SPJA queries in visualization tasks on both single and multiple relational tables can be transformed into probabilistic expressions and answered approximately by inference via MSPNs.

3 BLINKVIZ: A LEARNING-BASED APPROACH

The overall model architecture of BlinkViz is illustrated in Figure 4. Our model consists of two modules, combining the unsupervised and supervised learning modules. First, the data-driven unsupervised module uses several MSPN models trained with different data samples to capture the general distribution of a given dataset. Each MSPN model can be considered as a basic distribution function of the original dataset and can be used to infer probabilities like marginal probability and expectations. Then, in the supervised learning module, to enhance the accuracy of these individual models, we extract some features from MSPNs, which are related to both queries and data, feeding them into a neural network with residual structures. Finally, it takes the estimations of several individual MSPN models as input, aggregates MSPNs's statistic information, and outputs refined predictions.

3.1 Learning Data Distribution by MSPNs

BlinkViz aims to train a machine learning model to learn the data distribution and then can approximately answer visual requests by leveraging this model. As introduced in Section 2, the MSPN model might be a good option for our fundamental requirements. Hence, in this paper, we choose it to represent the data distribution.

3.1.1 MSPN Model Training. Figure 5 shows the process of learning a MSPN. As described in 2.3, the learning algorithm [11] splits the whole dataset by attributes or by records recursively. Each round of recursion determines what kind of split operation needs to be performed. If there is more than one attribute, they would be split into different clusters by attributes according to their correlations measured by a randomized dependency coefficient (RDC) [21]. And

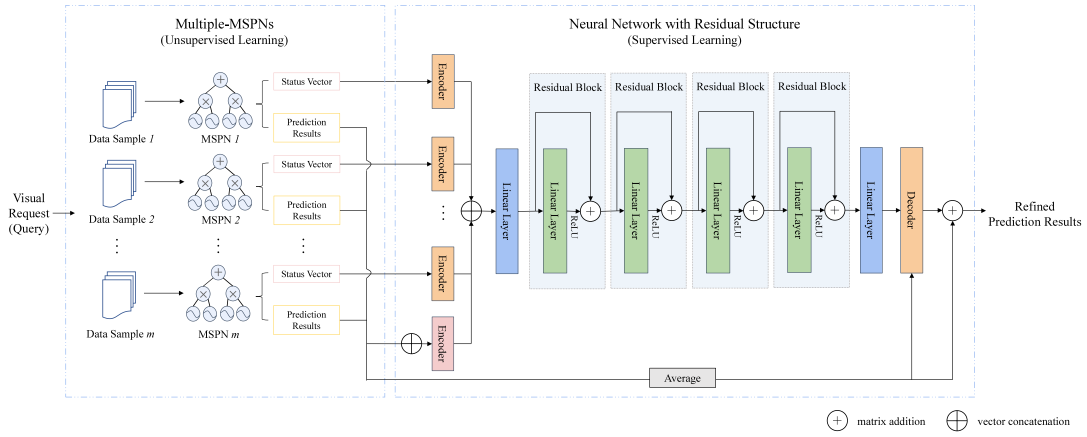


Figure 4: The architecture of Neural-enhanced Mixed Sum-Product Network model

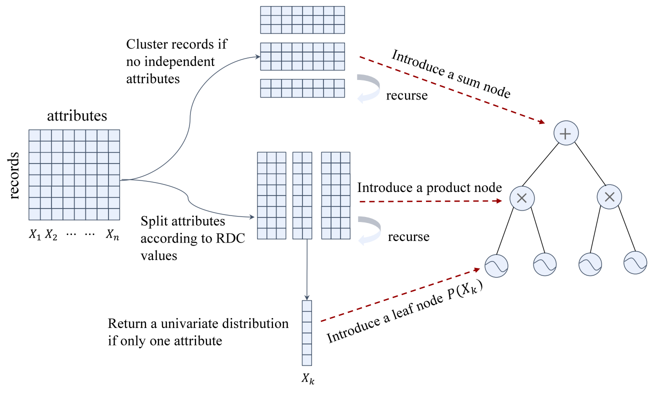

Cluster records if no independent attributes
Introduce a sum node
recurse
attributes
records
$X_1\ X_2\ \cdots\ \cdots\ X_n$
Introduce a product node
Split attributes according to RDC values
recurse
Return a univariate distribution if only one attribute
Introduce a leaf node $P(X_k)$
X_k

Figure 5: The process of learning a MSPN

these clusters are combined by a product node. Then, in each cluster, if the amount of records is larger than a threshold, records would also be clustered into independent groups by rows, which are united by a sum node. Finally, if only one attribute is left and the number of records is less than the given threshold, a univariate distribution of this attribute is returned and introduces a leaf node of this attribute.

3.1.2 Exact Inference. With an MSPN available, given a visual request, we first rewrite the data query into probabilistic expressions as demonstrated in Section 2.3. Then, the conditions in the query are evaluated on every relevant leaf node. Afterward, each probability and expectation involved in the expression is computed via MSPN bottom up. Finally, it returns the result of the entire expression. Here we discuss how to handle all kinds of SPJA queries occurred in visualization tasks.

- **AVG** An AVG query can be considered as computing a conditional expectation. For the example Flights table, given a query, e.g., *SELECT AVG(distance) FROM Flights WHERE origin=JFK*, which can be transformed into $\mathbb{E}(distance|origin = JFK) = \mathbb{E}(distance \cdot 1_{origin=JFK})/P(origin = JFK)$. Both $\mathbb{E}(distance \cdot 1_{origin=JFK})$ and $P(origin = JFK)$ are convenient to infer by MSPN bottom-up, as illustrated in Section 2.3.
- **COUNT** Assuming that the total number of data records of the table is N, then the result of query e.g., *SELECT COUNT(*) FROM Flights WHERE origin=JFK* is $N \times \mathbb{E}(1_{origin=JFK})$, which could be computed in the same manner as AVG.
- **SUM** A SUM query can be treated as AVG*COUNT, for example, *SELECT SUM(distance) FROM Flights WHERE origin=JFK* is equal to *SELECT AVG(distance)· COUNT(*) FROM Flights WHERE origin=JFK*. It can also be computed in the above way, but we need to infer through 3 passes.

In addition, for dataset consists of multiple tables, MSPNs are learned over the full outer join of the tables. Thus, a learned MSPN not only can answer queries on a single table, but also queries on joins of multiple tables.

3.2 Neural-Enhanced Module

For the sake of efficiency, like most machine learning models, MSPNs are trained with data samples. A single model can only capture one side of the data but cannot learn all the features of the underlying data exactly. Thus, only utilizing the MSPN model has two folds of limitations. For one thing, queries with low selectivity frequently occur in exploratory visual analytics, since they often produce interesting and valuable insights. However, for these low-selectivity queries, a single MSPN model might inevitably yield results with high error in the tails of the distribution because some population is ignored or even missing in the visualization due to their rareness. The lower the selectivity of the query, the worse it will be. For another, since MSPN is a data-driven model, it will not consider the characteristics of workloads. Thus, for some queries, e.g., low-selectivity queries, the accuracy of approximate visualizations can only depend on the learned data distribution. To improve the accuracy, we propose to integrate several MSPN models using a neural network to enhance the model as shown in Figure 4.

3.2.1 Model Design. The design of the neural-enhanced module is to unify the features of both data and queries by taking advantage of both the data-driven and query-driven models. The data-driven

MSPN model learns from data directly, which has better generalization ability but is easy to miss information about rare populations, especially for those low-selectivity queries. On the other hand, current query-driven models [8, 19, 32] aim to learn a map between queries and results, which identify patterns in the workload but lack generalization. We notice that when an MSPN makes an inference for a given query, each node of the MSPN keeps specific probabilistic values (which we name as '*status vector*'), which can be considered as a status of the model under corresponding query. By extracting them as the statistical features of MSPNs and feeding them into the neural network, the network can obtain both data and query-related information and get more accurate results.

Besides, the reason why we use multiple MSPN models is that multi-MSPNs can capture a more comprehensive picture of data distribution than a single MSPN and balance the errors caused by different data samples. The training data of MSPNs comes from different data samples, overlapping with each other and the information of rare populations are captured independently. Compared with naive ensembles of MSPNs, we incorporate several MSPNs with a neural network. The neural network learns a non-linear combination of individual MSPNs rather than simple arithmetic averaging or voting, which can eliminate some effects of certain anomaly predictions that lead to significantly deviation from the true result.

As shown in Figure 4, for a given query, each MSPN produces a preliminary prediction and yields a *status vector* during inference. The status vectors are concatenated together after the embedding layer and then are fed into the neural network with the preliminary predictions for training. The encoders, linear layers and decoders are all six-layer perceptrons (MLP) with residual structures. During training, we use exact query results as ground truth.

In addition, there are two reasons why we add residual modules in the network. One is to protect the integrity of the information in the propagation procedures by passing the input directly to the output by adopting several residual blocks in neural networks. Another is to avoid situations where the error fluctuates too much. To this end, we put the average of several preliminary predictions in the decoder as a guideline. As a result, it learns the residual instead of the complete output. Incorporating residual modules can make the learning objective simplified and easy to train. Thus, by leveraging such a neural-enhanced model, we can refine the prediction results to get better results.

3.2.2 *Model Training.*

The inputs of the neural network are the predictions and aforementioned status vectors of MSPNs. When a query arrives, each MSPN makes inference bottom-up, and each node corresponds to certain probability values in this process. For example, as shown in Figure 3(b), when computing the probability in the first pass, we extract the probabilities by pre-order traversal of the nodes and flatten them into a status vector $\boldsymbol{v} = [0.56, 0.8, 0.4, 0.8, 1, 0.4, 1]$. The vector $\boldsymbol{v}$ can be considered as a status of the model under current query. By extracting them as the statistical features of MSPNs and feeding them into the neural network, the network can obtain both data and query-related information and get more accurate results.

In the training stage, we generate 100,000 queries according to the visual request template and evaluate them by several trained MSPNs and PostgreSQL respectively, attaining the above features and ground truth. To keep the numerical stability, we also adopt min-max scaling and log transformation for data normalization. The neural network is randomly initialized without pretraining. We set 3 MSPNs and 4 residual blocks by default. And we use the Adam optimizer [18] in PyTorch with a learning rate of 0.00025. The loss function is L1 loss.

4 EVALUATION

4.1 Experimental Setup

4.1.1 *Hardware and Platform.*

We implement BlinkViz based on SPFlow library [24]. All experiments are conducted on a Ubuntu Linux 18.04.4 LTS machine with Intel Xeon Silver 5215 CPU, Nvidia Titan RTX GPU (24GB), 64GB RAM, and 3.3TB HDD disk.

4.1.2 *Datasets.*

We conduct experiments on both a real-world dataset *Flights* [3] and a synthetic dataset *Star Schema Benchmark* (SSB) [26]. 1) **Flights** has a single table with 12 attributes and contains flight delay information. We use the data generator from [9] to scale up this dataset, by default to 500 million records (43GB). 2) **SSB** is a multi-table (5 tables) dataset to demonstrate the performance of complex queries with join clauses. By default, the fact table (lineorder) in SSB has 300 million records (33GB).

4.1.3 *Workloads.*

We use two workloads in experiments. 1) *Training workload*: we generate 100,000 queries from the visual request template to train the neural network in BlinkViz. 2) *Test workload*: we use ten typical example queries F1-F10 on Flights. For SSB dataset, we use the standard SSB queries S1.1-S4.3 to evaluate the performance. Both the workloads cover various aggregation functions, GROUP BY clauses, and selectivities.

4.1.4 *Baselines.*

In the experiments, we mainly compare BlinkViz with two sampling-based methods, which are often used in the existing sampling-based approximate visualization systems. All samples are stored in memory.

- *Random Sampling-based Method*: a method that executes queries on a random sample.
- *Stratified Sampling-based Method*: a method that executes queries on a set of stratified samples. For Flights, we generate two stratified samples based on the attribute combinations $(year\_date, unique\_carrier)$ and $(origin\_state\_abr, dest)$, respectively. For SSB, we generate two stratified samples based on the attribute *lo_orderdate* and *lo_suppkey*, respectively.

4.1.5 *Performance Metrics.*

We use three metrics to evaluate the performance.

- *Latency*: the response time of visual request (query), i.e., the time cost for getting the query result.
- *Accuracy*: the relative error $RE(q) = \frac{|\theta - \hat{\theta}|}{\theta}$ is used to evaluate the accuracy of a specific query q, where θ is the exact result from the original data and $\hat{\theta}$ is the approximate result based on samples or the learned model. For group-by queries, the relative error is the average error upon each group. Specifically, we set the relative error upon a group to 100% when the group is missed in the result.

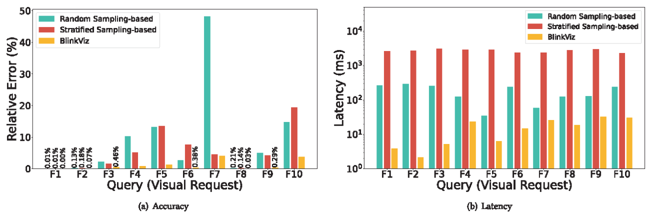


Figure 6: Performance on Flights dataset

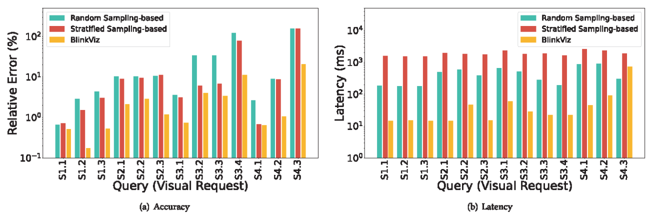


Figure 7: Performance on SSB dataset

- *Memory Footprint Size*: the size of memory used to store samples for sampling-based methods or the model for BlinkViz.

In the following experiments, the latency and accuracy is the average result of 10 separate experiments with the same experimental settings. By default, the sampling rate in the sampling-based methods is set at 1%.

4.2 Experimental Results

4.2.1 Performance on Flights and SSB datasets. In this set of experiments, we compare the performance of *random sampling-based* method, *stratified sampling-based* method, and BlinkViz on both Flights and SSB datasets in the default data size.

Figure 6 shows the performance of the three methods on the Flights dataset with 500 million records. As shown in Figure 6(a), the relative error of BlinkViz is lower than that of sampling-based methods. This is because BlinkViz learns the data distribution not only for the large subpopulations by using MSPN but also for the rare subpopulations by using the neural network. Specifically, for query F7, the relative error of the random sampling-based method is significantly higher than that of the stratified sampling-based method and BlinkViz, since the random sample cannot support query F7 very well. As shown in Figure 6(b), the latency of BlinkViz is significantly lower than that of sampling-based methods. All typical example queries can be evaluated within tens of milliseconds. This is because BlinkViz leverages model inference to answer queries instead of querying on the data or samples. The latency of the stratified sampling-based method exceeds 1000ms, since it needs extra computations to rewrite the approximate results.

Figure 7 shows the performance of the three methods on the SSB dataset with 300 million records. As shown in Figure 7(a), the relative error of BlinkViz is also lower than that of sampling-based methods. For queries S3.4 and S4.3, the relative error of sampling-based methods is significantly higher than that of BlinkViz, since the

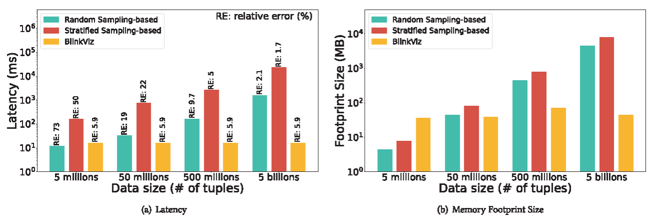


Figure 8: Scalability on latency and memory footprint size with different data sizes on Flights dataset

samples cannot provide enough tuples to answer the queries which have low selectivity. As shown in Figure 7(b), overall, the latency of BlinkViz is lower than that of sampling-based methods. For S4.3, the latency of BlinkViz is a little bit higher, since the number of groups in S4.3 is much more than that in other queries. The more groups in a query, the higher the time overhead required by MSPNs in BlinkViz. In summary, compared with the two sampling-based methods, BlinkViz can return results faster (responding in subseconds) while keeping a lower error rate.

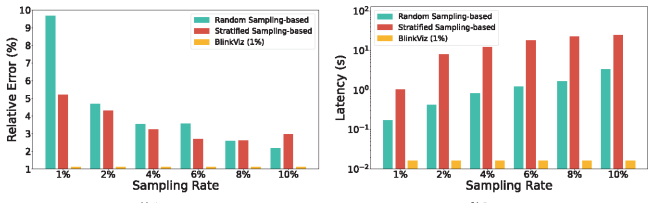
Table 1 shows the training cost of BlinkViz on the default size of Flights and SSB datasets. For both datasets, we find that the training time of MSPNs is very short compared to that of the neural network, since the training time of MSPNs relies on the number of columns in a dataset rather than the size of training workload. Therefore, when the data is updated, we can retrain the MSPNs in BlinkViz online and periodically retrain the neural network offline. Besides, the training time of MSPNs on SSB is longer than that on Flights because the MSPNs on SSB are trained on a 5-table join, which has more columns than Flights. The same reason causes the size of MSPNs on SSB to be larger than that on Flights. Furthermore, the size of the neural network on Flights and SSB is similar since the number of parameters in the neural network is fixed.

4.2.2 Scalability Evaluations. We scale up the data size of the Flights dataset from 5 million to 5 billion records and conduct experiments to evaluate the scalability of BlinkViz and sampling-based methods. As shown in Figure 8, BlinkViz has good scalability on both latency and memory footprint size. In contrast, the latency and memory footprint of sampling-based methods increase linearly along with the increase in data size. As shown in Figure 8(a), although when the data size is small (e.g., 5 million), the latency of the random sampling-based method is lower than that of BlinkViz, its relative error is very high. This is because a small size of the sample upon a low sampling rate (1% in this setting) cannot guarantee an acceptable accuracy. Even if we increase the sampling rate to 10%, the relative error of the sampling-based methods (Stratified/Random: 2.1%/2.2%) is still higher than that of BlinkViz (1.1%), and the delay exceeds that of BlinkViz at the same time. In other words, there is a trade-off between latency and accuracy for sampling-based methods. As for BlinkViz, its latency is mainly the model inference overhead, which is only related to the number of nodes in the MSPNs and the number of parameters in the neural network.

As shown in Figure 8(b), the memory footprint size of the sampling-based methods increases as the data size increases. The footprint size of the stratified method is larger than that of the random

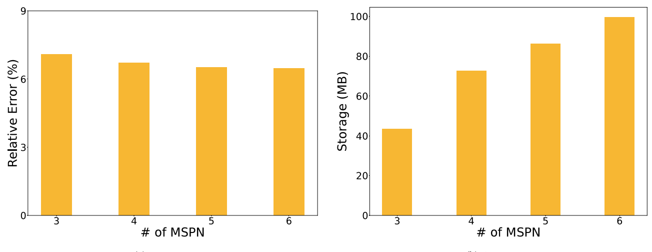
Table 1: Training cost of BlinkViz on Flights and SSB datasets

Dataset	# of records	Data Size	Training Time			Model Size		
			MSPNs	Neural Network	Total	MSPNs	Neural Network	Total
Flights	500 millions	43GB	128s	3h48min	3h50min	40.5MB	29.9MB	70.4MB
SSB	300 millions	33GB	578s	5h7min	5h17min	229.9MB	30MB	259.9MB



(a) Accuracy (b) Latency

Figure 9: Performance with different sampling ratios on Flights dataset



(a) Accuracy (b) Storage

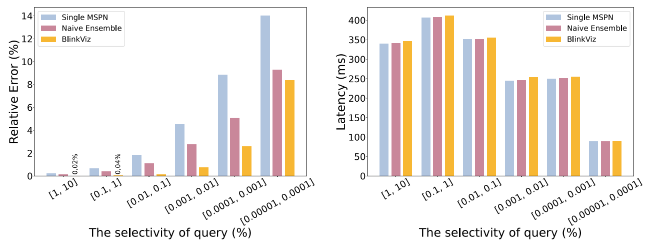
Figure 10: Effects of number of MSPN used in BlinkViz

method since it has two stratified samples by default. The memory footprint of BlinkViz keeps scalable because the size of MSPN is bounded by the number of columns in a dataset and the sample size [14] instead of the original data size. Besides, the neural network size in BlinkViz is stable since the number of parameters in the neural network is fixed. The scalability characteristics of BlinkViz make it particularly suitable for big data application scenarios.

4.2.3 Effects of Sampling Rate on Sampling-based Methods. We evaluate the influence of sampling rate on the performance of sampling-based methods, varying the sampling rate from 1% to 10% on Flights dataset. Specifically, the sampling rate of BlinkViz is fixed at 1%.

As shown in Figure 9(a), when the sampling rate increases, the average accuracy of the sampling-based methods grows as well but always worse than BlinkViz (1%). Meanwhile, in Figure 9(b), there is a sharp rise in the query latency of sampling-based methods as the sampling rate increases, which is unacceptable to users' interactive analytics. The results indicate that even if a larger sampling rate would certainly improve the accuracy of the sampling-based methods, BlinkViz with 1% sampling rate performs better than them, presenting unique advantages in both aspects.

4.2.4 Effects of Number of MSPN Used in BlinkViz. In this experiment, we evaluate the effects of the number of MSPN on the query accuracy and storage overhead on the Flights dataset. As shown in



(a) Accuracy (b) Latency

Figure 11: Effectiveness of neural network module

Figure 10(a), consistent with our intuition, the query accuracy of BlinkViz is improved with the increase of the number of MSPN. This is because additional MSPNs can provide more comprehensive data distribution. At the meantime, in Figure 10(b), the model size gets larger when adding additional MSPNs. However, even if combining six MSPNs together, the total model size does not exceed 100M, which takes up little storage resources compared with samples and is very convenient for transmission between the server and the client for offline use. Thus, users can trade off storage and accuracy by adjusting the number of MSPNs based on their own budget.

4.2.5 Ablations Studies. We conduct a set of ablation experiments to evaluate the effectiveness of the structure of BlinkViz, comparing the performance on average query accuracy, latency and training cost. We vary the selectivity of queries from 0.00001% to 10%, generate 1,000 queries for each selectivity interval and conduct experiments on the Flights dataset. When generating these queries, we calculate the data distributions under different group-by attribute combinations and leverage them to generate queries with different selectivity by changing the value ranges of where conditions.

We start from demonstrating the utility of using multiple MSPNs and neural network module, comparing BlinkViz with single MSPN model and naive ensemble of MSPNs (the arithmetic average of multiple MSPNs' predictions). Figure 11(a) shows the average accuracy of the queries with various selectivities. We can find that the relative errors of single MSPN, naive ensemble of MSPNs and BlinkViz all increase along with the decrease in selectivity. Since adopting multiple MSPNs is able to capture a more comprehensive picture of the dataset, the naive ensemble approach achieves higher accuracy than single MSPN. And the average accuracy of BlinkViz is higher than that of naive ensemble, because the errors with large deviation of the original single models still lead to weighty errors after averaging while the non-linear combination of multiple predictions through neural network can redress the balance. For latency, as shown in Figure 11(b), BlinkViz has a similar average latency to other methods. This means that the neural network module adds just a little bit of overhead by leveraging GPU acceleration.

Table 2: The effect of residual structure

Residual	Training Time	Relative Error	Storage
Yes	3h48min	7.03%	43.37MB
No	12h35min	8.77%	43.37MB

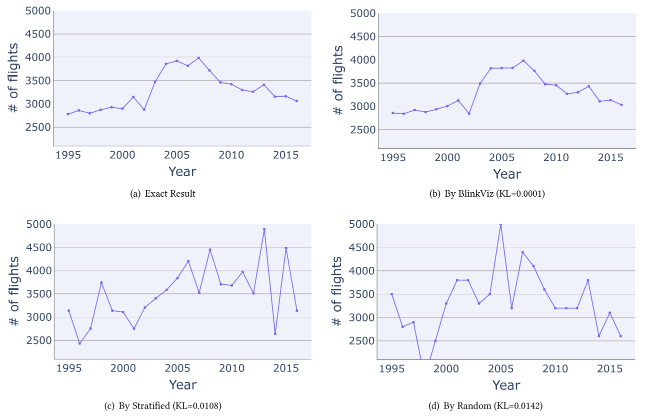


Figure 12: *Case 1* - # of flights departing from Los Angeles to JFK Airport each year

Meanwhile, we conduct experiments on neural network with and without residual modules to study the effects of residual structure in our model on the same dataset. As illustrated in Table 2, without the residual modules, the training time is three times longer than using residual structures, indicating that residual blocks speed up the convergence of the model. This is mainly because that the shortcut connection of residual structure reduces the loss of information, achieving high accuracy of predictions.

4.3 Demo Cases

To evaluate the effectiveness of BlinkViz from a visual perspective, we use two cases to demonstrate the visualizations under different methods. And we use the KL divergence [13] to quantify the difference between the exact visualization and approximate visualizations produced by different methods for a visual request.

Case 1. As shown in Figure 12, the visualization produced by BlinkViz (Figure 12(b)) is quite similar to the exact result (Figure 12(a)), and the KL divergence between them is very small, since BlinkViz has high accuracy in answering approximate visual requests. In contrast, the visualizations produced by the sampling-based methods (Figure 12(c) and Figure 12(d)) look very different from the exact result and the KL divergence between them is large.

Case 2. As shown in Figure 13, from a visual perspective, the approximate visualizations produced by the three methods look similar to the exact result (Figure 13(a)). But if we look closely, we can still observe the differences (noted by red circles). Overall, BlinkViz still performs best with only one obvious different point. However, as shown in Figure 13(c) and Figure 13(d), the approximate visualizations produced by the sampling-based methods have more obvious differences from the exact result than that of BlinkViz.

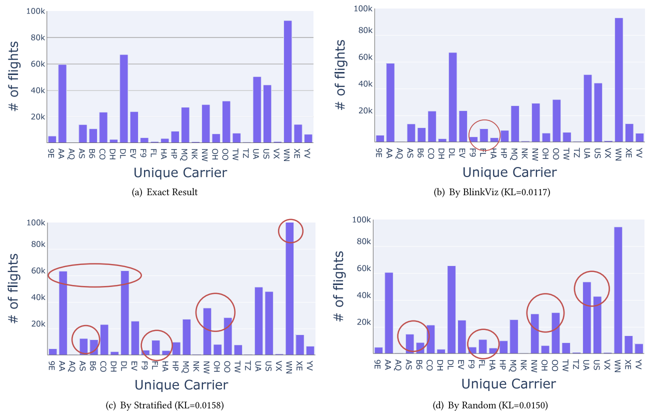


Figure 13: *Case 2* - # of flights whose *airtime* >1000 minutes and *departure delay* >1500 minutes.

5 RELATED WORK

Approximate Visualization. For faster response to visualization requests, existing visualization systems [4, 10, 12, 17, 27, 29, 30, 38] use samples instead of the original data to produce approximate visualizations. They make a trade-off between the time cost and accuracy of visualizations. For example, IFocus [17] progressively samples until the relative heights of the bars in the charts approach the actual size with a high probability. When the volume of data is large, the time overhead and memory footprint size of query processing in these sampling-based visualization systems will inevitably be large. Obviously, existing sampling-based approximate visualization methods cannot scale well enough for the data size.

Machine Learning-based Visualization. In order to achieve more effective visualizations, a series of studies apply machine learning techniques to solve visualization-related problems and present many opportunities [35]. The visualization pipeline could be divided into three parts: data processing, visualization display, and human interaction. In the stage of data processing, [36] proposes a perception-based supervised method to reduce high-dimensional data to 2D projections. In the visualization display stage, there are many studies using ML techniques to recommend visualizations, such as DeepEye [22], VizML [15], Data2Vis [7], etc. Finally, in the human interaction stage, machine learning-based methods are also emerging to predict the user's next exploration behavior [5, 25] and automatically generate insights [6, 16]. All these works are orthogonal to the learning-based approach proposed in this paper.

6 CONCLUSION

This paper proposes BlinkViz, a learning-based approximate visualization approach using neural-enhanced MSPNs. Compared to traditional sampling-based methods, BlinkViz is fast, scalable, and lightweight. It can respond in sub-seconds, even on a large dataset with hundreds of millions of data records, while keeping a low error rate. BlinkViz can effectively improve the user experience of interactive visual analytics in the big data era.

ACKNOWLEDGMENTS

We would like to thank our anonymous reviewers for their insightful feedback and comments. This work was supported by Shanghai Key Laboratory of Data Science and funded by National Natural Science Foundation of China No. 62072113 and No. 62272106.

REFERENCES

[1] 2013. Apache Superset. https://superset.apache.org/.
[2] 2013. Tableau Online. https://www.tableau.com/products/cloud-bi.
[3] 2020. Flights Dataset. https://github.com/IDEBench/IDEBench-public/blob/master/data/flights.zip. Accessed: 2021-12-06.
[4] Daniel Alabi and Eugene Wu. 2016. Pfunk-h: Approximate query processing using perceptual models. In *Proceedings of the workshop on human-in-the-loop data analytics*. 1–6.
[5] Eli T Brown, Alvitta Ottley, Helen Zhao, Quan Lin, Richard Souvenir, Alex Endert, and Remco Chang. 2014. Finding waldo: Learning about users from their interactions. *IEEE Transactions on visualization and computer graphics* 20, 12 (2014), 1663–1672.
[6] Zhutian Chen, Yun Wang, Qianwen Wang, Yong Wang, and Huamin Qu. 2019. Towards automated infographic design: Deep learning-based auto-extraction of extensible timeline. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 917–926.
[7] Victor Dibia and Çağatay Demiralp. 2019. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications* 39, 5 (2019), 33–46.
[8] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek R. Narasayya, and Surajit Chaudhuri. 2019. Selectivity Estimation for Range Predicates using Lightweight Models. *Proc. VLDB Endow.* 12, 9 (2019), 1044–1057. https://doi.org/10.14778/3329772.3329780
[9] Philipp Eichmann, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2020. Idebench: A benchmark for interactive data exploration. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1555–1569.
[10] Danyel Fisher, Igor Popov, Steven Drucker, and MC Schraefel. 2012. Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1673–1682.
[11] Robert Gens and Domingos Pedro. 2013. Learning the structure of sum-product networks. In *International conference on machine learning*. PMLR, 873–880.
[12] Tao Guo, Kaiyu Feng, Gao Cong, and Zhifeng Bao. 2018. Efficient Selection of Geospatial Data on Maps for Interactive and Visualized Exploration. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 567–582. https://doi.org/10.1145/3183713.3183738
[13] Jeffrey Heer and Michael Bostock. 2010. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 203–212.
[14] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2019. DeepDB: Learn from Data, not from Queries! *Proceedings of the VLDB Endowment* 13, 7 (2019).
[15] Kevin Hu, Michiel A Bakker, Stephen Li, Tim Kraska, and César Hidalgo. 2019. Vizml: A machine learning approach to visualization recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.
[16] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5648–5656.
[17] Albert Kim, Eric Blais, Aditya Parameswaran, Piotr Indyk, Sam Madden, and Ronitt Rubinfeld. 2015. Rapid sampling for visualizations with ordering guarantees. In *Proceedings of the vldb endowment international conference on very large data bases*, Vol. 8. 521.
[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980
[19] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf
[20] Zhicheng Liu and Jeffrey Heer. 2014. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2122–2131.
[21] David Lopez-Paz, Philipp Hennig, and Bernhard Schölkopf. 2013. The randomized dependence coefficient. *Advances in neural information processing systems* 26 (2013).
[22] Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. 2018. Deepeye: Towards automatic data visualization. In *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 101–112.
[23] Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. 2018. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[24] Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. 2019. SPFlow: An easy and extensible library for deep probabilistic learning using sum-product networks. *arXiv preprint arXiv:1901.03704* (2019).
[25] Alvitta Ottley, Roman Garnett, and Ran Wan. 2019. Follow the clicks: Learning and anticipating mouse interactions during exploratory data analysis. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 41–52.
[26] Patrick O'Neil, Elizabeth O'Neil, Xuedong Chen, and Stephen Revilak. 2009. The star schema benchmark and augmented fact table indexing. In *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 237–252.
[27] Yongjoo Park, Michael Cafarella, and Barzan Mozafari. 2016. Visualization-aware sampling for very large databases. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 755–766.
[28] Hoifung Poon and Pedro Domingos. 2011. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 689–690.
[29] Sajjadur Rahman, Maryam Aliakbarpour, Ha Kyung Kong, Eric Blais, Karrie Karahalios, Aditya Parameswaran, and Ronitt Rubinfield. 2017. I've seen" enough" incrementally improving visualizations to support rapid decision making. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1262–1273.
[30] Anish Das Sarma, Hongrae Lee, Hector Gonzalez, Jayant Madhavan, and Alon Y. Halevy. 2012. Efficient spatial sampling of large geographical tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 193–204. https://doi.org/10.1145/2213836.2213859
[31] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 341–350.
[32] Ji Sun and Guoliang Li. 2019. An End-to-End Learning-based Cost Estimator. *Proc. VLDB Endow.* 13, 3 (2019), 307–319. https://doi.org/10.14778/3368289.3368296
[33] Saravanan Thirumuruganathan, Shohedul Hasan, Nick Koudas, and Gautam Das. 2020. Approximate query processing for data exploration using deep generative models. In *2020 IEEE 36th international conference on data engineering (ICDE)*. IEEE, 1309–1320.
[34] Jiayi Wang, Chengliang Chai, Jiabin Liu, and Guoliang Li. 2021. FACE: a normalizing flow based cardinality estimator. *Proceedings of the VLDB Endowment* 15, 1 (2021), 72–84.
[35] Qianwen Wang, Zhutian Chen, Yong Wang, and Huamin Qu. 2021. A Survey on ML4VIS: Applying MachineLearning Advances to Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* (2021).
[36] Yunhai Wang, Kang Feng, Xiaowei Chu, Jian Zhang, Chi-Wing Fu, Michael Sedlmair, Xiaohui Yu, and Baoquan Chen. 2017. A perception-driven approach to supervised dimensionality reduction for visualization. *IEEE transactions on visualization and computer graphics* 24, 5 (2017), 1828–1840.
[37] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep unsupervised cardinality estimation. *Proceedings of the VLDB Endowment* 13, 3 (2019), 279–292.
[38] Jia Yu and Mohamed Sarwat. 2020. Turbocharging Geospatial Visualization Dashboards via a Materialized Sampling Cube Approach. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1165–1176. https://doi.org/10.1109/ICDE48307.2020.00105