DeepSPACE: Approximate Geospatial Query Processing with Deep Learning

Dimitri Vorona

Technical University of Munich
vorona@in.tum.de

Thomas Neumann

Technical University of Munich
neumann@in.tum.de

Andreas Kipf

Technical University of Munich
kipf@in.tum.de

Alfons Kemper

Technical University of Munich
kemper@in.tum.de

ABSTRACT

The amount of available geospatial data grows at an ever faster pace. This leads to a constantly increasing demand for processing power and storage in order to provide data analysis in a timely manner. At the same time, a lot of geospatial processing is visual and exploratory in nature, thus having bounded precision requirements. We present DeepSPACE, a deep learning-based approximate geospatial query processing engine which combines modest hardware requirements with the ability to answer flexible aggregation queries while keeping the required state to a few hundred KiBs.

CCS CONCEPTS

• **Information systems** → **Database query processing**; • **Mathematics of computing** → *Maximum likelihood estimation; Density estimation.*

KEYWORDS

geospatial, deep learning, approximate query processing.

ACM Reference Format:

Dimitri Vorona, Andreas Kipf, Thomas Neumann, and Alfons Kemper. 2019. DeepSPACE: Approximate Geospatial Query Processing with Deep Learning. In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19)*, November 5–8, 2019, Chicago, IL, USA. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3347146.3359112>

1 INTRODUCTION

With the growing amount of available geospatial data new requirements for query processing emerge [3]. The highly variable nature of the information presented in the data requires human supervision to discern interesting patterns. This leads to a processing pattern characterized by a high number of ad-hoc queries which ideally should be answered quickly and cheaply, enabling an interactive mode of exploration.

There are multiple ways to meet this challenge. Specialized geospatial indexes can trade high-end hardware requirements for high throughput of millions of data points per second [7]. Other

systems recognize the trade-off between the available hardware and required precision and provide approximate query results based on (online or offline) samples. Such approaches often deliver poor results under very selective queries since the sample size needs to grow inversely with the selectivity to provide the same precision.

In this work we present an alternative approach: **DeepSPACE** (**Deep Geo-Spatial Autoregressive Conditional Estimator**), a compact model which captures the distribution of the data and allows the user to discern interesting patterns while keeping the computation requirements to a level provided even by leanest of today's client devices like smartphones.

We present an unsupervised versatile training regime which works on any kind of geospatial data enriched with additional information about the data points. Using the example of New York City taxi data [11] we show that our model can answer typical exploratory queries with reasonable precision, enabling visual data exploration without the need for fast hardware or connectivity to a backend server.

We base our model on the recent advances in neural distribution estimation [4]. A special kind of autoregressive neural networks makes it possible to calculate arbitrary conditional distributions effectively providing a query interface for the underlying data.

We use space-filling curves to discretize geographic locations to facilitate the learning and the querying of the data distributions. In effect, we partition the space using a quad-tree-like structure which allows the model to capture the spatial locality relationship of the conditional distributions.

This paper makes the following contributions: (1) we introduce a deep-learning-based approach to approximately answer geospatial aggregation queries which enables interactive data exploration, (2) we show how our architecture can handle heterogeneous queries and data and allows user to extract rich information about the underlying data distributions, and (3) we compare our approach to sample-based baselines and show that we are competitive in the precision of query results even at high sample sizes.

Note: This work extends a 4-page paper that will be published in ACM SIGSPATIAL 2019.

2 RELATED WORK

Use of approximate query processing in domains that have relaxed requirements to the precision of the results (e.g., visualization) is a popular topic in database research. The aim is to provide the answer for arbitrary queries in lower (ideally bounded) time. A recent overview of the modern development in this area is presented in [9].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGSPATIAL '19, November 5–8, 2019, Chicago, IL, USA

© 2019 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-6909-1/19/11.

<https://doi.org/10.1145/3347146.3359112>

The authors identify two general approaches to approximate query processing (AQP): online sampling and offline synopsis generation.

Some systems opt for pre-computing a succinct synopsis of the data which can be used to efficiently approximate query results. Wang et al. [16] present an index structure which is enriched with a random sample of points that qualify the filter conditions of the previous queries. Such index can subsequently be used to answer approximate queries efficiently.

There has been a recent surge in the number of model-based methods for approximate query processing. Kulessa et al. [8] use a sum-product-network-based model to estimate aggregate query results. The authors explore both a direct conditional evaluation, similar to our approach, as well as generation of stratified samples. Similarly, Thirumuruganathan et al. [13] use variational autoencoders to generate samples from the learned joint data distribution. Yang et al. [17] propose the use of an autoregressive model to generate stratified samples and also support range predicates but do not address geospatial predicates.

Another direction is to trade precision for evaluation performance while using the total dataset and utilizing high-end hardware. Kipf et al. [7] use hierarchical grid approximations of polygons to achieve a throughput of tens of millions data points on a single core. Zacharitou et al. [14] propose to leverage the rendering pipeline of the GPU to provide interactive response times over large datasets. In contrast to these approaches, ours does not strictly require expensive hardware and has only a modest memory footprint.

3 BACKGROUND

We base our approach on a number of advances in the areas of geospatial location encoding and distribution modeling. In the following, we provide the necessary background to understand the DeepSPACE architecture.

We utilize space-filling curves to discretize the locations. The idea is to project the geographical surface on a 2d Euclidean surface, split it into nested cells and enumerate the cells according to their position on the space-filling curve. We opt for the Hilbert curve, which yields a nested cell structure similar to a quad-tree which achieves the best locality preservation. Our implementation is based on Google's S2 library [12].

The hierarchical structure allows a natural interpretation as a conditional distribution: each subsequent level can be seen as a categorical distribution conditioned on the joint distribution of the preceding cells. This allows the model to exploit the locality properties of the data and improves training.

Autoencoders [5] are a type of unsupervised deep learning architecture which aims to learn an efficient encoding for the given data. Autoencoders have many uses, one particular is neural density estimation.

The output of a standard autoencoder does not represent a valid probability distribution. This can be alleviated by enforcing the *autoregressive property* [4]. The autoregressive property holds if each output $\hat{x}_i$ only depends on the inputs $x_{0:i-1}$. By enforcing this property we get a valid probability distribution over the input. The model parameters can then be trained using maximum likelihood estimation.

4 DEEPSPACE

In the following, we introduce the DeepSPACE architecture which is able to utilize succinct pre-computed synopses to approximately answer a variety of geospatial queries. DeepSPACE supports various data types, and can answer common aggregation queries with selection predicates typical for the geospatial domain.

4.1 Model Architecture

DeepSPACE utilizes the autoregressive architecture to model a series of conditional distributions which describe the data. In particular, we build on the ideas of masked autoencoders [4] to provide an efficient implementation for training and inference.

We extend the autoregressive autoencoder with the notion of *sub-net modules*. A *sub-net module* ϕ_i can be considered a function which takes a subset of input attributes $x_{0:i-1}$ and produces the parameters of the target output distribution $\hat{x}_i$. The input attribute x_i is a contiguous set of neurons which describe the input of a data type. A sub-net module ϕ_i is fit using maximum likelihood estimation on the input attribute x_i .

Figure 1: DeepSPACE’s autoregressive architecture. The input layer neurons contain encoded data values, the output layer neurons produce the parameters of the output corresponding distributions. The hidden layer neurons are given order numbers which are used to decide which inputs and outputs are connected to them. The lines denote connections to the individual neurons.

DeepSPACE embeds the sub-net modules in a single neural network as pictured in Figure 1. Input and output neurons of a subnet are labeled with the subnet’s index. Special inputs and outputs are given placeholder values which are smaller or larger than any neuron index which forces a full or prevents any connection to the neighboring layer, respectively. The attributes which are not used as input are not connected to any neurons, the corresponding output distributions, on the other hand, are connected to every output except itself. The architecture can be trivially extended to multiple hidden layers by inserting additional hidden layers and assigning indexes to the neurons as described above.

The presented architecture imposes an ordering on the input attributes. This means that ϕ_i cannot be evaluated independently of

the input attributes j for $j < i$. While, due to the product rule, any ordering of the input attributes returns the same result, we must be able to “skip” certain attributes if we are not interested in an output distribution conditional on them. To achieve this DeepSPACE utilizes the training techniques from [15] to create an *order agnostic* representation of data. We organize the connections from the input layer and to the output layer depending on which attributes are used as conditional inputs and which are used as outputs. Concretely, we set the indexes of the “skipped” input neurons and the output-only neurons to a placeholder value higher than any other neuron index, thus enforcing the desired connectivity pattern (cf. Figure 1). To facilitate model training, we provide an additional input mask which denotes the attributes on which the filters are defined (cf. [15]).

4.2 Handling Heterogeneous Data

DeepSPACE can handle arbitrary data distributions by encoding their parameters in the corresponding input and output blocks. The key insight is that different data types correspond to different target distributions which can be approximated using maximum likelihood parameter fitting.

The categorical data corresponds to the categorical distribution. Its parameters are logits of the possible classes. The distribution parameters are found using categorical cross-entropy loss minimization.

Since we discretize the geospatial data, we handle it as a custom case of hierarchical categorical data. Each quad-tree layer is a 4-ary categorical variable, with subsequent layers conditionally dependent on the immediately preceding layer. This allows the network to exploit the locality of data.

The continuous attributes can be represented by any continuous distribution though Gaussian distribution or a mixture thereof often fits well enough. The fitting is done using log-likelihood maximization on the parameters. It also can be useful to fit the Gaussians on logarithms of the attribute values.

Datetime can be either seen as a continuous attribute or a discrete attribute encoding a value extracted from the continuous timestamp. We opt for the second approach since it facilitates the processing of the typical geospatial queries.

4.3 Query Processing

Our model can answer a variety of conjunctive queries typical in a geospatial setting. A single selection operation of applying a predicate $Pred$ on an attribute X_i can be interpreted as imposing the evidence $P(x|x \in X_i \wedge Pred(x))$ on the joint prior $P(X)$, i.e., $P_{\sigma}(X) \sim P(x|x \in X_i \wedge Pred(x)) \times P(X)$. In other words, $P(x|x \in X_i \wedge Pred(x))$ is the proportion of the tuples which qualify for the predicate $Pred(x)$, $P(x)$ is the data distribution before applying the filter.

Using the product rule we can recursively generalize this to N selection operations:

$$P_{\sigma,N}(X) = \prod_{i=1}^N P_{\sigma,i-1}(x|x \in X_i \wedge Pred(x))$$

with $P_{\sigma,0} = 1$. The distribution $P_{\sigma,i}$ is exactly the conditional distributions learned by the sub-net module ϕ_i .

We support filter predicates on any attribute subset. The output distributions, which can be roughly thought of as aggregations on

columns in the SELECT clause of a SQL query, provide basis for the evaluation of the aggregations. For example, a sum can be thought of as the product of the number of the elements in the selected region and the mean of the fitted distribution.

Since geospatial data is a special case of categorical data, it is handled in a similar fashion. When a query defines a selection of an area described by a specific cell, we apply a selection sequentially on each level up to the targeted one (top down).

5 EVALUATION

To assess the performance of DeepSPACE, we evaluate it on a number of common tasks and compare with a sampling-based approach.

5.1 Dataset And Model Training

We choose the publicly available New York City (NYC) taxi rides dataset [2] for the evaluation. This dataset consists of approximately 350,000 taxi rides per day, containing the information about date, time, and location of the origins and destinations of the taxi rides.

We train a DeepSPACE model on the data from January 2016 which consists of around 10 million records. To demonstrate the versatility of our approach, we choose *day of month*, *day of week*, and *hour* as datetime, *pickup location* as geospatial, and *total fare* as continuous attributes. See Section 4.2 for details on encoding of the data types.

We train a DeepSPACE model with two hidden layers of 386 exponential linear units (ELU) [1] each. This corresponds to a model size of around 520 KiB. The model is trained using the Adam optimizer [6] with a learning rate of 10^{-4} on mini-batches of size 1024.

5.2 Geospatial Query Processing

We compare the model performance with a sampling-based approach with different sample sizes on the q-error metric [10]. The q-error [$q \geq 1$] is the factor between the estimated and the true cardinality (or vice versa).

As the workload we choose queries filtering on the data location using a predicate similar to *ST_Contains* which works on the single cells and additional date time predicates selecting a day of the week of the date. We expect this to model the natural exploratory query workload where the user is interested in a subset of information limited by the time and area.

Table 1 presents the results. The results are grouped based on query selectivity. The most selective group ($N < 100$) represented 64% of queries and had median selectivity of 1.19×10^{-6} . At this selectivity our model provides better performance than sampling 10% of the data or less. The queries with results between 100 and 1000 constituted 26% of the workload (median selectivity 2.50×10^{-5}). At this group our model loses slightly to sampling 10%. A similar picture presents itself in the least selective group ($N \geq 1000$, median selectivity 1.79×10^{-4}) in which 10% of the queries fell. Over the whole query workload, DeepSPACE offered the best performance with a median q-error of 1.25 compared to 1.28 of the sampling of 10% of the datasets (1.60 versus 2.46 in mean).

	$N < 100$			$100 \leq N < 1000$			$N \geq 1000$		
	$\emptyset$	50th	95th	$\emptyset$	50th	95th	$\emptyset$	50th	95th
DeepSPACE	1.83	1.42	3.95	1.21	1.14	1.61	1.10	1.07	1.28
sample 0.1%	23.8	13.0	81.0	228	171	718	256	1.53	1600
sample 1%	16.3	7.00	66.0	22.3	1.51	158	1.22	1.14	2.25
sample 10%	3.18	1.69	11.0	1.19	1.13	1.54	1.06	1.04	1.16

Table 1: Q-errors on the query workload for the COUNT(*) aggregate, predicates on geospatial location and the datetime columns. The result are grouped by the number of the entries N which qualify the predicates.

		$N < 100$			$100 \leq N < 1000$			$N \geq 1000$		
		$\emptyset$	50th	95th	$\emptyset$	50th	95th	$\emptyset$	50th	95th
DeepSPACE	<i>MEAN</i>	0.23	0.15	0.73	0.06	0.06	0.16	0.05	0.04	0.11
	<i>SUM</i>	0.46	0.35	1.30	0.17	0.13	0.46	0.10	0.07	0.26
sample 1%	<i>MEAN</i>	1.72	2.00	2.00	0.51	0.29	2.00	0.12	0.16	0.33
	<i>SUM</i>	1.81	2.00	2.00	0.69	0.50	2.00	0.21	0.16	0.57
sample 10%	<i>MEAN</i>	0.94	0.92	2.00	0.11	0.08	0.52	0.04	0.00	0.09
	<i>SUM</i>	1.10	0.91	2.0	0.19	0.15	0.52	0.06	0.05	0.18

Table 2: sMAPE metric for the estimation of the MEAN and SUM aggregates.

5.3 Querying Continuous Attributes

We additionally evaluate the performance on the continuous attributes using the aggregation functions **SUM** and **MEAN**, using the symmetric mean absolute percentage error (sMAPE), defined as

$$\sum 2 \times |S_{i,true} - S_{i,est}| / (S_{i,true} + S_{i,est})$$

where Q is the number of queries, $S_{i,true}$ and $S_{i,est}$ are true and estimated results of i -th query, respectively. We choose sMAPE because it is bounded ($[0, 2]$) and almost symmetric in respect to under- and over-estimations. For this experiment, we skip evaluation on the sample size of 10 thousand entries since the results were vastly inferior to other approaches.

The results are presented in Table 2. DeepSPACE provides overall best precision for the queries with up to 1000 qualifying entries. Only in the group with more than 1000 entries, the sampling of 10% of the data provides slightly better estimates.

6 SUMMARY AND FUTURE WORK

The constantly increasing size of the available geospatial data presents new challenges to query processing systems. We have presented DeepSPACE, a deep learning-based approximate query processing engine which provides the user with quick and precise estimates while requiring only a small state size of few hundred KiBs.

We have introduced a novel modular model architecture which is able to support the most common data types and can be easily extended to support arbitrary data distributions. Our evaluation on the NYC taxi dataset showed that DeepSPACE provides better precision on the query workloads typical for the geospatial domain than our sampling baseline.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 725286).

REFERENCES

- [1] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- [2] N. Y. N. Y. T. Commission and Limousine. New york city taxi trip data, 2009-2018, 2019.
- [3] A. Eldawy and M. F. Mokbel. The era of big spatial data. *PVLDB*, 10(12):1992–1995, 2017.
- [4] M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 881–889, 2015.
- [5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [6] D. P. Kingma and J. Ba. 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*, 2015.
- [7] A. Kipf, H. Lang, V. Pandey, R. A. Persa, P. A. Boncz, T. Neumann, and A. Kemper. Approximate geospatial joins with precision guarantees. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1360–1363, 2018.
- [8] M. Kullessa, A. Molina, C. Binnig, B. Hilprecht, and K. Kersting. Model-based approximate query processing. *CoRR*, abs/1811.06224, 2018.
- [9] K. Li and G. Li. Approximate query processing: What is new and where to go? - A survey on approximate query processing. *Data Science and Engineering*, 3(4):379–397, 2018.
- [10] G. Moerkotte, T. Neumann, and G. Steidl. Preventing bad plans by bounding the impact of cardinality estimation errors. *PVLDB*, 2(1):982–993, 2009.
- [11] *TLC Trip Record Data*. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.
- [12] Google S2 library. <http://s2geometry.io/>.
- [13] S. Thirumuruganathan, S. Hasan, N. Koudas, and G. Das. Approximate query processing using deep generative models. *CoRR*, abs/1903.10000, 2019.
- [14] E. Tzirita Zacharatos, H. Doraiswamy, A. Ailamaki, C. T. Silva, and J. Freire. GPU rasterization for real-time spatial aggregation over arbitrary polygons. *PVLDB*, 11(3):352–365, 2017.
- [15] B. Uria, I. Murray, and H. Larochelle. A deep and tractable density estimator. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 467–475, 2014.
- [16] L. Wang, R. Christensen, F. Li, and K. Yi. Spatial online sampling and aggregation. *Proc. VLDB Endow.*, 9(3):84–95, Nov. 2015.
- [17] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica. Selectivity estimation with deep likelihood models. *arXiv preprint arXiv:1905.04278*, 2019.