AIDA: A Spatial Data Augmentation Tool for Machine Learning Dataset Preparation*

Sara Migliorioni
Department of Computer Science
University of Verona
Verona, Italy
sara.migliorini@univr.it

Alberto Belussi
Department of Computer Science
University of Verona
Verona, Italy
alberto.belussi@univr.it

Abstract

The use of machine and deep learning techniques for dealing with spatial data is progressively increasing as the amount of such kind of information consistently grows. At the same time, the quality of the obtained results strictly depends on the quality of the training data. In regression and classification tasks, the balancing of the training set with respect to both the characteristics of the input data and the ground truth values is essential to correctly capture all the eventualities and cases in the right way. However, as already pointed out in the literature, producing balanced training sets is not simple, even when they are synthetically generated. This demonstration presents a tool for producing balanced training sets for spatial operation estimation, which starts from the synthetic generation of spatial datasets resembling real-world situations, with respect to distribution and other spatial characteristics, and then apply spatial queries for obtaining a first collection on which balancing analysis and spatial augmentation techniques are applied to obtain a final balanced collection with respect to specific metrics. This tool is a step towards the generation of good-quality training sets for different spatial query optimization and evaluation models.

CCS Concepts

• **Computing methodologies** → **Machine learning approaches**.

Keywords

Spatial data, augmentation, balancing, machine learning

ACM Reference Format:
Sara Migliorioni and Alberto Belussi. 2025. AIDA: A Spatial Data Augmentation Tool for Machine Learning Dataset Preparation. In *The 33rd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '25), November 3–6, 2025, Minneapolis, MN, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3748636.3762790

*This study was carried out within the Interconnected Nord-Est Innovation Ecosystem (iNEST) and received funding from the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS00000043). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

SIGSPATIAL '25, Minneapolis, MN, USA

ACM ISBN 979-8-4007-2086-4/2025/11
https://doi.org/10.1145/3748636.3762790

1 Introduction

Machine Learning (ML) and Deep Learning (DL) techniques are progressively exploited in the spatial domain as an increasing amount of spatial data is becoming available. Such models are typically used for classification or regression tasks, for instance, in the context of query optimization or cost evaluation [2]. However, the accuracy of the obtained results and the generalization capability of the models strictly depend on the dimension and quality of the used training and test sets. In this regard, the generation of synthetic datasets resembling the spatial characteristics of real-world data is a common approach, and some tools, like the SpiderWeb [4] one, have been proposed to perform such an operation. However, as discussed in [1], the balancing of the input collection in terms of spatial characteristics of the contained geometries is not enough to guarantee a balancing of the ground through values, especially when they refer to the results or performances of spatial operation computation. Moreover, the production of a good training set is a time-consuming activity requiring not only the generation of the input dataset collections, but also the execution of a great number of spatial operations on them for obtaining the desired ground truth values. For this reason, in [1], some metrics have been proposed to evaluate the balancing of both the input collections and the ground truth values, as well as some augmentation techniques able to increase the presence of instances in under-represented classes, without requiring the re-computation of the time-consuming spatial operations. The range query operation has been chosen as a proof-of-concept spatial operation, and three augmentation techniques are proposed and evaluated.

Starting from the methodology presented in [1], in this demonstration, we present AIDA, a spAtIal Data Augumentation tool which can be freely downloaded from a GitHub repository[1]. This tool aims to streamline the process of generating balanced collections of spatial datasets for training machine learning and deep learning models able to estimate the cost of typical spatial operations, like range queries and spatial joins [2]. This tool collects some recent theoretical results into a unified software that can be easily exploited by researchers in the production of their training sets. In the remainder, Sect. 2 describes in detail the generation and augmentation pipeline supported by the AIDA software, then in Sect. 3 some hints about the architectural solution and some screenshots of the graphical interface are presented. Finally, Sect. 4 summarizes the characteristics of the AIDA tool and proposes some future extensions.

[1] https://github.com/smigliorini/aida-augumentation-tool

2 Spatial Data Augmentation Pipeline

The general pipeline of the spatial augmentation technique is depicted in Fig. 1, it encompasses a set of steps, denoted as TX, each one producing an output OX, and taking as input some data provided by users, denoted by DX, or the output of a previous task.

The first task T1 consists in the generation of a collection of spatial datasets starting from the specification of a set of desired characteristics, namely: the kind of spatial distribution (i.e. uniform, diagonal, Gaussian, parcel, etc), the MBR of the entire dataset, the cardinality of the dataset, the kind of geometry (i.e., point, box, polygon, etc), the representation format (i.e., CSV, WKT), together with other parameters which are tailored for each specific distribution, like the buffer size for the diagonal or Gaussian distribution. These parameters can be specified directly through the user interface or uploaded through a CSV file for massive and batch data generation. For the spatial generation, we exploit the use of the SpiderWeb tool [4] through a set of HTTP calls. The generated datasets are placed into a directory specified by the user, and are identified by a unique name containing some relevant characteristics of the input, like the distribution, the ordinal, and so on.

The second task T2 takes the generated datasets and computes a spatial index on them by using the Beast library (Scalable Exploratory Analytics on Spatio-temporal Data) [3]. We refer to a very efficient spatial index, called R*-Grove [5], provided by this library. The computation of the spatial index essentially consists of subdividing a big dataset into a set of partitions, each one containing only the geometries intersecting a certain MBR, together with a master file index summarizing the set of generated partitions in terms of their MBR and the corresponding file name. In the original implementation, the number of generated partitions is computed to guarantee that each one will have a uniform and fixed dimension in bytes. In the provided tool, the computation of the index has been slightly modified to allow the user to specify the number, or alternatively the cardinality, of the desired partitions. This choice is particularly useful, since the index plays a key role not only for improving the spatial query computation T3, but also during the spatial augmentation T5. Indeed, AIDA allows one to specify desired partitions that are small in terms of byte size but are big enough in terms of the number of contained geometries.

The desired spatial query is then computed on the indexed spatial datasets T3, producing the various ground truth values of interest. With particular reference to the experiments performed in [1], the considered spatial operation is the range query, and the generated ground truth values are: the selectivity, the number of MBR tests, and the computation time. By combining the original spatial datasets O1 with the obtained ground truth values and the query parameters O3, the original input collection O4 is produced, on which the balancing analysis T4 is performed.

As described in [1], the computation of the balancing metrics on input and target variables consists of computing the estimation of the fractal dimension E_q, with $q = 0$ and $q = 2$, on each characteristic of interest. In particular, while E_0 describes whether the entire spectrum of possible values is represented, E_2 describes whether such a spectrum is represented in a uniform way. Alg. 1 details the metrics computation: for each dataset D_i, we compute its spatial distribution through the computation of the value E_2 on its geometries (line 4), and on its spatial extension which is captured by the average area of its geometries f_a (line 5), the average length on the x and y axis of its geometries f_{l_x} and f_{l_y}, respectively (lines 6-7). Given that measures for all datasets D_i in input collection $\mathcal{D}$, the overall balancing metrics for the input variables are computed (lines 9-16). Moreover, the E_q values are computed for each ground trough values (line 18-25). Finally, the overall balancing metrics O5 for both the input and the target variables are returned (line 26).

Algorithm 1: Balancing Analysis

1 **function** *BalancingMetrics*($C = \{D_i, O_i, T_i\}_{i=1}^{k}$) :
2 $\mathcal{D} \leftarrow \{D_1, \dots D_k\}$;
3 **forall** $D_i \in \mathcal{D}$ **do**
4 $E_2(D_i) \leftarrow computeE2(D_i)$;
5 $f_a(D_i) \leftarrow (\sum_{g \in D_i} area(g))/|D_i|$;
6 $f_{l_x}(D_i) \leftarrow (\sum_{g \in D_i} MBR(g).x_2 - MBR(g).x_1)/|D_i|$;
7 $f_{l_y}(D_i) \leftarrow (\sum_{g \in D_i} MBR(g).y_2 - MBR(g).y_1)/|D_i|$;
8 **end**
9 $B_0(\mathcal{D}, E_2()) \leftarrow computeE0(E_2(D_1), \dots E_2(D_k))$;
10 $B_2(\mathcal{D}, E_2()) \leftarrow computeE2(E_2(D_1), \dots E_2(D_k))$;
11 $B_0(\mathcal{D}, f_a()) \leftarrow computeE0(f_a(D_1), \dots f_a(D_k))$;
12 $B_2(\mathcal{D}, f_a()) \leftarrow computeE2(f_a(D_1), \dots f_a(D_k))$;
13 $B_0(\mathcal{D}, f_{l_x}()) \leftarrow computeE0(f_{l_x}(D_1), \dots f_{l_x}(D_k))$;
14 $B_2(\mathcal{D}, f_{l_x}()) \leftarrow computeE2(f_{l_x}(D_1), \dots f_{l_x}(D_k))$;
15 $B_0(\mathcal{D}, f_{l_y}()) \leftarrow computeE0(f_{l_y}(D_1), \dots f_{l_y}(D_k))$;
16 $B_2(\mathcal{D}, f_{l_y}()) \leftarrow computeE2(f_{l_y}(D_1), \dots f_{l_y}(D_k))$;
17 $n \leftarrow$ number of target variables;
18 **forall** $j \in \{1 \dots n\}$ **do**
19 $\mathcal{V}_j \leftarrow \emptyset$;
20 **forall** $T_i \in \mathcal{D}$ **do**
21 $\mathcal{V}_j \leftarrow \mathcal{V}_j \cup \{T_i.v_j\}$;
22 **end**
23 $B_0(\mathcal{V}_i) \leftarrow computeE0(\mathcal{V}_j)$;
24 $B_2(\mathcal{V}_i) \leftarrow computeE2(\mathcal{V}_j)$
25 **end**
26 **return**
$\langle B_0(\mathcal{D}, E_2()), B_2(\mathcal{D}, E_2()), B_0(\mathcal{D}, f_a()), B_2(\mathcal{D}, f_a()),$
27 $B_0(\mathcal{D}, f_{l_x}()), B_2(\mathcal{D}, f_{l_x}()), B_0(\mathcal{D}, f_{l_y}()), B_2(\mathcal{D}, f_{l_y}()),$
28 $B_0(\mathcal{V}_1, B_2(\mathcal{V}_1) \dots, B_0(\mathcal{V}_n, B_2(\mathcal{V}_n)\rangle$
29 **end**

In case the balancing metrics O5 reveal that the ground truth values are not uniformly represented, namely, the user can proceed with the partitioning T5 of the spectrum of possible values into discrete intervals of interest, so that it is possible to compute the number of instances belonging to each interval and identify which one has to be augmented. Such subdivision is decided by the user through the parameter *class intervals* D3, and could depend on the problem at hand. Alg. 2 summarizing the partitioning procedure T5: given a collection C, the interval $\mathcal{T}$ of the ground truth values, and the number of desired classes n, the algorithm identify the elements $\langle D_i, O_i, T_i\rangle \in C$ belonging to each partition class.

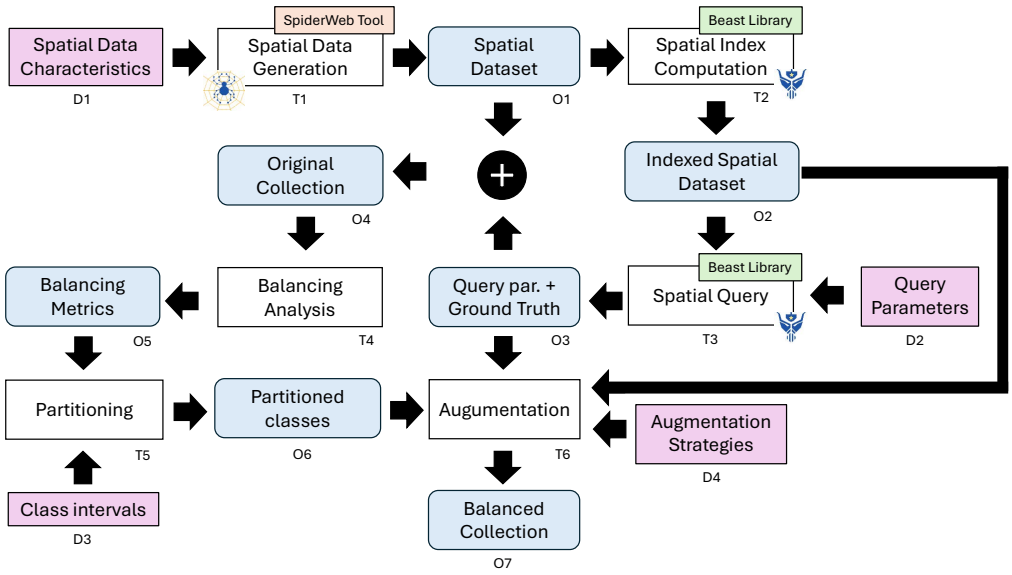

Figure 1: Pipeline of the spatial augmentation technique.

Algorithm 2: Spatial augmentation

```
1  function Partitioning(C = {⟨D_i, O_i, T_i⟩}_{i=1}^k}, v, T, n) :
2      δ ← (T.t_end − T.t_start)/n;
3      forall h ∈ {1, ... n} do
4          P_h ← ∅;
5      end
6      forall i ∈ {1, ... k} do
7          h ← ((T_i.v − T.t_start)/δ) + 1;
8          P_h ← P_h ∪ {D_i}
9      end
10 end
11 function Augumentation(P, A, θ) :
12     forall C_h ∈ P do
13         if |C_h| > θ then
14             perform an undersampling of C_h;
15         end
16         else if |C_h| < θ then
17             forall i ∈ {1, |C_h| − θ} do
18                 a ← chose an augmentation action in A;
19                 D ← chose a random dataset in C_h;
20                 C' ← a(D);
21                 C_h ← C_h ∪ {C'};
22             end
23         end
24     end
25 end
```

The partitioned classes O6 together with the augmentation strategies D4 identified by the user, are then passed to the augmentation step T6, which exploits the spatial index O2 previously collected to speed up the generation process. This procedure is detailed in Alg. 2: given partitioning $\mathcal{P}$, the set of possible augmentation operation $\mathcal{A}$, and a threshold number of representative for each class θ, the function *Augumentation* performs the augmentation of the underrepresented class, or the sampling of the overrepresented one. The application of the augmentation operation in line 20 produces the final balanced collection O7.

The produced balanced collection O7 could be applied in an ML/DL estimation pipeline, like the one in [2], which is based on the computation of the spatial embeddings, as exemplified in Fig. 2. The encoding of such spatial embeddings is based on the preliminary computation of the dataset histograms. For this reason, AIDA also supports the operation T6, while the computation of the spatial embedding T7 is left behind as a future extension.

3 AIDA Tool

The AIDA Tool has been designed as a RESTful application where a graphical user interface developed in React calls some backend functions in Python, relative to tasks T1, T4, T5, T6, and some in Scala, relative to the Beast functionalities, namely T2 and T3.

Fig. 3 illustrates the general structure of the application, where the top menu reports the operations composing the pipeline of the spatial augmentation and allows the user to follow the entire process. In the figure, the dataset generation process is depicted, where the various characteristics of each input dataset could be alternatively specified in a manual way through a table or in a batch mode through a CSV file. In particular, besides the set of common parameters, like the MBR of the dataset and cardinality, a set of configurations can be specified based on the chosen distribution.

At the end of each step, the results produced by the corresponding task are saved into a separate directory for being easily downloaded by the user or used as input for the following step. A preview function is also available to give the user some hints about the generated dataset, spatial index, or histogram computation.

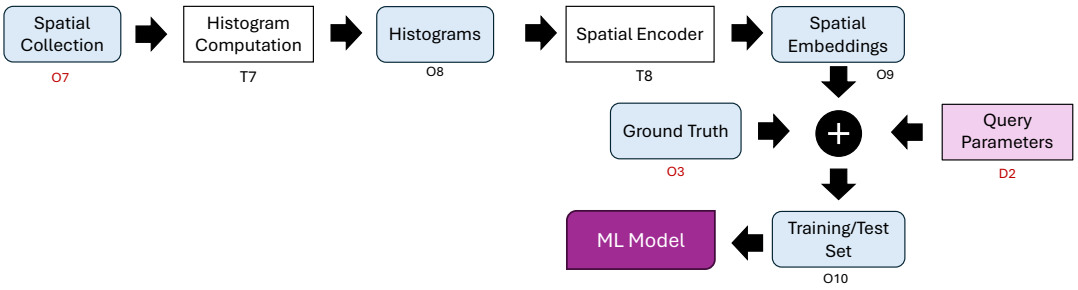

Figure 2: Final pipeline of the training and test set generation: labels in red referred to the pipeline in Fig. 1.

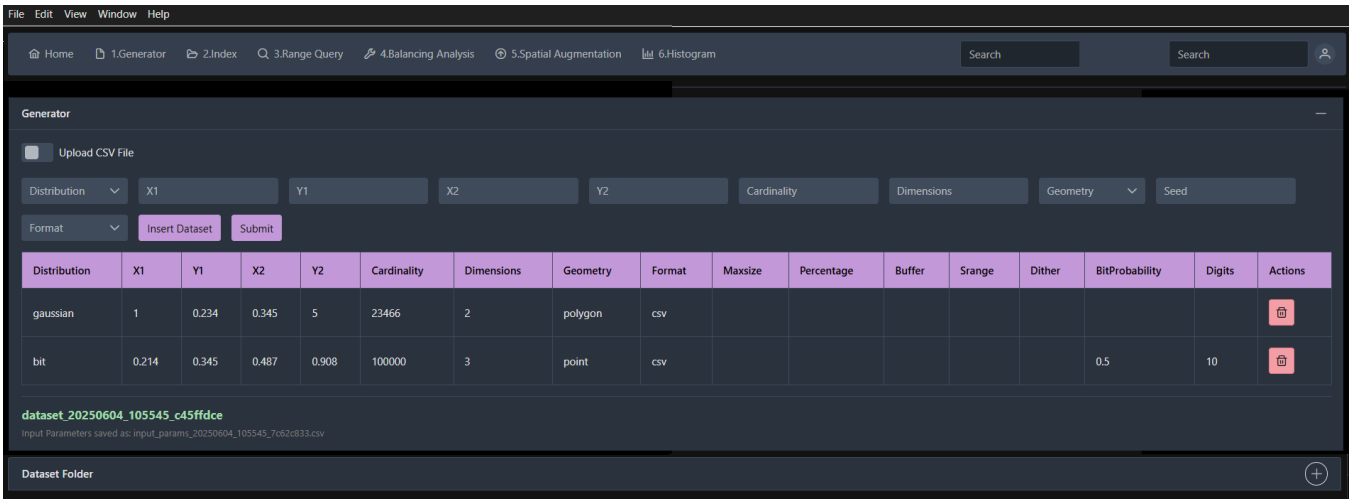

Figure 3: Section for the generation of synthetic spatial datasets.

The graphical interface has been developed in order to ensure its responsiveness while long-running computations are in progress, and specific checks have been implemented to prevent concurrent operations from degrading the data currently used by others.

The front-end and the back-end parts could be installed on a single machine, resembling a desktop application, or they can be subdivided into a server-side backend installed in a high-performance server and a web-based front-end application.

4 Conclusion

This paper demonstrates a tool, called AIDA, for the generation of synthetic balanced training and test sets for spatial machine learning or deep learning models. The tool collects some recent theoretical results published in the literature related to the augmentation of spatial datasets and the definition of balancing metrics. The main objective of this tool is to streamline the process typically used for producing synthetic training and test sets, resembling the characteristics of real-world data. As future work, we plan to extend the tool by introducing a plug-and-play mechanism for both the augmentation techniques and the production of the spatial embeddings. In this way, users could implement their own augmentation functionalities, or embedding construction, and invoke them through the AIDA tool.

Acknowledgments

We would like to thank the bachelor's degree students, Pietro Dudine, for developing the user interface, and Daniele Losco, for optimizing the first version of the augmentation algorithms.

References

[1] Alberto Belussi, Diego Garofolo, and Sara Migliorini. 2024. Augmentation Techniques for Balancing Spatial Datasets in Machine and Deep Learning Applications. In *Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems* (Atlanta, GA, USA) *(SIGSPATIAL '24)*. Association for Computing Machinery, 91–101. doi:10.1145/3678717.3691230

[2] Alberto Belussi, Sara Migliorini, and Ahmed Eldawy. 2024. A Generic Machine Learning Model for Spatial Query Optimization based on Spatial Embeddings. *ACM Trans. Spatial Algorithms Syst.* 10, 4, Article 36 (2024), 33 pages. doi:10.1145/3657633

[3] Ahmed Eldawy, Vagelis Hristidis, Saheli Ghosh, Majid Saeedan, Akil Sevim, A.B. Siddique, Samriddhi Singla, Ganesh Sivaram, Tin Vu, and Yaming Zhang. 2021. Beast: Scalable Exploratory Analytics on Spatio-temporal Data. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21)*. 3796–3807. doi:10.1145/3459637.3481897

[4] Puloma Katiyar, Tin Vu, Ahmed Eldawy, Sara Migliorini, and Alberto Belussi. 2020. SpiderWeb: A Spatial Data Generator on the Web. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20)*. 465–468. doi:10.1145/3397536.3422351

[5] Tin Vu and Ahmed Eldawy. 2020. R*-Grove: Balanced Spatial Partitioning for Large-Scale Datasets. *Frontiers in Big Data* 3 (2020). doi:10.3389/fdata.2020.00028