The Journal of Open Source Software

StreamGen: a Python framework for generating streams of labeled data

Laurenz A. Farthofer [1,2]

1 Kai - Kompetenzzentrum Automobil- und Industrieelektronik GmbH, Austria **2** Institute of Computer Graphics and Vision, Graz University of Technology, Austria

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

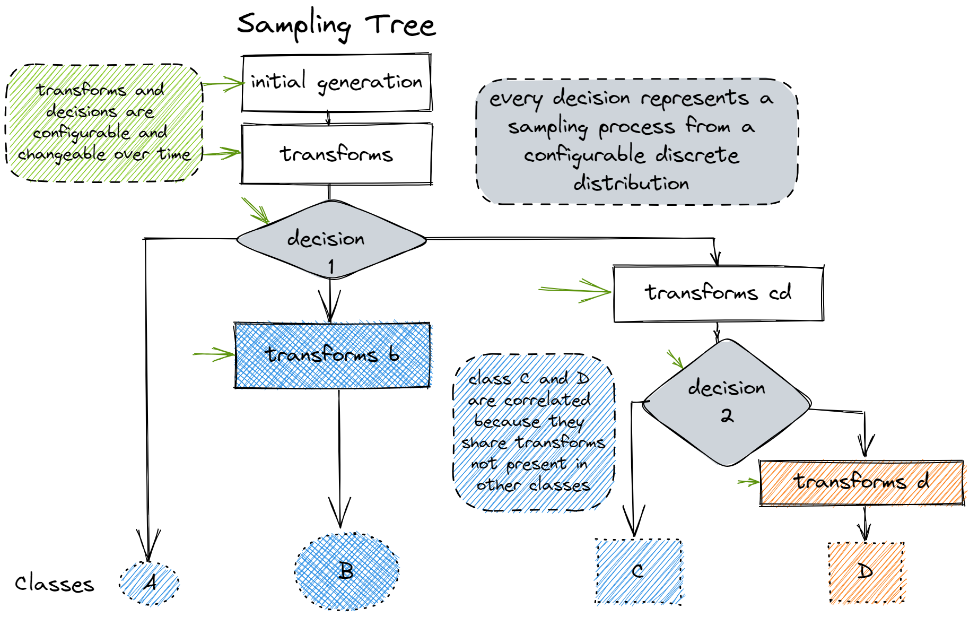
Sampling Tree
transforms and decisions are configurable and changeable over time
initial generation
transforms
every decision represents a sampling process from a configurable discrete distribution
decision 1
transforms cd
transforms b
class C and D are correlated because they share transforms not present in other classes
decision 2
transforms d
classes
A
B
C
D

Figure 1: A tree of sampling functions and transformations as a new data structure and framework for synthetic data generation. Samples are generated by traversing the tree from the root to the leaves. Each path through the tree represents its own class-conditional distribution. Each branching point represents a categorical distribution which determines the path to take for a sample during the tree traversal. By changing the parameters of the transformations over time, such trees can represent evolving distribution suitable to generate data streams (see Figure 2).

Summary

StreamGen is a framework for generating streams of labeled, synthetic data from trees composed of sampling functions and transformation monoids (see Figure 1).

Due to the expensive nature of the labelling process, researchers and machine learning practitioners often rely on existing datasets and stochastic data augmentation pipelines like `torchvision.transforms.Compose` objects in torchvision (maintainers & contributors, 2016). While such methods and datasets are enough to study learning from static domains, emerging research fields like continual learning study learning on long streams of data, representing evolving experiences. StreamGen addresses this need by giving researchers a tool to model time-dependent, diverse class-conditional distributions.

Such distributions can be represented through the use of a tree data structure (or other more general linked structures like directed acyclic graphs (DAG)) to store sampling functions and

transformations. Samples are generated by traversing the tree from the root to the leaves. Each branching point represents a categorical distribution which determines the path to take for a sample during the tree traversal.

Such a tree comprised of fixed transformations represents a static, class-conditional distribution. In order to extend the framework to evolving distributions (streams), either the **parameters** of the stochastic transformations or the **topology** of the tree needs to be changed over time (see Figure 2). Due to the complexity of designing and reasoning about evolving topologies, the first release of StreamGen (version 1.0) focuses on static tree topologies and only schedules the parameters of the transformations and the probabilities of the branching points.

StreamGen implements the following **abstractions** and utility functions to design data streams:

- Classes and functions to construct, schedule and visualize time-dependent parameters
- A selection of custom nodes based on anytree's `NodeMixin` (c0fec0de, 2016)
- A `SamplingTree` class with:
  - A pythonic short-hand construction via nested lists and dictionaries
  - Parameter scheduling and configuration of all nodes via one `update()` call
  - Multiple sampling strategies (stochastic traversal, stratified, pruned)
  - Visualizations using graphviz (Gansner & North, 1997)
- Stream abstraction to use datasets created with StreamGen in CL frameworks like avalanche (Lomonaco et al., 2021) or continuum (Douillard & Lesort, 2021)

The documentation also contains different stream generation examples:

1. Multi-class time series with different data drifts (covariate, prior-probability and concept shift)
2. An analog version of the WM811k dataset (Wu et al., 2015) (binary images) with covariate shift for Domain Adaptation research
3. A defect density wafer map dataset with geometrically generated patterns

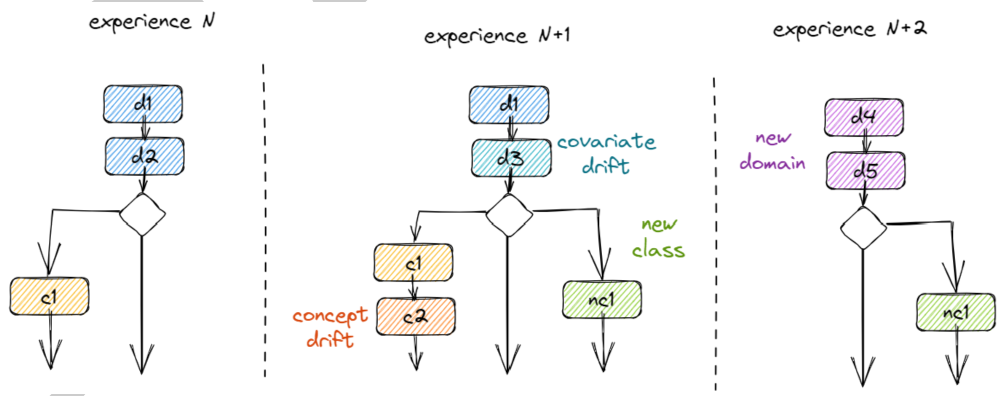

Figure 2: Changes in the topology of the tree of transformations are one possibility to represent evolving (time-dependent) distributions with different data drift scenarios. In this example, the sampling function and transformations labelled as d represent the *domain* - the data distribution common to all classes. c and nc represent class-specific transformations responsible for adding the class-specific parts of the signal. By changing the parameters or the transformations of the tree's root (the sequence of transformations before the first branching point), **covariate shift** can be simulated. By changing the probabilities of the branching points, (class-) **prior-probability shift** can be simulated. Concept shift is modelled through a change in the class-specific branches. New classes can be introduced by adding branches to the tree and new domains can be introduced by exchanging the trunk of the tree.

Statement of need

Most machine learning systems rely on *stationary, labeled, balanced and large-scale* datasets. **Incremental learning** (IL), also referred to as **lifelong learning** (LL) or **continual learning** (CL), extends the traditional paradigm to work in dynamic and evolving environments (Masana et al., 2022). This requires such learning systems to acquire and preserve knowledge continually.

Existing CL frameworks like avalanche (Lomonaco et al., 2021) or continuum (Douillard & Lesort, 2021) construct data streams by *splitting* large datasets into multiple *experiences* (a scenario called class-incremental learning), which has a few shortcomings:

- Real data streams are rarely comprised of disjoint experiences
- Such constructed scenarios offer limited insight into factors of the stream other than the class distribution, which are required to study learning scenarios with fewer constraints on the stream properties like domain adaptation or class-incremental scenarios with repetition. Some researchers even argue that the dominance of class-incremental scenarios has lead to the proposal of several rather complex methods, that completely fail in more realistic, unconstrained scenarios with repetition (Cossu et al., 2021)
- The evaluation of continual learners on such scenarios is not trivial as evident by the wealth of proposals (Ven et al., 2024)

To answer different research questions in the field of CL, researchers need knowledge and control over a variety of factors of the underlying data distribution including:

- Class distributions
- Novelties and outliers
- Complexity and evolution of the background domain
- Semantics of the unlabeled parts of a domain
- Class dependencies
- Class composition (for multi-label modelling)

A more economical alternative to collecting and labelling streams with desired properties is the **generation** of synthetic streams (Lu et al., 2018). Some mentionable efforts in that direction include augmentation based dataset generation like ImageNet-C (Hendrycks & Dietterich, 2019) or simulation-based approaches like the EndlessCLSim (Hess et al., 2021), where semantically labeled street-view images are generated by a game engine, that procedurally generates the city environment and simulates drift by modifying parameters (like the weather and illumination conditions) over time.

StreamGen builds on these ideas and provides researchers with a general and intuitive framework to generate data streams without constraints on the stream characteristics and the full knowledge of underlying distributions. It will lay the foundation for more directed and efficient research on Continual Learning.

Future work

The generation of multi-label samples requires loops and cycles for a compact and convenient representation. Such scenarios are still representable with tree data structures by unrolling these cycles through many redundant paths and transformations. A representation using less restricted types of graphs presents an interesting future extension to the framework.

StreamGen already defines protocols and base classes to include different sampler concepts in the future.

More declarative ways to build schedules and distributions with special characteristic represent other promising extensions.

Acknowledgements

This work was funded by the Austrian Research Promotion Agency (FFG, Project No. 905107).

Special thanks to Benjamin Steinwender, Marius Birkenbach and Nikolaus Neugebauer for their valuable feedback.

I also want to thank Infineon and KAI for letting me publish this project under a permissive and open license.

Finally, I want to thank my university supervisors Thomas Pock and Marc Masana for their guidance.

References

c0fec0de. (2016). *Anytree: Python tree data library*. GitHub. https://github.com/c0fec0de/anytree

Cossu, A., Graffieti, G., Pellegrini, L., Maltoni, D., Bacciu, D., Carta, A., & Lomonaco, V. (2021). *Is Class-Incremental Enough for Continual Learning?* arXiv. http://arxiv.org/abs/2112.02925

Douillard, A., & Lesort, T. (2021). *Continuum: Simple Management of Complex Continual Learning Scenarios*. arXiv. https://doi.org/10.48550/arXiv.2102.06253

Gansner, E., & North, S. (1997). An Open Graph Visualization System and Its Applications to Software Engineering. *Software - Practice and Experience - SPE*, *30*. https://doi.org/10.1002/1097-024X(200009)30:11%3C1203::AID-SPE338%3E3.0.CO;2-N

Hendrycks, D., & Dietterich, T. (2019). *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations*. arXiv. https://doi.org/10.48550/arXiv.1903.12261

Hess, T., Mundt, M., Pliushch, I., & Ramesh, V. (2021). *A Procedural World Generation Framework for Systematic Evaluation of Continual Learning*. arXiv. https://doi.org/10.48550/arXiv.2106.02585

Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T. L., De Lange, M., Masana, M., Pomponi, J., Van De Ven, G. M., Mundt, M., She, Q., Cooper, K., Forest, J., Belouadah, E., Calderara, S., Parisi, G. I., Cuzzolin, F., Tolias, A. S., … Maltoni, D. (2021). Avalanche: An End-to-End Library for Continual Learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 3595–3605. https://doi.org/10.1109/CVPRW53098.2021.00399

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, 1–1. https://doi.org/10.1109/TKDE.2018.2876857

maintainers, T., & contributors. (2016). *TorchVision: PyTorch's Computer Vision library*. https://github.com/pytorch/vision

Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & Weijer, J. van de. (2022). *Class-incremental learning: Survey and performance evaluation on image classification*. arXiv. https://doi.org/10.48550/arXiv.2010.15277

Ven, G. M. van de, Soures, N., & Kudithipudi, D. (2024). *Continual Learning and Catastrophic Forgetting*. arXiv. http://arxiv.org/abs/2403.05175

Wu, M.-J., Jang, J.-S. R., & Chen, J.-L. (2015). Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets. *IEEE Transactions on Semiconductor Manufacturing*, *28*(1), 1–12. https://doi.org/10.1109/TSM.2014.2364237