A Comparative Study of Continual, Lifelong, and Online Supervised Learning Libraries

Logan Cummins, Brad Killen, Somayeh Bakhtiari Ramezani, Shahram Rahimi, Sudip Mittal
Mississippi State University
Starkville, MS
{nlc123, bmk99, sb3182}@msstate.edu,
{rahimi, mittal}@cse.msstate.edu

Maria Seale
U.S. Army Eng. Research and Dev. Center
Vicksburg, MS
maria.a.seale@erdc.dren.mil

Abstract

Machine learning has shown to be a crucial part of big data analytics; however, it lacks when the data is continuously streaming in from the system and changing too much from the original training data. Online learning is machine learning for streaming data that arrives in a sequential order where the model updates after every data point. While machine learning relies on well-established libraries such as PyTorch and Keras, the libraries for online learning are less well known, but they are here to serve similar purposes of reproducibility and reducing the time from research to production. Here, we compare different libraries for online learning research, specifically supervised learning. We compare them on the axes of developmental experience and benchmark testing as researchers. Our comparison as developers takes maintenance, documentation, and offerings of state-of-the-art algorithms into account. As this is not necessarily free of bias, we also use benchmarks known to online learning to gather power usage, RAM usage, speed, and accuracy of these libraries to get an objective view. Our findings show that Avalanche and River, including River-torch, are among the best libraries in terms of performance and applicability to the research in supervised online learning.

Introduction

Machine learning (ML) has proven itself over time to be an important aspect to big data analytics (Athmaja, Hanumanthappa, and Kavitha 2017). Recently, ML has been applied to many real-time tasks such as COVID-19 modeling and prediction (Farooq and Bazaz 2021; Lalmuanawma, Hussain, and Chhakchhuak 2020), prognostics of machinery (Wen et al. 2022; Cummins et al. 2021; Adhikari, Rao, and Buderath 2018), and more. Most of these applications follow the same offline learning paradigm where a model is trained and deployed. While successful, this aspect of offline learning causes trouble for future use.

Offline ML assumes complete data availability (Losing, Hammer, and Wersing 2018) which is not always true, especially with real-time data. Real-time data can shift form quickly with a high rate of incoming data (Hammer, He, and Martinetz 2014; Benczúr, Kocsis, and Pálovics 2018). This would require possible model retraining which is inefficient both in time and space costs (Hoi et al. 2021). One way of overcoming these challenges is to transform into a different paradigm, online learning.

Online learning (OL) is a sub-field of ML that includes a family of techniques devised to learn models incrementally from data as it becomes available to the network (Hoi et al. 2021; Benczúr, Kocsis, and Pálovics 2018). The main component shared amongst OL methods is the idea that the models are trained with data that are received in real-time then potentially discarded. Many techniques store little previously seen data such as Learning without Forgetting and Replay, and some store none, which makes these networks scalable in terms of space. OL also saves space by using simpler architectures like Naive Bayes and Perceptron. OL also saves in time as retraining is not needed once the data changes form (Hoi et al. 2021). For more information on OL as a sub-field of ML, we divert to an extensive review done by Hoi *et al.* that looked at OL from all angles from algorithms to applications (Hoi et al. 2021).

With all of the work in developing algorithms for OL, the development seems to be unique to the individual researcher's style and coding preferences. This makes comparisons between methods more difficult as there is not a shared framework. The field of ML has been overcoming this issue with the developments of popular libraries like Keras (Chollet and others 2015) and PyTorch (Paszke et al. 2019) which have seen wide success. Similarly, libraries have come to life to assist the OL community. These libraries provide many state-of-the-art algorithms as well as benchmarks to use for comparison; however, most of these libraries are not introduced to the research community and light should be shown upon them, so they are applied more to research.

This study aims to compare multiple different libraries for continual, lifelong, online learning, and more. These different learning approaches are similar enough that comparing their libraries is within reason. In order to make a quantitative comparison between the performance of different libraries, the present work looks at supervised learning. The rest of this paper is structured as follows. The following section provides a background that defines and discusses the intricacies of continual, lifelong, and online learning. Next information is provided about the different libraries in the analysis. This is followed by the analysis criteria which is

broken down into development comparison and benchmark comparison. Finally, results of the comparative analysis are presented, and a discussion about the findings concludes the article.

Background

OL consists of algorithms that generally learn from data instances one at a time (Hoi et al. 2021). This important distinction of how the algorithms learn is the way that OL overcomes some of the drawbacks of batch learning previously mentioned. Continual learning, also known as lifelong learning, encompasses ML algorithms with the ability to learn new knowledge overtime while retaining already learned information (Parisi et al. 2019; Parisi and Lomonaco 2020; De Lange et al. 2022). While this approach is similar to OL, because of the algorithms' tendencies to learn from batches instead of instances, it is not considered OL (Hoi et al. 2021). An extension to continual learning called online continual learning has been brought to the spotlight in recent years with its ability to modify continual learning mechanisms to learn strictly from data instances one at a time. These works are descibed in more depth by Parisi and Lomonaco (Parisi and Lomonaco 2020). The present work considers online continual learning as one of the important players in this field, thus this technique is included as an integral part of our research.

Incremental learning is a collection of techniques that are suitable to learn from data streams while keeping space and computational costs down (Yang, Gu, and Wu 2019; Gepperth and Hammer 2016; Ade and Deshmukh 2013). These methods can be used in both online and offline settings, i.e. learning from single data points or small batches. Incremental learning can be thought of as a branch of OL (Hoi et al. 2021) where the key difference is the specific goal of keeping space and computational costs small in incremental learning.

Finally, interactive learning aims to bring humans, i.e. users or experts, into the online training process. This provides domain knowledge in the learning process, effective communication, and continuous improvements for learning efficacy (Hoi et al. 2021). While notable a method of OL, this is left out of the experiment due to our experiment set up.

All of these different types of learning perform similarly when learning from data streams. As this is not the typical training performance behind ML, several libraries have been created to facilitate this task. The next section provides descriptions of the most prominent libraries used in this field. While authors are cognisant of the differences between OL and similar methods of learning discussed above, for the sake of simplicity, in the next section we will simply refer to these libraries as "Online Learning" libraries.

Online Learning Libraries

Avalanche

Avalanche is an end-to-end library for continual learning based on PyTorch (Lomonaco et al. 2021) available on Github [1] from the Continual AI organization. Avalanche provides a framework for prototyping and comparing different continual learning techniques, including online continual learning. This approach towards creation of a platform for continual learning has led to the adoption of design principles of modularity, scalability, reproducibility, and more (Lomonaco et al. 2021). Avalanche's architecture, depicted in Fig. 1, is broken down into 5 main components: benchmark, model, training, evaluation, and logging.

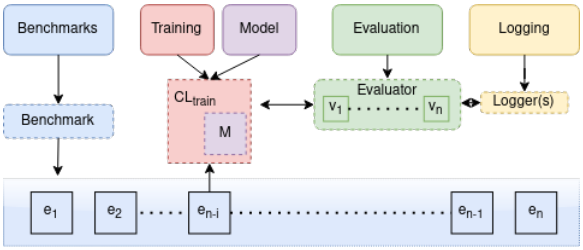

Figure 1: Avalanche Architecture (Lomonaco et al. 2021)

The *Benchmarks* component of Avalanche contains a wide variety of datasets and set-ups for continual learning. The *Model* component houses all of the base models that Avalanche provides such as Convolutional Neural Networks. The *Training* component contains plugins that allow customizability to the learning strategy such as Learning without Forgetting (LwF). The *Evaluation* module provides metrics to measure performance. Finally, the *Logging* component provides a service to display or store training and evaluation information.

Library for Online Learning Algorithms (LIBOL)

LIBOL is an OL library created by Hoi *et al.* (Hoi, Wang, and Zhao 2014) and is publically available on Github [2]. This library contains algorithms that were popular at the time of its publication implemented in C++ and MATLAB. The authors provided a few datasets to showcase all of the different functionalities. However, LIBOL has not been maintained since 2014. In 2018, a user converted LIBOL into Python [3], but they only converted and did not add. This keeps the functionality of LIBOL limited but worth including.

River

River is an ML library for data streams and incremental learning that provides state-of-the-art learning methods, data generators, performance metrics, and evaluators (Montiel et al. 2021). This library, which is on Github [4], came from the collaboration of the creators of scikit-multiflow (Montiel et al. 2018) and creme (Halford et al. 2019). River's architecture is depicted in Fig. 2 and can be broken down into three main sections: Dataset, Pipeline, and Metric.

The *Dataset* portion provides datasets and scripts needed to create datasets. The *Pipeline* module contains both the

[1] https://github.com/ContinualAI/avalanche
[2] https://github.com/LIBOL/LIBOL
[3] https://github.com/LGuitron/LIBOL-python
[4] https://github.com/online-ml/river

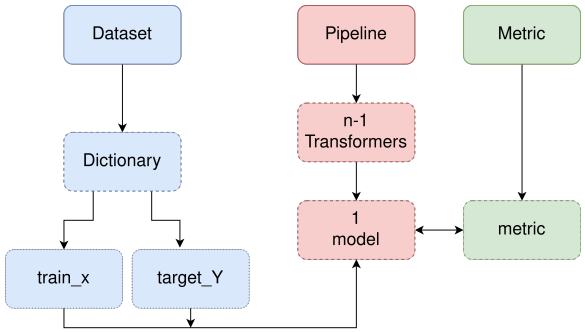

Figure 2: River Architecture (Montiel et al. 2021)

models that River provides and the different transforms one could apply to the dataset, such as scaling and drift detection. Finally, the *Metric* section provides different metrics that measure the performance of the model.

River-torch

River-torch is an extension to River maintained on Github [5]. It provides a wrapper that allows development of a PyTorch model to be inserted into the River pipeline. River-torch allows for developers and researchers with PyTorch experience to test models that are not natively supported in the River framework.

Vowpal Wabbit (VW)

The VW project was created by Microsoft Research and Yahoo! Research and hosted on Github [6]. This library focuses on fast learning algorithms for many areas of research including interactive learning and OL. VW aims to perform computationally efficient learning via a number of methods that include bounding the memory usage and formatting the input data in a free-form manner. This project is available in many languages such as R, Python, and Java and provides a purely OL environment for testing datasets via scikit-learn and their available set of algorithms.

Other Libraries

As previously stated, this study only focuses on supervised learning approaches in OL; however, there are a number of other libraries and functionalities that fall outside our scope. We will provide a brief overview of these libraries.

ContinualAI has a library dedicated to continual reinforcement learning known as Avalanche-RL [7]. Additionally, VW has a wide range of packages for interactive and active learning; it also supports contextual bandit learning algorithms. Additionally, River has unsupervised learning capabilities.

Comparative Analysis

Development Comparison

To analyze the different libraries, we broke the comparison down into two methods: model development and benchmarking. The development aspect of our comparison looks at features that would be useful to a user. This can be portrayed in many different scenarios: a product developer, a researcher looking to test the state-of-the-art algorithms and the likes. With this perspective, we create the bases for the development comparison: maintenance, documentation, and state-of-the-art. For maintenance, we are looking for how actively (i.e., date and frequency) the library is updated by the creators. Documentation is considered as a way of instructing the users on how to effectively use the library in the form of examples and comments. Finally, the state-of-the-art offerings of libraries is rather subjective. We examined the literature, e.g. (Hoi et al. 2021; De Lange et al. 2022; Parisi et al. 2019; Parisi and Lomonaco 2020; Gepperth and Hammer 2016; Losing, Hammer, and Wersing 2018; Shahraki et al. 2022; Yang, Gu, and Wu 2019; Sahoo et al. 2017) to gauge what algorithms should be present. These algorithms are listed in Table 1.

Benchmark Comparison

The benchmark comparison looks at the performance of the libraries under similar scenarios. This is used as a method of observing aspects such as power usage and memory usage of the libraries. We compare the runtime, memory usage, and power usage of the model creation and training. We do not account for the impact of loading the dataset into memory and testing in this analysis. We used the Python libraries memory-profiler [8] and pyRAPL [9] to gather information about RAM usage and power usage respectively.

In order to maintain consistency, control of the data source and formatting, we used datasets that are native to the Avalanche library. The data was then transformed to the formats necessary for the other libraries outside of the analysis. The workstation used for the testing had 125 GB of RAM, an 80 core Intel®Xeon Gold 6230 CPU, and was running Ubuntu 20.04.5 LTS.

Firstly, we use the Split MNIST dataset (Deng 2012) which is made up of 60,000 28x28 handwritten digits in 10 classes, and comprises of 50,000 training images and 10,000 test images. The second dataset is the CIFAR-10 dataset (Krizhevsky, Hinton, and others 2009) which consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. Finally, we used the Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset (Wah et al. 2011), which our version that was used for this work consists of 100 categories of birds and 5,864 128x128 images.

Due to the differences in the available architectures between the libraries, we kept the architectures similar where

[5] https://github.com/online-ml/river-torch
[6] https://github.com/VowpalWabbit/vowpal_wabbit
[7] https://github.com/ContinualAI/avalanche-rl
[8] https://github.com/pythonprofilers/memory_profiler
[9] https://github.com/powerapi-ng/pyRAPL

Table 1: State of the Art Algorithms From the Literature

Algorithm	River	Avalanche	LIBOL	VW
SVM (Yang, Gu, and Wu 2019; Losing, Hammer, and Wersing 2018)	**Yes**	No	No	No
Naive Bayes (Yang, Gu, and Wu 2019; Losing, Hammer, and Wersing 2018)	**Yes**	**Yes**	No	No
RF (Yang, Gu, and Wu 2019; Losing, Hammer, and Wersing 2018)	**Yes**	No	No	No
Neural Networks (Yang, Gu, and Wu 2019; Sahoo et al. 2017)	**Yes**	**Yes**	**Yes**	**Yes**
ART (Gepperth and Hammer 2016)	No	No	No	No
RBF (Gepperth and Hammer 2016)	**Yes**	No	No	No
ELM (Gepperth and Hammer 2016)	No	No	No	No
Ensemble (Gepperth and Hammer 2016; Parisi et al. 2019)	**Yes**	No	No	No
LVQ (Gepperth and Hammer 2016; Losing, Hammer, and Wersing 2018)	No	No	No	No
KNN Classifier (Gepperth and Hammer 2016)	**Yes**	No	No	No
Hoeffding Tree (Shahraki et al. 2022)	**Yes**	No	No	No
Very Fast Decision Tree (Shahraki et al. 2022)	**Yes**	No	No	No
Hoeffding Adaptive Tree (Shahraki et al. 2022)	**Yes**	No	No	No
Extremely Fast Decision Tree (Shahraki et al. 2022)	**Yes**	No	No	No
LASVM (Losing, Hammer, and Wersing 2018)	No	No	No	No
SGD (Losing, Hammer, and Wersing 2018)	**Yes**	**Yes**	**Yes**	**Yes**
Learn++ (Losing, Hammer, and Wersing 2018)	No	No	No	No
LWF (Parisi et al. 2019; Parisi and Lomonaco 2020; De Lange et al. 2022)	No	**Yes**	No	No
EWC (Parisi et al. 2019; De Lange et al. 2022)	No	**Yes**	No	No
AR1 (Parisi et al. 2019; Parisi and Lomonaco 2020)	No	**Yes**	No	No
Dynamic Architectures (Parisi et al. 2019)	**Yes**	**Yes**	No	**Yes**
GWR (Parisi et al. 2019; Parisi and Lomonaco 2020)	No	No	No	No
CWR (Parisi and Lomonaco 2020)	No	**Yes**	No	No
CWR+ (Parisi and Lomonaco 2020)	No	No	No	No
Replay (Parisi and Lomonaco 2020)	No	**Yes**	No	No
SI (Parisi and Lomonaco 2020; De Lange et al. 2022)	No	**Yes**	No	No
ICARL (Parisi and Lomonaco 2020; De Lange et al. 2022)	No	**Yes**	No	No
GEM (De Lange et al. 2022)	No	**Yes**	No	No
VCL (De Lange et al. 2022)	No	No	No	No
EBLL (De Lange et al. 2022)	No	No	No	No
MAS (De Lange et al. 2022)	No	**Yes**	No	No
IMM (De Lange et al. 2022)	No	No	No	No
PackNet (De Lange et al. 2022)	No	No	No	No
HAT (De Lange et al. 2022)	No	No	No	No
Perceptron (Hoi et al. 2021)	**Yes**	**Yes**	**Yes**	No
Winnow (Hoi et al. 2021)	No	No	No	No
PA (Hoi et al. 2021; Lu, Zhao, and Hoi 2016)	**Yes**	No	**Yes**	**Yes**
OGD (Hoi et al. 2021)	No	No	**Yes**	**Yes**
Cost Sensitive (Hoi et al. 2021)	No	No	No	**Yes**
Truncated GD (Hoi et al. 2021)	**Yes**	No	No	No

possible. Below are descriptions of the architectures used in the experiment. For brevity, the input sizes change between datasets to accommodate for the dimensionality of the inputs, and the output sizes are the amount of classes in the dataset.

Avalanche Architecture The Avalanche architecture used in this work is composed of an MLP architecture with 256 nodes in the hidden layer. In order to use Avalanche's features, the LwF was used to tackle catastrophic forgetting. While this would go against the OL mentality of minimizing disk usage, this is one method that continual learning can offer to online continual learning. The network is also trained in an online manner where it only receives one input datum at a time.

LIBOL Architecture The LIBOL architecture utilizes their native MLP with uniform update. The information about the hidden layers is not stated in the source code. Since this architecture is desgined for multiclass classification, the output layer adapts to the different number of classes. Just as all LIBOL algorithms, this one is trained in an online setting by nature of the library.

River Architecture Our architecture for River consists of a MLP combined with River's OneVSRest (OVR) classifier. This allows growth in the architecture as it comes across more classes. This is one way for OL to allow higher accuracy as the network changes to maintain an accurate performance.

River-torch Architecture Because River-torch allows for PyTorch models, the architecture is most similar to the Avalanche architecture. The Avalanche MLP architecture was copied over to River-torch. This allowed for the same architecture to be trained through River as opposed to Avalanche.

Table 2: Experimental Results

Dataset	Library	Accuracy	Runtime (s)	RAM Usage (MiB)	Power Usage (J)
Split MNIST	Avalanche	95.09%	409.9269	3284.1	62097.158
	LIBOL	86.27%	21.12	**0.1**	2444.786
	River	90.23%	161.0113	0.7	19279.709
	River-torch	**95.51%**	323.7411	15.3	37254.4664
	VW	90.22%	**5.7848**	11.5	**684.8273**
CIFAR10	Avalanche	**36.71%**	369.6757	2030.0	250662.586
	LIBOL	17.22%	102.5437	**0.8**	14454.985
	River	15.10%	546.5383	**0.8**	66258.0617
	River-torch	35.25%	800.6992	15.4	94745.7943
	VW	19.13%	**23.0954**	11.4	**2779.7661**
CUB-200-2011	Avalanche	1.19%	91.13	1656.4	12797.333
	LIBOL	**3.56%**	277.6035	**0.8**	262552.5097
	River	1.05%	3323.0912	108.4	390255.9768
	River-torch	1.05%	1060.7573	138.4	229119.0817
	VW	1.04%	**16.4296**	8.7	**2001.0211**

VW Architecture The VW architecture is the most dissimilar of the lot as VW does not natively support MLP. However, its architecture is more similar to River's architecture. It is a OneAgainstAll (OAA) classifier where multiple simple classifiers are used to learn individual classes, and these outputs are combined to determine which class the datum belongs.

Results and Discussion

In this section, we present the results of our development and benchmark comparisons. The benchmark comparison results are summarized in Table 2. The best performances of each category for each dataset are presented in bold font face.

Development Comparison

As developmental experiences differ between developers, this is a description of our unique experiences. While these are anecdotal, we believe valuable knowledge can come from the experience of others, so we choose to share our experience while trying to maintain a level of objectivity.

Maintenance Avalanche, River, and River-torch are currently in development; meaning they are all prior to version 1.0 releases. These are all being updated very regularly with multiple additions happening within a month. VW is on version 9.6.0 as of writing, and it sees regular updates as well. This library has existed for much longer, so it is more polished than the others. Finally, LIBOL has not seen an update in many years. We feel it is safe to say that no updates will be coming to the LIBOL library anytime in the near future which limits its usability in our perspective.

Documentation The documentation seems to be of equal quality as the maintenance of the libraries. As the most developed library, VW has a very in-depth documentation on their wiki page that goes over all of the functionalities in great detail. As the youngest library, River-torch has very little documentation, but when combined with River's documentation, they have a decent portal for documentation and examples. Some functionality was in the descriptions, but it just seems to not be finished at the time. Avalanche is on the same level of documentation as River. It mostly is complete, but there is some functionality unexplained in the documentation. Finally, LIBOL's documentation is simple, elegant, and straight to the point. It provides everything one needs to run their models.

State-of-the-Art When looking at state-of-the-art offers from the libraries, Table 1 summarizes the findings from the literature in comparison to the offerings of the libraries. Table 3 defines the abbreviations used in Table 1.

In regards to incremental learning, River, and by extension River-torch, includes most of the state-of-the-art algorithms. Some of the other algorithms, such as Learn++, are in the works as discovered through the discussion boards, so this library is only going to grow to continue supporting the new algorithms. Regarding continual learning, Avalanche also supports a majority of the desired algorithms. This library follows similarly to River where the discussions show that algorithms that are not currently implemented, such as PackNET and HAT, are in the works. Finally for OL specific algorithms, those found in (Hoi et al. 2021), no one library seems to support many algorithms, but there are not many algorithms to support. VW supports the most specifically OL algorithms. Avalanche is leaning towards support for online continual learning, so some of these algorithms may show up in the future. River has also made efforts in implementing purely OL algorithms, so it follows a similar story.

Benchmark Comparison

As previously stated, Table 2 shows the results of our benchmark analysis, where the higher accuracy is better. Lower runtime, RAM usage, and power usage are also better. Finally, RAM usage is measured in Mebibytes as the output of the memory-profiler library is Mebibytes. Mebibytes are slightly larger, roughly 1.049 times the size, than Megabytes. Therefore, they can be thought of as similar.

Split MNIST Overall, each library performed well in accuracy where River-torch and Avalanche performed the best

at roughly 95% accuracy. These are both using the same code for the architectures, so we can say the PyTorch implementation brings noticeably better performance than the non-PyTorch implementations. VW brings efficient computation with the fastest runtimes and least power usage. As this one was not using a MLP, the most efficient library using a MLP was LIBOL with second best runtime and power usage and the best RAM usage.

CIFAR10 All-in-all the performance accuracies of each library are not good in this dataset with these setups. Avalanche performs slightly better than River-torch with 36.71% accuracy while the others perform in the 15% to 20% accuracy range. Just like for Split MNIST, VW performed the fastest while using the least amount of power. The second best is still LIBOL. RAM usage actually saw a tie between LIBOL and River which was the second best performing in this category for Split MNIST.

CUB-200-2011 None of these libraries perform well on this dataset with these setups. The only library that performs better than random chance is LIBOL with a 3.56% accuracy. VW has the best performance runtime and power usage which is followed by Avalanche. Similar to the other datasets, LIBOL manages to keep its RAM usage very low.

Discussion

These architectures were not optimized for optimal performance, rather they were optimized for similarity of the problem setup as to offer some insights of the inner workings of the different libraries for OL. With this, we want to bring up the differences in performances in relation to the problem setups.

The Split MNIST and CIFAR10 datasets are exactly the same in number of data instances and classifications. The differences are slightly more features in CIFAR10 and colored images in CIFAR10. These features have shown to be detrimental to the performance of the architectures. It minimally affected RAM usage, but power and runtime were greatly impacted. These measurements were generally increased by 2- to 7-folds.

Another finding is that the PyTorch based libraries, i.e. Avalanche and River-torch, have an interesting performance trajectory compared to the others. Avalanche shows to be the slowest and most power hungry library until the very high dimensional dataset (CUB-200-2011). River-torch shows to be in the upper rankings in runtime and power usage, but once it is tested with the CUB-200-2011, it has increased at a slower rate than the non-Pytorch libraries.

The final item to address is the consistent high RAM usage of Avalanche. One possible reasoning is how Avalanche loads the data. While not certain, there could be a data loading process in the training function. If this is true, the other libraries had the data loaded into the code explicitly while Avalanche might have a generator or pointer to the data. This would have to be tested explicitly, but this is an after thought to the analysis.

Table 3: Nomenclature for Table 1

Abbreviation	Meaning
SVM	Support Vector Machine
RF	Random Forest
ART	Adaptive Resonance Theory
RBF	Radial Basis Function
ELM	Extreme Learning Machine
LVQ	Learning Vector Quantization
KNN Classifier	K-Nearest Neighbors Classifier
LASVM	Online Approximate SVM
SGD	Stochastic Gradient Descent
LWF	Learning without Forgetting
AR1	Architect & Regularize
EWC	Elastic Weight Consolidation
GWR	Grow When Required
CWR	Copy Weight with Reinit
CWR+	CWR + Mean-shift and Zero Initialization
SI	Synaptic Intelligence
ICARL	Incremental Classifier & Representation Learning
GEM	Gradient Episodic Memory
VCL	Variational Continual Learning
EBLL	Encoder Based Lifelong Learning
MAS	Memory Aware Synapses
IMM	Incremental Moment Matching
HAT	Hard Attention to the Task
PA	Passive Aggressive Network
OGD	Online Gradient Descent

Conclusion

Machine learning (ML) is an important field in big data analytics. As the field moves towards streaming data for training and analysis, we too must move our focus towards different approaches, namely online learning (OL). ML has popular libraries such as PyTorch and Keras, but OL libraries have yet to peak in popularity. We have provided two analyses of different OL libraries: a development comparison where we look at the developmental process and a benchmark comparison where we compare the performance of the libraries on three different datasets. Any of the libraries can be used, but we believe that the Avalanche, River, and River-torch are the future as they are continuously growing for online supervised learning.

Acknowledgement

This work by Mississippi State University was financially supported by the U.S. Department of Defense (DoD) High Performance Computing Modernization Program, through the US Army Engineering Research and Develop Center (ERDC) Contract #W912HZ21C0014. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Army ERDC or the U.S. DoD.

This paper and the research behind it would not have been possible without the exceptional support by the Predictive Analytics and Technology Integration (PATENT) Laboratory at Mississippi State University.

References

Ade, R. R., and Deshmukh, P. R. 2013. Methods for incremental learning: a survey. *International Journal of Data Mining & Knowledge Management Process* 3(4):119.

Adhikari, P.; Rao, H. G.; and Buderath, M. 2018. Machine learning based data driven diagnostics & prognostics framework for aircraft predictive maintenance. In *10th International Symposium on NDT in Aerospace Dresden, Germany*.

Athmaja, S.; Hanumanthappa, M.; and Kavitha, V. 2017. A survey of machine learning algorithms for big data analytics. In *2017 International conference on innovations in information, embedded and communication systems (ICIIECS)*, 1–4. IEEE.

Benczúr, A. A.; Kocsis, L.; and Pálovics, R. 2018. Online machine learning in big data streams. *arXiv preprint arXiv:1802.05872*.

Chollet, F., et al. 2015. Keras.

Cummins, L.; Killen, B.; Thomas, K.; Barrett, P.; Rahimi, S.; and Seale, M. 2021. Deep learning approaches to remaining useful life prediction: A survey. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–9.

De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; and Tuytelaars, T. 2022. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* 44(7):3366–3385.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29(6):141–142.

Farooq, J., and Bazaz, M. A. 2021. A deep learning algorithm for modeling and forecasting of covid-19 in five worst affected states of india. *Alexandria Engineering Journal* 60(1):587–596.

Gepperth, and Hammer. 2016. Incremental learning algorithms and applications. *European symposium on artificial*.

Halford, M.; Bolmier, G.; Sourty, R.; Vaysse, R.; and Zouitine, A. 2019. creme, a python library for online machine learning.

Hammer, B.; He, H.; and Martinetz, T. 2014. Learning and modeling big data. In *ESANN*, 343–352. Citeseer.

Hoi, S. C. H.; Sahoo, D.; Lu, J.; and Zhao, P. 2021. Online learning: A comprehensive survey. *Neurocomputing* 459:249–289.

Hoi, S. C. H.; Wang, J.; and Zhao, P. 2014. LIBOL: A library for online learning algorithms. *J. Mach. Learn. Res.* 15(1):495.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Lalmuanawma, S.; Hussain, J.; and Chhakchhuak, L. 2020. Applications of machine learning and artificial intelligence for covid-19 (sars-cov-2) pandemic: A review. *Chaos, Solitons & Fractals* 139:110059.

Lomonaco; Pellegrini; Cossu; and others. 2021. Avalanche: an end-to-end library for continual learning. *Proc. Estonian Acad. Sci. Biol. Ecol.*

Losing, V.; Hammer, B.; and Wersing, H. 2018. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing* 275:1261–1274.

Lu, J.; Zhao, P.; and Hoi, S. C. H. 2016. Online passive-aggressive active learning. *Mach. Learn.*

Montiel; Read; Bifet; and Abdessalem. 2018. Scikit-multiflow: A multi-output streaming framework. *J. Mach. Learn. Res.*

Montiel, J.; Halford, M.; Mastelini, S. M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A.; Gomes, H. M.; Read, J.; Abdessalem, T.; et al. 2021. River: machine learning for streaming data in python. *The Journal of Machine Learning Research* 22(1):4945–4952.

Parisi, G. I., and Lomonaco, V. 2020. Online continual learning on sequences. In Oneto, L.; Navarin, N.; Sperduti, A.; and Anguita, D., eds., *Recent Trends in Learning From Data: Tutorials from the INNS Big Data and Deep Learning Conference (INNSBDDL2019)*. Cham: Springer International Publishing. 197–221.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Netw.* 113:54–71.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32.

Sahoo, D.; Pham, Q.; Lu, J.; and Hoi, S. C. H. 2017. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*.

Shahraki, A.; Abbasi, M.; Taherkordi, A.; and Jurcut, A. D. 2022. A comparative study on online machine learning techniques for network traffic streams analysis. *Computer Networks* 207:108836.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wen, Y.; Rahman, M. F.; Xu, H.; and Tseng, T.-L. B. 2022. Recent advances and trends of predictive maintenance from data-driven machine prognostics perspective. *Measurement* 187:110276.

Yang, Q.; Gu, Y.; and Wu, D. 2019. Survey of incremental learning. In *2019 Chinese Control And Decision Conference (CCDC)*, 399–404.