Deep Online Storage-Free Learning on Unordered Image Streams

Andrey Besedin[1(✉)], Pierre Blanchart[1], Michel Crucianu[2], and Marin Ferecatu[2]

[1] CEA, LIST, Laboratoire d'Analyse de Donnes et Intelligence des Systemes, Digiteo Labs Saclay, 91191 Gif-sur-Yvette Cedex, France
{andrey.besedin,pierre.blanchart}@cea.fr

[2] Centre d'études et de recherche en informatique et communications, Le CNAM, 292 rue Saint-Martin, 75003 Paris, France
{michel.crucianu,marin.ferecatu}@cnam.fr

Abstract. In this work we develop an online deep-learning based approach for classification on data streams. Our approach is able to learn in an incremental way without storing and reusing the historical data (we only store a recent history) while processing each new data sample only once. To make up for the absence of the historical data, we train Generative Adversarial Networks (GANs), which, in recent years have shown their excellent capacity to learn data distributions for image datasets. We test our approach on MNIST and LSUN datasets and demonstrate its ability to adapt to previously unseen data classes or new instances of previously seen classes, while avoiding forgetting of previously learned classes/instances of classes that do not appear anymore in the data stream.

Keywords: Deep learning · GAN · Data streams · Classification

1 Introduction

In recent years methods based on Deep Learning have become state of the art in numerous applications, such as image and signal classification [7], object detection [10] and segmentation [4], natural language processing [2,11] and many others. Despite its popularity and efficiency, most of the currently existing applications are based on offline learning where all the data are constantly available during training. On the other hand, scenarios where data arrive continuously in large quantities and have to be integrated into the learning models in real time are starting to get more and more attention from the Machine Learning community.

In this context, the main problem is that most of the Deep Learning methods are prone to forgetting the concepts that are no longer represented by the dataset they are trained on. In literature this phenomena is known as catastrophic forgetting [8]. The current solution for this problem is to store the whole

A. Monreale et al. (Eds.): ECML PKDD 2018 Workshops, CCIS 967, pp. 103–112, 2019.
https://doi.org/10.1007/978-3-030-14880-5_9

dataset to be able to reuse all the data samples at any moment. At the same time, training Neural Networks is based on gradient backpropagation, which is slow and often requires passing through the dataset many times.

The described problems impose hard constraints for applications which require continuous learning to adapt to changing environments, and, especially, for distributed applications on devices with limited storage and computational resources, like smartphones or small private servers.

In this paper we introduce a method that uses Generative Adversarial Networks [3] to model the real data distribution and replace the necessity of storing and reusing historical data when performing online classification learning on data streams. We test our method on MNIST and LSUN datasets and show that it allows to efficiently train classifiers on multi-class streams of data with possible concept drifts [12] with no need of retraining the model on historical data.

The present proposal is based on our previous work [1] and improves upon it in several ways: first, we extend our framework to handle data coming continuously and in random order, which corresponds to a much more realistic situation; second, we test the framework on the much larger LSUN database [1] with more complex data; third, we quantify the loss (in classification accuracy) when using our method compared to the offline situation. We also study forgetting/classification improvement behavior of our approach on classes that were initially present or appear at some moment of the stream.

The rest of the paper is organized as follows: In Sect. 2 we present our method for online classification of unordered streams without data storage, followed in Sect. 3 by the experimental validation. Section 4 concludes the paper by a discussion of our work and suggests several directions for further development.

2 Proposed Method

The goal of this work is to develop a method allowing to train classifiers on data streams with time changing environment described by the set of the data classes currently present in the stream. The main issue with such a task is that neural network based classifiers tend to forget already learned classes if the corresponding data is removed from the training set, which is a realistic scenario in evolving data streams.

Intuitively, there are two possible ways to handle the problem of forgetting in Neural Networks. From one hand, one could try to control the way the backpropagation works when updating the classifier and to avoid strong modifications of the weighs of the network that are important for correct classification of those classes. From the other hand, forgetting is caused by the absence of corresponding data. Storing and reusing the data itself helps in batch learning, but is hardly feasible in the continuous massive data stream setup. Storing only partial information from missing data or some representation of it should help. In this paper we focus on second idea and propose to train generative models in order to replace the necessity of storing and reusing of historical data.

[1] http://lsun.cs.princeton.edu/2017/.

2.1 Replacing Original Data by Generators

To avoid storing historical data we train generative models to learn the distribution of the original data and use them to produce synthetic data samples to replace the original ones. To do so, we use Generative Adversarial Networks that have recently shown excellent ability to learn data distributions on image datasets and generate samples that are very similar to the original images. More specifically, we use a Deep Convolutional version of GAN (DCGAN [9]) in which, comparing to original GAN and its other convolutional modifications, all the pooling layers are replaced by stride convolutions. At the same time, the proposed architecture does not have any fully connected layers, uses Batch Normalization [6] and ReLU activation function are replaced by LeakyReLU in the discriminator network. All the described changes show better stability during training and allow to learn higher resolution models.

2.2 Batch Classification on Generated Data

In our previous work [1] we introduced quantitative metrics to evaluate generative and representative capacities of generative models on a given dataset. We demonstrated that, according to those metrics, DCGANs are able to represent well the original data and to generalize over it, i.e. allow a classifier trained on generated data to have good generalization abilities on unseen test data.

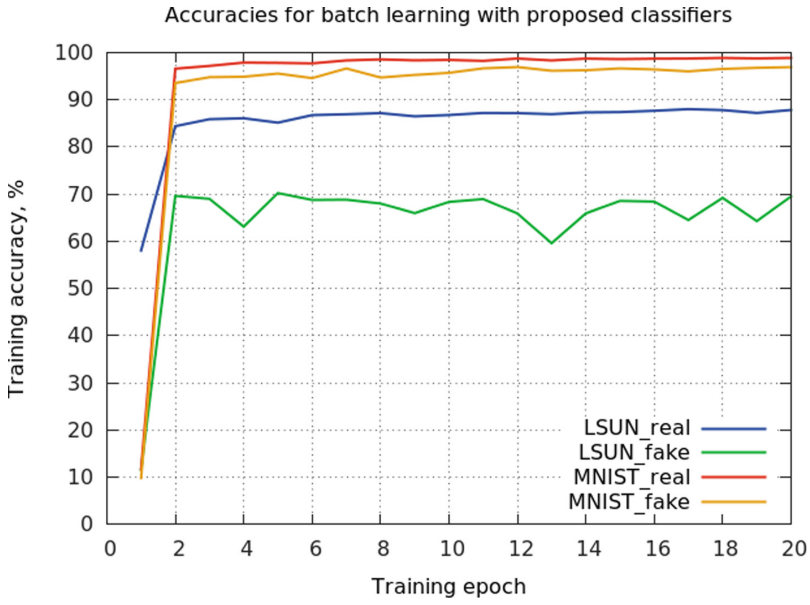

Fig. 1. Batch training accuracy on the original validation data for MNIST and LSUN dataset when trained on real vs. generated data

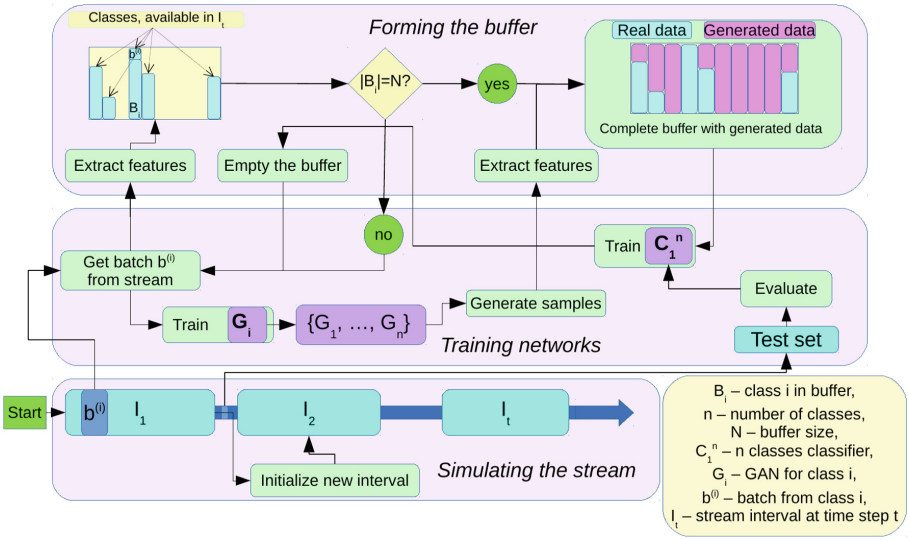

Fig. 2. Stream classification scheme (for LSUN dataset, on MNIST no feature extraction is performed), described in this work. Stream is represented as an infinite sequence of data intervals. Each processing block is described in corresponding subsection of Sect. 2.3. All the green boxes represent the processes, described in details in the paper. (Color figure online)

In order to check if DCGAN-based generators can represent more complex datasets, we train it on LSUN in batch mode, one generator per class. We then train two classifiers: first one on the original data that was used to train the generators, and second one on purely generated data produced by pretrained DCGANs. We then test obtained classifiers on validation set consisting of previously unseen real images and compare the classification accuracies of both. From Fig. 1 we observe that using generated data to train a classifier results in a decrease in classification accuracy (MNIST: 99.14% → 97.16%; LSUN: 88.69% → 70.22%), especially for LSUN dataset. Nevertheless, we find this decrease an acceptable trade-off to be able to pass to completely online classification training scenario with no necessity to store historical data, especially taking into account the data complexity of LSUN dataset.

2.3 Online Learning on Data Streams

Simulating Data Stream. In this work we consider the case where data arrive continuously in stream. Let $E = \bigcup_{k=1}^{\infty} E_k$ be an environment emitting data continuously in time, where E_k represents the subset of E corresponding to class i. We will make an assumption that data is sampled with the examples of unique type and format (e.g. RGB images of same size, sound recordings of given length, etc.). We will consider that each class, when it appears in the stream, lasts for

some period of time and emits at least b samples. We then can say that we receive data from stream in the form of batches of size b, where each batch contains only the elements of one class. Since the datasets we use are static and our goal is to work with data streams, for our experiments we need to define the way data will arrive during training.

We start by assuming that the stream is divided into time intervals. Every interval contains at least two and at most M distinct data classes. Each time a new interval is started we remove several classes from the previous interval and add new classes from E, so that the new interval always contains at least one class from the previous one (to simulate environment continuity) and never exceeds M classes. Every class, when it appears in the stream, emits a random number of batches. The durations of each interval and sub-interval, corresponding to a given class, are taken randomly from corresponding predefined ranges.

Forming the Buffer. Let us also initialize a data buffer B of size $N \times b$, that will serve to collect data from batches in order to use it later to train a classifier. We will fill in the buffer until the number of batches of one of the classes reaches the buffers limit size. After that we complete buffer to have equal number of images for each class by generating samples from all the pretrained generators (Fig. 2, Forming the buffer), and send obtained data to train the classifier. We then empty the buffer and start filling it again.

Network Training. When starting the online training on stream we consider that we already have pretrained generative models for some of the classes from the dataset, as well as the pretrained classifier for those data classes. Each time a new class appears in the stream we initialize a new GAN for it. We train the GANs with batches of corresponding classes directly when they appear in stream. The classification network is trained each time the data buffer is complete and its performance is evaluated on the test set at the end of each stream interval. Figure 2 shows the full schematic representation of the proposed framework.

3 Experimental Results

3.1 Datasets and Data Preparation

To test the hypothesis proposed in this paper we perform our evaluations first on the MNIST dataset, which is usually used as a baseline dataset in many ML-based studies in image analysis, and then on the LSUN dataset to check its performance on more complex data.

In every experimental setup, independently from the dataset, we train one generator per data class.

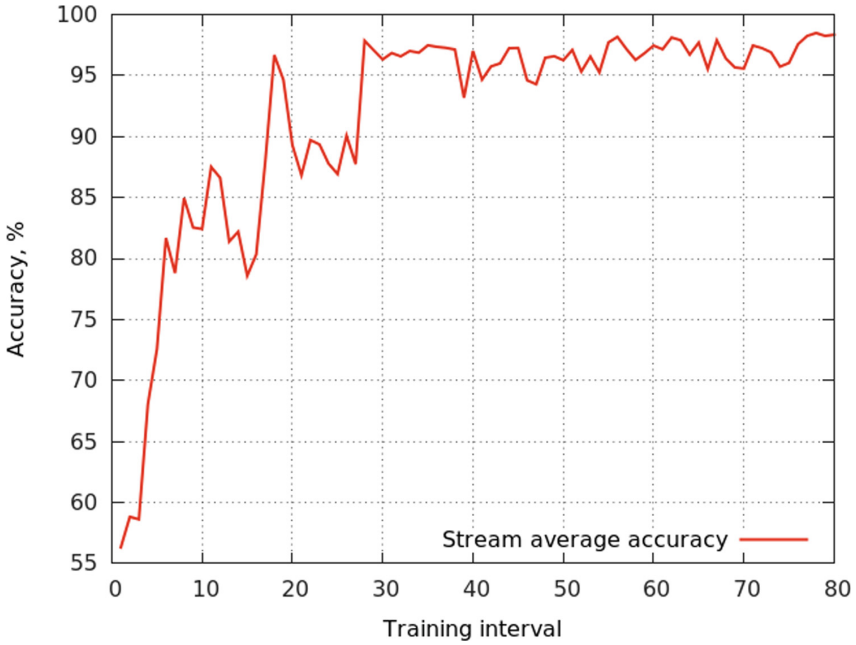

Fig. 3. Classification accuracy during online stream training for MNIST dataset

MNIST is a collection of gray-scale images of hand-written digits of 28×28 pixels each. The images include 10 data classes, each corresponding to a separate number from 0 to 9. The training set includes 6000 images per class and the test set 1000 images per class. No spatial transformation was applied on MNIST for either classification or GAN training. The classification network we use on MNIST consists of two convolutional with max pooling layers (with resp. 16 and 32 feature maps using 4×4 kernels), followed by three fully connected layers (512×512, 512×128 and 128×10) with ReLU activation function except for the output layer.

LSUN is a collection of RGB images of size at least 256×256 pixels each. The dataset includes 10 classes of scenes (bedroom, bridge, church outdoor, classroom, conference room, dining room, kitchen, living room, restaurant and tower), with the smallest class containing around 126k images and the biggest one over 3 millions of images. We extracted 5k images from each class to use them as a validation set for classification, the rest of the images were used to form the stream and train both generative models and classifier.

Every image is transformed to square shape by cutting its sides. DCGAN in its original formulation does not work on big size images, but works perfectly well on images of size 64×64 pixels and less. Also, since our goal was to simulate a data stream with only unique samples, we needed to perform some

data augmentation on the dataset. For these reasons, we rescaled LSUN images to the size of 96×96 pixels and randomly cropped them to 64×64 pixels each time they appear in stream.

Training state-of-the-art classifiers for large complex datasets usually requires very deep network architectures and takes a lot of time and resources. Getting a performance similar to the state-of-the-art batch learning scenario was not the intention of this work, so, we used a few shortcuts that allowed us to speed up training, to the expense of a slight decrease in classification accuracy. More precisely, we rescaled the original images, as well as the generated 64×64 pixels images, to a 224×224 size, and, passed them through the convolutional layers of the ResNet-200 network ([5]), a convolutional network pretrained on the ImageNet dataset. The latter was thus used as a feature extractor. Four fully-connected layers with ReLU activations (except for the output layer) were added on top of the feature extraction block to form the 10-class classification network ($2048 \times 1024 \rightarrow 1024 \times 512 \rightarrow 512 \times 128 \rightarrow 128 \times 10$).

3.2 Online Classification

We performed online classifier training on MNIST and LSUN datasets, which were streamed in the way described in Sect. 2.3.

In our online-learning-on-stream scenario, we achieved a maximum accuracy of 98.64% on MNIST (Fig. 3) and 77.59% for LSUN on 10 classes, which is comparable to the results of our batch experiments where the classifiers are trained on only generated data from pretrained DCGANs. Comparing the results of online classification on stream with batch classification allow us to quantify the loss of performance, to the best of our knowledge there exist very few works on online classification on evolving streams of complex data, and no established baseline for evaluation.

The online training on a stream is quite unstable for the LSUN dataset and classification accuracy varies a lot from one interval to the other as can be seen from the accuracy standard deviation that is plotted in light blue on Fig. 4. Still, performing mean filtering of classification accuracy over several successive intervals results in a curve that shows stable progression (the results for LSUN presented on Fig. 4 are averaged/mean-filtered over 80 successive training intervals). To find the reasons for the training instability in the stream scenario, we performed a test in a similar scenario with the only difference that the original images from the stream were used only to train generative models, while the classifier was trained with only generated data at the end of each interval (Fig. 5). The yellow curve on the diagram of Fig. 5 represents the average classification accuracy as a function of the average number of images ($\times 1000$) seen by each DCGAN. The red and blue lines represent respectively the average accuracy over pretrained classes and the average accuracy over the classes appearing at some point of the stream. We can see on Fig. 5 how the amount of images fed to the DCGAN improves the classification accuracy: each generative model received about 1M real images before it started to have a positive influence on classification training, while after 5M of images per class we do not see global

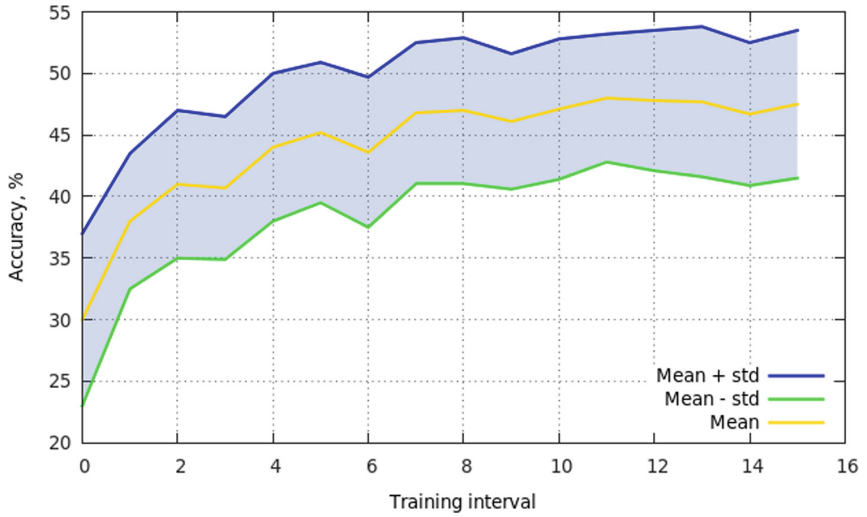

Fig. 4. Classification accuracy during online stream training for LSUN dataset. Each point on the graph corresponds to the average value over 80 training intervals (Color figure online)

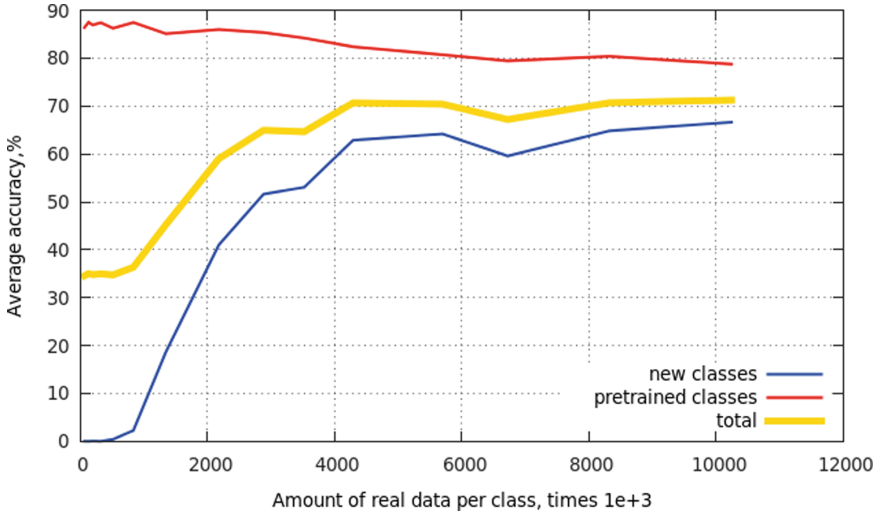

Fig. 5. Average classification accuracy during stream training when the classifier is trained only on generated data. The curves correspond respectively to the average performance over all classes (gold), the average performance over classes pretrained before the beginning of the stream (red), and, the average performance over classes introduced during the stream (blue) (Color figure online)

improvements in classification accuracy. The training in such a scenario appeared to be much more stable than when using a mixture of real and generated data to train the classifier.

We think that the training instability on the LSUN dataset is due to the complexity of the database (for the MNIST dataset, the training is rather stable), but, also, to multiple image rescaling steps rendered necessary by the incapacity of DCGANs to work directly with full-size 256×256 images. The classification accuracy during training might also be limited due to the fact that we use a pretrained network on ImageNet to perform the feature extraction step for the classifier, and that this feature extraction layer is not retrained in our stream scenario.

4 Conclusions

In this work we presented a new method for online classification on data streams. We defined streaming scenarios on the MNIST and LSUN datasets, and validated our online-learning-on-stream method on these datasets by showing that it is able to efficiently learn to classify complex image data from a time-evolving stream, with no need to store historical data. We also showed that DCGAN-based models are able to generate samples representative enough to replace the real data when training a classifier. Our online-learning-on-stream method showed on one side a strong capacity to adapt to unseen data classes appearing at different time of the stream, and, on the other side, did not lead to catastrophic forgetting of previously seen data.

The current approach requires quite a large amount of data per class which are not always available in the case of real data streams. We plan to tackle this problem in a future work by thinking of more reactive and efficient retraining procedures for the classification model (and eventually the generative models), able to retrain a network on a few data without forgetting of previously seen data.

References

1. Besedin, A., Blanchart, P., Crucianu, M., Ferecatu, M.: Evolutive deep models for online learning on data streams with no storage. In: 2nd ECML/PKDD 2017 Workshop on Large-Scale Learning from Data Streams in Evolving Environments (2017)
2. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
3. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
4. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. arXiv preprint arXiv:1703.06870 (2017)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

6. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
8. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: the sequential learning problem. Psychol. Learn. Motiv. **24**, 109–165 (1989)
9. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
10. Sermanet, P., Kavukcuoglu, K., Chintala, S., LeCun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3626–3633 (2013)
11. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
12. Webb, G.I., Hyde, R., Cao, H., Nguyen, H.L., Petitjean, F.: Characterizing concept drift. Data Min. Knowl. Discov. **30**(4), 964–994 (2016)