

# Evolutionary deep models for online learning on data streams with no storage

Andrey Besedin<sup>1</sup>, Pierre Blanchart<sup>1</sup>, Michel Crucianu<sup>2</sup>, and Marin Ferecatu<sup>2</sup>

<sup>1</sup> Laboratoire d'Analyse de Données et Intelligence des Systèmes,  
CEA Saclay, 91191 Gif-sur-Yvette Cedex, France

<sup>2</sup> Centre d'études et de recherche en informatique et communications,  
Le CNAM, 292 rue Saint-Martin, 75003 Paris, France  
andrey.besedin@cea.fr, pierre.blanchart@cea.fr,  
michel.crucianu@cnam.fr, marin.ferecatu@cnam.fr

**Abstract.** In recent years Deep Learning based methods gained a growing recognition in many applications and became the state-of-the-art approach in various fields of Machine Learning, such as Object Recognition, Scene Understanding, Natural Language processing and others. Nevertheless, most of the applications of Deep Learning use static datasets which do not change over time. This scenario does not respond well to a number of important recent applications (such as tendency analysis on social networks, video surveillance, sensor monitoring, etc.), especially when talking about online learning on data streams which require real-time adaptation to the content of the data. In this paper, we propose a model that is able to perform online data classification and can adapt to data classes, never seen by the model before, while preserving previously learned information. Our approach does not need to store and reuse previous observations, which is a big advantage for data-streams applications, since the dataset one wants to work with can potentially be of very large size. To make up for the absence of previous data, the proposed model uses a recently developed Generative Adversarial Network to drive a Deep Convolutional Network for the main classification task. More specifically, we propagate generative models instead of the data itself, to be able to regenerate the historical training data that we didn't keep. We test our proposition on the well known MNIST benchmark database, where our method achieves results close to the state of the art convolutional networks trained by using the full dataset. We also study the impact of dataset re-generation with GANs on the learning process.

**Keywords:** Deep Learning, Data Streams, Adaptive Learning, GAN

## 1 Introduction

Most of the recent research in Deep Learning community has been focused on the case of static datasets, where the training data is first acquired and then used to train a model [6]. However, nowadays there is a growing demand for real-time processing and analysis of continuously arriving huge amounts of data. One of the main challenges when dealing with online learning on data streams is that the dataset one wants to process

is potentially of a very large size, which raises the issue of storage and memory management during training. Another difficulty is that data should be processed in real time while new samples are still continuously arriving. The problem here is that most of machine learning algorithms converge to solution quite slowly and sometimes need several epochs of training over the whole dataset, which is impossible in the online learning setup. Moreover, not storing and reusing historical data usually results in the phenomenon of catastrophic forgetting [7]. An even bigger problem can be caused by concept drifts [11], more specifically by changes in data distribution in time, which may force the algorithm to over-fit on new data and discard useful information, learned from previous observations.

In this paper we mostly consider the problem of storing previous observations and adapting to changes in data content (e.g. new classes become available). To address the storage problem we make use of Generative Adversarial Networks (GANs) [3] which, in the past few years, acquired increased attention in the Machine Learning community due to their ability to learn data representations and generate synthetic samples that are almost indistinguishable from the original data.

Despite their popularity, GANs until now were almost not studied from the point of view of their ability to generalize outside the training set. In this paper we study the notions of generalizability and representativity of generative models and propose quantitative metrics to evaluate them. By generalizability of generative model we mean its capacity to focus on learning concepts and patterns and become a representation of the data distribution rather than reproducing data samples it encounters during training. The term representativity is used to describe the ability of generative models to represent the original dataset it was trained on with all its internal variability. We also investigate the ability of designed online classification system to adapt to the concept drift caused by incremental appearance of new classes of data during learning.

We validate our method on the MNIST database, often used for benchmarking. This choice was mainly guided by the need to have a well-known benchmark for comparison with both offline and online methods using deep neural networks. Our experiments show that recent generative approaches using GANs have excellent ability to generalize and discard the necessity of storing and reusing historical data to perform online training of deep classification models.

To summarize, the contribution of this work is twofold. First, we propose and evaluate a new network architecture that uses GANs to allow to train online classifiers without storing the incoming data while being able to adapt to new classes in the incoming data stream and at the same time to avoid catastrophic forgetting. Second, we justify the usage of GANs in proposed context and make a quantitative evaluation of how the replacement of real data by generated data influences the learning process.

The rest of the paper is organized as follows: In Sec. 2 we motivate our study and present previous results in online classification using deep neural networks and recent approaches for adaptive multi-task learning, in Sec. 3 we give the detailed presentation of our online classification approach and in Sec. 4 we demonstrate the validation results.

## 2 Related work

There are very few works in the literature that investigate the possibility to train Deep Neural Networks for online classification on data streams.

In [1] the authors address the problem of exploding storage demand in the online learning setup by using generative capacities of Deep Belief Networks (DBNs). In their approach, the authors train DBNs to generate samples similar to the original data, and then use those samples instead of the real data to train a classifier. The drawback of proposed method is the poor generative performance of DBNs on image datasets. It causes a big decrease in classification accuracy, compared to the offline setting with a static training dataset, and results in the performance being far beyond the current state-of-the-art on the MNIST dataset which they used to benchmark their method.

In a more recent work [9] the authors propose a method, Progressive Networks, designed for effective knowledge transfer across multiple sequentially arriving tasks. The evaluation of presented method is showed for the reinforcement learning problems, but the authors state its possible application to a wide range of Machine Learning challenges. In the proposed approach, a single deep neural network is initialized and trained for the first given task. Each time a new task appears, a new neural network with the same architecture as previous ones is added. At the same time, directed connections from all previous models to the current models are initialized and serve as knowledge transfer functions. During the back-propagation, those connections are updated together with the weights of the current model to adjust the influence of previous models on the current one. The pool of historical models stay unchanged and is only used for the forward pass. The big limitation of this method is that the size of parameter space increases rapidly when new tasks are added.

Another approach to deal with changing environment is proposed in [2]. Introduced model, Pathnet, is represented by a huge Neural Network with embedded agents, that are evolving to find the best matching paths through the network for a current task. After the task is learned and the new one is introduced, parameters of the sub-network containing the optimal pathway are "frozen" not to be modified by back-propagation when training for new tasks. PathNet can be seen as an evolutionary version of Progressive Networks, where the model learns its own architecture during training.

Both PathNet and Progressive Networks approaches showed good results for learning on sequences of tasks and can be considered as a good alternative to fine-tuning to accelerate learning. However, they don't provide a way to solve the problem of data storage for streams since every task for these algorithms should be fully described by its complete environment, which is not the case for data streams with only a part of all data classes available at each time point.

Unlike previously described methods, our model is able to learn incrementally on massive multi-class streaming data, is adaptable to changes in data distribution and has no need in excessive historical data storage.

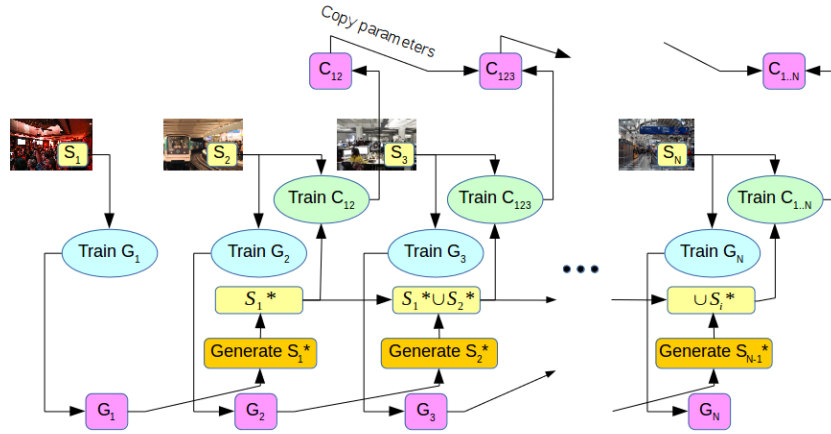


Fig. 1: Schematic representation of our online learning approach. Original data is presented to the model class by class. Each time new class of data appears we start training a new generator modeling that class. At the same time we train a classifier on the generated data from the previously learned classes and the original data from the new class that come from the stream.

### 3 Proposed approach

To represent the process of online learning we model the streams in a way that the data arrives continuously with distinct classes coming separately one by one. On fig. 1 we show proposed framework.

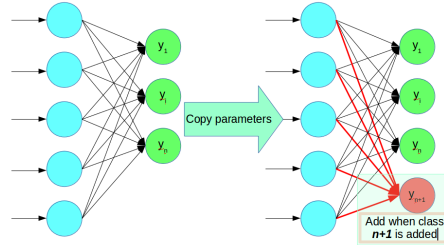


Fig. 2: Adding a node to the output layer and initializing the connections with the previous layer in the online learning scenario when new data class appears in the stream.

Let  $S = \{S_i | i = 1, \dots, N\}$  be the partition of real data into  $N$  distinct classes. In our learning framework we take the first coming class  $S_1$  from  $S$  and train a generator  $G_1$ , able to represent this data. We save  $G_1$  and discard  $S_1$ . Then we start training  $G_2$  on

the data from  $S_2$ , and in parallel train a classifier  $C_1^2$ , feeding it with samples from  $S_1^*$  – synthetic data, generated by  $G_1$ , and newly arriving real data from  $S_2$ . After that, data from  $S_2$  are discarded. We continue this procedure with all available classes from  $S$ , one by one, each time generating equal batches of data from all the previously trained generators. Each time a new class is added we also add a node to the output layer of the classifier and initialize its connections with the previous layer (fig. 2). The rest of the network weights are copied from the previous state. See algorithm 1 for pseudo-code.

**Require:**  $S = \bigcup_{i=1}^{\infty} S_i$  : data stream, with  $i$  - class number

**Require:**  $n$  : number of already learned classes

**Require:**  $G_i$  : generative model for class  $i$

**Require:**  $C_1^n$  : classification model for data from  $\bigcup_{i=1}^n S_i$

$G_1 \leftarrow$  initialize model

$n \leftarrow 1$

**while** are receiving samples from  $S$  **do**

$d \leftarrow$  get batch from  $S_j$ ,  $j$  - current class

**if**  $j = n + 1$  **then**

$n \leftarrow n + 1$

$G_n, C_1^n \leftarrow$  initialize models

**if**  $n > 2$  **then**

$C_1^n \leftarrow$  copy parameters from  $C_1^{n-1}$

**end if**

**end if**

$d^* \leftarrow \bigcup_{i=1..n}^{i \neq j} d_i^*$  generate synthetic data from  $\{G_i\}$

$C_1^n \leftarrow$  train with  $d \cup d^*$

$G_j \leftarrow$  train with  $d$

**end while**

**Algorithm 1:** Online learning model, proposed in Sec. 3

**Model architecture** All the experiments in this paper used a DCGAN [8] model for the generative network, both for the online and offline training settings. DCGANs follow the same training logic as GANs to the difference that DCGAN generator and discriminator networks are built from convolutional layers and have a set of topological constraints to ensure better convergence. Compared to the original GANs, DCGANs show higher stability during training and tend to produce more visually meaningful samples when trained on image datasets.

The classification model hyper-parameters were adjusted according to two criteria:

- the performance of the model should be comparable to the state-of-art
- the network should not be too deep to allow real-time training.

The model we retained consists of one convolutional layer followed by two fully-connected linear layers. Each layer except the last one is followed by a Rectified Linear

Unit activation function. Batch normalization [4] and dropout [10] are applied during training on all layers except the output layer. Stochastic Gradient Descent using Adam [5] is used to perform model parameters optimization. The classification network described in this paragraph is used for all experimental scenarios.

## 4 Experiments

To test proposed method we used the well-known MNIST<sup>3</sup> dataset of hand-written digits composed of gray-level images of 32x32 pixels, which is classically considered for static off-line approaches. It contains 60000 images in the train set and 10000 images in the test set. The dataset includes 10 classes corresponding to digits from 0 to 9. This database is widely used as a baseline in NN benchmarking and there already exist a lot of pre-trained convolutional models that provide state-of-the-art results on it.

The main goal of this section is to investigate the performance of our approach, more precisely to see how the data generated from DCGANs performs when used to train networks in an online scenario (Sec. 4.3). However before proceeding to that we first make sure that the enabling assumption is correct: that is, we make sure that trained generators are able to represent well the initial training data and generalize on the data distribution (Sec. 4.1 and Sec. 4.2).

The measure we use in this work to evaluate the fitness of the classifier is the classification accuracy, usually computed as the mean of the normalized diagonal values of the confusion matrix. This measure is well known and used in many research works (for example in [1]).

### 4.1 Generalizability of GANs

The ability to generalize on the whole data distribution when having only a small number of data samples available for training is one of the main characteristics that any learning system is expected to have. In the case of classification algorithms measuring the generalization capacities of a given model is very straightforward and can be evaluated by the difference between the classification accuracies on the training and validation sets.

Creating a generative model that would focus on learning concepts rather than memorizing single data samples is a problem of a very high importance, since it can be viewed as the machine's capacity to generalize to unseen instances (similar to what creativity and imagination is for human intelligence). In other words, we are more interested in creating a model that would be able to "imagine" objects, that are similar to those from data distribution, rather than just reproducing the data samples it has seen during training, especially in the online learning context where data distributions tend to change in time. This brings us to the question of defining and measuring the generalizability of the generative model. Unfortunately the measure of the model's capacity to generalize cannot be directly transferred from a classification model to a generative model. Instead, we propose a new notion of generalizability that captures much the same principles, but adapted to the generative case.

---

<sup>3</sup> <http://yann.lecun.com/exdb/mnist/>

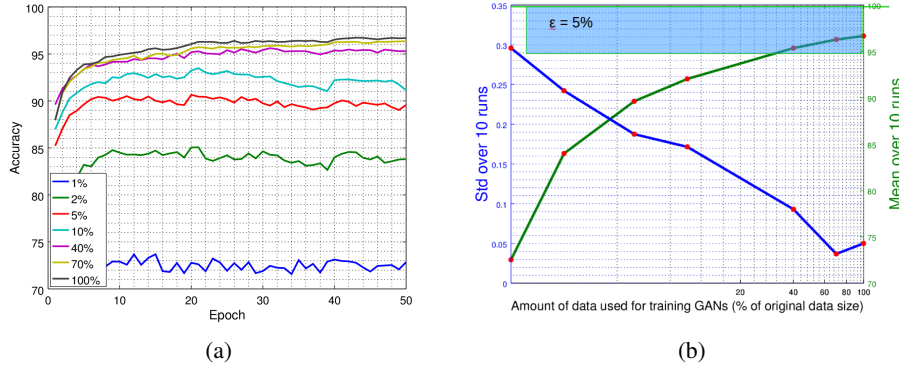


Fig. 3: Results of the generalizability test on MNIST (see Sec. 4.1). (a) Classification accuracy for different GANs support sizes as a function of training time. Average over 10 runs; (b) Mean/std of the classification accuracies for different GANs support sizes over 10 runs after 50 training epochs for the generalizability tests. Blue box represents the area in which the generalization error does not exceed 5%

We shall say that a generative model  $G$  trained on some support subset  $D^{supp}$  of a dataset  $D$ , with  $D^{val}$  being the validation set of  $D$  such that  $D^{val} \cap D^{supp} = \emptyset$ , generalizes well on  $D$  over some measure  $\mu$ , evaluating the semantical resemblance of samples from two datasets, if the similarity over  $\mu$  between  $D^{val}$  and  $D^{Gen}$  - the data sampled from  $G$ , approaches the similarity between  $D^{val}$  and  $D \setminus D^{val}$ . In a more formal way:

$$|\mu(D \setminus D^{val}, D^{val}) - \mu(D^{Gen}, D^{val})| < \epsilon,$$

where  $\epsilon$  is the parameter that determines the quality of generalization.

Choosing the metric to measure the semantical similarity between two datasets is not straightforward. In our study we decided to use a neural network based classification model for this purpose. Since neural networks are known and much appreciated for their ability to learn the abstractions and internal data representations, the mean classification accuracy of this model when trained on one dataset and tested on another one represents a desired property.

One of the main assumptions on the generalization capacities of any machine learning algorithm is that to improve it one often would want to get a bigger training set with more representative data samples. In our experimentations we adopt this idea and adapt it to generative models case.

In following experiment  $D$  is the MNIST dataset and  $G$  is the set of generative models  $G_1, \dots, G_{10}$  - one for each data class. To evaluate the ability of  $G$  to generalize on unseen content we designed a test that consists in varying the size of the set  $D^{supp}$  used to train generative models from 60 (1% of  $D$ ) to 6000 (100%) samples per class and comparing the mean classification accuracy of the classifier, trained on the data generated by  $G_i$ , to the one trained and tested on the original data.

Fig. 3a represents classification accuracy as a function of training time averaged over 10 runs of classifier training on generated data  $G$ , with  $G$  trained on the datasets of different sizes. We can see from the figure that the classification performance on validation set significantly improves when increasing the size of the support set to train the generative models. We observe that with only 1% support size the classification accuracy is pretty high (70-75%) and with a support size of 5% the value goes up to 90%. To quantify this effect, Fig. 3b shows the curve of the improvements through all the tested support size values and makes a link with the generalizability definition proposed earlier, with  $\epsilon = 5\%$  and  $\mu(D \setminus D^{val}, D^{val}) = 99.6\%$ . We can see from the figure that using only 40% of the initial dataset (2400 samples per class) allows us to obtain a highly generalizing generative model with the generalization error below 5% (the values in the blue box). We can also observe that the curve keeps going up which means that having more data for training the generative models should improve the final classification accuracy.

Fig. 4 shows random examples of synthetic samples, generated by DCGAN when trained on different amount of original images from MNIST dataset. We see that starting for 10% support size, the generated data is visually very consistent, confirming the results above.

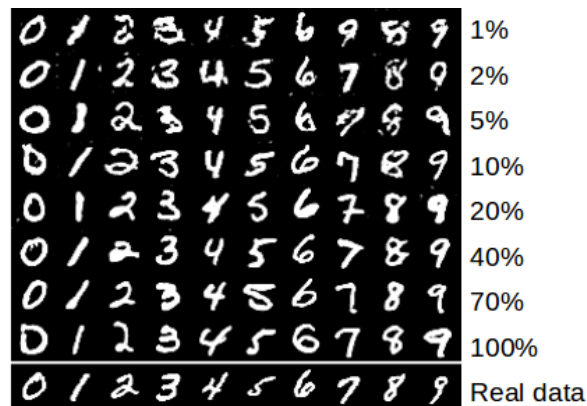


Fig. 4: Samples, produced by DCGAN-based generator, when using 1 to 100% of the original MNIST dataset to train it

## 4.2 Representativity of Generative Models

Another important question we needed to answer before passing to online scenario is how much data do we need to sample from pretrained generators in order to represent the full richness of the information, learned by generative models from the original dataset. This problem is essential since the amount of data we need to generate while training online classifiers influences directly the learning reactivity and it would be



reasonable to sample the smallest amount of data, that at the same time wouldn't affect too much the final classification accuracy.

Similar to the experiment, described in Sec. 4.1 we trained one generative model for each class in MNIST dataset and used the generated data to train a classifier with the difference that this time we used the full dataset as a support for generative models training.

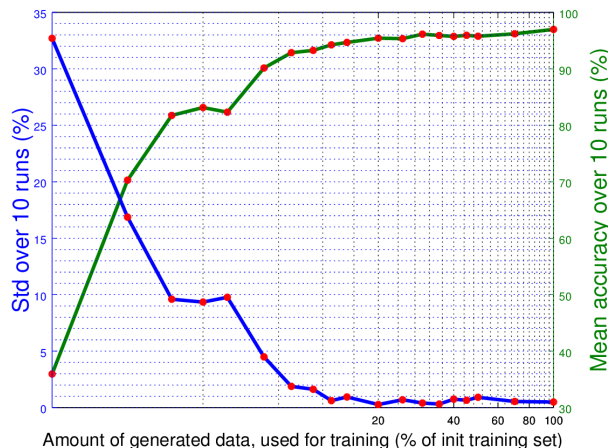


Fig. 5: Results of the representativity test on MNIST (Sec. 4.2). Mean/std of the classification accuracies for different amount of generated data used to train the classifier. Average over 10 runs after 50 training epochs

To verify the representative capacities of DCGAN, we studied the influence of the amount of generated data on the final training accuracy by varying its size from 1 to 100 % of the original dataset. Fig. 5 represents mean and standard deviation of the classification accuracy over 10 runs for each chosen size of generated dataset. We can see that for MNIST dataset, generating samples in the amount of only 30% of the original dataset is enough to reach almost the same training accuracy and stability, represented by low standard deviation, as for the case of generating 100% of the original dataset size.

Comparing to the training on original data, where with the classification model we use we obtain the accuracy of  $F^{orig} = 99.6\%$ , training on generated data reduces the maximum accuracy down to  $F^{gen} = 97.2\%$ . We can consider this decrease as the price we pay for not storing the data. Based on these two values we can introduce the metric to evaluate representativity of generative model:

$$R_M(D) = \frac{F_M^{gen}(D^{val})}{F_M^{orig}(D^{val})} = 0.976,$$

where  $M$  is the model we evaluate and  $D^{val}$  is the validation set - the subset of initial data that was not used to train the generative model. In these notations, the value of  $R_M(D)$  close to 1 represents the case of  $D$  being well represented by  $G$ , we would talk about bad representativity when  $R_M(D) \ll 1$  and  $R_M(D) > 1$  corresponds to the case where generative models not only work as the memory to store data representations, but also act as a filtering mechanism that extracts useful information from data samples.

### 4.3 Online classification using data regeneration

One of the possible limitations of our online classification method, described in Sec. 3, is that to avoid forgetting we need to continuously generate data from all the previously learned classes when receiving samples from new classes. The dependency between the amount of generated data and total number of classes in the dataset may become a problem for classification tasks with large number of classes. On the other hand, synthetic data in our model is used to ensure generalization and stability for learning process and should not be considered as the main source of information for the parameters update. Generating too much of synthetic data can thus reduce the importance that the model gives to original data we receive in streaming mode.

Similar to the tests on representativity of generated models, described in Sec. 4.2, we evaluate the performance of our algorithm depending on the amount of data we generate. The only difference is that in case of data streams we cannot know in advance the size of the dataset, neither the total number of classes.

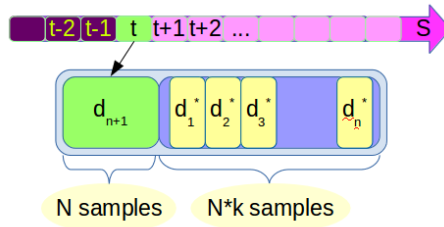


Fig. 6: Schematic representation of the way batches for online training are organized.  $N$  is the size of real data batch, coming from stream,  $n$  is the number of already learned classes.

To deal with the outlined remarks, we design our experiments in the following way. Each time we receive a batch of stream data of size  $N$ , we generate  $\frac{\min(n,k)*N}{n}$  data samples for each previously learned class, where  $k$  is a parameter, fixed in the beginning of each experiment, and  $n$  is the current number of learned classes, so that total volume of generated data is equal to  $\min(k, n) * N$  (Fig. 6). The size of generated data batch depends on number of classes we have already learned only when  $n \leq k$ . In each

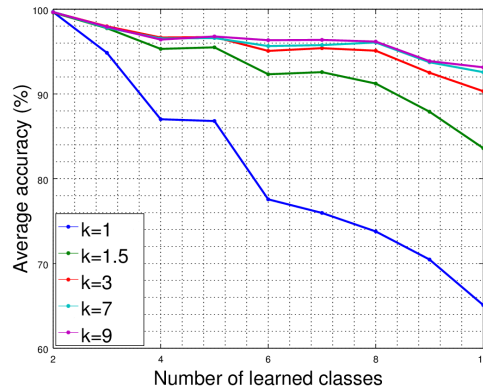


Fig. 7: Accuracy of our online-learning algorithm, described in Sec. 4.3 for different values of scaling parameter  $k$  for data regeneration

experiment, fixed value of parameter  $k$  in the range between 1 and 9 is taken. Value of 1 corresponds to the case where the total amount of generated data is equal to the size of received batch, while the value of 9 in the 10 classes classification problem represents the case where for each already learned data class we generate the amount of data, equal to the size of received data.

An important aspect to mention is that running the tests directly in streaming mode with just random initialization resulted in a poor performance. To overcome this problem, the classification network was bootstrapped by pre-training for several epochs on the first two data classes and then passing to online mode.

Fig. 7 represents the results of online classification. The graph shows the performance of proposed learning algorithm on the evolving dataset with incrementally added classes of data. Each line is an average over 50 independent runs, corresponding to one of the five tested values of  $k$ .

Our method achieves a classification accuracy above 90% in a completely online adaptive scenario starting from  $k = 3$ , which is close to the state-of-the-art performance in the offline learning setting. The performance increases for higher values of  $k$ , i.e., bigger sizes of generated data. As a comparison point, in a similar experimental online setting on the 10 classes MNIST dataset, [1] obtained only 60% accuracy. We can also see that with  $k \geq 3$  the accuracy decreases a little with every new added class, but complete forgetting never happens.

## 5 Conclusion and perspectives

In this work we developed a framework for online training of a Deep Neural Network classifier on data streams with no storage of historical data. The proposed model uses data regeneration with DCGANs to compensate for the absence of the historical data and avoid catastrophic forgetting in the online learning process on data stream.

We justify the choice of DCGAN generative models by showing that they have generalizability and representativity abilities: our experiments confirm that DCGAN-generated data can be used instead of the original data to train a classifier with good generalization properties. These properties allow us to train a classification model that obtains the accuracy above 90% in a completely online learning mode.

In a future part of this work, we will tackle the problem of designing more efficient training methods with less generated data to increase learning reactivity. Training both generative networks and classifier requires indeed having a sufficiently large amount of original data from each presented class, which is often not the case for the real-life applications with dynamic datasets. We will also measure the forgetting effect and find ways to limit its impact. We also plan to validate our online learning scheme on larger and more complex databases.

## References

1. R. Calandra, T. Raiko, M. Deisenroth, and F. Pouzols. Learning deep belief networks from non-stationary streams. *Artificial Neural Networks and Machine Learning–ICANN 2012*, pages 379–386, 2012. [3](#), [6](#), [11](#)
2. C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. [3](#)
3. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. [2](#)
4. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. [6](#)
5. D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. [6](#)
6. Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. [1](#)
7. M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989. [2](#)
8. A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. [5](#)
9. A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. [3](#)
10. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. [6](#)
11. G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016. [2](#)