Evolutive deep models for online learning on data streams with no storage

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

[1] Laboratoire d'Analyse de Données et Intelligence des Systèmes,
CEA Saclay, 91191 Gif-sur-Yvette Cedex, France
[2] 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 big number of recent applications (such as tendency analysis on social networks, video surveillance, sensor monitoring, etc.), especially when working with 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 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. An even bigger problem comes from changes in data distribution in time [10], 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 introduce the notions of generalizability and representativity of generative models and propose quantitative metrics to evaluate them. Here by generalizability of generative model we mean its capacity to focus not only on memorizing data samples it encounters during training, but on learning concepts and patterns and become a representation of the data distribution itself. The term representativity in its turn is used here to describe the ability of generative models to represent the initial dataset it was trained on with all its internal variability. We also investigate the ability of designed online classification system to adapt to concept drifts by incrementally adding 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. Second, we study the impact of dataset regeneration with GANs on the learning process.

The rest of the paper is organized as follows: In Sec. 2 we motivate our study and present related work, in Sec. 3 we describe our model and in Sec. 4 we present the validation results. We conclude the paper in Sec. 5 by an analysis of our main results and perspectives for further work.

2 Related work

Until now, there are very few studies that investigate the possibility to train Deep Neural Networks for classification in the online training scenario.

In [1] the authors address the problem of exploding storage demand in the online learning setup by using the generative capacities of Deep Belief Networks (DBNs). They train DBNs to generate samples from the training data distribution, and use those

at a later stage for retraining and classification. 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 [8] 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 proposed approach, a single deep neural network is initialized and trained for the first given task. Each time new task is added, new neural network with the same architecture as previous ones is initialized. 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 "freezed" 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 Progress 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.

3 Proposed approach

In our experiments we use image datasets that are classically considered for static offline approaches. On fig. 1 we present our framework for online learning. Let $S = \{S_i | i = 1, .., N\}$ be the distribution of real data, where each S_i represents a separate data class. 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_{12}, 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

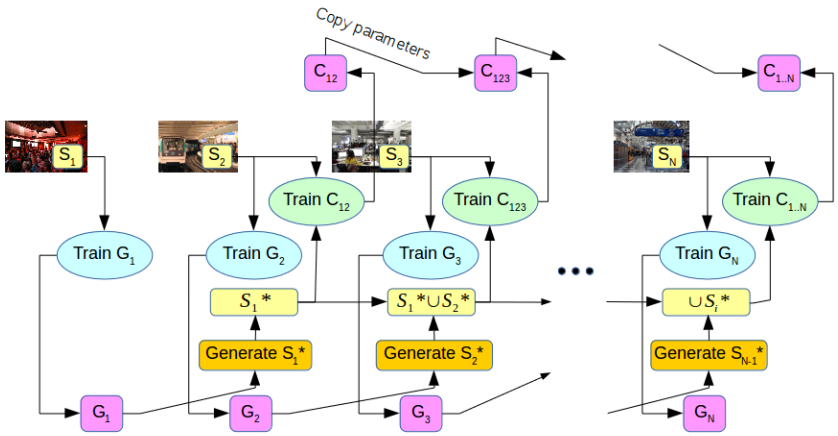


Fig. 1: Schematic representation of our online learning approach. Data from the original distribution is presented to the model class by class. Each time new class is presented we simultaneously train a generator, modeling its distribution, and a classifier for all the previously learned classes. See text for details.

classes from S, one by one, each time generating equal batches of data from all the previously learned generators. Each time a new class is added we also add a node to the output layer of the classifier and re-initialize its connections with the previous layer. The rest of the network weights are copied from the previous state. See algorithm 1 for pseudo-code.

To train generators, we use the DCGAN architecture [7], which follows the same training logic as GANs. The main difference of DCGANs is that both generator and discriminator networks are built from convolutional layers and have a set of topological constrains 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.

To choose an architecture for the classification model we followed two criteria: (1) the performance of the model should be comparable to the state-of-art and (2) it should be shallow enough to be trained in real-time. The model we retained consists of 1 convolutional layer followed by two fully-connected linear layers. Each layer except the last one is followed by a Rectified Linear Unit activation function and batch normalization is used [4]. Dropout [9] is applied during training on all layers except the last one. Adam [5] is used to perform model parameters optimization.

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 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 $\{G_i\}$
    $C_{1..n} \Leftarrow$ feed with $d \bigcup d^*$
    $G_j \Leftarrow$ feed with d
  end while

Algorithm 1: Online learning model, proposed in this paper. See Sec. 3 for details

4 Experiments

To test proposed method we used the well-known MNIST[3] dataset of hand-written digits under the form of gray-level images of 32x32 pixels. There are 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 for it.

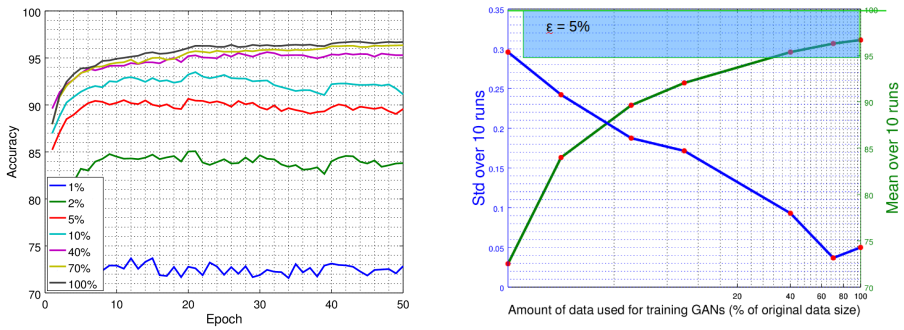
Before using generated data from DCGANs to train networks in online scenario we had to make sure that trained generators are able to represents well the initial training data and generalize on the data distribution.

4.1 Generalizability of GANs

The ability to generalize on the whole data distribution having only a small part of it 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 of the classification accuracy on training versus validation sets.

Creating a generative model that would be focused on learning concepts rather than memorizing single data samples is a problem of a very high importance, since it can be

[3] `http://yann.lecun.com/exdb/mnist/`



(a) (b)

Fig. 2: Results of the generalizability test on MNIST (see sec. 4.1). (a) Learning curves for different GANs support sizes, averaged 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 with $\epsilon = 5\%$

considered as the machine's creativity and imagination, which is a essential part of 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 generative model. Unfortunately the measure of the model's capacity to generalize cannot be directly transfered to generative models case from the classification task, as well as the notion of generalizability itself has to be adapted.

We will say that a generative model G trained on some support subset D^{supp} of data distribution D generalizes well on D over some measure M, evaluating the semantical resemblance of samples from two datasets, if

$$|M(D \backslash D^{val}, D^{val}) - M(D^{Gen}, D^{val})| < \epsilon,$$

where D^{Gen} is data, sampled from G, D^{val} is the validation subset of D, such that $D^{val} \cap D^{supp} = \emptyset$ and ϵ is the parameter, that determines the generalization quality.

Choosing the metric to measure the semantical likelihood 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 classification accuracy (mean of the diagonal values of the confusion matrix) of this model when trained on one dataset and tested on another one represents well desired properties.

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. Since the further experiments in the online scenario are

performed with the use of DCGAN architecture to train generative models, from here on we will consider only this example of generative model in our discussion. However, it is important to mention that all the experiments and theoretical parts of our study can be generalized to any type of generative models.

In following experiment D is the MNIST dataset and G is the set of generative models $G_1, \cdots, G_{10}$ - 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 amount of original data D^{supp} used to train generative models on D from 60 (1% of the original data) to 6000 (100%) samples per class and comparing the accuracy of the classifier, trained on the data generated by G_i, to the one trained and tested on the original data. The classification model we use for this and all following experiments is the shallow neural network described in sec. 3.

Fig. 2a represents averaged learning curves over 10 runs of classifier training on generated data, with G trained on the datasets of different sizes. We can see from the figure the confirmation of our hypothesis. The classification results on validation set significantly improve when increasing the support set for training generative models. Fig. 2b shows the curve of these improvements through all the tested support size values and makes a link with the generalizability definition proposed earlier, with $\epsilon = 5\%$ and $M(D \backslash D^{val}, D^{val}) = 99.6\%$. We can see from the figure that using only 40% (2400 samples per class) of the initial dataset allows us to obtain a highly generalizing generative model with the generalization error below 5%. We can also observe that the curve keeps going up and having more data for training the generative models would ideally improve the final classification accuracy.

Fig. 3 shows random examples of synthetic samples, generated by DCGAN when trained on different amount of MNIST dataset original images.

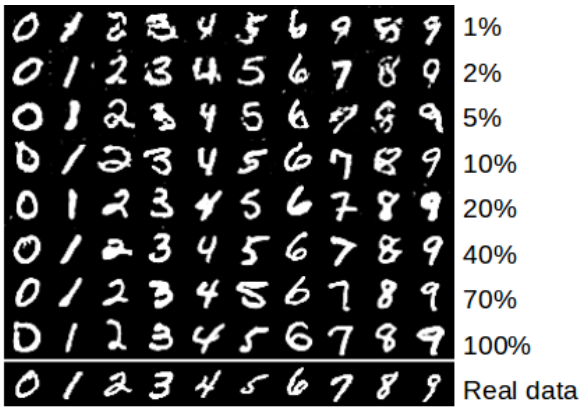


Fig. 3: Samples, produced by DCGAN-based generator, when using 1 to 100% of the original MNIST dataset to train it

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 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.

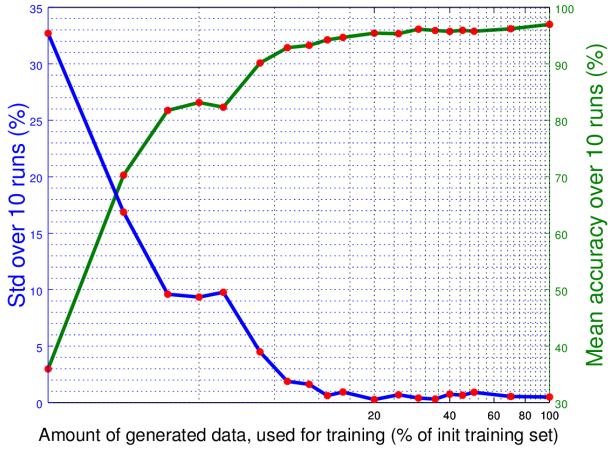


Fig. 4: Results of the representativity test on MNIST (see sec. 4.2). Mean/std of the classification accuracies for different DCGANs support sizes over 10 runs after 50 training epochs

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. 4 represents mean and standard deviation of the classification accuracy over 10 runs for each chosen size of generated dataset. We can clearly 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, as for the full-sized regeneration.

Comparing to the training on original data, where with the shallow network that we use for classification 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 $R_M(D) << 1$ 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. Having the dependency between the amount of generated data and total number of classes in the dataset can become a big problem for the classification tasks with hundreds and thousands of classes. From 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.

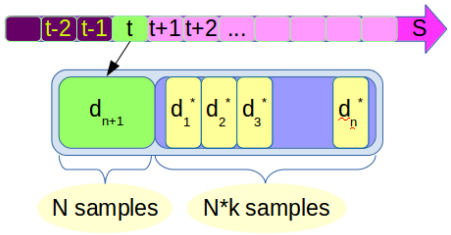


Fig. 5: Schematic representation of the way batches for online training are organized. N is the size of real data batch, coming from stream, n is the number of already learned classes.

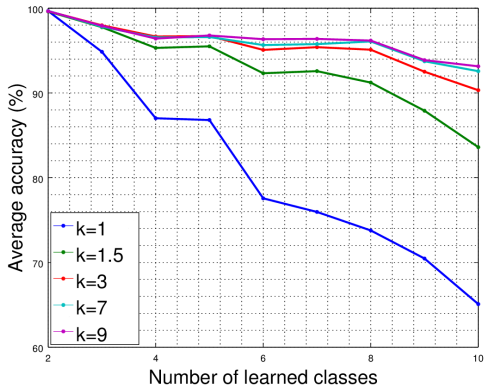


Fig. 6: Accuracy of our online-learning algorithm, described in sec. 4.3 for different values of scaling parameter k for data regeneration

To deal with the outlined remarks, we design our experiments in a 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. 5). The size of generated data batch depends on number of classes we have already learned only while $n \leq k$. In each experiment, fixed value of parameter k in the range between 1 and 9 is taken. Value of 1 here 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.

It is important to mention that in our experiments classification network was pre-trained for several epochs on first two data classes before passing to online mode. Running test directly in streaming mode with just random initialization resulted in a very poor performance for the whole duration of introduced streams.

Fig. 6 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 and corresponds to one of 5 different values of k. Our method achieves the above 90% accuracy in a completely online adaptive scenario already with $k = 3$ and keep growing. However, test on the dataset with more data classes are needed to confirm these results.

5 Conclusion and perspectives

In this work we developed a Deep Neural Network based method for online data classification, making use of generative models to avoid storage of the historical data, and proposed a quantitative way to evaluate the performance of latter. Despite the ability

of designed method to efficiently adapt to new data classes and deal with the problem of excessive storage for massive data streams, our model suffers from several limitations. First of all, the algorithm needs an important amount of data to be generated on each training step, which can become computationally expensive on datasets with a very big number of classes. From the other hand, training both generative networks and multi-class classifiers requires having a sufficiently big amount of original data from each presented class, which is not the case for most of the real-life applications with dynamic datasets.

To solve these problems, our future work will include the research on more efficient training methods with less generated data to increase learning reactivity. Finding possible solutions to control the way information is propagated through the model in order to give more importance to the new coming data and at the same time avoid forgetting is also among our main priorities.

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. 2
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. 4
5. D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4
6. Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1
7. A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 4
8. 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
9. 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. 4
10. 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