Data Augmentation On-the-fly and Active Learning in Data Stream Classification

Kleanthis Malialis[a, b], Dimitris Papatheodoulou[a], Stylianos Filippou[a],
Christos G. Panayiotou[a, b] and Marios M. Polycarpou[a, b]
[a] *KIOS Research and Innovation Center of Excellence*
[b] *Department of Electrical and Computer Engineering*
University of Cyprus, Nicosia, Cyprus
Email: {malialis.kleanthis, papatheodoulou.dimitris, filippou.stylianos, christosp, mpolycar}@ucy.ac.cy
ORCID: {0000-0003-3432-7434, 0000-0002-1922-1781, 0000-0001-8123-8309, 0000-0002-6476-9025, 0000-0001-6495-9171}

***Abstract*—There is an emerging need for predictive models to be trained on-the-fly, since in numerous machine learning applications data are arriving in an online fashion. A critical challenge encountered is that of limited availability of ground truth information (e.g., labels in classification tasks) as new data are observed one-by-one online, while another significant challenge is that of class imbalance. This work introduces the novel Augmented Queues method, which addresses the dual-problem by combining in a synergistic manner online active learning, data augmentation, and a multi-queue memory to maintain separate and balanced queues for each class. We perform an extensive experimental study using image and time-series augmentations, in which we examine the roles of the active learning budget, memory size, imbalance level, and neural network type. We demonstrate two major advantages of Augmented Queues. First, it does not reserve additional memory space as the generation of synthetic data occurs only at training times. Second, learning models have access to more labelled data without the need to increase the active learning budget and / or the original memory size. Learning on-the-fly poses major challenges which, typically, hinder the deployment of learning models. Augmented Queues significantly improves the performance in terms of learning quality and speed. Our code is made publicly available.**

***Index Terms*—incremental learning, active learning, data streams, class imbalance, neural networks.**

I. Introduction

Nowadays, in numerous applications information is becoming available in an online or streaming fashion. Applications include monitoring of critical infrastructure systems (e.g., leakage detection in water distribution networks [1]), security (e.g., spam filtering [2]), environmental monitoring [3], and recommender systems [3]. Deploying online learning algorithms in real-world applications to train predicting models on-the-fly is impeded by a series of open challenges and problems.

A key challenge is the label availability for a classification task as data are arriving online. Acquiring the labels can be expensive or even impossible in some real-time tasks. Class imbalance is another challenge, which refers to the problem of having a skewed data distribution [2]. Imbalance may render a traditional learning model ineffective as its predictive power on minority class examples declines significantly.

To cope with learning from limited-labelled data, we focus on active learning, a paradigm in which the model queries an oracle (typically, a human expert) for the ground truth information of selected examples [4]. Active learning is part of several successful industrial systems, such as, Google's method for labelling malicious advertisements [5], and NVIDIA's [6] and Tesla's [7] methods for their autonomous vehicles.

A recently proposed method called ActiQ uses online active learning in synergy with a multi-queue memory [8]. It was shown to address the aforementioned challenges, however, it typically performs well when the queue lengths are sufficiently large. Overall, the ActiQ method has the following limitations, which we try to address in this paper: (i) In online environments where data are sampled from a long, potentially infinite, sequence, it is impractical to assume that previous examples whose label had been requested, will always be available; (ii) To initialise the multi-queue system, it requires a large amount of historical labelled data; (iii) Its memory requirements are high. As a result, serious deployment concerns are raised. This work makes the following key contributions.

1) We propose the Augmented Queues method which significantly extends ActiQ by incorporating on-the-fly augmentation in synergy with active learning. It overcomes ActiQ's limitations (listed above), and we show its applicability using image and time-series augmentations.
2) We perform an extensive study in which we include two active learning methods, image and time-series datasets, two types of neural networks (standard, VGG). We also examine the roles of the active learning budget, memory size, and imbalance level. Models using Augmented Queues have access to more labelled data without the need to increase the budget and / or the original memory size. Augmented Queues significantly improves the learning quality and speed. Our code is made available[1].

This work has been supported by the European Research Council (ERC) under grant agreement No 951424 (Water-Futures), by the European Union's Horizon 2020 research and innovation programme under grant agreements No 883484 (PathoCERT) and No 739551 (TEAMING KIOS CoE), and from the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

[1] https://github.com/kmalialis/augmented_queues

The paper is organised as follows. Section II provides the background material. Related work is presented in Section III. Augmented Queues is described in Section IV. The experimental setup and results are presented in Sections V and VI respectively. We conclude in Section VII.

II. Preliminaries

Online learning considers a data generating process that provides at each time t a sequence of examples $S = \{(x^t, y^t)\}_{t=1}^{T}$, where the number of steps is denoted by $T \in [1, \infty)$ and data are typically sampled from a long, potentially infinite, sequence [3]. The examples are drawn from an unknown probability distribution $p^t(x, y)$, where $x^t \in \mathbb{R}^d$ is a d-dimensional vector in the input space $X \subset \mathbb{R}^d$, $y^t \in \{1, ..., K\}$ is the class label in the target space $Y \subset \mathbb{Z}^+$, and $K \geq 2$ is the number of classes.

An online classifier receives a new instance x^t at time t and makes a prediction $\hat{y}^t$ based on a concept $h : X \rightarrow Y$. In **online supervised** learning, the classifier receives the true label y^t, its performance is evaluated using a loss function and is then trained based on the loss incurred. The process is repeated at each step. In online applications, however, the label cannot be typically provided, as it's either impossible (e.g. in real-time applications) or expensive and thus impractical.

To address this issue, an alternative paradigm is active learning [4], which deals with strategies to selectively query for labels from a human expert according to a pre-defined "budget" $B \in [0, 1]$, for example, $B = 0.3$ means that 30% of the arriving instances can be labelled. A budget spending mechanism must ensure that the labelling spending $b \in [0, 1]$ does not exceed the allocated budget.

In **online active** learning [9], a classifier is built that receives a new instance x^t at time t. At each time step the classifier calculates the prediction probability $\hat{p}(y|x^t)$. The classifier outputs the best prediction probability $h(x^t) = \max_y \hat{p}(y|x^t)$ and the predicted class $\hat{y}^t = \operatorname{argmax}_y \hat{p}(y|x^t)$. A given active learning strategy $\alpha : X \rightarrow \{False, True\}$ decides if the true label y^t is required, which is assumed that the oracle will provide. The classifier is evaluated using a loss function and is then trained based on the loss incurred. Typically, training occurs only when $\alpha(x^t) = True$.

One major challenge encountered in some streaming applications is the presence of infrequent events, also known as **class imbalance** [2], [10]. It occurs when at least one class is under-represented, thus constituting a minority class. In binary classification, imbalance is defined as follows:

$$\exists y_0, y_1 \in Y \quad p^t(y = y_0) \gg p^t(y = y_1), \tag{1}$$

where y_1 represents the minority class.

III. Related Work

Online supervised learning methods that address imbalance are grouped as [2] (i) resampling methods, e.g., Oversampling-based Online Bagging (OOB) [11], Adaptive REBAlancing (AREBA) [12], [13], and Hybrid-AREBA (HAREBA) [14]; and (ii) cost-sensitive learning methods, e.g., CSOGD [15]. Despite their effectiveness, they rely on continual supervision. This work focuses on online active learning, and augmentation.

A. Online active learning

1) Querying strategies: The most widely used active learning strategy is uncertainty sampling, where the learner queries the most uncertain instances, which are typically found near the decision boundary [16]. Most existing strategies assume that the training set $U \subset X$ is already available (offline active learning) [17]. One way to measure uncertainty [4] is to first find the instance x_q with the least confident best prediction:

$$x_q = \underset{x \in U}{\operatorname{argmin}}\, h(x) \tag{2}$$

where $h(x) = \max_y \hat{p}(y|x)$ and request its label if it satisfies the following condition:

$$h(x_q) < \theta, \tag{3}$$

where θ is a threshold which is typically fixed. Work on online active learning is limited. The arriving x^t is queried if:

$$h(x^t) < \theta, \tag{4}$$

where $h(x^t) = \max_y \hat{p}(y|x^t)$ and θ is a fixed threshold. This is called a *fixed uncertainty sampling* strategy [9].

This strategy may not perform well if the threshold is set incorrectly, or if the classifier learns enough so that the uncertainty remains above the fixed threshold most of the time. In [9] a variable uncertainty sampling strategy is proposed, which uses randomisation to ensure that the probability of labelling any instance remains above zero. This is termed *randomised variable uncertainty sampling (RVUS)* strategy and the threshold is modified as follows:

$$\theta = \begin{cases} \theta(1 - s) & \text{if } h(x^t) < \theta_{rdm} \text{ \# request label} \\ \theta(1 + s) & \text{if } h(x^t) \geq \theta_{rdm} \text{ \# don't request} \end{cases} \tag{5}$$

where s is a step size parameter, $\theta_{rdm} = \theta * \eta$ where η follows a Normal distribution $\eta \sim N(1, \delta)$ with a standard deviation of δ.

A recently proposed method called ActiQ [8], has achieved state-of-the-art results in imbalanced scenarios by combining the RVUS strategy with a multi-queue data storage. In this work we propose the Augmented Queues method, which significantly extends ActiQ to allow data augmentation.

For a comprehensive review the interested reader is directed towards [4]. Lastly, we adopt the widely-used budget spending mechanism from [9]. Due to space constraints we direct the interested reader to [9], and to our released code.

B. Data augmentation

Augmentation is applied to a dataset to expand its size by artificially creating variations of the data [18]. It enhances the diversity of the dataset which could improve the learning performance, and improve generalisation.

1) Image augmentation: Such techniques include:

Geometric techniques [19] alter the pixel positions of images; some examples include scaling, cropping, flipping, padding, rotation, and image translation.

Colour-space augmentations [18] alter the colour properties of images by changing their pixel values, such as, changes to brightness, contrast, saturation, and hue of the images.

Random erasing [20] helps to prevent overfitting as it forces the model to learn more descriptive features by cutting-off random patches from the images. Patches can be masked with pixel values of 0s, 255s or mean pixel values, nonetheless, it was shown that random noise masking yields the best results.

2) Time-series augmentation: Such techniques include:

Window Slicing is a time domain transformation method, for extracting slices from a time series, and assigning the same class y as the time series used for slice extraction [21].

Time Warping [22] is a time domain transformation method, which aims to disrupt a pattern in the temporal domain either by using a randomly located fixed window [21] or by using a smooth warping path.

IV. Augmented Queues

The overview of the Augmented Queues method is shown in Fig 1. At any time t, the classifier observes an arriving instance x^t, and then provides a prediction $\hat{y}^t$; this is shown in yellow colour. If the active learning strategy does not request the ground truth, i.e., the class label, no training is performed and the algorithm waits for the next arriving example. If the strategy requests the ground truth, this is provided by an oracle (typically, a human expert) as shown in green colour. The example (x^t, y^t) is then appended to the relevant queue in Q^t. Before training, the data augmentation process is initiated to create the augmented queues A^t. The neural network is then trained using both Q^t and A^t. This is shown in orange colour. We describe below each individual element, as well as a discussion on the computational aspects of the method.

A. Individual elements

Multi-queue memory: The method uses multiple first-in-first-out (FIFO) queues which will be populated by instances queried by the active learning strategy. At any time t we maintain a set of K queues, one for each class as follows:

$$Q^t = \{q_1^t, q_2^t, ..., q_K^t\} = \{q_c^t\}_{c=1}^K, \tag{6}$$

where $K \geq 2$ is the number of classes. All queues are of the same capacity M and a queue corresponding to class c is defined as follows:

$$q_c^t = \{x_{c,1}, x_{c,2}, ..., x_{c,M}\} = \{x_{c,i}\}_{i=1}^M, \tag{7}$$

where $x_{c,i} \in \mathbb{R}^d$, and for any two $x_{c,i}, x_{c,j} \in q_c^t$ such that $j > i$, $x_{c,j}$ has been observed more recently in time.

As in the original *ActiQ* work, we assume the initial availability of M labelled examples per class. Balanced queues ensure robustness to class imbalance; the assumption is not needed for problems in which imbalance does not exist, as the queues will be populated at a similar rate. As this may be difficult to have in practise, the purpose of this work is to restrict M to a very small number e.g. up to ten, and then apply data augmentation. We argue that for the vast majority of applications this is realistic and practical.

Augmented memory: Let $F = \{f_o\}_{o=1}^{|F|}$ be a set of $|F|$ available transformation functions $f_o : X \to X$, such that, an original example x is augmented to $x^* = f_o(x)$. The number of transformations $|F|$ is task-dependent, for instance, like the ones described in Section III-B. Augmentation is initiated every time before the model is trained, that is, when the active learning strategy receives a class label. Let's define the augmented multi-queue memory as follows:

$$A^t = \{a_1^t, a_2^t, ..., a_K^t\} = \{a_c^t\}_{c=1}^K, \tag{8}$$

where $K \geq 2$ is the number of classes.

Each augmented queue $a_c^t \in \mathbb{R}^{M \times N}$ has capacity $|a_c^t| = M \times N$ where c is the class and N is the number of augmentation transformations per example. It is defined as:

$$a_c^t = \{x_{c,i}^* \in \mathbb{R}^N | \forall i \in [1, M]\} \tag{9}$$

where $x_{c,i}^*$ contains all N augmented examples generated from the original example $x_{c,i}$ as follows:

$$x_{c,i}^* = \{x_{c,i,j}^* = f_o(x_{c,i})\}_{j=1}^N \tag{10}$$

where each time the transformation function is selected randomly from the set of functions $f_o \in_R F$.

Class prediction: The neural network predicts the class of each arriving instance x^t as shown below. The multi-queue memory is not used in the prediction process.

$$\hat{y}^t = \underset{y \in \{1,...,K\}}{\operatorname{argmax}} \ \hat{p}(y|x^t) \tag{11}$$

Active learning strategy: The proposed Augmented Queues method, like ActiQ, uses the RVUS [9] active learning strategy as shown in Eq. (5).

Incremental learning: Recall that the neural network is trained only when the active learning strategy receives the class label. Prior the training at time t, the data augmentation process is initiated to create the augmented queues A^t from the original queues Q^t. We then merge the two queues as follows: $Q^{*t} = Q^t \cup A^t$. The output of the neural network is provided to a softmax output unit and its cost function is:

$$J^t = \frac{1}{|Q^{*t}|} \sum_{x_i \in Q^{*t}} l(y_i, \hat{P}_{x_i}), \tag{12}$$

where $y_i \in \{1, ..., K\}$ is the ground truth and $\hat{P}_{x_i} = \{\hat{p}(1|x_i), ..., \hat{p}(K|x_i)\}$ are the prediction probabilities for each class. The loss function used is the cross entropy $l(y, \hat{P}_x) = -\sum_{c=1}^K \mathbb{I}_{y=c} \ \log \ \hat{p}(c|x)$, where $\mathbb{I}_{condition}$ is the identity function that returns 1 if the condition is satisfied. The neural network h will be updated incrementally based on the cost incurred, that is, $h^t = h^{t-1}.train(J^t)$. The pseudocode of the Augmented Queues method is provided in Algorithm 1.

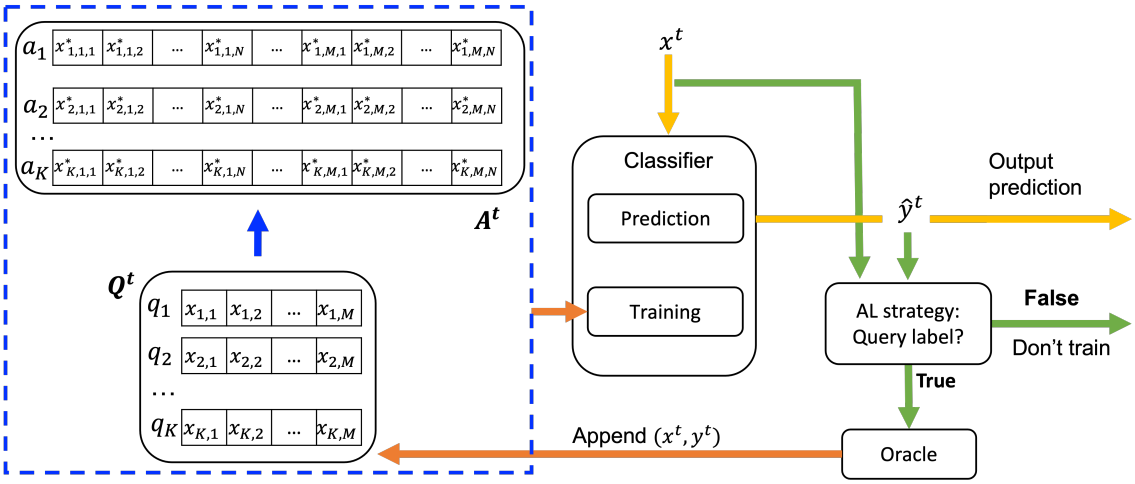


Fig. 1: The overview of the proposed method, Augmented Queues

Algorithm 1 Augmented Queues

Input:
a: active learning strategy
B: labelling budget
K: number of classes
M: memory / queue size
D: initial labelled examples ▷ $|D| = K \times M$
Initialisation:
init queues $Q^0 = FIFOs(capacity = M, init = D)$
create model h^0
init budget expenses $b^0 = 0$
Main:
for each time step $t \in [1, \infty)$ **do**
  receive instance $x^t \in \mathbb{R}^d$
  predict class $\hat{y}^t$ using Eq. (11)
  $Q^t = Q^{t-1}$
  $h^t = h^{t-1}$
  if $b^{t-1} < B$ **then** ▷ expenses within budget
    if $a(x^t, h(x^t)) == True$ **then** ▷ AL Eq. (5)
      receive true label y^t
      append example $Q^t = Q^{t-1}.append(x^t, y^t)$
      create augmented queues A^t using Eq. (8)
      calculate cost J^t using Eq. (12)
      incremental training $h^t = h^{t-1}.train(J^t)$
  update budget expenses b^t

Augmented Queues is robust to imbalance due to the *separate* and *balanced* queues per class. Propagating previously observed examples in the most recent training set is a form of oversampling. The augmentation is performed in such as way that A^t also contains separate and balanced queues per class.

B. Computational aspects

In online or streaming environments where data are arriving from a long, potentially infinite, sequence of data, it is unrealistic to expect that all previously observed examples will always be available during learning. Therefore, a learning model should use no more than a fixed amount of memory for any storage [23]. The multi-queue memory Q^t has a fixed size of $Q^t = K \times M$. As discussed, the queue length is kept to a minimum (e.g., $M = 10$) while the number of classes K is task-dependent. The augmented memory A^t is fixed as well with size $|A^t| = N \times |Q^t|$, where N is the number of augmentation transformations per example. Importantly, it reserves memory only at training times.

Predicting the class of an arriving instance x^t only requires a forward propagation pass of the neural network. Recall that the multi-queue memory is not used in the prediction process.

Training is only performed at certain times according to the budget, that is, the model is updated every time a class label has been received. Following the recommendation by [8] to avoid overfitting, the model is updated once at each training time, i.e., the number of epochs is set to 1. This is a parameter of the optimiser (e.g., gradient descent) that corresponds to a one-pass over the entire batch (or each mini-batch) of the data.

V. Experimental Setup

A. Datasets

MNIST [24]: This widely used dataset is a collection of images of handwritten digits. Each image has a 1-colour channel (monochrome), and it depicts a digit between "0" to "9" (10 classes), which is centred in a 28×28 pixel-sized box. MNIST, typically, serves as a benchmark dataset for image classification, however, it is important to note that it can stress test online / streaming learning algorithms due to its high-dimensionality of 784 features. To examine the effect of imbalance, we create three variations of MNIST and focus on the challenging case of multi-minority scenarios [25]:

- MNIST-balanced: It refers to balanced scenarios with 5000 arriving examples per class.
- MNIST-imbalanced10: It refers to scenarios with 10% imbalance. Digit "0" is the majority class from which

5000 examples arrive. The rest are minority classes, with 500 arriving examples per class.

- MNIST-imbalanced1: It refers to scenarios with 1% imbalance, with 50 arriving examples per minority class.

Two Patterns [26]: A simulated time series dataset in which each class represents the presence of two patterns in a definite order, that describes upward and downward steps defined by time functions presented in [26]. There are 4 classes and 5000 samples as follows: down-down class with 1306 cases, up-down with 1248 cases, down-up with 1245 cases and up-up with 1201 cases. The number of features is 128.

uWave Gesture Library Z [27]: A time series dataset for a set of eight simple gestures generated from accelerometers using the Wii remote. The data consists of the Z coordinates of each motion. There are 8 classes with total 3582 samples, each with 315 features.

Each augmentation method' values are found in our code.

B. Methods

The active learning methods used are:

RVUS [9]: The seminal work which introduced the randomised variable uncertainty sampling strategy shown in Eq. (5). It uses a neural network (described below), and it is a one-pass learner as it does not use any memory. When used with data augmentation, it is applied to the most recent example for which the oracle provided its label.

ActiQ [8]: A state-of-the-art method which uses the RVUS strategy and a multi-queue memory. In its original form, no data augmentation is used.

Augmented Queues: The proposed method shown in Fig. 1. We will refer to it as ActiQ with data augmentation, which is applied to the memory elements as described in Section IV.

The neural networks used with the AL methods are:

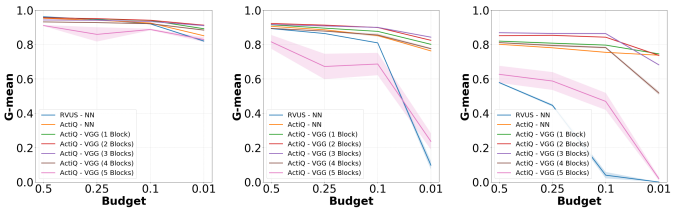
Standard Neural Network (NN) [28]: It is a standard fully-connected feed-forward neural network, trained using back-propagation. The hyper-parameters of NN for the MNIST, Two Patterns and uWave Gesture Library Z datasets are in our publicly available code.

VGG [29]: It is a convolutional network, distinguished by its simplicity, as it consists of a series of VGG-blocks which are a sequence of convolutional layers with padding, a non-linear activation function, and a pooling component. We focus on VGG-16 which is often used in practical applications due to its effectiveness and simplicity. It is composed of 5 blocks with a total of 13 convolutional layers and 3 fully-connected layers; its total number of parameters exceeds 100 million. The hyper-parameters of VGG are in our publicly available code. Note that the VGG will start learning from scratch, i.e., no pre-training on the ImageNet dataset will be performed.

C. Performance metrics and Evaluation method

A popular metric which is insensitive to class imbalance [10] is the geometric mean, defined as [30]:

$$G\text{-}mean = \sqrt[K]{\prod_{c=1}^{K} R_c}, \tag{13}$$



(a) Class balance (b) Imbalance 10% (c) Imbalance 1%

Fig. 2: The role of the budget and the model depth on the final performance ($t = 50000$, $t = 9500$, $t = 5450$) in MNIST

where K is the number of classes, and $R_c = N_{cc}/N_c$ is the recall of class c where N_{cc} is the number of examples correctly classified, and N_c is the total number of examples for this class. To compare learning methods in a sequential setting, we use the widely adopted *prequential evaluation with fading factors* method. The fading factor is set to $\xi = 0.99$. In all simulation experiments we plot the prequential $G\text{-}mean$ in every time step averaged over 20 repetitions, including the error bars displaying the standard error around the mean.

VI. Experimental Results

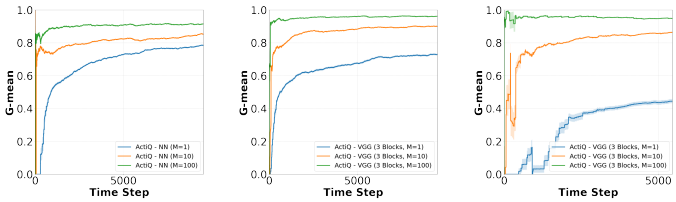
A. Role of the budget and the model depth

These experiments examine the behaviour of NN and VGG (1-5 blocks) under various budgets. The models have a queue length of 10. Figure 2a shows the performance on MNIST-balanced, whereas Figures 2b and 2c depict the performance on MNIST with 10% and 1% imbalance respectively. For each dataset, we examine 4 different budgets, 0.5, 0.25, 0.1, and 0.01, and we present the results on the final performance.

In Fig. 2a, the models achieve their peak performance on budget 0.5. A similar performance is achieved on smaller budgets, e.g., 0.25, 0.1, and even 0.01. The highest performing models are the VGGs with 2 and 3 blocks, which achieve almost identical performance. The VGGs with 1 and 4 blocks perform similar to the NN, with the exception of budget 0.01 where they achieve a higher score. While RVUS-NN and ActiQ-NN have similar performances, we notice a performance drop for the former on budget 0.01. Interestingly, the worst performing models for budgets 0.5, 0.25 and 0.1 is the VGG (5 blocks), while for budget 0.01 is the RVUS NN.

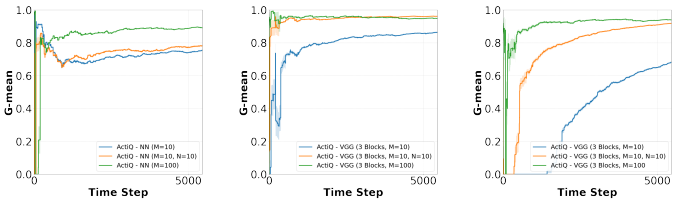
In Fig. 2b, similarly with the balanced dataset, the models achieve the best results when the budget is 0.5. The effect of the budget is slightly more prominent, especially on budget 0.01. The VGGs with 2 and 3 blocks achieve the best performance overall, while the latter achieves the best performance on the lowest budget. Most VGG networks outperform ActiQ-NN, with the exception of VGG with 5 blocks and RVUS-NN. Regarding RVUS-NN, the model outperforms the VGG with 5 blocks on budgets 0.5, 0.25 and 0.1, but on budget 0.01 we notice a significant drop of for RVUS-NN. This establishes the RVUS-NN as the worst performing model on budget 0.01.

In Fig. 2c, the best performing model is the VGG with 3 blocks for most budgets, followed by VGG with 2 blocks. However, on budget 0.01, the VGG (3 blocks) performs



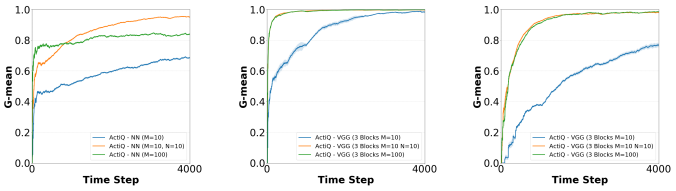
(a) ActiQ-NN (10% imbalance) (b) ActiQ-VGG (10% imbalance) (c) ActiQ-VGG (1% imbalance)

Fig. 3: The role of the memory size M for ActiQ in MNIST with budget $B = 10\%$.



(a) ActiQ-NN, $B = 10\%$ (b) ActiQ-VGG, $B = 10\%$ (c) ActiQ-VGG, $B = 1\%$

Fig. 4: The role of augmentation in MNIST with 1% imbalance



(a) ActiQ-NN, $B = 10\%$ (b) ActiQ-VGG, $B = 10\%$ (c) ActiQ-VGG, $B = 1\%$

Fig. 5: The role of augmentation in Two Patterns

worse than its shallower counterparts as well as the NN. As before, the deepest VGG model performs the worst along with RVUS-NN. Here, the effect of the VGG model depth is more prominent as the budget decreases. Important remarks are:

- As the budget decreases, the performance drops, attributed to the fewer labelled examples, therefore, the queues are populated by a slower rate with new examples. Training also occurs more infrequently.
- As imbalance becomes higher, the performance drops, attributed to the fact that the minority classes are populated by a smaller rate with new examples.
- The role of the model type plays a key role. As expected, the convolutional network VGG outperforms the standard NN on image classification tasks, however, the model depth plays an important role. Shallow architectures have limited capacity, while deeper architectures have a larger capacity but are typically more difficult to train. For instance, VGG with 3 blocks appears to provide a good trade-off, while the original VGG (5 blocks) is even outperformed by the standard NN.
- The one-pass learner RVUS yields the worst performance due to the lack of a memory component. Interestingly, in some cases it outperformed the original VGG (5 blocks).

B. Role of the memory size

We examine now the memory size's role; we compare NN, and VGG with 3 blocks which achieved the best overall performance in Section VI-A. The budget is fixed to 0.1. In these experiments, we present the learning curves, that display the performance at different time steps. Fig. 3a shows ActiQ-NN's performance in MNIST with 10% imbalance. Figs. 3b and 3c show ActiQ-VGG's performance in MNIST with 10% and 1% imbalance respectively. Important remarks are:

- By increasing the memory size, the performance is significantly improved. This is attributed to the larger training set, which could particularly aid larger models.
- In severely imbalanced scenarios the role of the memory size becomes even more important. As before, this is attributed to queues corresponding to the minority classes which are populated by a smaller rate with new examples.
- We stress out, however, that in online / streaming environments, a large memory size is not desirable. This constitutes a limitation of the original ActiQ.

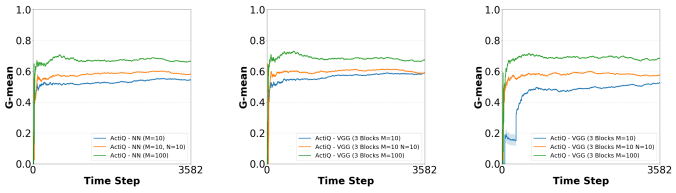
C. Role of data augmentation

We consider the datasets with 1% and 10% budget, and memory sizes of 10 and 100. The performance results of NN and the VGG (3 blocks) are presented in Figures 4, 5 and 6. The augmentation techniques applied are described in Section III-B and their value ranges are provided in our code.

In Fig. 4a, we compare the performance of the NN when we apply data augmentations to the MNIST samples. As a baseline, we use the NN with memory 10 without any augmentations, and we also include the NN with memory 100 with no augmentations. We notice that the use of data augmentation techniques yields slightly improved results when compared to the NN (M=10). We attribute this to the small capacity of the NN. The analogous plots for VGG (3 blocks) are shown in Figs. 4b and 4c. It is evident that data augmentations are very effective when applied to the VGG network. For both MNIST variations, the performance of the model with augmented queues of size 10, reaches the performance of the models with memory size 100, which is a significant improvement.

In Fig. 5a, we examine NN's performance when we apply augmentations to the Two Patterns samples. Again, as a baseline we consider the NN with memory 10 and 100, without any augmentation. It is evident, that data augmentation helped memory size 10 to outperform both memory size 10 and 100, without augmentation. Similarly, for VGG in Figs. 5b and 5c we notice that augmentation increases the VGG's performance compared to memory size 10 without augmentation. Importantly, in both VGG experiments, augmentation achieves similar performance as the memory size 100.

Fig. 6 shows the NN and VGG performance results on the uWave Gesture Library Z. We observe that for budget 10% as shown in Figs 6a and 6b, augmentation can slightly improve over memory 10 without augmentation. In Fig 6c with budget



(a) ActiQ-NN, $B = 10\%$ (b) ActiQ-VGG, $B = 10\%$ (c) ActiQ-VGG, $B = 1\%$

Fig. 6: The role of augmentation in uWave Gesture Library Z

1%, Augmented Queues performance increased significantly when compared with memory 10 without augmentations.

Important remarks are as follows:

- Augmentation has a drastic improvement on model performance. This is attributed to the increase of the training set, which appears to have a similar effect as if the original memory was increased. For the shallow model NN, an improvement was observed but to a lesser degree.
- The proposed method uses less space, as augmentation happens on-the-fly at training times only; augmented data do not reserve memory when they are not in use.

VII. Conclusion and Future Work

Learning online poses major challenges which hinder the deployment of learning models. We introduced Augmented Queues which addresses the problems of limited labelled data and class imbalance. Augmented Queues synergistically combines on-the-fly data augmentation with active learning. We demonstrate its applicability using image and time-series augmentations, and we show that it significantly improves the learning quality and speed. Future work will examine:

Non-stationary environments. While this work has focused on the critical challenges of limited label availability, class imbalance, and limited storage, other challenges exist, such as, concept drift [3], [23].

Augmentation in the latent space. One promising direction is to perform augmentation in the latent space, e.g., using Siamese networks [31]. This would be challenging as the latent space changes over time due to online incremental learning.

References

[1] E. Kyriakides and M. Polycarpou, Eds., *Intelligent monitoring, control, and security of critical infrastructure systems*. Springer, 2014, vol. 565.

[2] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4802–4821, 2018.

[3] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[4] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[5] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, "Detecting adversarial advertisements in the wild," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 274–282.

[6] NVIDIA-AI. Scalable active learning for autonomous driving. Accessed 25 Jan. 2022. [Online]. Available: https://medium.com/nvidia-ai/scalable-active-learning-for-autonomous-driving-a-practical-implementation-and-a-b-test-4d315ed04b5f

[7] A. Karpathy. Artificial intelligence for full self-driving. Accessed 25 Jan. 2022. [Online]. Available: https://www.youtube.com/watch?v=hx7BXih7zx8

[8] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Data-efficient online classification with siamese networks and active learning," in *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[9] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 27–39, 2013.

[10] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowledge and Data Engineering*, no. 9, pp. 1263–1284, 2008.

[11] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368, 2015.

[12] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Online learning with adaptive rebalancing in nonstationary environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4445–4459, 2021.

[13] K. Malialis, C. Panayiotou, and M. M. Polycarpou, "Queue-based resampling for online class imbalance learning," in *International Conference on Artificial Neural Networks (ICANN)*. Springer, 2018, pp. 498–507.

[14] K. Malialis, M. Roveri, C. Alippi, C. G. Panayiotou, and M. M. Polycarpou, "A hybrid active-passive approach to imbalanced nonstationary data stream classification," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2022.

[15] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2425–2438, 2014.

[16] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR'94*. Springer, 1994, pp. 3–12.

[17] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[18] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[19] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1542–1547.

[20] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.

[21] L. Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.

[22] T. Um, F. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, 2017, pp. 216—-220.

[23] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] S. Wang, L. L. Minku, and X. Yao, "Dealing with multiple classes in online class imbalance learning." in *International Joint Conference on Artificial Intelligence*, 2016, pp. 2118–2124.

[26] P. Geurts, "Contributions to decision tree induction: bias/variance tradeoff and time series classification," PhD Thesis, University of Liège, 2002.

[27] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "Uwave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive Mob. Comp.*, vol. 5, pp. 657–675, 2009.

[28] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag, 2006.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[30] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 592–602.

[31] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Nonstationary data stream classification with online active learning and siamese neural networks," *Neurocomputing*, 2022.