# SGDOL: Self-evolving Generative and Discriminative Online Learning for Data Stream Classification

Deeksha Aggarwal[1], J. Senthilnath[2], Uttam Kumar[1], Vivek Yadav[1],
Sushant Kulkarni[3], Md Meftahul Ferdaus[2], Li Xiaoli[2]
[1]Spatial Computing Laboratory, Center for Data Sciences, IIIT Bangalore, India.
[2]Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore.
[3]Indian Institute of Technology, Chennai, India.
{deeksha.aggarwal, uttam, vivek.yadav}@iiitb.ac.in, {J_Senthilnath, Ferdaus_Meftahul, xlli}@i2r.a-star.edu.sg,
ksushantk@gmail.com

***Abstract*—Data streams are usually non-stationary obtained from the same or different data sources. It needs to be processed sequentially, hence termed stream processing. Stream processing often demands evolving neural network architectures that can alter the number of nodes and layers on-demand to classify data streams in an online manner, known as evolving online learning. Traditional deep neural networks (DNNs) uses batch data processing, often limited by their static network structures and offline learning approaches while addressing data streams. In this work, we propose a novel evolving deep neural network framework, known as Self-evolving Generative and Discriminative Online Learning (SGDOL), which utilises an online learning approach to evolve both generator and discriminator network structure from scratch, and on-demand to classify data streams. The dynamic feature learning mechanism of autoencoder-based generative models have demonstrated its potential in learning latent feature representations from data streams. These latent features are fed to the evolving feed-forward DNN-based discriminator as input. The mechanism of adding or pruning nodes in the evolving architecture of discriminator supports in dealing with catastrophic forgetting problems; a new layer is added to the discriminator when a new concept appears in the data stream. To back these theoretical contributions of SGDOL, experiments were conducted using nine benchmark datasets and compared with ten state-of-the-art online learning algorithms. SGDOL performance measure of testing classification rates was better in seven datasets out of nine than the existing algorithms, which clearly indicates its ability to deal with the data stream.**

***Index Terms*—data stream, evolving architecture, generator, discriminator**

## I. Introduction

In the current era, data are generated by almost every electronic devices. To perform analytics and knowledge mining from such data in an online fashion, data are recorded in small chunks and sent simultaneously to relevant companies. The major source of such data comes continuously from sensors fitted in smart e-devices, internet users, meteorological stations, autonomous systems, etc. and known as data streams [1], [2]. Modelling a universal function approximator on data streams for classification tasks becomes challenging due to the following reasons: (i) all the samples are not fully observable since the data arrives in small chunks sequentially, (ii) feature distribution of the incoming unseen data may change over time, giving rise to concept drift (CD) [3]. A traditional deep neural network (DNN) which uses batch or offline learning techniques becomes infeasible for modelling the data streams because it requires the incoming data to be first stored in memory and then train a fixed pre-designed DNN by iterating over it multiple times, this violates point (i) mentioned above. Moreover, to handle concept drift, it is often seen in practice that these fixed-structured DNNs are manually re-designed and trained repeatedly whenever CD is noticed and classification performance decreases. These methods make these models slow and inconsistent in handling practical data streams as shown in Fig. 1. To cope with these challenges, the concept of online learning was introduced, where the DNNs are capable of automatically self-evolving their network structures in terms of adding or pruning the hidden neurons and hidden layers by observing the feature distribution of incoming data chunks. As a result, data stream classification using online learning methods have gained growing research attention, and a number of advanced models [4], [5], [6], [7], [8], [9] have been proposed recently. However, in the existing literature, we observed the following research gaps: (i) majority of them are not fully evolving networks i.e.,

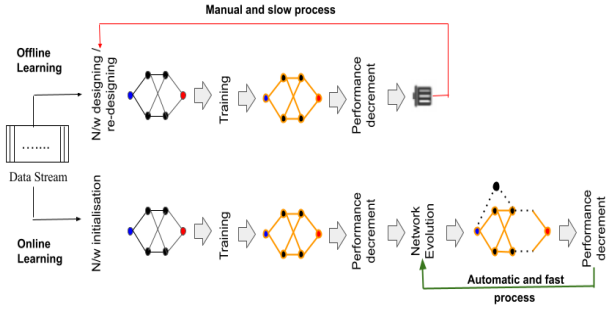


Fig. 1. Flow diagram showing the training of a general offline and online DNN on data streams

they either use a fixed-layered network structure [4], shallow network [8], or only grows the hidden neurons without pruning [7], (ii) proposed models like [9] using a fully evolving network structure but is not able to efficiently extract latent features from the complex data streams resulting in low accuracy. Motivated from these gaps, in this paper we aim to develop a fully evolving network structure named SGDOL (self-evolving generative and discriminative online learning) by combining generator and discriminator architecture in an online setting. A self-evolving autoencoder based generator is introduced as a pre-training step to improve the multi-layer perceptron (MLP) DNN predictive performance by providing latent feature representations. A self-evolving MLP is used as a discriminator model (MLP-DNN) which self-constructs its hidden neurons and hidden layers from scratch and on-demand for data stream classification. The generator also plays a vital role in providing meaningful information to the evolving DNN architecture to converge faster with better decision boundary. Similar combination with fixed structure has been proved to be effective in learning better decision boundary [10].

The main contributions of this work are listed below:
(i) The proposed SGDOL is a novel generator-discriminator based fully automatic deep neural network framework which self-evolves/adapts its network structure by adding or pruning hidden neurons and hidden layers on receiving data chunks.
(ii) Pre-training and fine-tuning operations are performed for efficient latent feature extraction using autoencoder as a self-evolving generator to handle complex data streams.
(iii) Incorporation of hidden layer(s) growing mechanism with a dynamic voting weight given to each layer for selective weight update policy to retain the old knowledge, and that is also efficient at the new knowledge i.e. capable of handling catastrophic forgetting.

The rest of the paper is organized as follows. Section II summarizes the recent and relevant works on streaming data classification. Section III presents the proposed SGDOL framework with its architectural details. Section IV provides a thorough description of the empirical study along with the details of datasets, experimental set-up, results, and major findings. Finally, we conclude in Section V.

## II. Related works

Online learning has become an interesting area of research due to its high scalability and memory-efficient dynamic modeling. Algorithms like perceptron [11], online gradient descent [12], passive aggressive [13], etc. have been proposed to learn linear models with online convex optimisation. However, these models cannot learn complex nonlinear feature space in real-world applications. This opens the challenge of learning DNN in an online mode on data stream problems.

Related works include the hedge backpropagation (HBP) online deep learning algorithm proposed in [6] that evaluated the performance of each hidden layer of the DNN using a hedging strategy and modified the backpropagation algorithm to allow knowledge sharing. Moreover, there are several algorithms proposed to address the catastrophic forgetting problem as follows: (i) by storing some amount of old data in memory, (ii) by constraining the network weights, or (iii) by extending the network capacity. The first method is often limited by space complexity problem and also depends on the type of data samples to be stored. The iCARL algorithm [14], elastic weight consolidation algorithm (EWC) [15], and dynamically expandable networks (DEN) are examples of the three methods, respectively. Also, EWC and HBP require a network initialization step and operate under a user-defined fixed network capacity, whereas DEN uses a fixed number of layers. Autonomous deep learning (ADL) proposed in [9] evolve both the neurons and layers of the DNN on-demand whenever there is concept change, whereas the Progressive Neural Network (PNN) [16] learns $K$ different tasks by introducing new columns while freezing the old ones. ADL and PNN still inhibit the ability to process noisy and unlabelled data streams which are often seen in practice [8]. Deep evolving denoising autoencoder (DEVDAN) [8] evolves its network neurons on-demand and is capable of extracting the meaningful latent representations from the data streams but has the limitation of having a fixed number of hidden layers and has no mechanism to prevent catastrophic forgetting efficiently. SGDOL framework overcomes issues of existing online deep learning algorithms.

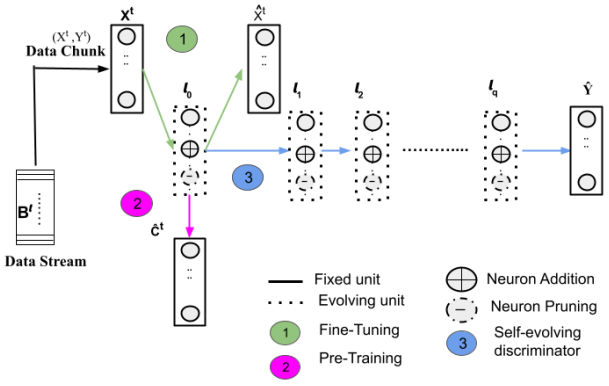


Fig. 2. SGDOL framework.

## III. SGDOL Framework

In this section, we discuss the proposed self-evolving architecture with online learning for data stream problems. SGDOL classifies the input data stream using: (i) a self-evolving generator to capture meaningful representation by adding or pruning the neurons, and (ii) a self-evolving discriminator to overcome catastrophic forgetting by evolving neurons and layers. The detailed flow of the SGDOL framework is shown in Fig. 2.

### A. Problem Statement

Consider an online classification task where labelled data (denoted by $B^t$) is arriving sequentially in small records called data chunks $t$, $B^t = (X^t, Y^t), \forall t = 1, \ldots, T$; $X^t \in \Re^{N \times d}$ and $Y^t \in \Re^N$. Here, $X^t$ are the input features, $Y^t$ are the labels, $N$ denotes the chunk size, $d$ is the number of

input features or attributes and $T$ is the last data chunk. The goal is to learn and capture the feature representation of the data stream efficiently using a self-evolving generator $\mathcal{G}$ followed by learning a generalized functional mapping $\mathcal{D}: X^t \rightarrow Y^t, \forall t = 1, \ldots, T$ using the self-evolving discriminator $\mathcal{D}$ so as to predict the target label on receiving a new data point in real-time. The performance of the learnt function $\mathcal{D}$ is evaluated by testing it on $(t+1)^{th}$ data chunk using cumulative error, $err = \frac{1}{T}\sum_{t=1}^{T} loss(Y \neq y_p)$. Here, $y_p$ is the reconstruction output of $X^t$ while training $\mathcal{G}$ where $loss$ is reconstruction loss and it is the predicted output while training $\mathcal{D}$ where $loss$ is cross-entropy loss.

### *B. Self-evolving Generator*

Training of the self-evolving generator ($\mathcal{G}$) is termed as pre-training which is built upon an autoencoder (AE). Here, AE learns a non-linear manifold in comparison with its linear counterpart called Principal Component Analysis (PCA) [17]. This property of AE is useful to extract robust and meaningful latent representations from the independent features ($X^t$) utilizing each streaming data chunk $B^t$ [18]. The reconstructed output ($\hat{X}$), carried out by the encoding-decoding scheme is formed with the sigmoid activation function, which can be expressed as follows:

$$\hat{X} = f_{(W^T,c)} = sigmoid(l_0 W^T + c) \tag{1}$$

$$l_0 = f_{(W,b)} = sigmoid(XW + b) \tag{2}$$

where $W \in \Re^{d \times n}$ is the weight matrix, $b \in \Re^n$, $c \in \Re^d$, are the bias of the hidden layer $l_0$ and $\hat{X}$ respectively, $n$ is the number of self-evolving hidden neurons.

Following the basic concept of bias-variance trade-off [19], the hidden layer ($l_0$) is evolved dynamically. Here, a hidden neuron ($h_n^g$) is added when the network has a high bias (underfitting condition), whereas the hidden neuron is pruned when the network has a high variance (overfitting condition). Here, we attempt to find the optimal network structure, which learns the incoming data efficiently and generalizes upon testing. Hence, approximate computation of network bias and variance upon receiving the streaming data chunk $B^t$ is needed and derived from the expectation of squared error ($E[e]$) between the ground truth ($y_g$) and the predicted ($y_p$) output as follows:

$$\begin{aligned} E[e] &= E[(y_g - y_p)^2] \\ &= E[y_p^2] - E[y_p]^2 + (E[y_p] - y_g)^2 \\ &= Var(y_p) + (Bias(y_p))^2 \end{aligned} \tag{3}$$

where, $(Bias(y_p))^2$ and $Var(y_p)$ are the bias and variance of the predicted output $y_p$ respectively.

In order to calculate $(Bias(y_p))^2$ and $Var(y_p)$, let, $y_g = X$ and $y_p = \hat{X}$, and to obtain the expectation of the squared reconstruction error, let $E[e] = E[(X - \hat{X})^2]$. Hence, $E[\hat{X}]$ is derived using "(1)", "(2)", "(3)" as follows:

$$E[\hat{X}] = sigmoid(E[l_0]W^T + c) \tag{4}$$

$$E[l_0] = \int_{-\infty}^{\infty} sigmoid(F)P(F)\,dF \tag{5}$$

where, $F = XW + b$ and $P(F) = \frac{1}{\sqrt{2 \times \pi(\sigma_F^t)^2}} \exp - \frac{(F - \mu_F^t)^2}{2(\sigma_F^t)^2}$ is the probability density function when normal distribution holds. Using probit approximation [20], "(5)" can be expressed as follows:

$$E[l_0] \approx sigmoid(\frac{\mu^t}{\sqrt{1 + \pi(\sigma^t)^2/8}}) \tag{6}$$

where $\mu^t$ and $\sigma^t$ are the mean and standard deviation of $X^t$. Using "(6)" and "(4)", $E[\hat{X}]$ can be calculated, and $(Bias(y_p))^2$ and $Var(y_p)$ can be calculated by substituting $E[\hat{X}]$ in "(3)".

The below dynamic conditions can be monitored in order to trigger the construction or destruction of a hidden neuron in the hidden layer ($l_0$) respectively:

$$\mu_{bias}^t + \sigma_{bias}^t \geq \mu_{bias}^{min} + \pi\sigma_{bias}^{min} \tag{7}$$

$$\mu_{var}^t + \sigma_{var}^t \geq \mu_{var}^{min} + 2\chi\sigma_{var}^{min} \tag{8}$$

In "(7)", $\pi = 1.3\exp(-(Bias(y_p))^2) + 0.7$ is a function of $Bias^2$ that governs the degree of confidence of sigma rule and revolves between the interval 1 to 2 providing a confidence level around $68.2\%$ to $95.2\%$. These conditions have been derived based on the $k$-sigma rule adopted from the theory of statistical process control [21] by directly utilizing the bias and variance of the network itself. They provide flexibility in growing and pruning of the hidden neurons while eliminating the involvement of problem specific thresholds. Here, $\mu_{bias}^t$, $\sigma_{bias}^t$ are the recursive mean and standard deviation of the $Bias^2$ up to the arrival of $t^{th}$ data chunk, whereas $\mu_{bias}^{min}$, $\sigma_{bias}^{min}$ are the minimum value of the $\mu_{bias}^t$, $\sigma_{bias}^t$ up to the arrival of $t^{th}$ data chunk but are reset whenever "(7)" is satisfied. Similarly, $Var(y_p)$ is utilized instead of the $(Bias(y_p))^2$ where, $\sigma_{var}^{min}$ is multiplied by 2 to avoid direct-pruning-after-growing problem. Here, $\chi$ revolves between 2 and 4, which is providing a confidence level of around 95.4% to 99.9%. In high bias problem, "(7)" will be satisfied which triggers the construction of a hidden neuron with Xavier initialized weights and on the other hand, in high variance problem, "(8)" will be satisfied which triggers the pruning of least contributing hidden neuron in producing the network output $y_p$ in $l_0$. The least contributing neuron ($i$) with the lowest $E[l_0]$ can be calculated using "(6)" and is formalised as, $prune : \min_{i=1,\ldots,n} E[l_0]_i$. This concept of the statistical contribution of the hidden neuron can be categorized as a performance estimation strategy of neural architecture search because it estimates the generalization power of the network on unseen data [22].

Once the pre-training is done for a particular data chunk $B^t$, single pass fine-tuning is conducted on the network to tune its learned weights utilising the true class labels $Y^t$. Fine-tuning also follows a similar hidden neuron growing and pruning mechanism of the same hidden layer ($l_0$) as that is followed in the pre-training step but in a supervised way. That is, squared predictive error $E(e) = E[(Y^t - \hat{C})^2]$ is

computed instead of the reconstruction error. The encoder part of the pre-trainer is connected to the softmax layer and the output weight matrix ($\theta \in \Re^{n \times c}$), bias ($\eta \in \Re^{c}$), $W$ and $b$ are further adjusted using Stochastic Gradient Descent (SGD) method with momentum in a single pass learning mode using cross-entropy loss function $loss(Y^t, \hat{C})$. The predicted label $\hat{C}$ of the fine-tuning step can be expressed as follows:

$$\hat{C} = \arg\max_{1,\ldots,c}(softmax(l_0\theta + \eta)) \tag{9}$$

$$W, b, \theta, \eta = \arg\min_{W,b,\theta,\eta} \sum_{t=1}^{T} \frac{1}{T} loss(Y^t, \hat{C}) \tag{10}$$

### C. Self-evolving Discriminator

Once the pre-trainig is done by the self-evolving generator, the self-evolving discriminator $\mathcal{D}$ is trained to find a generalised decision boundary using a self-evolving MLP-DNN architecture. The hidden layer $l_0$ consisting of latent features becomes the first hidden layer of $\mathcal{D}$ and all the intermediate hidden layers (termed as $l_q$) are self-evolved and interspersed by a sigmoid activation function. Here, $q \in 0, \ldots L-1$ and $L$ is the total number of hidden layers in the network at a particular time. The output vector $\hat{C}_d$ of this network is as follows:

$$\hat{C}_d = \arg\max_{o=1,\ldots,c} \hat{Y}_o; \hat{Y} = \sum_{q=0}^{L-1} \beta^{(l_q)}.y^{(l_q)} \tag{11}$$

$$y^{(l_q)} = softmax(Ws^{(l_q)}h^{(l_q)}) + bs^{(l_q)}, \forall q = 0, \ldots, L-1 \tag{12}$$

$$h^{(l_q)} = sigmoid(W^{(l_q)}h^{(l_q-1)}) + b^{(l_q)}, h^{(l_0)} = X \tag{13}$$

where, $y^{(l_q)}$ is the unique multi-class probability observed for every $q^{th}$ hidden layer. The network parameters of the $q^{th}$ hidden layer is denoted as $W^{(l_q)} \in \Re^{m^l_q \times m^{l_q-1}}, b^{(l_q)} \in \Re^{m^{l_q}}, Ws^{(l_q)} \in \Re^{c \times m^{l_q}}, bs^{(l_q)} \in \Re^{c}$, where $m^{l_q}$ and $m^{l_q-1}$ are the number of hidden neurons in the $q^{th}$ and $(q-1)^{th}$ hidden layer respectively. Each hidden layer is assigned with a normalized voting weight $\beta^{(l_q)}$; $\sum_{q=0}^{L-1} \beta^{(l_q)} = 1$, which is dynamically adjusted by a unique penalty or rewarding factor $\rho^{(l_q)} \in [0, 1]$. $\rho$ plays an important role in adapting the network whenever there is a CD in the data stream. The value of $\rho^{(l_q)}$ is inversely proportional to the rate of adapting the network with the current drift. Hence, $\rho$ is dynamically adjusted to represent the performance of each hidden layer using a step size $\epsilon$ as in "(14)". The value of $\rho$ is set to either $\rho + \epsilon$ or $\rho - \epsilon$ whenever the $q^{th}$ hidden layer returns a correct prediction or an incorrect one, respectively. This also considers the fact that the voting weight $\beta^{(l_q)}$ of a hidden layer embracing relevant representation should decrease slowly when making misclassification while that embracing irrelevant representation. On the other hand, it should be increased slowly when returning the correct prediction.

$$\rho^{(l_q)} = \rho^{(l_q)} \pm \epsilon \tag{14}$$

$$\begin{aligned} \beta^{(l_q)} &= min(\beta^{(l_q)}(1 + \rho^{(l_q)}), 1) \\ &= \beta^{(l_q)}.\rho^{(l_q)} \end{aligned} \tag{15}$$

**Algorithm 1** SGDOL Model

**Input:** Input data chunks; $B^t = (X^t, Y^t)$,t $\leftarrow 1 \ldots T$ is the incoming data chunks

**Output:** $\mathcal{D}$: Discriminator Model

*Initialisation:*

$l_0 \leftarrow generativeLearning(B^t)$ // Algorithm 2

$\mathcal{D} \leftarrow$ initialize discriminator network (MLP-DNN)

$l_w \leftarrow l_1$

$y^{(l_q)} \leftarrow$ Softmax score for the q-th layer (Equation (12))

$\Gamma \leftarrow$ Maximum information compression index

$\delta \leftarrow 0.05$

*/* Processing */*

1: **for** t = 1 to T **do**
2: **Discriminating Testing Phase:**
Execute forward propagation using $\mathcal{D}$
3: **Discriminating Training Phase:**
4: **if** ($q > 1$ and $\Gamma(y^{(l_i)}, y^{(l_j)}) > \delta, \forall i, j = 1, \ldots, L; i \neq j$) **then**
5: Prune hidden layer with minimum $\beta$
6: **end if**
7: drift status = Execute drift detection // Section III-C
8: **if** drift status == DRIFT **then**
9: Add hidden layer $l_{new}$ to $\mathcal{D}$
10: $y^{(l_{new})} \leftarrow y^{(l_w)}$
11: **end if**
12: $l_w \leftarrow \arg\max_q(y^{(l_q)})$
13: Calculate: $\mu^t_{bias}$, $\sigma^t_{bias}$, $\mu^{min}_{bias}$, $\sigma^{min}_{bias}$, $\mu^t_{var}$, $\sigma^t_{var}$, $\mu^{min}_{var}$, $\sigma^{min}_{var}$ // Section III-B
14: **if** ($\mu^t_{bias} + \sigma^t_{bias} \geq \mu^{min}_{bias} + \pi\sigma^{min}_{bias}$) **then**
15: Add neuron in $l_w$
16: Reset $\mu^{min}_{bias}$ and $\pi\sigma^{min}_{bias}$; grow $\leftarrow 1$
17: **else**
grow $\leftarrow 0$
18: **end if**
19: **if** ($\mu^t_{var} + \sigma^t_{var} \geq \mu^{min}_{var} + 2\chi\sigma^{min}_{var}$), (grow == 0) and ($q > 1$) **then**
20: Prune neuron from $l_w$ having least value of $\beta$
21: Reset $\mu^{min}_{var}$ and $\pi\sigma^{min}_{var}$
22: **end if**
23: Calculate loss and update weights and biases using Equation (10)
24: **end for**

The winning layer termed as $l_w$ is a hidden layer embracing the highest $\beta$ value and $y^{(l_w)}$ is the multi-class probability observed by the wining layer. Starting from one hidden neuron and one hidden layer, the hidden neurons in $l_w$ are grown or pruned recursively following the same mechanism discussed in section III-B in a supervised way. That is, squared prediction error $E(e) = E[(Y^t - \hat{C}_d)^2]$ is computed. In SGDOL, multiple hidden layers are added ($L = L + 1$) by using a drift detection scenario (DDS) to deal with the CD problem in the streaming data set. The drift is signalled using online and non-parametric drift detection method [23] based on hoeffding's error bounds by monitoring the drop

**Algorithm 2** $generativeLearning(B^t)$

**Input:** $B^t$: Input data chunks
**Output:** $l_0$ : Hidden layer
*Initialisation:*
$\mathcal{G}$ = Initialize generative autoencoder
*/* Processing */*
**for** t = 1 to T **do**
  Execute feed forward propagation using $\mathcal{G}$
  Calculate: $\mu_{bias}^t$, $\sigma_{bias}^t$, $\mu_{bias}^{min}$, $\sigma_{bias}^{min}$, $\mu_{var}^t$, $\sigma_{var}^t$, $\mu_{var}^{min}$, $\sigma_{var}^{min}$ // Section III-B
  **Generative Training Phase:**
  **if** $(\mu_{bias}^t + \sigma_{bias}^t \geq \mu_{bias}^{min} + \pi\sigma_{bias}^{min})$ **then**
    Add neuron in $l_0$
    Reset $\mu_{bias}^{min}$ and $\pi\sigma_{bias}^{min}$; grow $\leftarrow$ 1
  **else**
    grow $\leftarrow$ 0
  **end if**
  **if** $(\mu_{var}^t + \sigma_{var}^t \geq \mu_{var}^{min} + 2\chi\sigma_{var}^{min})$ and (grow == 0) **then**
    Prune neuron from $l_0$ having least value of $\beta$
    Reset $\mu_{var}^{min}$ *and* $\pi\sigma_{var}^{min}$
  **end if**
  Calculate loss and update weights and biases.
**end for**

in accuracy. Here, predefined thresholds ($error_W, error_{drift}$) and two accuracy matrices ($F$ and $H$) are used to signal drift status. The condition $|H - F| > error_{drift}$ signals a drift status and the condition $error_W < |H - F| < error_{drift}$ signals a warning status, otherwise the status is stable. Once the drift is signalled, a new hidden layer will be formed and is trained using the current data batch $B^t$. Meanwhile, the network parameters of other hidden layers are frozen to preserve the old knowledge. The dynamic voting weight adaptation $\beta$ and selective weight update method prevents catastrophic forgetting.

SGDOL employs hidden layer pruning mechanism by eliminating the most redundant layers as they do not open the manifold of learning problem to a unique representation. The redundant layers are identified by finding the correlation between hidden layers through the utilization of the Maximum Information Compression Index (MICI) [24] method and yields the below pruning condition:

$$\Gamma(y^{(i)}, y^{(j)}) > \delta, \forall i, j = 1, \ldots, L, i \neq j \tag{16}$$

where, $\delta$ acts as a threshold and is directly proportional to the maximum correlation index between two hidden layers. If "(16)" is satisfied, the hidden layer with the lowest $\beta$ value will be pruned as $\beta$ is expected as an appropriate indicator of a hidden layer performance as discussed before in this section. Consequently, the hidden layer weight dimensions and network connections are updated and resulting in dynamic resource allocation. Interestingly, the hidden neuron and layer pruning can be seen as dropout scenario in the realm of deep learning. Moreover, SGDOL has linear time and space complexity which depend on the chunk size $N$ and number of data chunks $T$ for a particular dataset. The proposed SGDOL framework is described in Algorithm 1.

TABLE I
DATASET PROPERTIES.
IA: INPUT ATRIBUTE; C: CLASSES; DP: DATA POINTS; CD: CONCEPT DRIFT; PPS: PRIOR PROBABILITY SHIFT; CS: CONCEPT SHIFT; IF: IMBALANCE FACTOR

| CD Source | CD Type | Dataset | IA | C | DP | IF |
|---|---|---|---|---|---|---|
| PPS | Recurring | Permuted MNIST | 784 | 10 | 70K | 0.09 |
| | Sudden | SEA | 3 | 2 | 100K | 0.26 |
| | Gradual; Incremental | Hyperplane | 4 | 2 | 120K | 0.00 |
| CS | Sudden | Rotated MNIST | 784 | 10 | 65K | 0.09 |
| | Sudden; Recurring | Occupancy | 7 | 2 | 20K | 0.53 |
| | | KDDCup 10% | 41 | 2 | 500K | 0.60 |
| | | SECOM | 590 | 2 | 1567 | 0.86 |
| - | - | MNIST | 784 | 10 | 70K | 0.09 |
| - | - | Hepmass 19% | 28 | 2 | 2M | 0.00 |

## IV. EXPERIMENTS

### A. Concept Drift and Dataset Description

Concept Drift (CD) is a phenomenon in which the statistical properties of the data stream changes which leads to a change in the joint distribution $P_t(X, Y) \neq P_{t+1}(X, Y)$ [3]. These changes might be caused by the changes in hidden variables which cannot be measured directly [25]. Prior probability shift (PPS) and covariate shift (CS) are the two sources of CD. PPS occur due to drift in feature space when $P_t(X) \neq P_{t+1}(X)$ while $P_t(Y|X) = P_{t+1}(Y|X)$. Whereas, CS occurs when drift results in decision boundary change that leads to decrease in learning accuracy when $P_t(X) = P_{t+1}(X)$ while $P_t(Y|X) \neq P_{t+1}(Y|X)$. In this paper, three types of CD have been considered for experimentation: (i) Sudden drift (SD): when a new concept occurs within a short span of time, (ii) Gradual drift (GD): when a new concept gradually replaces an old one over a period of time, (iii) Recurring drift (RD): when an old concept reoccurs after some time [3], [26].

The learning performance of SGDOL is evaluated using nine datasets i.e. Rotated MNIST [27], Permuted MNIST [28], SEA [29], Hyperplane [30], Occupancy [31], KDDCup [32], SECOM [33], MNIST [34] and Hepmass [35]). The first seven datasets are non-stationary having different types and sources of CD. The last two datasets are stationary where MNIST has high input attributes with multi-class classification problem. In our experiments, we have utilized 10% and 19% of the data points of the KDDCup and Hepmass dataset, respectively. The properties of each dataset are summarized in Table I. In our experiment, the SECOM dataset is used as non-stationary by considering sudden CD. Here, the data chunks are balanced in the initial chunks, as the minority class expires, the data chunks contain only the majority class. SGDOL performance is also evaluated on three imbalanced datasets having an imbalance factor (IF) greater than 0.5. The degree of sample imbalance as measured through an IF is defined as:

$$IF = 1 - \frac{c}{S} \times \min_{j=1,..,c} S_j \tag{17}$$

where $S_j$ is the total number of samples in class $j$, $c$ is the number of target classes and $S = \sum_{j=1}^{c} S_j$.

### B. Results and Discussion

In this section, SGDOL performance on different datasets are compared with ten state-of-the-art online learning algorithms: DEVDAN [8], ADL [9], HBP [6], EWC [15], PNN [16], OMB [36], pEnsemble [24], pENsemble+ [37], I. Bagging and I. Boosting [38].

***Experimental Settings:*** Performance of the algorithms were evaluated using four criteria: classification rate (CR), the mean number of hidden neurons in each hidden layer (HN), the mean number of hidden layers (HL) and the total number of parameters (NoP) in the SGDOL framework. CR is obtained by evaluating SGDOL's performance on the next data chunk (trained on $B^t$ and tested on un-trained incoming data chunk $B^{t+1}$) to test its generalization power. The results of pENsemble, pENsemble + , I. Bagging and I. Boosting are not reported in some of the problems because they are not scalable enough in high dimensional data. SGDOL algorithm was executed using MATLAB 2020b with the Intel(R) Xeon(R) CPU E5-1650 @3.20 GHz processor and 16 GB RAM. It starts the learning process from scratch having one hidden neuron and one hidden layer. Here, the SGD method is used to adjust the parameters with learning rates of 0.01 and a momentum coefficient of 0.95 for every dataset. The hyperparameter values such as $[\delta, \epsilon]$ are set to [0.05, 0.001] in all the experiments, whereas dynamic parameters $\beta$ and $\rho$ are initialized to 0. The datasets were min-max normalized, and the chunk size ($N$) was set to 1000 except for the SECOM dataset, where it was set to 10 due to few data points (1567).

***Numerical Results:*** Table II shows the comparison between eleven algorithms on nine datasets. The proposed SGDOL is able to achieve state-of-the-art classification accuracy on seven datasets when compared to other algorithms. The self-evolving generative network of SGDOL takes care of dimensionality reduction for the high dimensional dataset like SECOM, permuted MNIST, rotated MNIST and MNIST while reducing their dimension by 97.75%. SGDOL shows the highest classification rate when tested on the non-stationary dataset having different types and sources of CD. This is because SGDOL not only learns the latent representation of the streaming dataset efficiently but also solves the forgetting problem while learning the self-evolving decision boundary through its self-evolving discriminator.

To further analyse the learning behaviour of SGDOL under concept drift and its ability to handle catastrophic forgetting, Fig. 3 shows the evolution of accuracy over the entire data stream of three non-stationary datasets i.e. SEA, Permuted MNIST and Hyperplane having sudden, recurring and gradual and incremental concept drifts respectively. The overall positive increment trend in the accuracy curves and small length drips in the accuracy shows the fast adapting and knowledge retaining behaviour of SGDOL with the incoming concept drift. This behaviour of SGDOL helps combat CD and catastrophic forgetting problems.

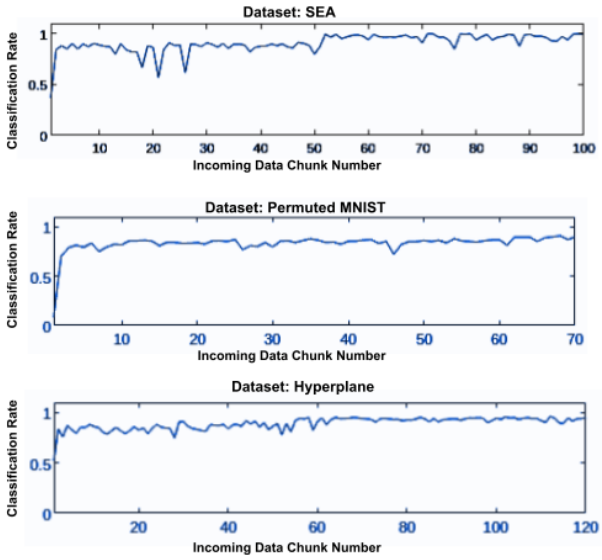


Fig. 3. Performance analysis under different types of concept drift condition for SEA, Permuted MNIST and Hyperplane dataset.

**MNIST dataset: Visualising learned Representations**: To visualise the learned representations in 2D space for the MNIST dataset, we have applied the t-SNE method [39] on the last hidden layer activations of DEVDAN, ADL and SGDOL models as shown in Fig. 4. Fig. 4(A), shows the t-SNE plot for raw data distribution where each class clusters have high intra-class separation/low cluster compactness and low inter-class separation, which is more challenging to classify. Whereas Fig. 4(B)-(D) shows the learned representation of the dataset using DEVDAN, ADL and SGDOL, respectively.

In Fig. 4(B), the class clusters are compact but the inter-class separation is low, whereas, in Fig. 4(C), the inter-class separation is high but the compactness or intra-class separation is still high as compared to Fig.4(D) where the inter-class separation is high as well as intra-class separation is low. Hence, this shows that SGDOL was able to capture the higher-level features which are useful in class discrimination giving the highest accuracy of 79.04% for MNIST dataset as reported in table II.

**Comparison of online deep learning models' performance on a highly imbalanced SECOM dataset**: The SECOM dataset is a fault detection problem in semiconductor manufacturing processes and is characterized by hundreds of different types of operations and the identification of those operations that lead to faults in the final products. The three main challenges in this dataset are: (i) high-dimensionality: 591 signals (features) generated from the semiconductor device to sense anomalies, (ii) class imbalance problem: only 6.6% fail class having high IF value of 0.867, (iii) less number of data points: 1567 total samples with pass class having 1463

TABLE II
COMPARATIVE PERFORMANCE STUDY OF SGDOL

| Dataset | Metrics | SGDOL | DEVDAN | ADL | HBP | HAT | PNN | OMB | I. Boosting | I. Bagging | pENsemble | pENsemble+ | L++.NSE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rotated MNIST | CR | **79.04 07.82** | 76.48 ± 9.7 | 73.97 ± 9.92 | 7.61 ± 4.5 | 65 ± 12 | 57 ± 13.9 | 26 ± 6 | - | - | - | - | - |
| | HN | 46.95 ± 13.22 | 48.7 ± 9 | 66.98 ± 9.3 | 100 | 60 | 750 | - | - | - | - | - | - |
| | HL | 2 | 1 | 1.14 ± 0.4 | 20 | 2 | 3 | 3 | - | - | - | - | |
| | NoP | (74.28 ± 22.13)K | (38 ± 8)K | (18 ± 7.5)K | 270K | 24.9K | 530K | - | - | - | - | - | - |
| Permuted MNIST | CR | **82.63 ± 14.28** | 76.67 ± 14 | 79.8 ± 14.6 | 12.08 ± 0.1 | 66 ± 16 | 65 ± 13.9 | 11 ± 6 | - | - | - | - | - |
| | HN | 38.45 ± 6.85 | 67.8 ± 16.8 | 20 ± 5 | 100 | 60 | 750 | - | - | - | - | - | - |
| | HL | 2 | 1 | 1 | 20 | 2 | 3 | 3 | - | - | - | - | - |
| | NoP | (54.46 ± 11.53)K | (53 ±14)K | (16 ± 4)K | 270K | 24.9K | 530K | - | - | - | - | - | - |
| MNIST | CR | **89.44 ± 3.96** | 86.12 ± 7.8 | 86.07 ± 8.22 | 11.50 ± 0.5 | 78 ± 12 | 68 ± 13.4 | 29 ± 5 | - | - | - | - | - |
| | HN | 57.41 ± 12.93 | 68.9 ± 14.9 | 108 ± 6.2 | 100 | 60 | 750 | - | - | - | - | - | - |
| | HL | 2 | 1 | 1.2 ± 0.4 | 20 | 2 | 3 | 2 | - | - | - | - | - |
| | NoP | (53.4 +- 36)K | (54 ± 13)K | (17 ± 5)K | 270K | 24.9K | 530K | - | - | - | - | - | - |
| SEA | CR | **92.28 ± 6.36** | 91.12 ± 7.11 | 92 ± 6.49 | 62.06 ± 2.1 | 75 ± 10 | 83 ± 6 | 88 ± 4 | 79.6 ± 6.18 | 87.3 ± 10.2 | 91.61 ± 5.6 | 92 ± 6 | 91.93 ± 5.9 |
| | HN | 29.05 ± 7.23 | 23.7 ±7.2 | 21 ± 4 | 100 | 10 | 33 | - | - | 100 | 2 | 2.51 ± 0.81 | 10 |
| | HL | 2 | 1 | 1.01 ± 0.1 | 20 | 2 | 3 | 2 | - | - | 1 | 2 ± 1 | - |
| | NoP | (1.36 ± 0.24)K | (144.8 ± 44.81)K | (359 ± 253)K | 192K | 72 | 353 | - | 100 | - | 24 | 60.3 ± 19.43 | 101 |
| Hyperplane | CR | **92.34 02.55** | 91.19 ± 3.28 | 92.26 ± 2.67 | 72.80 ± 0.5 | 92.26 ± 2.67 | 86 ± 6 | 87 ± 4 | 74.78 ± 3.54 | 81.39 ± 2.2 | 91.65 ± 2.42 | 87.6 ± 6.2 | 90.45 ± 2 |
| | HN | 9.38 ± 1.32 | 16 ± 2.3 | 9.44 ± 1 | 100 | 9.44 ± 1 | 42 | - | - | 100 | 4.8 ± 2.4 | 2.76 ± 0.47 | 10 |
| | HL | 2 | 1 | 1 | 20 | 1 | 3 | 2 | - | - | 2.4 ± 1.2 | 3 ± 2 | - |
| | NoP | (137 ± 19.3)K | (114 ± 18.8)K | (69.1 ± 7)K | 192K | 69.1 ± 7 | 0.5K | - | 120 | - | 57.88 ± 28.7 | 54.68 ± 10.92 | 121 |
| Occupancy | CR | 92.26 12.90 | 90.72 ± 15.9 | 87.97 ± 17.27 | 76.90 ± 0.45 | 72 ± 34 | 71 ± 34 | **99 ± 0.2** | 56.65 ± 34 | 86.02 ± 15.2 | 89.30 ± 23.4 | 89.33 ± 24 | 94.65 ± 11 |
| | HN | 44.62 ± 10.62 | 35.68 ± 6.45 | 21.75 ± 11 | 100 | 20 | 30 | - | - | 100 | 2.5 ± 1.2 | 2.3 ± 0.49 | 10 |
| | HL | 2 | 1 | 1 | 20 | 2 | 3 | 2 | - | - | 2 ± 1.4 | 1.4 ± 0.5 | - |
| | NoP | (1.7 ± 0.534)K | (257.8 ± 117)K | (177 ± 87)K | 192K | 162 | 302 | - | 8 | - | 30 ± 14 | 27.7 ± 0.48 | 8 |
| KDDCup 10% | CR | **99.84 ± 00.15** | 98.29 ± 6.4 | 99.83 ± 0.2 | 91.04 ± 0.15 | 99.6 ± 1 | 99 ± 1 | 97 ± 0.6 | 98.55 ± 0.53 | 99.5 ± 0.4 | 99.3 ± 0.4 | 96.7 ± 6 | - |
| | HN | 28.03 ± 1.50 | 63.64 ± 13.74 | 36 ± 2 | 100 | 60 | 60 | - | - | 100 | 1 | 1 | |
| | HL | 2 | 1 | 1 | 20 | 2 | 3 | 2 | - | - | 1 | 1 | - |
| | NoP | (3.53 ± 0807)K | (513 ± 111)K | (1.6 ± 87)K | 194K | 2K | 2K | - | 500 | - | 12 | 12 | - |
| Hepmass 19% | CR | **83.90 01.91** | 83.39 ± 2 | 83.04 ± 1.8 | 82.17 ± 0.1 | 76 ± 4 | 70 ± 4 | 78 ± 1 | 80.11 ± 98.21 | 78.3 ± 2.2 | 82.6 ± 1.9 | 82.3 ± 2.2 | - |
| | HN | 7.44 ± 0.45 | 10.88 ± 0.5 | 99 ± 4.5 | 100 | 40 | 18 | - | - | 100 | 2.01 ± 0.69 | 2.01 ± 0.69 | - |
| | HL | 2 | 1 | 2 ± 0.7 | 20 | 1 | 3 | 2 | - | - | 2.01 ± 0.69 | 2.01 ± 0.69 | - |
| | NoP | (0.83 +- 0.13)K | (340 ± 17)K | (730 ± 378)K | | 1K | 324 | - | 2K | - | 24.14 ± 8.23 | 24.14 ± 8.23 | - |
| SECOM | CR | 93.65 +- 16.62 | 93.65 ± 16.62 | **93.72 ± 16.55** | 57.31 ± 1.1 | - | - | - | - | - | - | - | - |
| | HN | 7.68 ± 1.2 | 14.21 ± 1.39 | 2.82 ± 0.39 | 100 | - | - | - | - | - | - | - | - |
| | HL | 2 | 1 | 1 | 20 | - | - | - | - | - | - | - | - |
| | NoP | (7.85 ± 3.92)K | (8.42 ± 0.15)K | (1.61 ± 0.05)K | 251K | - | - | - | - | - | - | - | - |

TABLE III
ACCURACY ASSESMENT FOR SECOM DATASET

| SECOM dataset majority:minority class distribution in each data chunk | Metrics | SGDOL | DEVDAN | ADL |
|---|---|---|---|---|
| 50:50 | Majority class accuracy | 93.50 | **98.97** | 97.26 |
| | Minority class accuracy | **91.34** | 14.42 | 39.42 |
| | F1 | 0.96 | 0.96 | 0.96 |
| | Sensitivity | **0.99** | 0.94 | 0.95 |
| | Specificity | 0.5 | 0.5 | 0.5 |
| | G-Mean | **0.70** | 0.68 | 0.69 |
| 60:40 | Majority class accuracy | 94.25 | **100** | 99.58 |
| | Minority class accuracy | **53.84** | 0 | 3.84 |
| | F1 | 0.95 | 0.96 | 0.96 |
| | Sensitivity | **0.96** | 0.93 | 0.93 |
| | Specificity | 0.4 | 0 | 0.4 |
| | G-Mean | **0.62** | 0 | 0.61 |
| 70:30 | Majority class accuracy | 98.56 | **100** | 99.38 |
| | Minority class accuracy | **9.47** | 0 | 2.88 |
| | F1 | 0.96 | 0.96 | 0.96 |
| | Sensitivity | 0.93 | 0.93 | 0.93 |
| | Specificity | **0.3** | 0 | 0.25 |
| | G-Mean | **0.53** | 0 | 0.48 |

samples and fail class having 104 samples.

In Table II, it can be observed that among four online deep learning methods that are applied on the SECOM dataset, SGDOL performed second best in terms of overall classification rate. In order to further analyze SGDOL's performance on the minority class, this dataset was altered by combining three different ratios of majority and minority class samples in each chunk as 50:50, 60:40 and 70:30. These three cases introduce sudden drift in the data stream when all the minority class is extinct and also with different ratio of class imbalance in each data chunk. The performance of three cases evaluated on SGDOL, DEVDAN and ADL models and individual class accuracy along with other relevant metrics are shown in Table III. From this table, it is observed that SGDOL performs better in the classifying of fail class (minority class) than the ADL and DEVDAN in all three cases. Thus, much higher accuracy in the minority class and high G-Mean value is witnessed. Whereas, SGDOL also performed competitively in classifying the pass class (majority class). The high individual accuracy of both the majority and minority class of the SECOM dataset confirms that the SGDOL framework is able to capture latent feature distributions for the imbalanced datasets too.

### C. Ablation Study

In this subsection, we conducted the ablation study by modifying SGDOL's self-evolving generator and discriminator architecture in order to provide additional insight into the effect of each SGDOL's learning policy. Three cases are introduced as follows: case A: fixing the generator while evolving the discriminator, case B: fixing the discriminator while evolving the generator, and case C: fixing both generator and discriminator. This study was done on SEA and KDD cup datasets and the outcome of these cases are shown in Table IV. From this table, it was observed that in comparison with the SGDOL framework, the classification rate drops by 2%, 17.87%, 19.87% in all three cases, respectively. This is because each component of SGDOL overcomes different challenges of the data stream problem such as learning latent representations, handling CD and evolving the network capacity on-demand.

## V. CONCLUSION

This paper focuses on evolving online Deep learning architecture that overcomes challenges faced by fixed DNN architectures in dealing with data streams. The proposed SGDOL (self-evolving generative and discriminative online

TABLE IV
SUMMARY OF ABLATION STUDY FOR SGDOL.

| Data Set | Metrics | SGDOL: Proposed model with evolving generator and discriminator | Case A: Proposed model with fixed generator and evolving dicriminator | Case B: Proposed model with fixed discriminator and evolving generator | Case C: Proposed model with fixed discriminator and generator |
|---|---|---|---|---|---|
| SEA | CR | **92.28 ± 06.36** | 90.28 ± 08.71 | 85.73 ± 1.25 | 80.17 ± 5.2 |
| | HN | 29.05 ± 7.06 | 24.37 ± 7.21 | 21.5 ± 3.5 | 23.5 ± 2.5 |
| | HL | 2 | 2.06 ± 0.23 | 2 | 2 |
| | NoP | (1.36 ± 0.24)K | (1.56 ± 0.72)K | 126 ± 9 | 138 ± 15.6 |
| KDD Cup 10% | CR | **99.84 ± 00.15** | 98.83 ± 00.17 | 81.97 ± 4.30 | 79.97 ± 3.2 |
| | HN | 40.11 ± 1.53 | 34.01 ± 1.54 | 26.5 ± 1.12 | 13.5 ± 2.22 |
| | HL | 2 | 2 | 2 | 2 |
| | NoP | (3.53 ± 0.80)K | (2.60 ± 0.05)K | (2.17 ± 0.01)K | (1.6 ± 0.12)K |

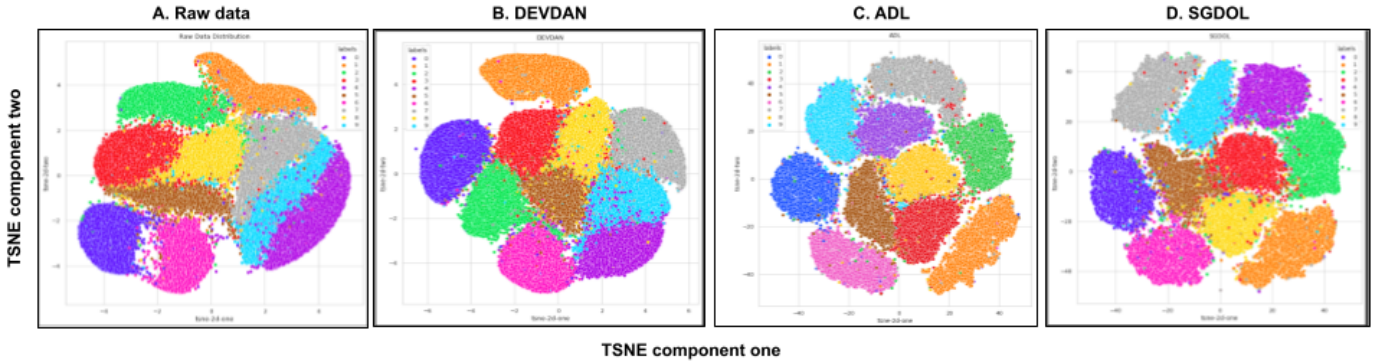


Fig. 4. t-SNE plots showing projection of last hidden layer activations of MNIST dataset after training on different models.

learning) utilizes the bias-variance concept in determining the optimal network complexity that evolves its network structure on-demand to tackle concept drift and catastrophic forgetting problem in streaming data. Our extensive experiments on nine benchmark datasets demonstrated the competitive performance of SGDOL against ten state-of-the-art online algorithms. In-depth analysis of SGDOL's performance on highly imbalanced SECOM dataset and ablation study proves its high efficiency in dealing with complex data streams in a real-world scenario and its network importance respectively.

## ACKNOWLEDGMENT

J. Senthilnath and Md Meftahul Ferdaus acknowledge funding from the Accelerated Materials Development for Manufacturing Program at A*STAR via the AME Programmatic Fund by the Agency for Science, Technology and Research under Grant No. A1898b0043.

## REFERENCES

[1] C. C. Aggarwal, Data streams: models and algorithms. Springer Science Business Media, 2007, vol. 31.

[2] J. Gama, Knowledge discovery from data streams. CRC Press, 2010.

[3] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama and G. Zhang, "Learning under Concept Drift: A Review", IEEE Transactions on Knowledge and Data Engineering, pp. 1-1, 2018. Available: 10.1109/tkde.2018.2876857.

[4] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A.G. Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell , "Overcoming catastrophic forgetting in neural networks", Proceedings of the National Academy of Sciences, vol. 114, no. 13, pp. 3521-3526, 2017. Available: 10.1073/pnas.1611835114.

[5] J. Senthilnath, A. Kumar, A. Jain, K. Harikumar, M. Thapa, S. Suresh, G. Anand and J.A. Benediktsson, "BS-McL: Bilevel Segmentation Framework With Metacognitive Learning for Detection of the Power Lines in UAV Imagery", IEEE Transactions on Geoscience and Remote Sensing, In press, 2021.

[6] Sahoo, Doyen, Q. Pham, J. Lu, and S. CH Hoi. "Online deep learning: Learning deep neural networks on the fly." arXiv preprint arXiv:1711.03705, 2017.

[7] Y. Jaehong, E. Yang, J. Lee, and S. Ju Hwang. "Lifelong learning with dynamically expandable networks." arXiv preprint arXiv:1708.01547, 2017.

[8] A. Andri, M. Pratama, E. Lughofer, and Y. Ong. "DEVDAN: Deep evolving denoising autoencoder." Neurocomputing 390, 2020, pp. 297-314.

[9] A. Andri, and M. Pratama. "Autonomous deep learning: Continual learning approach for dynamic environments." In Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 666-674. Society for Industrial and Applied Mathematics, 2019.

[10] L. Hugo, Y. Bengio, J. Louradour, and P. Lamblin. "Exploring strategies for training deep neural networks." Journal of machine learning research 10, no. 1, 2009.

[11] R. Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review 65, no. 6, 1958, pp. 386.

[12] Z. Martin. "Online convex programming and generalized infinitesimal gradient ascent." In Proceedings of the 20th international conference on machine learning (icml-03), pp. 928-936, 2003.

[13] C. Koby, O. Dekel, J. Keshet, S.S Shwartz, and Y. Singer. "Online passive aggressive algorithms.", 2006.

[14] R. Sylvestre-Alvise, A. Kolesnikov, G. Sperl, and C.H. Lampert. "icarl: Incremental classifier and representation learning." In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 2001-2010, 2017.

[15] K. James, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan et al. "Overcoming catastrophic forgetting in neural networks." Proceedings of the national academy of sciences 114, no. 13, 2017, pp. 3521-3526.

[16] R. Andrei A., 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.

[17] J. McClelland and D. Rumelhart, Learning Internal Representations by Error Propagation. Cambridge, Mass. [u.a.]: MIT Press, 1999, pp. 318–362.

[18] B. Dor, N. Koenigstein, and R. Giryes. "Autoencoders." arXiv preprint arXiv:2003.05991, 2020.

[19] E. Briscoe and J. Feldman, "Conceptual complexity and the bias/variance tradeoff", Cognition, vol. 118, no. 1, pp. 2-16, 2011. Available: 10.1016/j.cognition.2010.10.004.

[20] K. MURPHY, Machine learning: a probabilistic perspective. MIT Press, 2012.

[21] J. Gama, R. Fernandes and R. Rocha, "Decision trees for mining data streams", Intelligent Data Analysis, vol. 10, no. 1, pp. 23-45, 2006. Available: 10.3233/ida-2006-10103.

[22] E. Thomas, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey.". J. Mach. Learn. Res. vol. 20, no. 55, pp. 1-21, 2019.

[23] I.F. Blanco, J.C. Avila, G.R. Jimenez, R.M. Bueno, A.O. Diaz and Y.C. Mota, "Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds", IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 3, pp. 810-823, 2015. Available: 10.1109/tkde.2014.2345382.

[24] M. Pratama, W. Pedrycz, and E. Lughofer. "Evolving ensemble fuzzy classifier." IEEE Transactions on Fuzzy Systems 26, no. 5, 2018, pp. 2552-2567.

[25] A. Liu, Y. Song, G. Zhang and J. Lu, "Regional Concept Drift Detection and Density Synchronized Drift Adaptation", Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017. Available: 10.24963/ijcai.2017/317

[26] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy and A. Bouchachia, "A survey on concept drift adaptation", ACM Computing Surveys, vol. 46, no. 4, pp. 1-37, 2014. Available: 10.1145/2523813.

[27] D. Lopez-Paz and M. Ranzato, Advances in neural information processing systems. NIPS, 2017, pp. 6467–6476.

[28] R.K. Srivastava, J. Masci, S. Kazerounian, F.J. Gomez, and J. Schmidhuber, "Compete to Compute", In NIPS, 2013, pp. 2310-2318.

[29] W.N. Street, and Y.S. Kim. "A streaming ensemble algorithm (SEA) for large-scale classification." In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 377-382, 2001.

[30] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. "Moa: Massive online analysis, a framework for stream classification and clustering." In Proceedings of the First Workshop on Applications of Pattern Analysis, pp. 44-50. PMLR, 2010.

[31] L. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and CO 2 measurements using statistical learning models", Energy and Buildings, vol. 112, pp. 28-39, 2016. Available: 10.1016/j.enbuild.2015.11.071.

[32] S.J. Salvatore, W. Fan, W. Lee, A. Prodromidis, and P.K. Chan. "Cost-based modeling for fraud and intrusion detection: Results from the JAM project." In Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00, vol. 2, pp. 130-144. IEEE, 2000.

[33] A. Arthur, and D. Newman. "UCI machine learning repository.", 2007.

[34] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]." IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141-142, 2012.

[35] P. Baldi, P. Sadowski and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning", Nature Communications, vol. 5, no. 1, 2014. Available: 10.1038/ncomms5308.

[36] Y.H. Jung, J. Goetz, and A. Tewari. "Online multiclass boosting." arXiv preprint arXiv:1702.07305, 2017.

[37] M. Pratama, E. Dimla, T. Tjahjowidodo, W. Pedrycz, and E. Lughofer. "Online tool condition monitoring based on parsimonious ensemble+." IEEE transactions on cybernetics 50, no. 2, 2018, pp. 664-677.

[38] C.O. Nikunj, and J.R. Stuart. "Online bagging and boosting. Jaakkola Tommi and Richardson Thomas, editors." In Eighth international workshop on artificial intelligence and statistics, pp. 105-112. 2001.

[39] L.V. Maaten, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9, no. 11, 2008.