

Article

Hybrid-Based Machine Incremental Learning in K-Nearest Neighbor Heterogeneous Drifting Environment

Japheth Otieno Ondiek *, Kennedy Odhiambo Ogada and Tobias Mwalili

School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, P.O. Box 62000, Nairobi 00200, Kenya; tmwalili@jkuat.ac.ke (T.M.)

* Correspondence: japheth.otieno@students.jkuat.ac.ke

Abstract

The ability to continuously learn over time by incorporating new information while holding onto previously acquired expertise is known as incremental learning (IL). Although this concept is fundamental to human learning, existing machine learning techniques have a significant propensity to forget prior experience by overwriting previously learned patterns from classes. The continuous learning of new information in K-nearest neighbor (KNN) with lazy learning strategies compounds to loss of old knowledge upon learning new information and stability-plasticity dilemma. The change in new data points and data distributions in unforeseen ways impacts KNN's ability to adapt to changes in class label distribution, leading to concept drift. This experiment models a hybrid 3WDKNN-based incremental learning algorithm (ILA) designed for application in a heterogeneous and dynamically changing environment. This model addresses the limitations of KNN by overcoming computational costs and inefficiencies associated with loss of information in classes, while facilitating incremental learning to attain high predictive accuracy in crop yield datasets. The algorithm employs weighted voting to identify optimal assigned classes for the nearest neighbor and uses memory reconstruction strategy for class incremental learning until the memory is full without forgetting. Using weighted voting for the best assigned classes for the nearest neighbor, the algorithm uses a local mean vector to determine the best distances for the shortest-term incremental learning to achieve the highest performance accuracy in a concept drift environment. The hybrid 3WDKNN_ILA was developed and evaluated alongside advanced algorithms within the same dataset context. The model improves performance in incremental learning contexts by utilizing current concepts and minimizing errors on both current and recent data to avoid parameterization. The model achieves optimal efficient incremental learning by mitigating intentional loss and minimizing errors associated with valuable class information derived from aggregated mean values through class rectification and transfer. The hybrid model achieves the best efficient performance accuracy in all the tested weighted averages of 200W, 500W, and 1000W with tested set K values of 5, 9, and 13K. This hybrid model demonstrates performance accuracy of 97% at a value of 13K, whereas 3WD_KNN achieves 96% at 9K, HoKNN attains 89% at 13K, and 1KNN reaches 88% at 9K accuracy, respectively. The integrated novelty in the hybrid 3WDKNN_ILA proves superior in terms of computational efficiency, accuracy, and high-level incremental performance and learning in comparison with other tested models of algorithms.



Academic Editor: Douglas O'Shaughnessy

Received: 10 March 2026

Revised: 19 April 2026

Accepted: 20 April 2026

Published: 29 April 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](#)

[Attribution \(CC BY\)](#) license.

Keywords: increment learning algorithm; hybrid machine learning; K-nearest neighbor (KNN); concept drift; heterogeneous

1. Introduction

Machine learning offers powerful technological approaches including offering an effective broad range of pattern recognitions specifically to infer structural information; however, in digital data extraction, acquisition of representative training data is time-consuming and expensive. It is therefore uncommon to realize that most of these data are streamlined in batches over a period of time in a classical batch process where it is necessary to update the classifiers in an incremental way to accommodate the new digital data without interfering with classification in old data [1]. Incremental learning algorithms (ILA) are dynamic algorithms that represent reinforcement learning, supervised learning algorithms, and unsupervised learning algorithms in training data occasioned by gradual change in data attributes, adjustments of new class labels, and extension of models' knowledge [2]. The understanding of incremental learning is diverse according to Halder, [3]; for instance, in some cases incremental learning has been used to project the selection of the most promising classes. Incremental-based learning algorithms aim at minimizing generalization errors, especially with a growing training set [4].

In the field of machine learning (ML), there are well-established incremental learning strategies for time-varying data and unbounded streams; however, the majority of these incremental classifiers exhibit consistent limitations in solving multi-class instances, overlapping classes, and failed prediction for increased feature space [5]. Even so, incremental learning in KNN is fraught with several challenges due to the introduction of new data characterized by specific circumstances, imbalances, and noise. Although learning seeks to adapt to new data without necessitating retraining for new classes due to computational optimizations, the incorporation of new class instances incurs additional computational insensitivity and expenses. The intensity exacerbates if the neighborhood structure of classes alters, thus impeding the recalculation of distances and the re-identification of nearest neighbors for new data points. Therefore, the expanding datasets get intricate and expensive regarding restructuring and re-indexing for nearest neighbor searches. Other common challenges experienced in incremental learning strategies include concept drift [6]. Concept drift in KNN modeling is the process in which attributes, classes and properties changes due to the introduction of new properties in the model either when time series data or streaming data is introduced, leading to data loss of information or catastrophic forgetting. Catastrophic forgetting is a problem in continual learning whereby a model loses previously acquired knowledge when exposed to new tasks, complicating the retention of performance on earlier tasks throughout the acquisition of new ones. The understanding is that continuous learning applications, especially those dealing with limited resources, are not able to solve this problem since they experience catastrophic forgetting behavior [7]. Ultimately, the curse of dimensionality associated with high-dimensional data escalates progressively, necessitating greater processing resources and augmenting memory consumption in KNN models [3,8]. This is one of the prominent problems exhibited in streaming real-time data where the sample $(\bar{x}_1, y_1)$ experiences changes in statistics [5]. Consequently, the input distribution $(p(\bar{x}_1))$, which is referred to as virtual concept drift, changes the functionality $(p(y|\bar{x}_1))$, which is the real concept drift. In an ideal scenario, concept drift can happen in two different contexts: the first is where local concept drift happens—this involves a situation where changes in data statistics happens in specific data spaces and locality [5]. In addition, real concept drift occurs when new classes are introduced to existing data, creating conflicts in the classification. Second, concept drift can occur in KNN due to the stability–plasticity dilemma; this is a well-known constraint even in biological neural systems and artificial systems. In incremental learning, loss of information when new patterns overwrite old patterns creates what is universally called the stability–plasticity dilemma. Similarly, a combination of extreme incremental learning

for data streaming classification and dynamic incremental-supervised methods have failed to integrate small-size clusters of classes in concept drift [9].

Learning in streaming data or time series data which arrives over time, especially in changing environments, comprises prediction accuracy in memory resource-constrained applications in KNN applications and models [1,5]. This scenario creates real concept drift problems with impacts on classification performance created by a visually dissimilar object class. Equally, adaptation to incremental learning in KNN is compounded by the stability–plasticity dilemma as well as known constraints that also bedevil artificial and biological learning systems [4].

A number of approaches towards intelligent adaptation such as self-adjustments, incremental architecture or prototype-based model strategies have the potential to solve the crisis of resources [10]. Although class-incremental learning has been designed to integrate new data classes while alleviating loss of old knowledge upon learning new information, it often assumes the presence of labeled data for supervised training. This dependence on substantial labeled data poses significant difficulties in real-world data streams characterized by rapid data influx and constrained annotation resources. Additionally, promising studies involving a multilayer neuroadaptive reinforcement learning algorithm utilizing the actor–critic mechanism have been suggested for the tracking control of high-dimensional uncertain nonlinear systems [11]. These models mitigate uncertainty using actor multilayer neural networks and extended state observer-based disturbance estimators. The utilization of critic multilayer neural network-based reinforcement signals enhances the approximation efficacy of the actor multilayer neural networks and increases tracking performance in high-dimensional systems [12]. According to Belouadah [11], it is imperative that some machine learning models offer seamless memory transfer models or model parameters which store information of typical examples. Therefore, in predictive models, settings require optimum memory adaptation design since it directly reflects on the plasticity–stability dilemma [11]. Another challenge comprises model benchmarking, which fundamentally offer two distinctive features of assessing performance of incremental learning algorithms (ILA). First, in incremental vs. non-incremental learning, the inference of stationary learning distribution ($p(y|\mathbf{I}\bar{x}_1)$) for data is characterized by ($p(\bar{x}_1)$) [10]. This implies that whenever incremental learning algorithms are used for big datasets, there is eminent competition with parallelized batch algorithms where the method of choice informs evaluation of classification accuracy M_t on test set data.

Following the underlying limitations highlighted above, this study combines existing strengths to model a super KNN-based hybrid that equates to solving problems of incremental learning and concept drifting caused by time series variable datasets in crop yield prediction. Against this background, characterization of incremental learning for enhancing requirements of other existing algorithms, like in KNN, to improve their performance needs to be established. Satisfying the problem statements from this study, a number of conditions from incremental learning algorithms need to be implemented to support the hybrid model as follows: (1) the suggested improved hybrid model should be able to accommodate new classes introduced with different variable datasets without running out of computation power; (2) previously acquired knowledge about prediction and environmental prediction should not be forgotten; (3) the algorithms should not require access to data used to train the existing classifier, as this would slow down the iteration process and increase computing power; and (4) the algorithm should optimally learn new information from new data, including memory reconstruction and transfer of data for the classes. It is from these properties that the researcher would develop an indispensable hybrid model for crop prediction. The problem addressed in this study is designing a

hybrid machine model using a combination of incremental algorithm, KNN and concept drift detection technique.

To solve the existing problems occurring within the embedded KNN incremental learning environment, we frame the following research questions to prove our contributions to the study.

RQ_1: Does the new model provide better accuracy and performance in the context of catastrophic forgetting and concept drifting in incremental learning model in both short-term incremental learning (STIL) and long-term incremental learning (LTIL)?

RQ_2: How does the hybrid model store and retain exemplar data, or construct memory of learned features of old data through compression to preserve essential information in the old classes by improving learning in short-term incremental learning (STIL) and long-term incremental learning (LTIL)?

RQ_3: How does the proposed hybrid classification model remain more competitive in classification accuracy and efficiency than the existing state of abrupt concept drift detection for crop yield dataset distribution in comparison to other state-of-the-art models?

RQ_4: How do the dynamic weight size (w) and K values impact the meantime average accuracy performance of the hybrid 3WDKNN_ILA classification model in comparison to other state-of-the-art models?

Motivation and Contribution

First, our contribution models the adaptive hybrid KNN_ILA that addresses the memory challenges for the incoming data streams. In the experiment, an efficient 3WDKNN_ILA incremental learning in concept drift model is proposed. The model uses memory reconstruction strategy for class incremental learning where training data is conducted for a smaller number of classes in the STIL as the new classes with instances are progressively added until the memory is full without forgetting through rectification, cleaning and transfer of classes. The datasets for the old classes are then released to the LTIL memory for continuous learning without additional computational inputs and resources. This combination of approaches defines our first novelty towards achieving a hybrid model with three-way decision and incremental learning adaptation.

Second, we model the adaptive hybrid KNN_ILA that addresses the fitting challenges for the incoming data streams, employing dynamically updated weighted voting majority for the classifiers using the three-way decision approach for improving learning through adjusting errors on both current and past datasets automatically. This approach has the ability to handle incremental learning in the heterogeneous environment by handling concept drift variations created by the incoming classes and new content using transfer of existing classes or additional of new classes from short-term incremental to long-term incremental classes. In addition, our hybrid incremental learning model reduces error with incremental learning during operating in productions by improving time-complexity of K-nearest neighbors classifier using weighted and K values in incremental learning approach.

Third, our contribution encompasses creating a speed-up process of minimum distance for the KNN classifiers in the classification model through speed-up classification in the dynamic environment and in the search procedure for the nearest neighbors' procedure. We create a centroid local mean distance classifier using the Euclidean distance for classification by averaging the attribute values for nearest neighbors to enhance the incremental learning process. This is achieved when the new incoming classes to be classified are set such that the distance is set to the nearest local centroid to avoid a high computational cycle of scanning the entirety of the training data and avoid loss of crucial class information.

The subsequent sections of this document are organized as follows: Initially, we examine pertinent introductory literature on incremental learning contexts. Subsequently, we present a state-of-the-art literature review and then delineate our experimental technique and methods for modeling the hybrid algorithm. We then conduct detailed analysis of the model performance and discussions and conclude with a brief experimental conclusion and limitations.

2. Literature Review

This research investigates various models through investigating diverse literature relevant to incremental learning models. Since the modeling of the algorithm depends on incremental models, it is important to discuss the trends, generalizability, learning behavior and convergence speeds in the context of predictions. Multilayer Perceptron (MLP) and Radial Basis Function (RBF), which are common paradigms of neural networks, are typical of learning new information, discarding existing classifiers, and retraining the data using accumulated data [13]. In biologically inspired incremental learning, various terms such as lifelong learning, constructive learning and evolutionary learning have been used to predict the acquiring of new information.

Evolving in a dynamic environment calls for consistent learning and updating of their capabilities by injecting new data. Depending on one's theoretical information, incremental learning, lifelong learning and continual learning include the challenge of loss of old knowledge upon learning new information. Past comparative studies have given deep analysis on increasing data and model capacity but have fallen short of researching incremental algorithms. Jain et al. [14] proposed a set of properties in incremental algorithms that describe understanding including complexity, memory accuracy, timeliness, plasticity and scalability. According to Belouadah et al. [4], they conducted a series of incremental algorithms by integrating test methods in open resources and repositories and concluded that none of the existing class incremental algorithms are superior to the others in all experimental configurations. The difference was noted only when testing incremental learning without or with memory of past classes. This significantly opens the research problem in the area of incremental learning and its application in hybrid models.

Another model that is proposed is the growth model. This model, according to Belouadah et al. [15], derives its strengths from the increased size of deep models to integrate new knowledge. Yu [16] modeled the growing brain method anchored on deepening its network by increasing representation. Roy et al. [17] gravitated towards self-growth. This model is based on reorganization that features a hierarchy when new data arrives. Similarly, Rosenfeld [9] created Deep Adaptation Network (DAN) as a model growth approach where new tasks are populated with new parameters. On the other hand, Mallya et al. [18] introduced Pack Net which employs a pruning technique based on identifying redundant parameters as well as using them to train new tasks. However, the approach was proven to add more complexity to new tasks, thus creating redundancy. Interestingly, Memory Aware Synapses (MAS) which had limited application, such as face recognition, was introduced to identify the most important weights in the model. The model, having been designed to work on unlabeled data, thrives on looking at the sensitivity of the derived output function and not the loss of the task. Incremental Learning Vector Quantization (ILVQ) by Xu et al. [19] demonstrated a prototype-based classifier which does not depend on prior knowledge of the prototypes or initial value. This uses a threshold-based insertion scheme using the data distribution method to know the number of prototypes designed for each class. This approach received a drawback since it continuously increases the architecture of learned patterns in incremental learning, thus creating complexities.

On fixed representation, the method does not update the incremental state by deep representation; however, it can be classified as a basic variant of fine-tuned base methods. Therefore, Deep Shallow Incremental Learning (DeesiL) by Belouadah et al. [15] was introduced as a simple transfer learning scheme to incrementally learn new representations of classes using the battery of SVM and the first batch of classes. FearNet, proposed by Kemker and Kanan [20], also introduced biologically inspired incremental learning where separate networks are implemented for short-term and long-term measures to accommodate new classes. Subsequently, a decision framework was implemented to sieve the network which should be preferred for each test example. FearNet is promising, but its challenges arise when memory increases significantly with time, especially when detailed statistics of learned classes create a crisis for the algorithms.

Similarly, Hayes & Kanan [21] implemented a Deep Streaming Linear Discriminant Analysis (DeepSLDA) on an online algorithm approach based on training of classes, which are frozen afterwards. In the process of training, covariance mix and class-specific running vectors are updated. Another online approach called Replay using Memory INDEXing (REMIND) was introduced based on initial representation that is partially updated upwards afterwards. Further, it was developed and inspired by hippocampal indexing theory using vector quantization process to store compressed images, which would then be replayed for memory consolidation. Even though vector quantization is widely applied in unsupervised incremental learning [22], Adaptive Resonance Theory (ART) was constructed with variant vector quantization specifically as a tradeoff between stability and plasticity during online incremental learning.

As per the fine-tuning methods that employ distillations, Zhao et al. [23,24] attempted to reduce the loss of old knowledge upon learning new information observed in incremental learning. Self-distillation is derived from knowledge distillation, a scenario that imitates the relationship in network architecture of a student–teacher relationship. In the past, learning without forgetting was proposed to work without requiring memory of past classes [25]. It minimizes disparities by leveraging knowledge distillation of previous classes and current incremental learning states. Convergence is achieved when learning without forgetting performs an introductory setup by only training new ones and freezing past parameters. Classifier and Representation Learning (iCaRL), introduced by Rebuffi et al. [26], uses a combination of memory of past classes and Learning without Forgetting (LwF). Instead of using raw scores that have been predicted through the incremental network, classification is conducted using the nearest mean of the exemplar method. Furthermore, the external classifiers can be deployed to favor new classes through reducing prediction bias, occurring due to data imbalance between new and past classes. iCaRL is therefore reduced to its most vital components of distillation loss and fixed-size memory. Li et al. [27] surveyed another distillation-based system that trains new classes and a separate network for past classes thereafter, integrating them using double distillation loss. Additionally, Hou et al. [28] presented Learning a Unified Classifier Incrementally via Rebalancing (LUCIR) method based on (a) less-forget constraints that change usual distillation loss, (b) cosine normalization that balances new classes and past probabilities, and (c) inter-class separation that separates new and past classes based on [29]’s experimental findings. In this study, incremental learning algorithms and their characteristics, featuring model-based growth, fixed representation-based models, and fine-tuning based models, are referenced and summarized in Table 1 below.

Table 1. Summarized incremental learning algorithms and their characteristics.

Incremental Learning Algorithm	Characteristics of the Model	Model-Based Growth	Fixed Representation-Based Models	Fine-Tuning Based Models	References Reviewed
Progressive Neural Networks (PNN), Deep Adaptation Network (DAN), PackNet, Piggyback	Complexity	This model is based on adding parameters and weights to include new knowledge. The challenge experienced is the optimization of model growth on performance	Here, additional parameters are included to improve past classes' performance. What happens is that the model is fixed after initial non-incremental steps are conducted	This model is programmed to work with fixed structures. Here, parameters are affected by classification layers specifically to reduce the imbalance	[9,18,24,30]
Memory Aware Synapses (MAS), Self-Organizing Maps (SOMs), Neural Gas (NG), PROjection-PREdiction (PROPRE), Hybrid_KNN (HoKNN)	Memory	Memory is allocated to model parameters using weights instead of raw data for past classes	No updates of incremental learning are conducted, hence low dependency of past classes	Performance is subscribed and is dependent on the past memory	[31–33]
Dynamic Online Growing Neural Gas (DYNG), Topology-Preserving Knowledge Incremental (TOPIC), Incremental Learning Vector Quantization (ILVQ)	Accuracy	Have low performance accuracy unlike the FT-based ones. Performance growth comes closer to that of classical learning	Projects lower accuracy compared to FT since the model is not incrementally updated. Initial model should be learned with new classes	Knowledge distillation is used here, which has improved performance	[19,21,34,35]
Neural Gas with local Principal Component Analysis (NGPCA), Classifier Weights Scaling for Class Incremental Learning (ScaIL), Standardization of Initial Weights (SIW), Semantic Drift Compensation (SDC), Generative Adversarial Networks (GAN)	Timelines	In terms of timeliness, the complexity of model growth compares to FT-based methods	New knowledge is integrated in a timely manner, and the classifier weights layer needs to be trained	Classes are not recognizable, especially new classes, until they are retrained	[16,26,36]
Incremental via Rebalancing (LUCIR), PODnet, Bias Correction (Bic), Class Incremental Learning with Dual Memory (IL2M), Incremental synthetic aperture radar (SAR).	Plasticity	Challenge of minimizing the number of parameters needed to accommodate new tasks	Plasticity is limited and performance drops significantly in the event that tasks change	This model enables updates to new data as they are streamed in the model	[6,37]
Learn without Forgetting (LwF), Incremental Classifier and Representation Learning iCaRL, Learning without Memorizing (LwM)	Stability	The methods adapt well to tasks or new classes as the systems deployed have sufficient resources	Dependence on bounded memory is limited. This method can include large number of classes. Learned class weights can be in the initial state which can be reused later	Bounded memory determines the number of past classes which can be stored and, at the same time, new classes can still be recognized	[2,38,39]

Table 1. Cont.

Incremental Learning Algorithm	Characteristics of the Model	Model-Based Growth	Fixed Representation-Based Models	Fine-Tuning Based Models	References Reviewed
Adaptive Resonance Theory (ART), Deep Shallow Incremental Learning DeeSIL, FearNet, Deep Streaming Linear Discriminant Analysis (Deep SLDA), Replay using memory INDEXing (REMIND)	Global assessment	It is noted that this approach works better with new data and is not dependent on memory of the past. They are scalable to large datasets; however, they have complexity issues	Projects the advantage of transferring learning since model complexity can be updated on time as it is constant. Additionally, they have low dependency and can scale to a large number of classes. They are utilized when data variability is low and when updates are needed	FT becomes adequate when architecture complexities of plasticity and complexities are optimized. Here, timeliness is not optimal since they require network retraining. Scaling is also difficult as the memory will be full to accommodate fewer past classes. Are utilized when storage space for classes are available, when immediate use of updated parameters is not very essential	[16,18,23,26,40–42]

Source: Authors' own analysis and summary of incremental learning and their classification tasks.

3. Material and Methods

Incremental learning utilizes large amounts of input data. The suggested incremental learning process involves steps such as pre-processing, feature extraction, feature selection, and incremental learning, which are implemented in modeled hybrid K-nearest neighbor (KNN) and incremental learning configurations.

Experimental set-up: Using existing and well-known datasets, this experiment models and test the incremental learning-based KNN with other frequently used classifiers and well-known state-of-the-art algorithms in streaming data for incremental classes in the MOA framework implementations. The novel model implements a hybrid machine learning approach using the KNN algorithm and incremental learning models integrated with weighted features and the optimum K values. The experiment is modeled with a KNN contribution that deals with heterogeneous concept drift using real-world datasets. We implement hybrid 3WDKNN-based incremental learning under concept drift in consideration of limited processing time and memory consumption. In addition, the implementation is modeled based on the short-term incremental learning (STIL) and long-term incremental learning (LTIL) framework in heterogeneous and streaming datasets. The parameters for the STIL represent current concept drift of the dynamic sliding window with m examples of data streams, $M_{STL} = \{(x_i, y_i)\} \in \mathbb{R}^n \times \{\{1, \dots, C\} | I = 1, \dots, p\}$, and LTIL is based on the class-based timing threshold set in the model. The implementation method considers the new incoming data, along with acquired knowledge, which is transmitted to subsequent classes for ongoing learning, thereby enhancing the model's generalization capabilities. The implementation of LTIL and STIL is employed with weighted classifications of 200W, 500W, and 1000W at the K values of 5, 9 and 13 and the management of overlearning and underlearning in classes.

Our implementation considers changes in datasets, incremental learning shortfalls and concept drift in classes. In our experiment and modeling, we consider the combination of KNN classification and incremental learning for crop prediction which can be applied without any parameterization. This experiment also considers that there is the challenge of model adaptation at any time by processing the incoming order of datasets, thereby violating the hypothesis of sample independence for incoming data. The model employed

high-dimensional heterogeneous datasets and utilized a combination of complex hybrid models. To mitigate potential overlearning in LTIL for weighted average classification at 1000W, we developed our model utilizing a general regularization technique that enforces early learning by halting class overloading immediately upon an increase in generalization error during optimization steps. This improvement in our model addresses key computing limitations by preventing memory overloading, reducing iterations that lead to computational inefficiency and time wastage, and implementing an early stopping mechanism that does not alter the constraint parameter vector or the loss function affecting prediction space complexity.

The accuracy is measured by cumulative means of errors over time, correct prediction values, and loss functions. This accuracy method is preferred as it enables accurate metrics. Crop yield datasets from OMDENA (<https://datasets.omdena.com/dataset/crop-yield-prediction>, accessed on 10 April 2024) Cropyield_1 and KAGGLE Cropyield_2 were experimented on with one other object recognition dataset from CORE50 datasets [43], with different generator instances of $K = 5, 9, 13$ and weighted at 200W, 500W and 1000W from the MOA framework. Incremental learning was distributed equally in all datasets and simulated in abrupt concept drifts, and loss of old knowledge upon learning new information was implemented by a probability function embedded in the MOA framework. The means for all the datasets were uniformly executed at intervals of 95% confidence levels. Experimentation results were then extracted and analyzed using the MOA framework (<https://moa.cms.waikato.ac.nz/downloads/>), platform tool version 2024.07, University of Waikato, New Zeland downloaded on 10 August 2024. Reactive Drift Detection Method (RDDM) was chosen to generate catastrophic forgetting and abrupt concept drift on the integrated datasets with three sizes of K values at $K = 5, 9, 13$ instances. RDDM was chosen because it improves sensitivity when dealing with long concepts drifts by reducing the number of instances and accurately prompting detection of drifts. Figure 1 illustrates the hybrid 3WDKNN_ILA model configuration below.

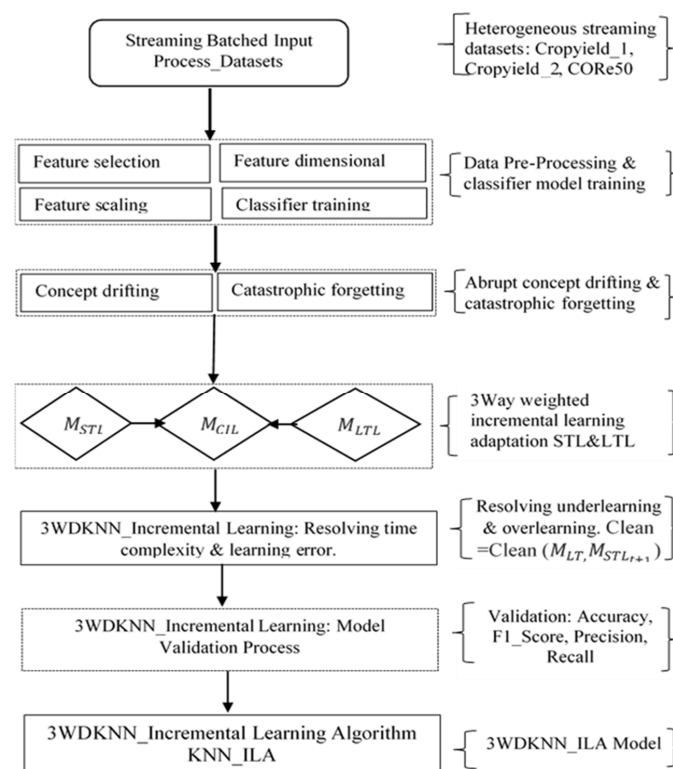


Figure 1. Authors' own illustrations and Hybrid 3WD K-nearest neighbor (KNN) and incremental learning model configuration.

3.1. Datasets

Data pre-processing: Two main testing crop yield datasets were imported from different datasets, OMDENA and KAGGLE. Mainly, the datasets considered from the (Massive Online Analysis) MOA framework were considered since they were already integrated in the framework, were mainly in data streaming, and had a history of incremental learning features and characteristics, which influenced the dataset selection. CORE50, specifically designed for assessing Continual Learning techniques in an Object Recognition context, was extracted from <https://vlomonaco.github.io/core50/index.html#dataset>, accessed on 10 April 2024. RDDM generator was used to create streaming sequence of batches for the two main crop yield datasets, Crop yield_Datasets1 [44] and Crop yield_Datasets2 [45] files respectively, based on $t + 1$ or $t - 1$. The datasets were integrated within the framework and were used to generate batched and streaming data that are readily used for loss of old knowledge upon learning new information and incremental learning. While splitting data, we continued checking on feature distribution because missing data could lead to incorrect assumptions. Data was processed through cleaning, normalization, transformation, and sorting of missing values and imbalanced data treatment. Data splitting was performed in the MOA framework for incremental learning and concept drift, considering the incoming datasets for modeling the KNN classifier were set as follows:

$$\text{Let } S = \{ \langle X_i, Y_i \rangle, \dots, \langle X_i, Y_i \rangle \}$$

Splitting S randomly into training sets and test sets proportionally for the modeling of the KNN classifier, the best classifiers are selected (considerations for the best classifier is categorized for the highest classification quality of the incoming datasets in extremely fast computational methods). Even though KNN does not build the model in training, the stored datasets are used to make decisions in the predictions; however, this is prone to memory loading due to storage of entire datasets.

For the training and test sets containing S and $S - s$ ($s > S$) dataset tuples, the datasets considered have the following properties as summarized in Table 2 below.

Table 2. Authors' own summary of number of instances, attributes, and classes for the tested datasets.

Datasets	#Instances	#Attributes	#Classes
CORE50 Datasets [46]	1000	300	10
CROP1_Datasets [44]	300	2	7
CROP2_Datasets [45]	300	4	6

In each of the tested algorithms and datasets, a hyper-parameter selection, computed accuracy values with 5-fold cross-validation, and 95% confidence intervals for all possible implementations around the mean were reported.

3.2. Data Processing and Feature Selection

In this experiment, the components of incremental learning techniques are subjected to sensitive changes. This means that if the training is deemed to be too short, then the components of the hybrid will not generate good results. On the other hand, if the training is too long, then it creates too much computational time, which results in poor generalization and overlearning. In this circumstance, generalizations and underlearning were prevented by establishing a validation setup that included an over-stopping criterion. Therefore, this experiment is divided into three sets: a training set, which is used to train the datasets; a validation set, which examines the quality of classes while the training is being done; and a check set, which is used to keep an eye on whether or not the datasets are being

overlearned. Last but not least, a test set is utilized at the conclusion of the training process in order to validate the results that were obtained.

Evaluation framework:

To ascertain accuracy of the KNN_ILA hybrid model, evaluation of the model's performance on the newly learned classes and the old past classes was performed to understand the overall effectiveness of the incremental learning approach in heterogeneous datasets and concept drift. The evaluation metrics used were accuracy, F_1 score, Precision and Recall to understand the performance accuracy and adaptability. We can note from the model that it swiftly learns new information in order to reach a certain level of performance. In the post-modeling evaluation, the hybrid algorithms were compared to other benchmarked processes based on the intended properties. The experiment benchmarked well-established incremental models utilizing KNN; however, they may not be as widely recognized as the classic ones proposed below. For instance, HoKNN (hybrid online KNN) was introduced as an efficient online learning method based on the KNN algorithm. This model has been effectively utilized in autonomous wall-following robot navigation and was selected for its robust incremental learning capabilities using various heterogeneous datasets for evaluation. Furthermore, an inverse-distance-weighted K-nearest neighbors (IKNN) model was used to benchmark with introduced parameters to enhance diagnostic sensitivity by accounting for feature-space distance variations. The 3WD_KNN is a developed version by the authors and was used to evaluate the improved and proposed hybrid 3WDKNN_ILA model.

4. Model Adaption: Incremental Learning in KNN

4.1. Assigning KNN Classifier Model

For the KNN classifier, the value consideration for K value neighbors is determined where the error is minimal. This process is outlined to set the best KNN classifiers for the nearest distance using the local mean vector and deriving the most votes for majority rule as a step towards assigning the best classes in the nearest neighborhood. The classes belonging to $X_i \in Y$ for the object $X_i \in Z$ are considered for the K -nearest neighbor for the object $X_i \in Z$.

The following implementation steps for determining the classes of K -nearest neighbors for the object class is employed.

Step 1: Determine the distance for classes based $d(X_i, X)$ from the objects S to S_i . The classes are conducted on ascending order of their values.

Step 2: Conduct selection of K objections for X_i , (K -nearest neighbors) that are closely related to object from X .

Step 3: Find the classes belonging to K -nearest neighbors for the objects X . Classes of nearest neighbours that are common are assigned to class object X . Apply the modified Euclidean distance metrics as follows:

Data class is tested using the closest distance and the local mean vector for the distances between the two classes, using the following equation:

$$d_{xt} = \sqrt{\sum_{j=1}^n (x_{sj} - y_{tj})^2} \quad (1)$$

In implementing the KNN algorithms, we employed weighted voting to enhance accuracy through improving object X to each nearest neighbor for object classes.

For each KNN, $X_i = i = (\overline{1, K})$ of objects in X voters to assign the class of $Y_{X_i, X}$; the resultant implementation of KNN algorithm is assigned to the class that derives the most votes from majority rule as depicted:

$$\alpha = \arg \max_{y \in Y} \sum_{t=1}^k |y_{X_i, X} - y| \quad (2)$$

Using the weighted voting rule, we note the object X distance of the nearest neighbor $X_i = (i = \overline{1, K})$. Hence, the smaller the distance, the more the votes of $X_i = i = (\overline{1, K})$ are considered the major contributors to the object classes. Therefore, the total contribution of object X for the classes of $Y \in Y$ in weighted voting is considered as

$$\alpha = \sum_{i=1}^k \frac{1}{d^2(X_i, x)} \cdot r_i \quad (3)$$

where $r_i = 0$ if $X_{Z_i, Z} \neq Y$ and $r_i = 1$ if $Y_{Z_i, Z} \neq Y$. This assessment aids in assigning the best classes in the nearest neighborhood and assigning classes.

4.2. Incremental Learning for Concept Drifting Data in KNN

The algorithm is designed to learn incrementally, ensuring that any previous data from earlier datasets is accounted for in the storage of classifiers generated from those datasets. Thus, the drift allows the algorithm to temporarily retain or discard knowledge through tuning for categorization and activation to identify new data.

Consider the algorithm in arriving training datasets where

$$D^t = X^t(i) \varepsilon X_j y^t(i) \varepsilon \gamma, i = 1, \dots, m^t \quad (4)$$

where t is depicted as the time index.

Therefore, $X^t(i)$ is considered the (i^{th}) instance for the incoming datasets, while $P^t(x, y)$ is drawn from the unknown distribution with consideration of time (t) .

Incremental learning at a time $(t + 1)$ is obtained in the datasets drawn from $P^{t+1}(x, y)$. Each time stamp may represent change or no change; therefore, the rate of this change is not assumed to be constant due to possible changes, and that is also not known.

Concept drift in classes occurs when the distribution of classes changes or occurs at the two times steps t_0 and t_1 . Here, we consider concept drift in classes if there is a change in dataset class and time, denoted as the class distribution $C_t(x, y)$ for changes in initial time t_0 and latest t_1 :

$$\exists \times C_{t_0}(x, y) \neq P_{t_1}(x, y)$$

Considering the drift

$$C_t(x, y) = C_t(x) C_t\left(\frac{y}{x}\right); \quad (5)$$

where $C_t(x)$ is the distribution of class features and $C_t(x/y)$ considers the probability of the classes.

We can consider the real drift or virtual drift, which are conceptually regarded as active concept drift or change t as detected, and the learning is discarded at this point. In a passive approach, there is a continuous adaptation of the model where it is not clearly known whether there is a drift occurring. In the architecture illustrations, we consider long-term incremental and short-term incremental learning methods. The assumption for LTIL and STIL is based on the ability of classes to learn as much as possible in the short term, with the determination of nearest neighbourhood and time, and transfer the rest of the

learning to the other classes further away based on time. We illustrate the general approach to classification of concept drift for classes.

To maximize the prediction accuracy, we model the current concept $P_t(x, y)$ including the earlier ones. $P_{t-1}(x, y) \dots P_1(x, y)$ is considered short-term incremental learning containing the current data concept and long-term incremental learning maintaining knowledge of past concepts, respectively. We exploit minimization of error on the current and recent data as much as possible to avoid any parameterization.

Batched Datasets

In our classification-based batched datasets, we employ an infinite sequence of data batches. We assume they arrive at different time stamps.

The sequence of arrival $S = (S_1, S_2, \dots, S_t)$ of the tuples $S_i = (x_i, y_i)$ are batched after another.

If we take t as the current time stamp of arriving data, we then need to predict label y_t for every input denoted as x_t , which is unknown.

We determine the loss $L(\hat{y}_t, y_t)$ from the prediction $(\hat{y}_t, y_t)$ based on the earlier model h_{t-1} denoted as,

$$h_{t-1} = \text{train}(h_{t-1}, S_t)$$

Calculating the error sequence based on the current time is noted by

$$E(S) = \frac{1}{t} \sum_{i=1}^t 1(h_{t-1}(x_i) \neq y_i) \quad (6)$$

In batched process datasets, streaming data inherit compounding challenges of model adaptation with incoming data requiring processing, thus violating the assumption of sample dataset independence.

In the prediction of the target variable, $y \in \{1, \dots, C\}$ is a set of features of $X \in \mathbb{R}^n$.

The model is based on the training set

$$D_{\text{train}} = \{(x_i, y_i) \mid (X_i, y_i) \mid i \in \{1, \dots, j\}\} \quad (7)$$

The notation adopted as alternate is the following:

$$y(x_i) = y_i$$

Subsequently, the model is based on the subsequent test set which we use on the notation as

$D_{\text{test}} = \{(x_i, y_i) \mid (X_i, y_i) \mid i \in \{1, \dots, j\}\} \dots$ with the labels remaining hidden. Hence, the prediction label is set $\hat{y}_i = h(x_i)$ at each point $x_i \in D_{\text{test}}$.

We calculate loss:

$$L(\hat{y}_i, y_i) = 1(\hat{y}_i \neq y_i) \quad (8)$$

In this experiment, the pseudocode in Algorithm 1 is summarized in the section below.

Algorithm 1: Determining the distance and training the model

```

Input: Determine the distance for classes based  $d(X_i, X)$  from the objects  $S$  to  $S_i$ .
from collections IMPORT Counter
DEFINE FUNCTION euclidean distance  $d(X_i, X)$  (point1, point2):
    RETURN np.sqrt(np.sum((np.array(point1) - np.array(point2))**2))
DEFINE FUNCTION knn_predict(training_data, training_labels, test_point, k):
    SET distances TO  $[d_{xt}]$ 
    FOR i IN range(len(training_data)):
        SET dist TO euclidean_distance( $D_{train\_point}$ ,  $tD_{train\_data}[i]$ )
        distances.append((dist, training_labels[i]))
    distances.sort(key=lambda x: x[0])
    SET k_nearest_labels TO [label FOR _, label IN distances[:k]]
    RETURN Counter(k_nearest_labels).most_common(1)[0][0]
DEFINE FUNCTION main():
    SET training_data TO [[1, 2], [2, 3], [3, 4], [6, 7], [7, 8]]
    SET training_labels TO ['S1', 'S2' ... 'St']
    SET test_point TO  $[x_i, y_i]$ 
    SET k TO [3, 5, 9, 13]
    knn_predict(training_data, training_labels, test_point, k)
IF __name__ EQUALS '__main__':
    main()

```

4.3. Incremental Learning Adaption Accuracy with Weights

We represent the learning sets as M_{ST} , M_{LT} , M_C .

Each incremental learning is subset-based.

The length of $\mathbb{R}^n \times \{1, \dots, C\}$ is to be adjusted in the adaptation process. Short-term learning (STL) represents the current concept drift with a dynamic sliding window with m examples of data streams $\{1, \dots, C\}$.

$$M_{STL} = \{(x_i, y_i)\} \in \mathbb{R}^n \times \{\{1, \dots, C\} | I = 1, \dots, p\}.$$

We combined incremental learning with memories using the following equation:

$$M_{CIL} = M_{STL} \cup M_{LTL} \dots$$

We imply that every set induces a classifier where the distance weighted is denoted by

$$\mathbb{R}^n \rightarrow \{1, \dots, C\}, KNN_{MSTL}, KNN_{LTL}, KNN_{MCL},$$

When the KNN function assigns a label for point x based on a given point x for a set of Z . This notation is given as

$$Z = \{(x_i, y_i)\} \in \mathbb{R}^n \times \{\{1, \dots, C\} | I = 1, \dots, n\}$$

Therefore,

$$KNN_Z(x) = \underset{y_i \in Z}{\operatorname{argmax}} \left\{ \sum_{x_i \in N_K(X, Z)} \frac{1}{d(x_i, x)} \mid C = 1, \dots, C \right\};$$

$d(x_1 x_2)$ is given as the Euclidean distance for two points.

Then, $N_K(X, Z)$ sets KNN of X in Z .

Considering weights W_{STL} , W_{LTL} , W_C , representing current concept drift. We define the complete incremental model for a given point (X) as

$$X \rightarrow \left\{ \begin{array}{l} KNN_{M_{STL}}(x) \text{ if } W_{STL} \geq \max(W_{LT}, W_{IC}) \\ KNN_{M_{LTL}}(x) \text{ if } W_{LTL} \geq \max(W_{STL}, W_{IC}) \\ KNN_{M_{IC}}(x) \text{ if } W_{IC} \geq \max(W_{STL}, W_{LTL}) \end{array} \right\} \quad (9)$$

We adopt this model incrementally every time. Whenever STIL is reduced in learning, its discarded knowledge is transferred to LTL for consideration in prediction.

We set the incremental learnings by representing sets IL_{ST} , IL_{LT} , M_C , as subsets of $\mathbb{R}^n \times \{1 \dots C\}$ of varying length for the adjusted adaptation process.

4.4. Incremental Learning in the Classification_Catastrophic Forgetting

When data samples arrive, the short-term learning process is triggered until S_{max} is exhausted. When this happens, the previous and oldest samples are discarded progressively for each incoming dataset. By holding onto the most recent concept for the STL after reduction and at every iteration, we denote

$$M_L = \{(X_t - 1 + 1, Y_{t-1+1}), \dots, (x_t, y_t)\}$$

where we consider

$$l \in \left\{ m, \frac{m}{2}, \frac{m}{4}, \dots \right\} \text{ and } l > S_{min}$$

$$M_{STL_{t+1}} = \underset{s \in \{M_m, M_{m/2}, \dots\}}{\operatorname{argmin}} \quad (10)$$

When STIL has few concepts to learn from, we consider the dataset sample θ_t to be discarded.

$$\theta_t = M_{STL_t}, M_{STL_{t+1}}$$

This happens as the new datasets with different data features and instances are let in. A set of A classes is cleaned by another set of B ($x_i, y_i \in B$)

$$\text{Clean} : (A, B, (x_i, y_i \in B)) \rightarrow \hat{A}$$

where $A, B, \hat{A} \subset \mathbb{R}^n \times \mathbb{R}$ and $(x_i, y_i) \in B$

We define $\hat{A}$ in the following steps:

- (i) Firstly, the K -nearest neighbor is determined through x_i by finding the maximum distance.

$$\Delta_x^* = \max \{d(x_i, x) | x \in N_K(x_i, B) \setminus (x_i, y_i)\}$$

- (ii) Secondly, we compute the maximum weighted difference for y_i and $y \in N_K(x_i, B \setminus (x_i, y))$.

$$\Delta^* = \max \left(\frac{y_i - y}{x_i, x / e\Delta_x^*} \mid y \in N_K(x_i, B \setminus (x_i, y)) \right)$$

- (iii) Thirdly, all class samples in A are determined within the Δ_x^* of x_i .

$$\mathbb{C} = \{(x, y) \in A | d(x_i, x) < \Delta_x^*\}$$

- (iv) Fourthly, weighted difference is computed for y_i and $y \in \mathbb{C}$.

$$\Delta_y = \left\{ \left(\frac{y_i - y}{x_i, x / e\Delta_x^*} \mid y \in \mathbb{C} \right) \right\}$$

Therefore, cleaning operations for the full set of B is set as follows:

Clean: $(A, B) \hat{A}_{|B|}$

This is iteratively applied to former cleaning for all datasets $(x, y) \in B$.

$$\hat{A}_0 = A$$

$$\hat{A}_{t+1} = \text{Clean}(\hat{A}, B(x_{t+1}, y_{t+1}))$$

When all the processes of cleaning have occurred in the nearest neighbor, the old datasets are discarded at θ_t which is then denoted as

Clean $(\theta_t, M_{STL_{t+1}})$

For the clean datasets,

$$\text{Clean} = \text{Clean}(M_{LT}, M_{STL_{t+1}}) \quad (11)$$

This outcome occurs when additional datasets are incorporated into LTL until it attains maximum capacity, resulting in the elimination of all samples. To prevent the intentional loss of valuable information for both LTIL and STIL samples classes, samples are seldom discarded immediately. Instead, the nearest samples from the respective classes are aggregated, and their mean values are sequentially established until $(M_{LT}, | < L_{max})$ is attained. A summarisation of how the pseudocodes operates for the experimentataion is depicted in Algorithm 2 and Algorithm 3 below.

Algorithm 2: Pseudocode for incremental learning with concept drift detection

Inputs: Data Stream S , Learner L , DriftDetector D , Window Size W

Output: Updated Model L

Initialize model L with initial data batch S_0

Initialize drift detector D

Initialize window $M = [M_{ST}, M_{LT}, M_C]$ (stores data for retraining)

For each new batch of $S = (S_1, S_2, \dots, S_t)$ of the tuples $S_i = (x_i, y_i)$

 // Step 1: Evaluate performance

 Prediction $P = L.\text{predict}(B_i)$

 Error $E = \text{CalculateError}(P, B_i.\text{true_labels})$

 // Step 2: Detect Drift

 Drift_Detected = $D.\text{update}(E)$

 If Drift_Detected:

 // Concept Drift Detected (Abrupt or Gradual)

$L = \text{RetrainModel}(M)$ // Retrain on recent representative data

 Reset(D)

 Clear(M_{CIL})

 Else:

 // No Drift, Incremental Update

$L = L.\text{partial_fit}(B_i)$ // incremental update with new data

 UpdateMemory(M_{ST}, M_{LT}, M_C // Add new data to sliding window

 Return Combined incremental learning with memories $M_{CIL} = M_{STL} \cup M_{STL}$

// INITIALIZATION

Initialize Model f_0 with parameters θ_0

Initialize Memory Buffer $M = \{\}$ // Stores representative samples from past

Set max_tasks, memory_size_per_class

// LONG-TERM INCREMENTAL LEARNING LOOP

Algorithm 2: *Cont.*

```

For task_id = 1 to  $M_{LT_{t+1}}$  max_tasks:
    Receive new data stream  $D_{\text{new}} = \{(x_i, y_i)\}$ 
    // 1. Memory Management (Optional: Keep representative old data)
    Update Buffer M using  $D_{\text{new}}$  and M (using memory allocation, herding or
    reservoir sampling)
    // 2. Combine New and Old Knowledge
    // To prevent forgetting, mix new data with a replay of past experiences
     $D_{\text{train}} = D_{\text{new}} \cup M$ 
    // 3. Incremental Training (Stability-Plasticity Balance)
    // Regularization based ensures old weights/outputs don't change drastically
    For epoch = 1 to max_epochs:
        Batch = Sample( $D_{\text{train}}$ )
        // Compute loss on new data
        Loss_new = CrossEntropy( $f(\text{Batch}_{\text{new}})$ , Labels_new)
        // Compute regularization loss to prevent forgetting (Knowledge
        Distillation)
        Loss_old = KnowledgeDistillation( $f(\text{Batch}_{\text{old}})$ ,  $f_{\text{previous}}(\text{Batch}_{\text{old}})$ )
        Total_Loss = Loss_new +  $\lambda$  Loss_old
        Update parameters  $\theta$  using Backpropagation
        // 4. Update Previous Model
         $f_{\text{previous}} = f_{\text{current}}$  (Copy model for distillation in next task)
    // FINAL MODEL Return  $KNN_{MSTL}$ ,  $KNN_{LTL}$ ,  $KNN_{MCL}$ 

```

4.5. Incremental Learning Accuracy, Rectification and Transfer of Classes

Consider that LTL has all original data containing concepts associated with STL. We need to conduct rectification and cleaning of LTL classified datasets. Since STL is already reduced in size, we do not discard the data as some important data may still be used for future predictions. This is what we assume to be the recurring drift for our future prediction. In this modeling, we do not relearn former concepts to reduce duplicating errors but enable the transfer of as much knowledge as much as possible for LTL. We model over operation as follows:

We rectify set of A by another set of

Step 1: $(x_i, y_i) \in B$

Rectify; $(A, B, (x_i, y_i)) \rightarrow \hat{A}$

Since $A, B, \hat{A} \subset \mathbb{R}^n \times \{1, \dots, C\}$ and $(x_i, y_i) \in B$.

We define $\hat{A}$ in two ways.

- (a) We find the KNN of x_i in $\frac{B}{(x_i, y_i)}$, then select y_i based on the labels and define

$$\theta = \max\{d(x_i, x) | x \in N_k(x_i, B) | (x_i, y_i)\}$$

- (b) KNN of $x_i \in A$ are cleaned as the operation progresses.

$$\hat{A} = A | x_j, y(x_j) | x_j \in N_k(x_i, A), d(x_j, x_i) \leq \theta, y(x_j) \neq y_i \}$$

Step 2: We do rectification operation for set B.

Rectification: $(A, B \rightarrow \hat{A}_{(B)})$

Since $A, B \rightarrow \hat{A}_{(B)} \subset \mathbb{R}^n \times \{1, \dots, C\}$

We then model this by cleaning the operations for the full set B.

$$(x_i, y_i) \in B = \{x_i, y_i\} \dots \dots \dots (x_{|B|}, y_{|B|})$$

$$\hat{A} = A$$

$$\hat{A}_{t+1} = Rectify(\hat{A}_t, B(x_i, y_i))$$

Adaptation for LTL is consistently updated in two forms at every incoming dataset sample (x_i, y_i)

$$M_{LTL_t} = Rectified(M_{LTL_t}, M_{LTL_t}(x_t, y_t))$$

For STL, and if θ_t is discarded, this is accommodated to LTIL after rectification. Therefore, long-term incremental learning (LTIL) is denoted as

$$M_{LT_{t+1}} = \left\{ \begin{array}{c} M_{LT_t} \cup clean(O_t, M_{ST_{t+1}}) \\ M_{LT_t} \end{array} \right\} \quad (12)$$

Step 3: Overall incremental learning adaptation. The incremental learning accuracy was noted over the last M_t samples, where $M_t = |M_{ST_t}|$ is the learning of the current STL.

Hence, the increment learning at the time stamp t for LTL is as follows:

$$W_{LT}^t = \frac{1 \left\{ i \in \{t - m_t + 1, \dots, t\} \mid KNN_{M_{LT_t}}(x_i) = y \right\}}{m_t} \quad (13)$$

4.6. Resolving Time Complexity and Learning Error

This experiment does not emphasize the overall hybrid model and excludes complex analysis associated with KNN time complexity in incremental learning. We stopped incremental learning when the error exceeded a certain threshold, which is the relative minimum error observed. We observed the KNN within an example of linear 0 (m) by evaluating each time up to $\log \frac{L_{max}}{L_{min}}$ for different learnings of STIL, where L_{max} and L_{min} are considered the maximum learning and minimum learning lengths of STIL. Overall, the worst scenario concerns complexity for the complete error for one STIL, denoted by the following:

$$\theta \sim L_{max}^2 \log \frac{L_{max}}{L_{min}} \text{ for the } k \text{ examples.}$$

To reduce case complexity, the error calculation can be computed incrementally if STIL S_{t-1} recalls the data stream.

$$M_{ST} = \{(x_i, y_i) \in \mathbb{R}^n \times \{1, \dots, C\} \mid i = t - m + 1, \dots, t\}$$

If $S_t = [S_{t-1}, (x_i, y_i)]$, then we compute the corresponding error as

$$E(S_t) = \frac{(t-1)E(S_{t-1}) + 1(h_{t-1}(x_t) \neq y_t)}{t} \quad (14)$$

Algorithm 3: Weighted 3WDKNN_ILA modeling algorithm

```

def incremental_learning( $M_{ST}$ ,  $M_{LT}$ ,  $M_C$ ):
    if  $M_{ST}$ :
        # Initialize combined memories
         $M_{IC} = M_{ST}.copy()$ 
        # Weight calculations
         $W_{STL}$ ,  $W_{LTL}$ ,  $W_C$ , = 200, 500, 1000 # Initialize weights
        # Output based on weights
        if  $W_{STL} \geq \max(W_{LTL}, W_C)$ :
            return KNN( $M_{ST}$ , x)
        elif  $W_{LTL} \geq \max(W_{STL}, W_C)$ :
            return KNN( $M_{LT}$ , x)
        elif  $W_C \geq \max(W_{STL}, W_{LTL})$ :
            return KNN( $M_C$ , x)
        # Update Drift Detector
        ADWIN.update(error)
        If ADWIN.detects_drift():
            For i in range(K): # K = 3,5,9,13
                new_learner = initialize_new_learner()
                Train new_learner on data in W
                E.add(new_learner)
    # Clean datasets and rectification
    pass # Implement cleaning logic
    # Rectify classes
    if rectify_classes(A, B):
        # Output incremental learning results
        return  $IL_{ST}$ ,  $IL_{LT}$ ,  $M_C$ .
```

4.7. Memory Reconstruction in Incremental Classes and Concept Drift

We model the hybrid mix by using a self-adjusting memory buffer for the representative classes in STIL with cleaning and rectification. The model progressively replays the stored classes and utilizes the knowledge fully before transferring the classes to long-term learning. When the drift is detected, the modeled memory reconstruction strategy for improving incremental learning classes is carried out by representing the sequence of incoming classes $C_1, C_2, \dots, C_t, C_{t+1}$. The memory for classes C_1^M for class C_1 containing random classes is established. Hence, the memory for class C_1^M combines the incoming classes for the sequence task from C_2 containing the random samples C_1^M . Therefore, many sequences for the incoming classes are denoted as $C_1^M, C_2^M, \dots, C_t^M, C_{t+1}^M$. This modeling considers that memory size for each class is equal to M and is restricted to avoid hardware overrun. By deploying new classes of incremental data for learning and the new instances that arrive with incoming data, the model denotes (C_1) with a set of C_i classes $\{C_1 \mid x = x_i, (i = 0, \dots, C_1)\}$ and the unique label of y ($y = 0, \dots, s - 1$). The C_1 dataset is thus represented in the equation below.

$$C_1 = \bigcup_{y=0(0, \dots, s-1)} \{x, y\}; x = (x_i), i = (0, \dots, C_i) \quad (15)$$

Fine-tuning C_i datasets is performed with reconstructed memory size set at M . Hence, given the $(t + 1)_{th}$ with z_{t+1} classes with previous memory, C_1^M with S_t classes in the C_t^M buffer memory is realized in the equation below:

$$C_t^M = \bigcup_{y = (0 \dots, S_t - 1)} \{x, y\}; x = (x_i), i = (0, \dots, K_t) \quad (16)$$

Equally, the realization of $(t + 1)_{th}$ incremental dataset classes is denoted in the equation below.

$$C_{t+1} = \bigcup_{y = (S_t \dots, S_{t+1})} \{x, y\}; x = (x_i), i = (0, \dots, C_{t+1}) \quad (17)$$

Therefore, incremental classes of the memory reconstruction are achieved for the first classes C_i , with the initiation memory C_1^M being split into $C_t^{M(1)}$ and $C_t^{M(2)}$. The incoming class C_2^M is broken into $C_t^{M(1)}$ and $C_t^{M(2)}$ incremental $(t + 1)_{th}$ classes for $C_{t+1} = \{C_{t+1}^{(1)} \cup C_{t+1}^{(2)}\}$ with Z_{t+1} and the previous memory allocation $C_t^M = C_1^{M(1)} \cup D_t^{M(1)}$ with S_t . This combination of memory for classes in continuous learning is illustrated as follows:

$$C_{t+1} = \left\{ \begin{array}{l} C_{t+1}^{(1)} = \bigcup_{y = (S_t - K_{t+1} \dots S_{t-1})} \{x, y\} \in C_{t+1}; x = (x_i), i = (0, \dots, C_{t+1}) \\ C_{t+1}^{(2)} = \bigcup_{y = (S_t \dots (N - 1))} \{x, y\} \in C_{t+1}; x = (x_i), i = (0, \dots, C_{t+1}) \end{array} \right\} \quad (18)$$

$$\left\{ \begin{array}{l} C_t^{M(1)} = \bigcup_{y = (0 \dots S_t - K_{t+1} - 1)} \{x, y\} \in C_t^M; x = (x_i), i = (0, \dots, C_t) \\ C_t^{M(2)} = \bigcup_{y = (S_t - K_{t+1} \dots (S_t - 1))} \{x, y\} \in C_t^M; x = (x_i), i = (0, \dots, K_t) \} \leq STIL. \end{array} \right\} \quad (19)$$

The model uses a memory reconstruction strategy for class incremental learning where training data is conducted for a smaller number of classes in the STIL, as the new classes with instances are progressively added until the memory is full without forgetting, as iterated in Equations (15)–(19). The datasets for the old classes are then released to the LTIL memory for continuous learning without additional computational inputs and resources.

5. Results and Discussions

This section analyzes and discusses experimental outcomes from model implementation in cropyield_1, cropyield_2, and CORe50 datasets using K values of K = 5, 9 and 13 with weighted classification of 200W, 500W and 1000W. This experiment evaluates the developed hybrid together with the state-of-the-art algorithms which include 3WD-KNN, HoKNN [25,32] and 1IKNN.

5.1. Classification Incremental Accuracy Performance with K Values [5K, 9K, 13K] for Long-Term and Short-Term Incremental Learning

The experimental analysis made a comparison in the accuracy levels of the incremental learning process in long-term incremental learning (LTIL) and short-term incremental learning (STIL) (summarized in Table 3) in both the K values and the weighted classification approach in the datasets. The hybrid model 3WDKNN_IL significantly outperforms the other tested and benchmarked algorithms in the STIL analysis. The hybrid model 3WDKNN_IL achieved a performance of 98.77 percent with a K value of 13K for the cropyield_1 datasets, while a K value of 9K achieved a performance of 97.61 percent. The algorithm models also perform accurately with cropyield_2 at 97.25 percent with 9K, while

the 5K value for the model performs closely at 97.02 percent. Overall, it is observed that 3WD_KNN closely performs at 94.01 percent in the 9K value instance, consistently followed by HoKNN and 1IKNN, respectively. In STIL, the average accuracy performance of the hybrid model 3WDKNN_IL improves with higher levels of K values, as shown in the trends outlined in Table 3 and Figure 2. In LTIL, the experimental analysis proves otherwise; the hybrid model 3WDKNN_IL has lower average accuracy than the performance recorded in STIL. The hybrid model 3WDKNN_IL achieves higher performance with CORE50 datasets at 94.87 with 9K values, followed by 94.03 at 5K. The hybrid model 3WDKNN_IL achieves the best performance at LTIL in 5K at 92.77 percent, as summarized in Table 4 and illustrated in Figure 3. The average performance accuracy does not reveal an increment or decrement with the increase or decrease in K values. This outline demonstrates that the presence of K values in the tested models does not affect LTIL performance. However, for LTIL, a repeatedly condensing set of the classes and prototype abstraction complicates the processing steps, creating a costly and slow classifier in the dynamic situation in a continuous environment and making the whole process inefficient.

Table 3. Summary of short-term incremental learning accuracy for models K values K = 5, 9, 13. In cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL& K VALUE SIZE	STIL	SHORT-TERM INCREMENTAL LEARNING (STIL)			
	DATASETS	3WD_KNN	3WDKNN_IL	HoKNN	1IKNN
IL 5K	CROPYIELD_1	94.05	97.02	93.67	86.33
	CROPYIELD_2	93.99	96.24	93.04	87.99
	CORE50	89.87	89.98	92.04	86.78
IL 9K	CROPYIELD_1	94.01	97.61	93.11	87.03
	CROPYIELD_2	93.94	97.25	93.01	85.71
	CORE50	88.93	87.03	91.77	86.98
IL 13K	CROPYIELD_1	90.01	98.77	92.09	88.01
	CROPYIELD_2	89.12	94.21	92.03	85.99
	CORE50	81.34	83.45	91.01	87.01

Table 4. Summary of accuracy performance for STIL implementation in cropyield_1, cropyield_2, and CORE50 datasets using K values of K = 5, 9 and 13 for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL& K VALUE SIZE	LTIL	LONG-TERM INCREMENTAL LEARNING (LTIL)			
	DATASETS	3WD_KNN	3WDKNN_IL	HoKNN	1IKNN
IL 5K	CROPYIELD_1	83.31	87.02	92.77	87.78
	CROPYIELD_2	84.97	86.24	91.05	86.04
	CORE50	89.43	89.02	94.03	87.87
IL 9K	CROPYIELD_1	84.43	87.61	84.79	85.75
	CROPYIELD_2	82.11	84.25	89.09	87.31
	CORE50	83.45	87.03	94.87	88.67
IL 13K	CROPYIELD_1	86.12	87.77	85.09	85.15
	CROPYIELD_2	83.12	84.28	84.88	85.19
	CORE50	92.11	83.45	88.03	86.07

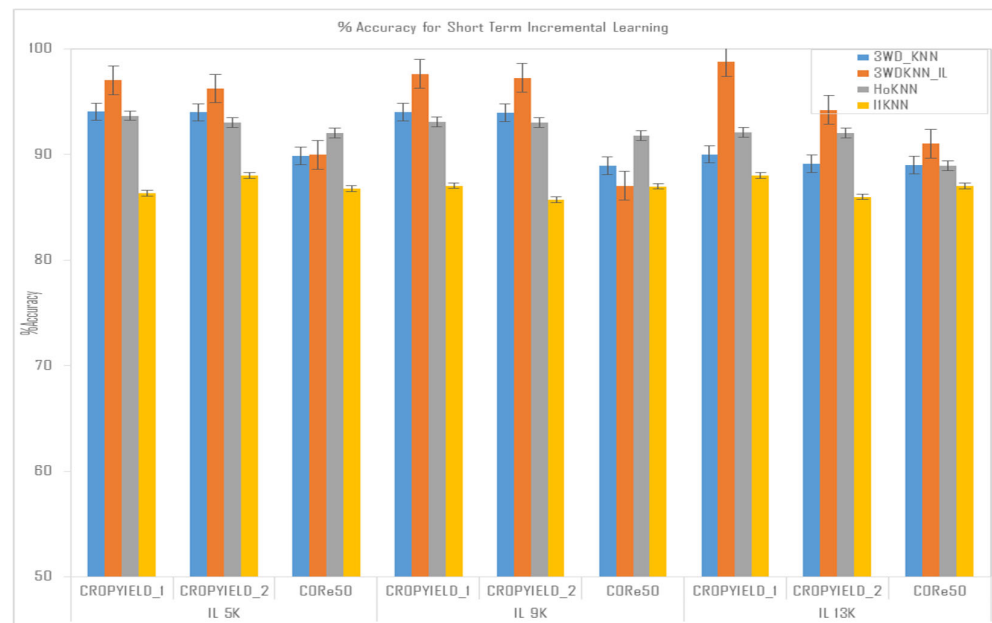


Figure 2. Accuracy performance for STIL implementation in cropyield_1, cropyield_2, and CORE50 datasets using K values of K = 5, 9 and 13 for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

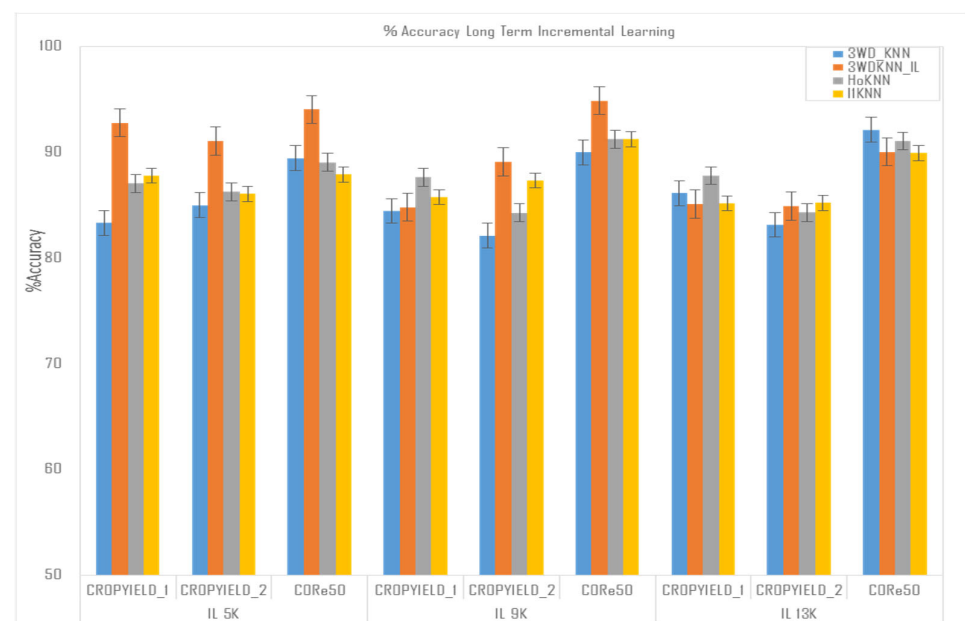


Figure 3. Accuracy performance for LTIL implementation in cropyield_1, cropyield_2, and CORE50 datasets using K values of K = 5, 9 and 13 for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

In experimental analysis, during STIL, the accuracy of incremental learning appears to be higher because the model can manage potential overlearning of classes, particularly when it accurately predicts all incremental learning within those classes but fails to learn incrementally for the test class targets. In the heterogeneous process during STIL, the incoming new data with knowledge gained are passed to the next subsequent classes for continuous learning, improving the model's ability to generalize. In the case of LTIL, analysis shows that most incoming new heterogeneous datasets cannot transition their class attributes and learn incrementally, which prevents them from generalizing and negatively impacts accuracy of the subsequent classes.

A further illustration with modeled incremental learning adaptation with K values set at [K = 5, 9, 15] in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN threshold is depicted in Figure 4. This model demonstrates the incremental learning adaptation of classes in real-time modeling with indications projecting how learned content is transferred from one class to the other. The distances between the boundary project the timelines that each individual class would take to learn incrementally, with further classes taking longer timelines that those classes near the boundary lines as determined by the local mean Euclidean distance. In addition, the experiment avoids locating the K-nearest neighbor in the conventional classifier which suffers from computational pre-training and complicated pre-processing of each class and data reduction. The local mean classification, which is independent of repeated processing training from high-dimensional and heterogeneous datasets where concept drift and loss of information occurs over changing dataset properties, is established by deriving the most votes for majority rule as a step towards assigning the best classes in the nearest neighborhood. Therefore, a hybrid 3WDKNN_IL model free incremental learning and classification algorithm that extends computational cost savings through extra classifications to achieve a high level of accuracy is proposed.

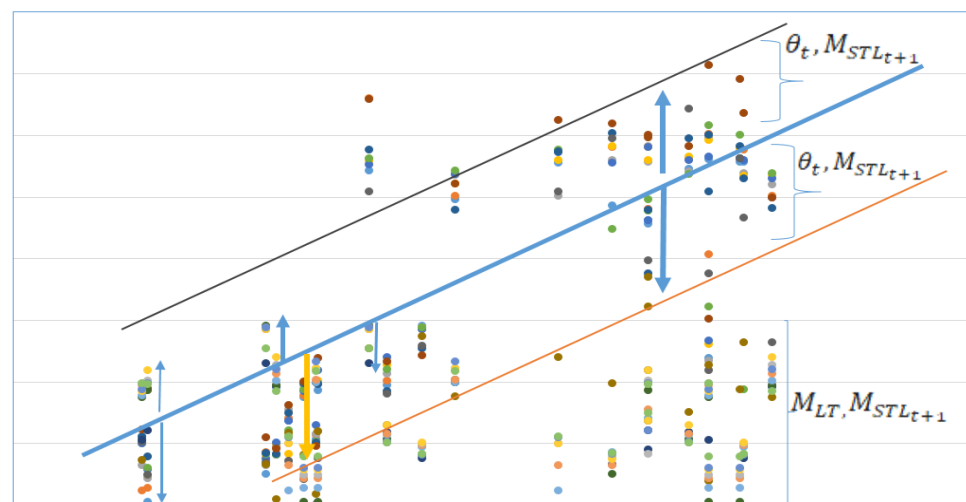


Figure 4. Incremental learning adaptation for different incoming classes with K values set at [K = 5, 9, and 13] in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN threshold.

5.2. Measure of Incremental Learning Efficiency (Epochs, Timelines)

The experiment was conducted to ascertain the incremental learning efficiency using ideal timelines due from the incoming datasets, or the transfer from the old datasets to the news datasets. This timeline measurements were captured from the ideal scenarios of K value instances of 5K, 9K and 13K environments. The illustrations in Figure 5 and summarization in Table 5 describe the comparison of the incremental learning pattern as per the timelines extracted for the K values for the hybrid 3WD_KNN, hybrid 3WDKNN_IL, and 1IKNN models. The hybrid 3WDKNN_IL model shows improved efficiency in timelines and epochs, way better than the other tested algorithms for all the datasets run. The experimental analysis shows that hybrid 3WDKNN_IL performs efficiently at the K value of 13K at 0.4667 s, closely followed by the K value at 9K with 0.4863 s and K value 5K at 0.5842 s, respectively. Close-range timelines were also realized in the performance of the hybrid 3WD_KNN with 5K value at 0.5421 s, 9K at 0.5578 s and 13K at 0.5795 s, respectively. The datasets for Cropyield-1 and Cropyield_2 have almost similar properties and class features, making them close to having almost similar timelines across the tested

models. The CoRe50 model had the slowest timelines in the experiment, attributable to repeated iterations and memory usage throughout the algorithms for hybrid 3WD_KNN, hybrid 3WDKNN_IL, and I1KNN. The lowest timeline achieved for CoRe_50 datasets was at 13K value at 1.9094 s for hybrid 3WDKNN_IL while the highest timeline for CoRe_50 was achieved at 6.2981 s for hybrid 3WD_KNN.

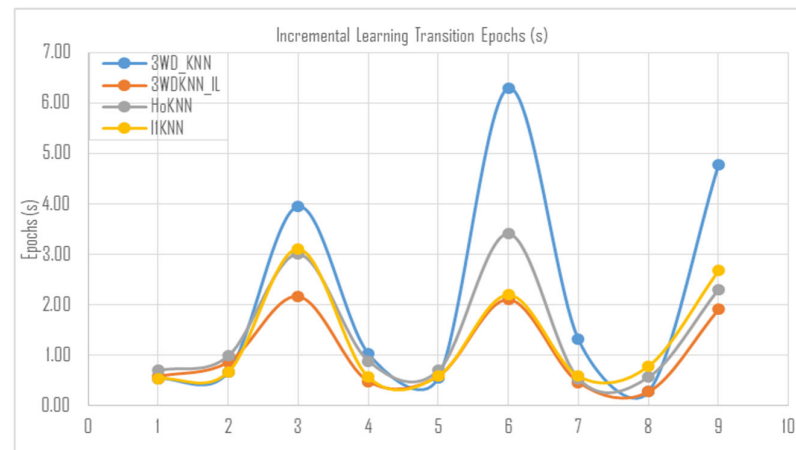


Figure 5. Incremental learning efficiency time (epochs) using K values [K = 5, 9, 13] in using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and I1KNN.

Table 5. Incremental Learning Efficiency (Epochs, Timelines) for models K values K = 5, 9, 13 in cropyield_1, cropyield_2, and CoRe50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and I1KNN.

IL& K VALUE SIZE	Epoch (s) Timelines Datasets	3WD_KNN	3WDKNN_IL	HoKNN	I1KNN
IL 5K	CROPYIELD_1	±0.5421	±0.5842	±0.7022	±0.5421
	CROPYIELD_2	±0.6701	±0.8669	±0.9874	±0.6701
	CoRe50	±3.9541	±2.1706	±3.0047	±3.1107
IL 9K	CROPYIELD_1	±1.0351	±0.4863	±0.8892	±0.5673
	CROPYIELD_2	±0.5578	±0.5904	±0.6987	±0.5904
	CoRe50	±6.2981	±2.1084	±3.4087	±2.2008
IL 13K	CROPYIELD_1	±1.3135	±0.4667	±0.5389	±0.5842
	CROPYIELD_2	±0.5795	±0.5789	±0.5673	±0.7863
	CoRe50	±4.7804	±1.9094	±2.3094	±2.6777

This experiment is able to achieve time-bound accuracy due to the classification distance of the test data points for the training data. The hybrid 3WD_KNN reduces the time required for processing by decreasing the number of training samples. Additionally, the hybrid 3WD_KNN method shifts to a direct instance selection approach using Euclidean distance for heterogeneous datasets, which enhances computational complexity and reduces training time for incoming datasets. To avoid computational iterations and repetitions, the hybrid 3WD_KNN model prevents misclassifications in the iterations. In essence, incremental learning occurs when noise is eliminated, reducing the error rate by removing outliers in the reduced datasets. The nearest neighbors of K are identified within the weighted Euclidean distance, saving significant computational and time complexity. This makes the model faster with optimal characteristics, thereby improving accuracy in incremental learning within limited timelines. At the same time, the hybrid 3WD_KNN model solves time complexities during the predictions and searches for the nearest neighbors and weighted classes, as shown in Figure 5, considering both training complexity $O(k \cdot n \cdot d)$, predictive time complexity $O(k \cdot n \cdot d)$, training space complexity, and prediction space complexity $O(1)$.

5.3. Weighted Classification Rolling Average Timelines and Model Efficiency During STIL and LTIL

Further experiments were conducted to determine the most efficient timelines and analyze how long the weighted classification error could take to ascertain optimal efficiency timelines related to incremental learning classification weighted training average error for the hybrid model from STIL to LTIL. We utilized Tables 6 and 7 to summarize classification rolling average accuracy and efficiency for the hybrid 3WDKNN_IL fused with equal K value and 200W and 500W weights. Tables 6 and 7 and Figures 6 and 7 summarize and illustrate the weighted 200W and 500W classification rolling average training error using K values [K = 5, 9, 15] in cropyield_1, cropyield_2, and CORE50 datasets for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN. In the experimentation of weighted 200W with K values of 5, 9, and 13, the algorithm's best performance was realized by hybrid 3WDKNN_IL at weighted 200K with 13K and 9K values at 0.01 s, respectively. The weighted 200W at 5K with 0.04 s achieved the highest weighted classification accuracy for the training error (see Table 6). In comparison, experimental analysis indicates that hybrid 3WD_KNN equally demonstrates remarkable results and performs better with a weighted 200W at 9K with 0.02 s, answering research question 3. Even though hybrid 3WDKNN_IL shows remarkable results in both the datasets tested, the variance with hybrid 3WD_KNN in tested Crop_1 and Crop_2 datasets is almost similar since the algorithms operate on the same parameters with datasets that have close resemblance in class properties and features. This finding confirms that hybrid 3WD_KNN is superior in performance compared with other state-of-the-art models as illustrated in Figure 5. Equally, CORE50, which was also used a benchmark dataset for the algorithms, proved less reliable since the weighted training average error was equally huge, showing that the algorithms are not able to resolve weighted time average accuracy between the STIL and LTIL, creating memory and time complexity issues for the models. Experimentation on CoRE_50 with weighted 200W at K values of 9 and 13 showed the most average classification training error at 0.39, creating high computation complexity through iterations for incremental learning between STIL and LTIL as shown in Table 6 and Figure 6 below. In this experiment, increased average test error was realized with higher K values, as shown in Table 6, due to reduced variance that led to higher bias, causing the model to predict less accurately, hence the inability to capture the complexity of the class datasets.

Table 6. Summary of weighted 200W classification rolling average training time using K values [K = 5, 9, 15] in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL& K VALUE SIZE	Classification Rolling Average Training	3WD_KNN	3WDKNN_IL	HoKNN	1IKNN
	Datasets				
IL 5K-200W	CROPYIELD_1	0.03	0.04	0.11	0.14
	CROPYIELD_2	0.03	0.05	0.11	0.12
	CORE50	0.34	0.36	0.24	0.16
IL 9K-200W	CROPYIELD_1	0.03	0.02	0.15	0.21
	CROPYIELD_2	0.03	0.02	0.14	0.13
	CORE50	0.35	0.39	0.28	0.21
IL 13K-200W	CROPYIELD_1	0.03	0.02	0.15	0.21
	CROPYIELD_2	0.03	0.02	0.14	0.13
	CORE50	0.35	0.39	0.28	0.21

Table 7. Summary of weighted 500W classification rolling average training time using K values [K = 5, 9, 15] in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL& K VALUE SIZE	Classification Rolling Average Training Error	3WD_KNN	3WDKNN_IL	HoKNN	1IKNN
	Datasets				
IL 5K-500W	CROPYIELD_1	0.02	0.02	0.1	0.18
	CROPYIELD_2	0.02	0.01	0.1	0.16
	CORE50	0.54	0.57	0.58	0.76
IL 9K-500W	CROPYIELD_1	0.02	0.01	0.15	0.21
	CROPYIELD_2	0.02	0.01	0.14	0.13
	CORE50	-	-	-	--
IL 13K-500W	CROPYIELD_1	0.03	0.01	0.17	0.24
	CROPYIELD_2	0.03	0.01	0.15	0.14
	CORE50	-	-	-	-

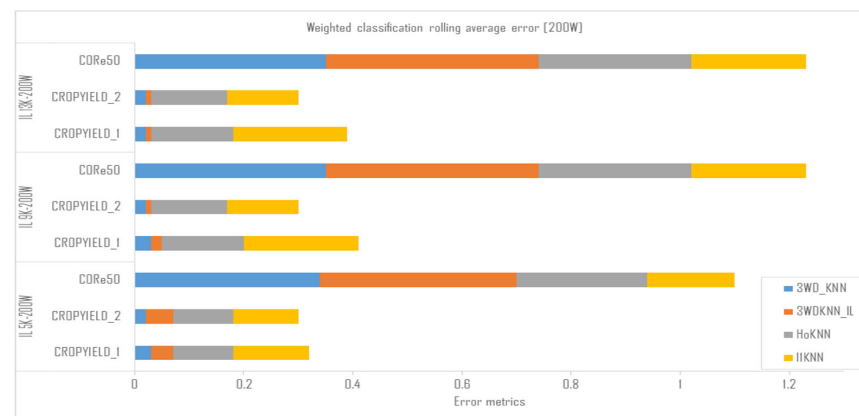


Figure 6. Illustration of Weighted 200W classification rolling average training time using K values [K = 5, 9, 15] in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

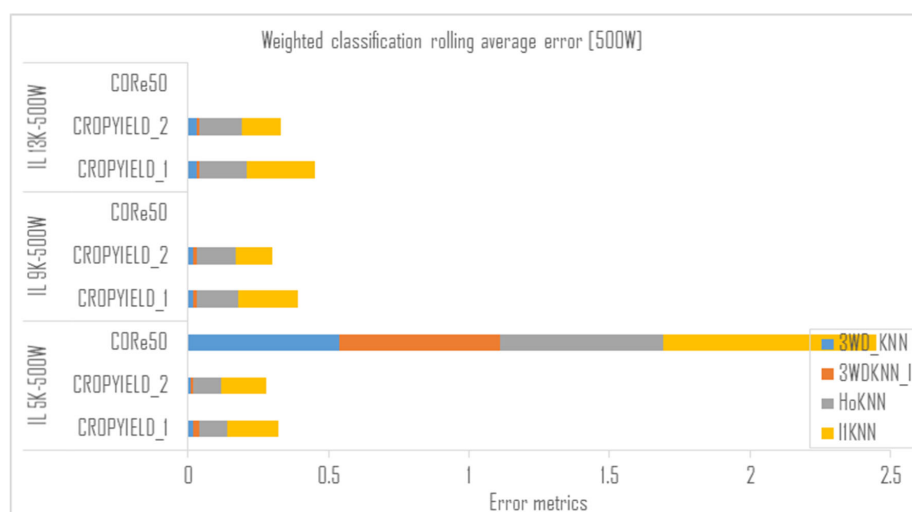


Figure 7. Illustration of Weighted 500W classification rolling average training error using K values [K = 5, 9, and 15] in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

To smoothen the classification boundaries, reduce training average error, and enhance performance, the experiment utilized a weighted value of 500W with similar parameters of K values of 5, 9, and 13 as summarized in Table 7 and illustrated in Figure 7. Table 7 and Figure 6 depict weighted 500W classification rolling average training error using K values [K = 5, 9, 15] in cropyield_1, cropyield_2, and CoRe50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN. From the experimental analysis, it could be deduced that a higher value for weighted 500W shows some improvement in average training error for the hybrid 3WD_KNN, confirming our answer to research questions 3 and 4. The results indicate that hybrid 3WD_KNN performs optimally at weighted 500W at K values of 5, 9, and 13 at 0.01 s, respectively. This shows that the optimal performance set of weighted average classification error is performed equally within the K values set. Our analysis implies that, with a higher weighted value of 500W, the K values do not significantly impact the classification average accuracy error between LTIL and STIL. Additionally, there is a slight performance improvement for the hybrid 3WD_KNN with weighted 500W at K values of 5, 9, and 13 for all the tested datasets, with the best performance being achieved at 0.02 s, which is similar to what was achieved with weighted 200W. It therefore shows that there is not much of a significant effect of K values on weighted classification average error in the performance of the tested algorithms and datasets used. As shown in Table 7 and illustrated in Figure 7, for the CoRe_50 datasets, null results were realized for hybrid 3WD_KNN, hybrid 3WDKNN_IL, and 1IKNN, indicating that the runs between LTIL and LTIL could have surpassed the computational runtime and memory due to too much time being taken between the two. Since our approach aimed to use the model as a benchmark for comparing the performance of hybrid 3WD_KNN, the computational cost threshold did not permit further analysis of the reasons behind the null results for some weighted 500W values.

Further illustration in Figure 8 demonstrates the pattern of weighted incremental learning adaptation, cleaning, and rectification with weighted difference sets of weights in 200W and 500W thresholds, using the hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN, and 1IKNN for the datasets. The analysis shows that with the weighted model, the concentration of datasets increases, improving the learning ability of the classes. However, this concentration of incoming classes and learning implies increased computation cost and inefficiency. Our model shows a high ability to retain knowledge and transfer new information for incremental learning, achieving this in the shortest time possible, as illustrated by the mathematical modeling in Figure 8 below. Classes that have already been learned in full are pushed further away as they retain learning properties, increasing our all-time learning accuracy despite the additional computational cost of retaining new information and knowledge in datasets.

5.4. Weighted [200W, 500W and 1000W] Average Accuracy Performance for STIL and LTIL in K Values [5, 9, and 13K]

Experimentation was conducted to understanding the modeling pattern and dataset behavior towards the fused weighted threshold and how this affects the performance of the incremental hybrid model in the heterogeneous environment with computational limitations and loss of old knowledge upon learning new information. The aim of the experiment was to answer research questions 1, 3 and 4 as well as determine the best accuracy performance level for the modeled hybrid model. A summary of our findings consists of accuracy performance for the STIL in 5K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CoRe50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN, as described in Table 8 with an illustration in Figure 9. The best weighted average accuracy in 5K value is confirmed by the hybrid model 3WD-KNN_IL at 98.22 percent at 500W threshold followed by 98.11 percent at weighted value of

200W. The average accuracy at weighted 200W and 500W is almost equal across the models; however, as the weighted average increases towards 1000W as illustrated in Figure 9, there is a decline in computational average accuracy, indicating that performance degrades with computational increase or costs. These finding answers research questions 2 regarding the hybrid model's computational efficiency with weighted and 5K value.

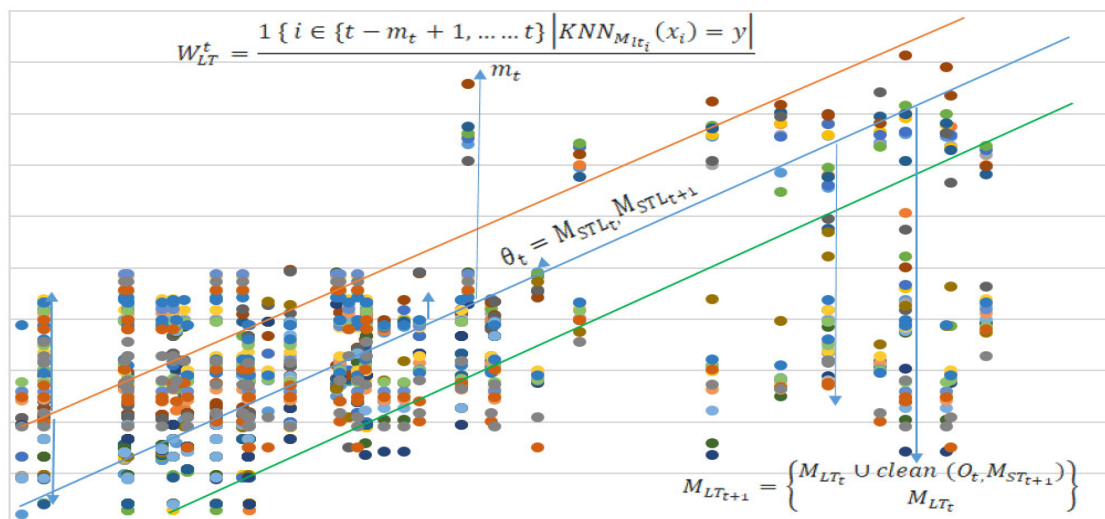


Figure 8. Weighted incremental learning adaptation, cleaning and rectification of different classes with weighted difference sets of weights in 200W, 500W threshold using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

Table 8. Percentage average accuracy performance for the STIL in 5K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL5K Weights (W)	STIL % Average Accuracy	3WD_KNN	3WDKNN_IL	HoKNN	1IKNN
	Datasets				
200W	CROPYIELD_1	94.97	98.11	94.31	89.06
	CROPYIELD_2	94.91	97.44	94.07	90.01
	CORE50	86.97	86.92	89.07	86.08
500W	CROPYIELD_1	95.47	98.22	94.97	89.07
	CROPYIELD_2	94.01	97.86	94.02	84.99
	CORE50	89.23	87.98	90.72	86.33
1000W	CROPYIELD_1	91.21	97.01	93.87	83.76
	CROPYIELD_2	89.09	93.11	91.77	84.05
	CORE50	89.90	89.07	88.95	87.98

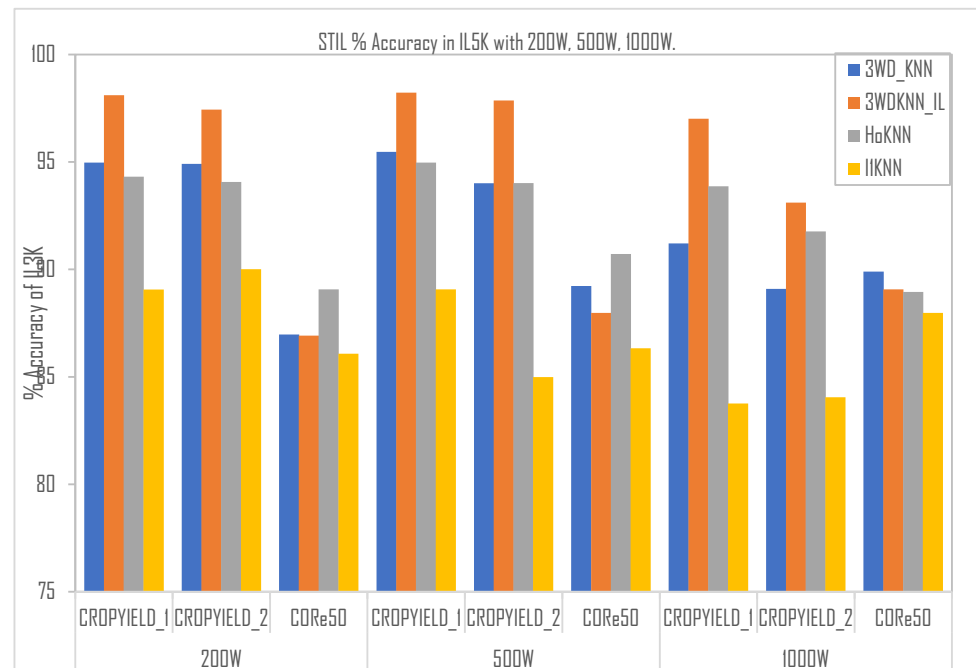


Figure 9. Computational average accuracy performance for the STIL in 5K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

Similar experimentation was conducted and incrementally learned using K values of 9K. Table 9 and Figure 10 below summarize the percentage average accuracy performance for STIL in 9K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN. The analysis shows that hybrid model 3WDKNN_IL performs exceptionally well in 9K values compared to 5K value with total average accuracy of 98.78 percentage weighted at 500W. This is followed by performance of 97.98 percent at the weighted value of 200W. From the analysis in Table 8, it can be derived that 3WD_KNN also shows significant performance values in regard to the accuracy of short-term incremental learning with K value at 9K. The general performance observation outline shows that the average accuracy performance improves at the weighted value of 200W and demonstrates peak performance with weighted 500W; the performance eventually dips with weighted 1000W due to computational costs associated with computing runs, as illustrated in Figure 10 below. A similar trend was also noted in the previous average accuracy performance for STIL in 5K value for weighted 200W, 500W and 1000W.

In the analysis for average accuracy performance for STIL in 13K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN (see summarized Table 10 and illustrated Figure 11), it is noted that the best performance was achieved by hybrid model 3WDKNN_IL in 97.61 percent at weighted 500W. Experimental analysis depicts that there is a slight dip in performance of all the tested models at the weighted value of 1000W. This description is best visualized in Figure 11 below, indicating the subpar performance as compared with the weighted performance at 5K and 9K with weighted values.

Table 9. Percentage average accuracy performance for the STIL in 9K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL9K	% Average Accuracy Datasets	3WD_KNN	3WDKNN_IL	HoKNN	I1KNN
200W	CROPYIELD_1	95.09	97.67	94.90	91.75
	CROPYIELD_2	94.91	97.98	94.09	91.76
	CORE50	84.23	85.44	89.01	84.98
500W	CROPYIELD_1	96.72	98.78	95.67	90.02
	CROPYIELD_2	95.05	97.97	95.11	84.03
	CORE50	87.65	86.92	87.56	85.90
1000W	CROPYIELD_1	94.81	97.01	94.71	84.98
	CROPYIELD_2	90.02	93.02	90.07	83.89
	CORE50	85.07	86.54	84.97	84.09

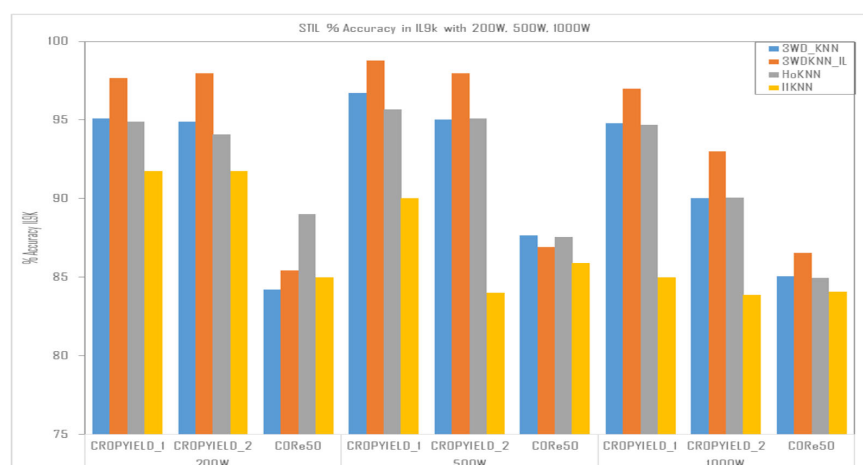


Figure 10. Illustration of computational average accuracy performance for the STIL in 9K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

Table 10. Percentage average accuracy performance for the STIL in 13K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL13K	% Average Accuracy Datasets	3WD_KNN	3WDKNN_IL	HoKNN	I1KNN
200W	CROPYIELD_1	94.98	96.66	93.75	87.91
	CROPYIELD_2	92.88	96.77	93.03	86.04
	CORE50	74.67	86.09	89.07	88.08
500W	CROPYIELD_1	93.77	97.61	93.11	85.99
	CROPYIELD_2	94.22	95.98	93.66	82.78
	CORE50	83.87	84.90	83.89	84.33
1000W	CROPYIELD_1	89.08	91.01	87.99	83.18
	CROPYIELD_2	87.05	88.77	86.81	77.04
	CORE50	82.34	81.22	82.04	83.09

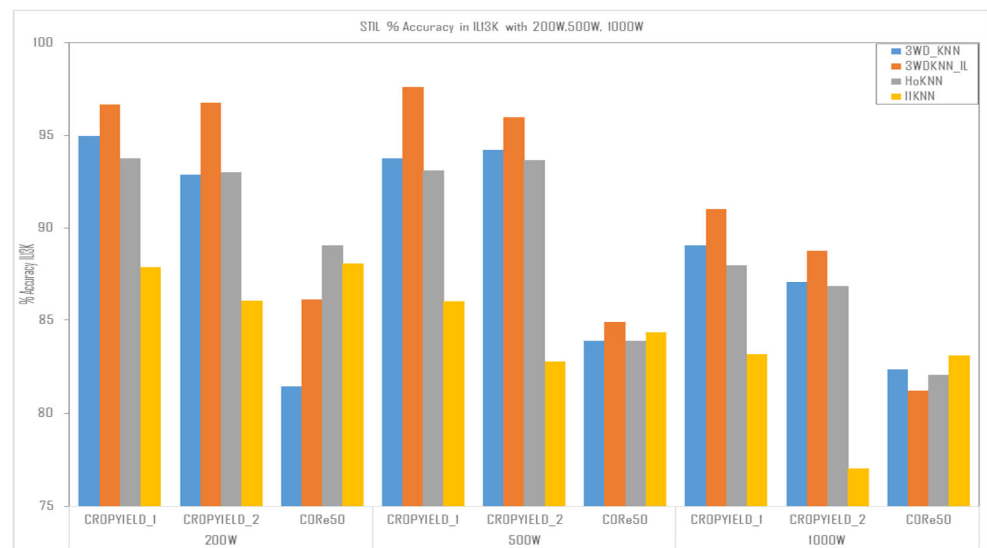


Figure 11. Computational average accuracy performance for the STIL in 9K value for weighted 200W, 500W and 1000W in cropyield_1, cropyield_2, and CORE50 datasets using hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

5.5. Classification Rolling Average Learning Error Pattern for LTIL and STIL

In this experiment and to answer research question 2, overlearning is achieved when the classification average training E_T is small and the testing error is large. This is caused by the incoming learned content from heterogeneous datasets presenting new information and instances for incremental learning. The classes then store and memorize new data until they are full and transfer the learned content to the next class datasets. Therefore, the model retains and stores exemplar data and old features through compression to preserve essential information of old classes that can be useful for incremental learning, as shown in Figure 12 below. If there are too many incoming streaming datasets with new information and slow transfer incremental learning, then that class drops learning through early stopping or takes longer to transfer learning data. Additionally, underlearning occurs when training and testing errors are large, occurring when poorly trained classes do not transfer any new learning from the incoming datasets, affecting accuracy. Figure 12 illustrates classification learning error patterns for LTIL and STIL implementation in cropyield_1, cropyield_2, and CORE50 datasets using K values of K = 5, 9 and 13 with weighted classification of 200W, 500W and 1000W in an ideal heterogeneous dataset.

This analysis shows that hybrid model 3WDKNN_IL performs much better with CORE50 datasets in LTIL settings. This experiment demonstrates that LTIL performs better with image datasets that require more time to learn and transfer new knowledge between classes, resulting in higher error rates. The long average error and time covered from one class to another in LTIL contributes to the lower performance average. In the analysis, we note that slow transfer of the learned content from one class to the other is caused by several factors ranging from different class settings to Euclidean distances from one point to the other. In this modeling, it is noted that the dropping of learning classes, especially when they exceed the average training error time, can cause the slow transition of learning from one class to the other. This experimental analysis was conducted to prove the incremental learning patterns associated with STIL and LTIL, answering research question 2.

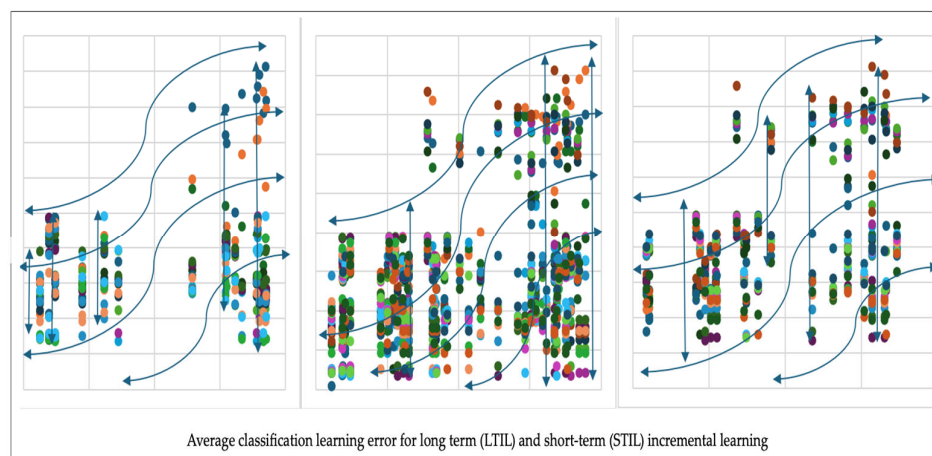


Figure 12. Classification learning error pattern for LTIL and STIL implementation. The model retains and stores exemplar data and old features through compression to preserve essential information of old classes that can be useful for incremental learning.

With LTIL, there is an increased validation error that increases exponentially, as shown in the tabulated Table 11 and the class learning trends in Figure 13, which is a result of the long distance between weighted classes that potentially drive expensive computational classes. In STIL, the experiment revealed that using a weight of 200W at the K value resulted in some underlearning, as it produced a lower training error. Even though this trend is recommended with the parameters of our selected K values and weighted classification averages, lower K values and weights could have potentially caused loss of low function at the output, causing validation error and training error and thus affecting overall accuracy through underlearning and correctly predicting the outputs.

Table 11. Classification rolling average training error in cropyield_1, cropyield_2, and CORE50 datasets using K values of K = 5, 9 and 13 for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

IL& K VALUE SIZE	Classification Rolling Average Training Error %	3WD_KNN	3WDKNN_IL	HoKNN	1IKNN
	Datasets				
IL 5K	CROPYIELD_1	10.74	10.00	0.90	−1.45
	CROPYIELD_2	9.02	10.00	1.99	1.95
	CORE50	0.44	0.96	−1.99	−1.09
IL 9K	CROPYIELD_1	9.58	10.00	8.32	1.28
	CROPYIELD_2	11.83	13.00	3.92	−1.60
	CORE50	−1.04	−4.20	−3.10	−4.24
IL 13K	CROPYIELD_1	3.89	11.00	7.00	2.86
	CROPYIELD_2	6.00	9.93	7.15	0.80
	CORE50	−3.11	−0.03	−1.08	−2.90

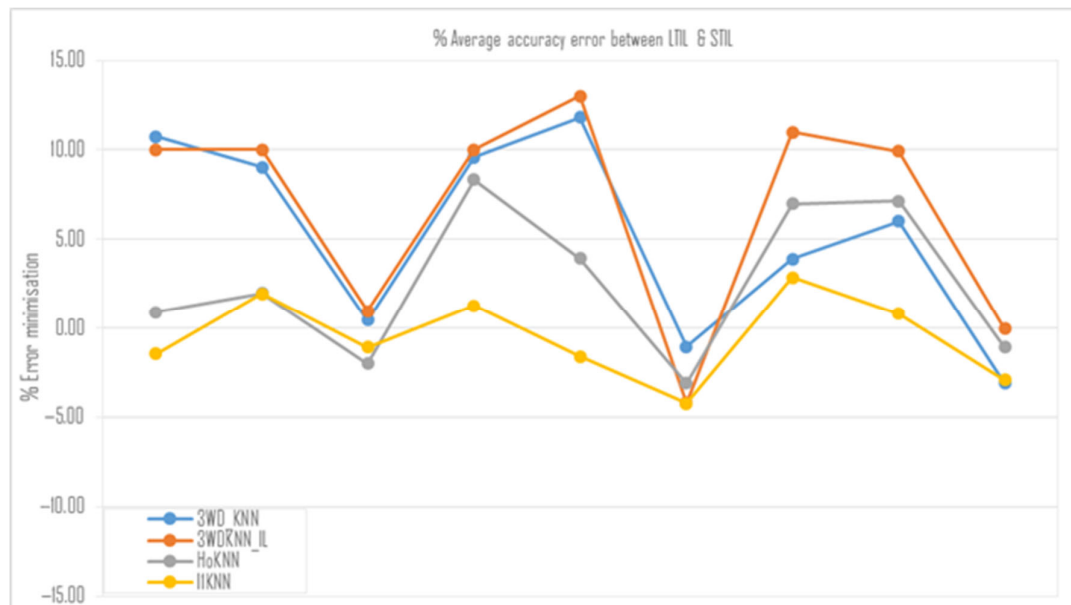


Figure 13. Computational average accuracy training error between LTIL and STIL in copyield_1, copyield_2, and CORE50 datasets using K values of K = 5, 9 and 13 for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

To address potential overlearning in LTIL for weighted average classification at 1000W, we built our model using a general regularization technique that strictly enforces early stopping learning by interrupting class overloading as soon as the generalization error increases during optimization steps. This improvement in our model solves critical computing limitations by avoiding and preventing memory overloading, reducing iterations causing computational inefficiency and time wastage, and creating an early stopping mechanism which modifies neither the constraint parameter vector nor the loss function (but retains) that creates prediction space complexity, thereby answering research question 2. In certain weighted classes, the experimental analysis proves that between a K value of 13K and 1000W, the early stopping rule appears to have occurred late, resulting in a loss of model generality due to slower training and a subsequent decline in prediction accuracy. In the experimental analysis as presented in Figure 12, it was observed that with weighted classes, there are classes that contain important incremental learning information from the incoming datasets. However, these classes cannot be utilized for subsequent class learning or prediction due to the maximization of the number of epochs, which is defined as the average error of the class incremental learning set measured post-epoch. Secondly, it is also observed that potential classes with incremental learning information cannot be guaranteed to carry important prediction information to the subsequent classes. The early stopping criterion considerably reduces costly computational procedures and enhances efficiency in the hybrid model and quality incremental learning in heterogeneous datasets.

6. Evaluation of Incremental Learning Models

In this experiment, an evaluation is performed to visualize the clear insights of accuracy, Precision, F1_Score and Recall. We use these evaluation results to analyze the incorrect and correct predictions in imbalanced datasets. Evaluations for the incremental learnings for the located hybrid algorithms were employed as below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1_Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

The accuracy prediction indicates that the model achieves high accuracy across numerous incremental learning predictions. Our model emphasizes the quality of positive predictions in hybrid incremental learning, utilizing Recall to assess the model's effectiveness in predicting actual positive instances. The incremental learning accuracy test was used for the hybrid 3WD_KNN, hybrid 3WDKNN_IL, HoKNN and I1KNN using K-instance values of K = 5, 9 and 13, as summarized in Table 12 and graphically represented in Figure 14. Similarly, the prediction was anchored on weighted classifiers in the experiment that consisted of hybrid 3WD_KNN, hybrid 3WDKNN_IL, HoKNN and I1KNN weighted on 200W and 300W for accuracy, Recall, Precision, and F1_Score as tabulated in the results in Table 13. Our experimental analysis for the weighted classifiers in 200W and 500W shows that modeled hybrid 3WDKNN_IL significantly shows improved and steady accuracy in all weighted evaluations criteria. Accuracy evaluation for 3WDKNN_IL for 200W was 0.98 and 500W with 0.98 while 3WD_KNN had 0.96 for 200W and 0.96 for 500W, respectively. In assessing Recall and the model's effectiveness in predicting actual positive instances, the experiment showed that hybrid 3WDKNN_IL for weighted 200W was at 0.97 and weighted 500W was at 0.98, while hybrid 3WD_KNN recorded 0.95 for weighted 200 and 0.95 for weighted 500W, respectively. For the precision of the positive and negative instances, the experiment showed that weighted 200W recorded 0.97 and weighted 500W recorded 0.98 while hybrid 3WD_KNN showed 0.94 for 200W and 0.95W for 500W, respectively. F1_Score experimental analysis revealed that higher Precision scores were recorded with both 200W and 500W achieving 0.99 for hybrid 3WDKNN_IL, while 3WD_KNN achieved 0.94 for 200W and 0.95 for 500W, respectively. The weighted classification evaluation for 200W and 500W in Recall, accuracy, Precision and F1_Score for HoKNN and I1KNN proved to be much lower as depicted in Table 13 below.

Table 12. Summary of classification evaluation for K values 5, 9, 13 in Recall, accuracy, Precision and F1_Score for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and I1KNN.

Classifier-K Instances	Accuracy %			Recall %			Precision %			F1_Score %		
	IL5K	IL9K	IL13K	IL5K	IL9K	IL13K	IL5K	IL9K	IL13K	IL5K	IL9K	IL13K
3WD_KNN	0.95	0.96	0.94	0.94	0.93	0.92	0.91	0.94	0.93	0.91	0.91	0.90
3WDKNN_IL	0.97	0.97	0.96	0.96	0.95	0.96	0.95	0.96	0.97	0.96	0.97	0.95
HoKNN	0.88	0.88	0.85	0.84	0.86	0.87	0.85	0.87	0.84	0.87	0.89	0.89
I1KNN	0.86	0.83	0.83	0.88	0.87	0.86	0.84	0.87	0.88	0.87	0.88	0.87

Table 13. Weighted classification evaluation for Recall, accuracy, Precision and F1_Score in weighted 200W, 500W for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and I1KNN.

Classifier-Weighted Classifiers	Accuracy %		Recall %		Precision %		F1_Score %	
	200W	500W	200W	500W	200W	500W	200W	500W
3WD_KNN	0.96	0.96	0.95	0.95	0.94	0.95	0.94	0.95
3WDKNN_IL	0.98	0.98	0.97	0.98	0.97	0.98	0.99	0.99
HoKNN	0.91	0.89	0.88	0.87	0.89	0.91	0.91	0.92
I1KNN	0.88	0.87	0.86	0.89	0.88	0.87	0.85	0.87

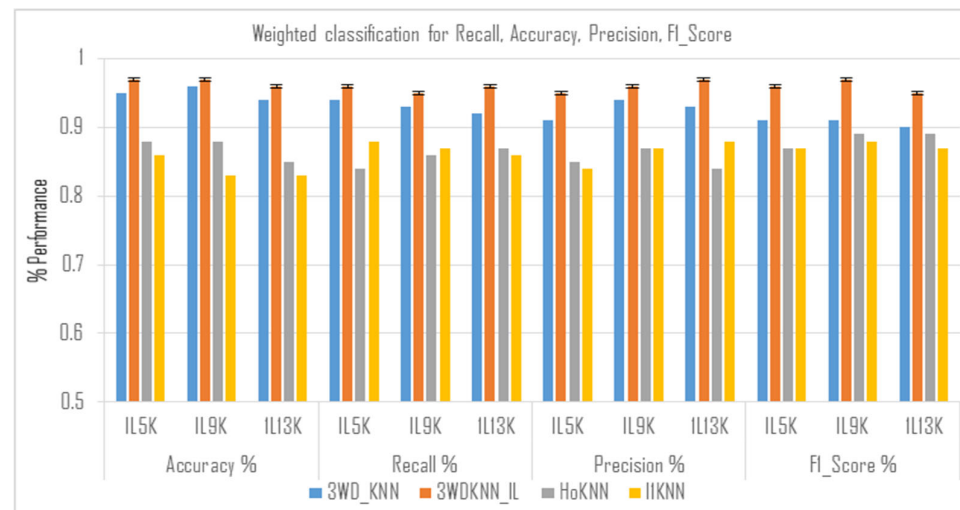


Figure 14. Weighted classification evaluation for Recall, accuracy, Precision and F1_Score using K values 5, 9, 13 in weighted 200W, 500W for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and I1KNN.

The results were analyzed to understand performance based on the choice of set K values of K = 5, 9, 13 for the classifiers of hybrid 3WD_KNN, hybrid 3WDKNN_IL, HoKNN and I1KNN. This evaluation was deemed crucial to help narrow down the best performing hybrid model based on the weighted values on the K-instances. Parameters of evaluation were set on the same criteria and standards in Recall, accuracy, Precision and F1_Score. Overall performance indicated that hybrid 3WD_KNN achieved better evaluated performance with accuracy of K5 = 0.97, K9 = 0.97 and K13 = 0.96 while 3WD_KNN achieved K5 = 0.95, K9 = 0.96 and K13 = 0.94, respectively. For the model's effectiveness in predicting actual positive instances with Recall, hybrid 3WD_KNN revealed K5 = 0.96, K9 = 0.97 and K13 = 0.96, while 3WD_KNN achieved K5 = 0.94, K9 = 0.93 and K13 = 0.92. Regarding experimental evaluation for the classification of K value on precision, hybrid 3WD_KNN revealed K5 = 0.95, K9 = 0.96 and K13 = 0.97 while 3WD_KNN achieved K5 = 0.91, K9 = 0.94 and K13 = 0.93, respectively. F1_Score for the hybrid 3WD_KNN revealed K5 = 0.96, K9 = 0.97 and K13 = 0.95 while hybrid 3WD_KNN achieved K5 = 0.91, K9 = 0.91 and K13 = 0.90. The summarized experimental analysis, as shown in Table 12 above, depicts that the hybrid 3WDKNN_IL has the capacity and superiority to show precision and high accuracy levels in all the evaluated models, including in comparison to the other state-of-the-art models of HoKNN and I1KNN in incremental learning settings. Therefore, we conclude that the weighted classification model of hybrid 3WD_KNN achieves better accuracy and superiority in the weighted 200W.

From the experimental evaluation, the lower dimensionality of data points for 200W resulted in a reliable hybrid 3WDKNN_IL model. Improvement from the Euclidean distance measure did not suffer from the curse of dimensionality during incremental learning, as attributed in the other incremental learning models. The containment of the dissimilar features and the ability of this hybrid model to utilize relevant weighted class features within the fastest, nearest Euclidean distance resulted in a lower dimensional vector feature. In essence, this was useful in eliminating the curse of dimensionality and error rate, resulting in a greater level of accuracy. The experimental model shows that the weighted values perform better with a choice of K = 5 as compared to K = 9 and K = 13 values; see the summarized trend in Table 12. The realization of creating too many noisy features with lowered values less than K = 5 led to the abandonment of fewer K values. This is because lower K values ($<K$) tend to create more noise and bias, resulting in misclassification accuracy. Additionally, areas with high data point concentration proved very sensitive to incremental

learning with the smaller values of K and could not tolerate small perturbations in highly dense data regions, becoming problematic and thus affecting the performance accuracy. In the conducted experiments, larger values of K beyond K = 15 resulted in complexity concerns due to memory constraints and underfitting of the incremental datasets, leading to inadequate generalization and elevated bias. The experiment excluded the utilization of bigger K values owing to considerations that affected the overall forecast. Another reason for avoiding bigger K values following the experimental test was the exclusion of numerous overlapping training sets of local class neighbors with incremental neighbors, which diminished the model's sensitivity to the specific class increment datasets. Likewise, the experiment could not surpass K = 13K, as bigger K values produced bias towards predominant values in the populated class data points, resulting in an imbalance in the distances from the query points. Additionally, a higher computational run time was required with the rising value of K. This led to experiment halting due to high K values, despite the potential for a greater degree of dependence.

In the experimentation, it was also realized that weighted values above 500W created a learning problem, and some results could not be realized due to memory lapse, computational limitations and complexity. Therefore, the experimentation ceased using the higher value of 1000W and implemented the experiment within 200W and 500W for the hybrid model, as summarized in Table 13 and Figure 15 above. To evaluate the statistical significance of the experimental results of the four classifiers using performance measures (accuracy, Recall, Precision, F1_Score) under two weight settings (200W and 500W), we conducted thorough statistical significance tests to ascertain paired *t*-tests comparing performance at 200W and 500W for each metric, ANOVA analysis of classifiers across all metrics, and estimations of effect size (Cohen's *d*) as summarized in Table 14 below.

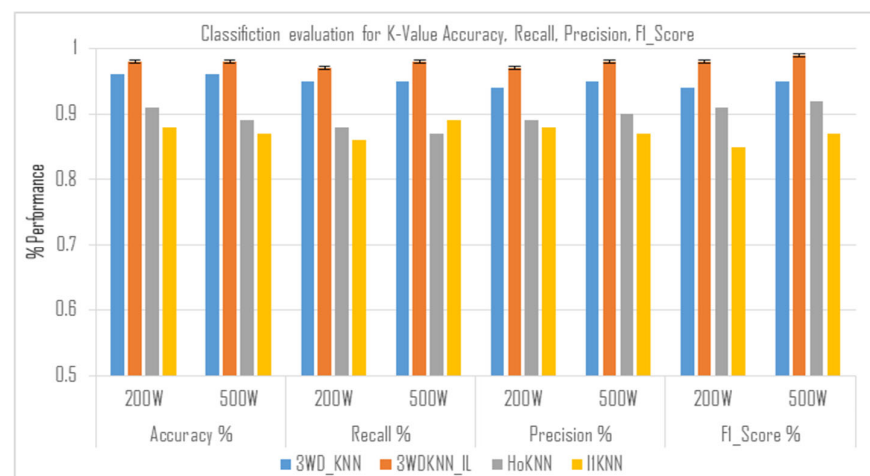


Figure 15. Illustration of classification evaluation for K values 5, 9, 13 in Recall, accuracy, Precision and F1_Score for hybrid model 3WDKNN_IL, 3WD_KNN, HoKNN and 1IKNN.

Table 14. ANOVA analysis of classifiers across all metrics, and estimations of effect size (Cohen's *d*).

Metric	Mean Diff	t-Statistic	p-Value	Significant?
Accuracy	0.0075	0.21	0.8445	No
Recall	−0.0075	−0.20	0.8521	No
Precision	−0.0075	−0.23	0.8297	No
F1_Score	−0.01	−0.26	0.8127	No

From the statistical analysis, there are no statistically significant differences between 200W and 500W conditions (all $p > 0.05$). In addition, effect size (Cohen's *d*) shows negligible effect sizes ($|d| < 0.2$), indicating that the differences between conditions have no

practical significance. One-way ANOVA shows that there are highly significant differences among the four classifiers with ($p \approx 1.1 \times 10^{-16}$), F-statistic of 134.91 and p -value < 0.0001 . The Pairwise Classifier Comparisons as summarized in Table 15 show all classifier pairs' statistically significant differences ($p < 0.05$).

Table 15. Pairwise Classifier Comparisons.

Comparison	p -Value
3WD_KNN vs. 3WDKNN_IL	1.5×10^{-6}
3WD_KNN vs. HoKNN	1.9×10^{-6}
3WD_KNN vs. I1KNN	4.0×10^{-10}
3WDKNN_IL vs. HoKNN	7.4×10^{-9}
3WDKNN_IL vs. I1KNN	5.2×10^{-12}
(Accuracy, Recall, Precision, F1_Score)	(Accuracy, Recall, Precision, F1_Score)
HoKNN vs. I1KNN	0.0039

Consequently, under the weight conditions of 200W versus 500W, there is a minimal and negligible impact on classifier performance. This statistical test, using (accuracy, Recall, Precision, F1_Score) for weights, showed no significance impact because weights are normalized in this hybrid model by providing a probability or total vote score; the classification decision is based on neighbor proximity rather than weight absolute value. As a result, both 200W and 500W will continue to significantly prioritize the single closest neighbor, making absolute weight magnitude unimportant if the same point is closest. The difference becomes significant only if the weighting function is sensitive to noise or if the dataset is highly dimensional, resulting in equidistant neighbors, making weight selection more important. However, 3WDKNN_IL (Mean: 0.98) exhibited superior performance, followed by 3WD_KNN (Mean: 0.95), HoKNN (Mean: 0.90), and I1KNN (Mean: 0.87) as the least effective performer.

7. Ablation Study

The investigation included ablation research to support the hybrid model's reasoning using the same original datasets. We studied ablation as a systematic technique for separating and evaluating model feature contributions using the cropyield_1, cropyield_2, and CORE50 datasets, along with accuracy, Recall, Precision, and F1_Score. We trained cropyield_1, cropyield_2, and CORE50 datasets using cleaning, normalization, transformation, and sorting of missing values, unbalanced data treatment, and tweaked hyperparameters before doing ablation research by gradually removing the most relevant attributes. Specifically, this study experimented with assessment components to analyze the datasets, confirming the hybrid model's reasoning and comparing the results to the datasets' original outcomes. Classifier instances for the datasets tested with K values for 5, 9, 13 all returned significantly low values, revealing that the developed model has quality and superior performance. Consequently, the prediction and accuracy results without incremental learning and concept drift detection resulted in unsatisfactory performance and very low accuracy as compared to results derived from the developed hybrid 3WDKNN_ILA model, as summarized in Table 16.

Table 16. Results of ablation study on proposed model using Recall, accuracy, Precision and F1_Score.

Classifier-K Instances	Accuracy %			Recall %			Precision %			F1_Score %		
	IL5K	IL9K	1L13K	IL5K	IL9K	1L13K	IL5K	IL9K	1L13K	IL5K	IL9K	1L13K
Crop_yield_1	0.61	0.59	0.55	0.67	0.66	0.62	0.61	0.58	0.60	0.59	0.61	0.58
CORe50	0.57	0.57	0.56	0.56	0.55	0.57	0.58	0.57	0.59	0.56	0.57	0.55
Cropyield_2	0.68	0.68	0.65	0.64	0.66	0.67	0.65	0.67	0.64	0.67	0.69	0.69

8. Conclusions

The hybrid 3WDKNN_ILA demonstrates improved performance and high accuracy in heterogeneous crop yield datasets, despite challenges such as loss of old knowledge upon learning new information, elevated computational costs, and limitations in learning capacity. This technique reduces memory consumption and storage needs by offloading historical datasets within a large-scale data streaming framework. By promptly employing the shortest distance metrics for subsequent classes, the incoming data avoids the computational costs associated with retraining. This adaptability facilitated the effective management of the changing dataset distribution, thus addressing concept drift during real-time implementation and continuous streaming. In conclusion, this experiment contributes widely to machine learning modeling and development in heterogeneous environments that enhance the accuracy and predictability of these models. This experiment makes improvements regarding adjusting errors in current datasets and past datasets automatically and in variations of concept drift in trading. The modeling of the hybrid 3WDKNN_ILA, which dynamically updates weighted voting majority for classifiers, shows that novelty in machine learning predictions has the potential to improve and enhance optimization in prediction models like crop yield predictions for short-term learning and long-term learning processes. We note that our hybrid incremental learning model overcomes learning errors and improves the time complexity of KNN classifiers in weighted and K values in the incremental learning approach. At the same time, the reduction of the nearest local centroid helps avoid high computational cycles that highly affect KNN optimization, especially for short-term incremental learning. These lead to high performance in KNN in heterogeneous environments, as proven in the evaluation and comparison of selected state-of-the-art models.

Author Contributions: Conceptualization, J.O.O. and K.O.O.; methodology, J.O.O.; software, J.O.O.; validation, J.O.O., K.O.O. and T.M.; formal analysis, J.O.O.; investigation, J.O.O.; resources, J.O.O.; data curation, J.O.O.; writing—original draft preparation, J.O.O.; writing—review and editing, J.O.O.; K.O.O.; T.M.; visualization, J.O.O.; supervision, K.O.O.; T.M.; funding acquisition, J.O.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset available on request from the authors. The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Al-Amin, S.T.; Nur, M.I.; Farooque, A.; Chen, G.; Varghese, R.; Ordonez, C. A Different VIM: Visualizing Incremental Machine Learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM '22)*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 4783–4787. [\[CrossRef\]](#)
2. Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; Fu, Y. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019*; pp. 374–382.
3. Halder, R.K.; Uddin, M.N.; Uddin, A.; Aryal, S.; Khraisat, A. Enhancing K-nearest neighbor algorithm: A comprehensive review and performance analysis of modifications. *J. Big Data* **2024**, *11*, 113. [\[CrossRef\]](#)
4. Belouadah, E.; Popescu, A.; Kanellos, I. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Netw.* **2021**, *135*, 38–54. [\[CrossRef\]](#)
5. Gepperth, A.; Hammer, B. Incremental learning algorithms and applications. In *Proceedings of the 24th European Symposium on Artificial Neural Networks, ESANN, Bruges, Belgium, 22–24 April 2016*; p. 1.
6. Li, Y.; Du, L.; Liu, H.; Guo, Y. Class-Incremental SAR Ship Detection and Classification via Context-Robust Exemplar Replay and Multi-Granularity Knowledge Distillation. *IEEE Trans. Aerosp. Electron. Syst.* **2025**, *61*, 9276–9289. [\[CrossRef\]](#)
7. Jakob, J.; Artelt, A.; Hasenjaeger, M.; Hammer, B. Interpretable SAM-KNN Regressor for Incremental Learning on High-Dimensional Data Streams. *Appl. Artif. Intell.* **2023**, *37*, 2198846. [\[CrossRef\]](#)
8. Adhikary, S.; Banerjee, S. Introduction to Distributed Nearest Hash: On Further Optimizing Cloud Based Distributed kNN Variant. *Procedia Comput. Sci.* **2023**, *218*, 1571–1580. [\[CrossRef\]](#)
9. Rosenfeld, A.; Tsotsos, J.K. Incremental Learning Through Deep Adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 651–663. [\[CrossRef\]](#)
10. Schnörr, D.; Schnörr, C. Learning system parameters from turing patterns. *Mach. Learn.* **2023**, *112*, 3151–3190. [\[CrossRef\]](#) [\[PubMed\]](#)
11. Liu, Y.; Su, Y.; Liu, A.-A.; Schiele, B.; Sun, Q. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020*.
12. Yang, G. Multilayer Neuroadaptive Reinforcement Learning of Disturbed Nonlinear Systems Via Actor-Critic Mechanism. *IEEE Trans. Autom. Sci. Eng.* **2025**, *23*, 1372–1381. [\[CrossRef\]](#)
13. Przybyła-Kasperek, M.; Marfo, K.F.; Sulikowski, P. Multi-Layer Perceptron and Radial Basis Function Networks in Predictive Modeling of Churn for Mobile Telecommunications Based on Usage Patterns. *Appl. Sci.* **2024**, *14*, 9226. [\[CrossRef\]](#)
14. Jain, M.; Kaur, H.; Gupta, B.; Gera, J.; Kalra, V. Incremental learning algorithm for dynamic evolution of domain specific vocabulary with its stability and plasticity analysis. *Sci. Rep.* **2025**, *15*, 272. [\[CrossRef\]](#)
15. Belouadah, E.; Popescu, A.; Kanellos, I. Initial Classifier Weights Replay for Memoryless Class Incremental Learning. *arXiv* **2020**, arXiv:2008.13710. [\[CrossRef\]](#)
16. Yu, L.; Twardowski, B.; Liu, X.; Herranz, L.; Wang, K.; Cheng, Y.; Jui, S.; Weiher, J. Semantic Drift Compensation for Class-Incremental Learning. *arXiv* **2020**, arXiv:2004.00440. [\[CrossRef\]](#)
17. Roy, D.; Panda, P.; Roy, K. Tree-cnn: A hierarchical deep convolutional neural network for incremental learning. *Neural Netw.* **2020**, *121*, 148–160. [\[CrossRef\]](#)
18. Mallya, A.; Lazebnik, S. *PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning*; IEEE: New York, NY, USA, 2018; pp. 7765–7773. [\[CrossRef\]](#)
19. Xu, Y.; Furao, S.; Hasegawa, O.; Zhao, J. An Online Incremental Learning Vector Quantization. In *Proceedings of the Advances in Knowledge Discovery and Data Mining, PAKDD 2009; Lecture Notes in Computer Science*; Theeramunkong, T., Kijirikul, B., Cercone, N., Ho, T.B., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; p. 5476. [\[CrossRef\]](#)
20. Kemker, R.; Kanan, C. FearNet: Brain-Inspired Model for Incremental Learning. *arXiv* **2017**, arXiv:1711.10563. [\[CrossRef\]](#)
21. Hayes, T.L.; Kanan, C. Lifelong Machine Learning with Deep Streaming Linear Discriminant Analysis. *arXiv* **2020**, arXiv:1909.01520. [\[CrossRef\]](#)
22. Zhang, C.; Jiang, C.; Xu, Q. Pretrained back propagation based adaptive resonance theory network for adaptive learning. *J. Algorithms Comput. Technol.* **2023**, *17*, 13. [\[CrossRef\]](#)
23. Zhao, B.; Xiao, X.; Gan, G.; Zhang, B.; Xia, S. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020*; IEEE: New York, NY, USA, 2020; pp. 13205–13214.
24. Zhao, H.; Bi, Y.; Tian, S.; Wang, J.; Zhang, P.; Deng, Z.; Liu, K. Self-Knowledge Distillation via Progressive Associative Learning. *Electronics* **2024**, *13*, 2062. [\[CrossRef\]](#)
25. Madi, S. Datafiles for Incremental Research.rar. Figshare. Dataset. 2023. Available online: https://figshare.com/articles/dataset/datafiles_for_incremental_research_rar/23623185/1?file=41 (accessed on 16 April 2025).
26. Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; Lampert, C.H. iCaRL: Incremental Classifier and Representation Learning. *arXiv* **2017**, arXiv:1611.07725. [\[CrossRef\]](#)

27. Li, S.; Su, T.; Zhang, X.-Y.; Wang, Z. Continual Learning with Knowledge Distillation: A Survey. *Neural Netw. Learn. Syst.* **2024**, *36*, 9798–9818. [\[CrossRef\]](#)
28. Hou, Q.; Leng, J.; Ma, G.; Liu, W.; Cheng, Y. An adaptive hybrid model for short-term urban traffic flow prediction. *Phys. A Stat. Mech. Its Appl.* **2019**, *527*, 121065. [\[CrossRef\]](#)
29. Phuong, M.; Lampert, C. Towards Understanding Knowledge Distillation. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 5142–5151.
30. Rusu, A.A.; Rabinowitz, N.C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; Hadsell, R. Progressive neural networks. *arXiv* **2016**, arXiv:1606.04671. [\[CrossRef\]](#)
31. Fan, X.; Zhang, S.; Xue, X.; Jiang, R.; Fan, S.; Kou, H. An Improved Self-Organizing Map (SOM) Based on Virtual Winning Neurons. *Symmetry* **2025**, *17*, 449. [\[CrossRef\]](#)
32. Madi, S.; Baba-Ali, A.R. A new hybrid incremental learning system for an enhanced KNN algorithm (hoKNN). *Evol. Syst.* **2024**, *15*, 1001–1019. [\[CrossRef\]](#)
33. Tao, X.; Hong, X.; Chang, X.; Dong, S.; Wei, X.; Gong, Y. Few-shot class-incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 12183–12192.
34. He, C.; Wang, R.; Shan, S.; Chen, X. Exemplar-supported generative reproduction for class incremental learning. In *Proceedings of the British Machine Vision Conference 2018, BMVC, Newcastle, UK, 3–6 September 2018*; Northumbria University: Newcastle, UK, 2018; p. 98.
35. Zhou, P.; Mai, L.; Zhang, J.; Xu, N.; Wu, Z.; Davis, L.S. M2KD: Multi-model and multi-level knowledge distillation for incremental learning. *arXiv* **2019**, arXiv:1904.01769. [\[CrossRef\]](#)
36. Migenda, N.; Möller, R.; Schenck, W. Adaptive local Principal Component Analysis improves the clustering of high-dimensional data. *Pattern Recognit.* **2024**, *146*, 110030. [\[CrossRef\]](#)
37. Douillard, A.; Cord, M.; Ollion, C.; Robert, T.; Valle, E. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the Computer Vision—ECCV 2020—16th European Conference Proceedings, Part XX*; Glasgow, UK, Vedaldi, A., Bischof, H., Brox, T., Frahm, J., Eds.; Springer: Berlin/Heidelberg, Germany, 23–28 August 2020; pp. 86–102.
38. Anna, V.; Joaquin, V.; Mohamed, R.B.; Thorsteinn, R. Learning to Learn without Forgetting using Attention. In Proceedings of the 3rd Conference on Lifelong Learning Agents (CoLLAs), Pisa, Italy, 29 July–1 August 2024. [\[CrossRef\]](#)
39. Tscheschner, B.; Veas, E.; Masana, M. Incremental Learning with Repetition via Pseudo-Feature Projection. *arXiv* **2025**, arXiv:2502.19922. [\[CrossRef\]](#)
40. Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the Computer Vision—ECCV 2018—15th European Conference Proceedings, Part III*; Munich, Germany, Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 8–14 September 2018; pp. 144–161.
41. Khawand, J.; Hanappe, P.; Colliaux, D. Continual Learning with Deep Streaming Regularized Discriminant Analysis. *arXiv* **2023**, arXiv:2309.08353. [\[CrossRef\]](#)
42. Lu, Y.; Tamas, F.; Penny, L. Exploring the roles of memory replay in targeted memory reactivation and birdsong development: Insights from computational models of complementary learning systems. *bioRxiv* **2024**. [\[CrossRef\]](#)
43. Lomonaco, V.; Maltoni, D. CORE50: A new Dataset and Benchmark for continual Object Recognition. In Proceedings of the 1st Annual Conference on Robot Learning, PMLR, Mountain View, CA, USA, 13–15 November 2017; Volume 78, pp. 17–26.
44. Crop Yield_1 Datasets. Available online: <https://datasets.omdena.com/dataset/crop-yield-prediction> (accessed on 10 May 2024).
45. Crop Yield_2 Datasets. Available online: <https://www.kaggle.com/datasets/patelris/crop-yield-prediction-dataset> (accessed on 10 May 2024).
46. CORE50 Datasets; CORE50, Designed for Continual Object Recognition, Consists of 50 Household Items Organized Into 10 Categories: Plug Adapters, Mobile Phones, Scissors, Light Bulbs, Cans, Glasses, Balls, Markers, Cups, and Remote Controls. Classification Can Be Conducted at the Object Level, Comprising 50 Classes, or at the Category Level, Consisting of 10 Classes. the Default Task Is Significantly More Challenging Because of the Difficulty in Differentiating Objects Within the Same Category at Specific Locations. the Dataset Was Collected over 11 Independent Sessions, Consisting of 8 Indoor and 3 Outdoor Environments, Each with Diverse Backgrounds and Lighting Conditions. Each Session and Item Was Documented as a 15-Second Video (at 20 fps) Utilizing a Kinect 2.0 Sensor, Yielding 300 RGB-D Frames. Available online: <https://vlomonaco.github.io/core50/index.html#dataset> (accessed on 10 April 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.