An Efficient Bayesian Neural Network for Multiple Data Streams

Ming Zhou, Yiliao Song, Guangquan Zhang, Bin Zhang and Jie Lu

Decision System&e-Service Intelligence (DeSI) Lab
Australian Artificial Intelligence Institute, *Faculty of Engineering and Information Technology*
University of Technology Sydney, *Sydney, Australia*
{*Ming.Zhou, Bin.Zhang-4*}*@student.uts.edu.au,* {*Yiliao.Song, Guangquan.Zhang, Jie.Lu*}*@uts.edu.au*

***Abstract*—Spatial and temporal data such as multiple data streams often have concept drift problems, which refers to changes of the data distributions over time. Once concept drift occurs, a stationary machine learning predictor will probably be invalid because the testing data have a different data distribution from that of the training data. Recent studies in data streams aim to address this issue by concept drift adaptation techniques. Concept drift adaptation methods update the predictor by time and have been validated to provide accurate real-time prediction for single data stream. However, how to handle multiple relevant data streams is still an unsolved challenge in this area, considering many real-world applications evolving multiple data streams simultaneously. To fill this gap, we here present a predicting network for multiple data streams, named *MuNet*, which can provide real-time predictions for multiple data streams with considering the dependency between data streams. In *MuNet*, an online-learned Bayesian neural network (BNN) is designed as a *connector* between streams. The proposed *BNN-connector* can continuously use the real-time information from *only one* base stream to correct the stationary predictor for the other streams, which efficiently lowers down the computational cost.**

***Index Terms*—Bayesian Neural Network, Spatial and temporal data mining, concept drift, multiple streams**

I. Introduction

Streaming data is widespread in many practical scenarios such as finance, transportation, weather, social, etc., where infinite data instances arrive in a sequential way [1]. The streaming data mining has been one of the most important research topics in recent years [2], especially how to overcome the intrinsic uncertainties and obtain accurate real-time predictions have attracted increasing attention in this area [3]. One of the most common uncertainties in data streams is the occurrence of concept drift [4]. Concept drift refers to the phenomenon that data distributions change over time [5]. For example, the weather forecast of "rain/no rain" largely depends on whether it is in the dry season or wet season [6].

In conventional machine learning methods, the training set and test set are assumed to have the same probability distribution [7], so that a predictor trained with the training set can still be used to predict the testing set. Once concept drift occurs, the trained predictor will be probably invalid to predict the testing data because they have different or even opposite data patterns, resulting in a decrease in the prediction accuracy of the predictor for the coming data [8], [9]. In data stream prediction, when concept drift occurs frequently, the predictor is not applicable if it is trained with only fixed historical data [10].

Current research shows that concept drift handling techniques can solve this problem efficiently [11]. The detection, the quantification and the adaptation are currently challenging problems in the field of concept drift [12]. Drift detection is to detect when the concept drift occurs [13] by monitoring the learner error [14] or a designed statistic [15]. Regarding the quantification of conceptual drift, it involves retrieving the time when the drift occurred, the severity of the drift, and the area of the drift. Most of the drift detection algorithms can measure time and severity of the drift, but whether the specific location of the drift can be determined depends very much on the nature of the predictor itself [16]. The purpose of concept drift adaptation is to integrate the latest data machine statistical distribution information into the predictor to maintain or improve the prediction accuracy. There are usually two strategies to adjust the predictor [17]. The first is to retrain the predictor with the latest data to adapt to the new distribution, and the second is to update the original predictor with new data training [18], [19].

So far, the concept drift handling techniques have been validated in many single data stream cases [20], [21]. However, how to solve multiple data streams with concept drift is still an unsolved question. In addition, data streams in real scenarios basically do not exist independently, and data mining on multiple related data streams is the real dilemma [22]. Taking a traffic scene as an example, the bus routes of different lines in a city may overlap or intersect. The operating information of these lines is concurrent in time, and there is a certain degree of dependency between each other.

To fill the research gap in multiple streams, this paper proposes a concept drift adaptation method called $MuNet$ for multiple dependent data streams. Especially, we discuss the scenario that multiple data streams contain the concurrent drift, that is, the similar concept drift would occur simultaneously for all these data streams. Taking a traffic scenario as an example, when there is traffic congestion on overlapping lines, all buses on the same road section will be affected simultaneously.

In $MuNet$, we propose a novel adaptation strategy for mul-

tiple data streams with concurrent drift and we use Bayesian neural networks [23], [24] to implement this strategy. The main idea of the proposed adaptation strategy is to randomly pick one stream as the base stream. We train and update the predictor for this base stream by using a drift adaptation method designed for the single data stream. As for the other streams, we only train an initial predictor with a batch of historical data and this predictor will not be stored in the memory without updating. Then, the Bayesian neural network would be designed as the *connectors* between the base stream and other streams. These *connectors* can continuously learn the drift information from the base stream and use the learned information to correct the prediction results for other streams. As it takes less time to update the connector than updating the predictor, the computational cost has been largely decreased when predicting data streams rather than the base stream. It is also easy to be extended to other scenarios for application, such as the monitoring of water quality and water level in a certain water area, the speed prediction of each wind turbine in a certain wind power station, etc. The data streams recorded by multiple sensors establish a more comprehensive regional monitoring system, and $MuNet$ can maximize the operating efficiency of this system and minimize its computation and time costs.

The remainder of this paper will cover the following: Section II reviews related works. Section III presents the proposed FDA matrix. Section IV reports the empirical studies of our method on the synthetic data streams. The conclusions and future works are drawn in Section V.

II. Related Work

Concept drift is a common phenomenon in non-stationary data streams. The data stream is constantly updated over time, but due to various factors, the historical data distribution and the future data distribution may deviate, i.e., $D_{0,t}(X, y) \neq D_{0,t+1}(X, y)$ [25]. In fact, the concept drift on data streams is inevitable, especially in the context of the current high-speed generation, dissemination and update of information [26]. Following the inconsistency of data distribution, there are also changes in the statistical information of various target objects.

Recent research on concept drift adaptation is devoted to solving this problem [27]. Experiments have verified that by continuously updating target variables and adjusting the model over time, the concept drift adaptation can provide accurate real-time predictions for a single data stream. A popular variable sliding window algorithm is ADWIN [28], which checks all possible window cuts and calculates the optimal window size based on the difference in data distribution between two sub-windows. Thus using the latest window data to train the new model. DELM [29] expands on the traditional Extreme Learning Machine (ELM) algorithm [30]. When concept drift may occur, more nodes will be added to its hidden layer to improve the model's fitting ability. Similarly, FP-ELM [31] introduces forgetting parameters on the basis of ELM to ensure that the model adapts to concept drift. SAGA [32] is an informed adaptation method, which develops a drift region-based data sample filtering method to update the obsolete model and track the new data pattern accurately. A drift region-based data sample filtering method [33] based on competence model is used to distinguish drift instances from noise. The drift gradient in the algorithm is defined on a segment in the training set. When only one new instance is available at each point in time, it can accurately quantify the increase in the distribution difference between the old segment and the latest segment. Adaptive Random Forest (ARF) [34] is one of them. It uses ADWIN algorithm to extend Random Forest to decide when to replace an obsolete tree. The IADEM-3 model [35] is based on the classic online decision tree model. As related research shows that the Hoeffding boundary used in VFDT [36] may not be suitable for the split calculation of nodes, in the IADEM-3 algorithm, the sum of independent random variables is used to correct the problem caused by the Hoeffding boundary. Meanwhile, the error rate of the subtree is used to detect concept drift and tree pruning.

In order to eliminate the influence of concept drift, it is necessary to design and build complex adaptive algorithms. Existing work can indeed provide accurate prediction results for online data streams by retraining all data, or retraining local latest data through sliding windows, or continuously incrementally updating and adjusting the model, but their shortcomings are also obvious. They can be only limited applied on a single data stream [37]. Practically, the concept drift on multiple data streams is more complicated [38]. In some prediction tasks, the data analysis of multiple data streams needs to be performed synchronously [39]. Under the premise of extremely fast information stream update speed, setting up a separate concept drift processing module on each data stream will not only increase the amount of computation exponentially, but also greatly extend the time required for the prediction task [40], [41].

Therefore, in this paper, we consider the scenario of multiple dependent data streams with concurrent drift. That is, data streams with the same target variable in a specific area are affected by similar factors. When they experience concept drift, their drift time, degree of drift and amount of drift will all have a strong correlation, which means that there is a mapping relationship between them. However, processing each data stream individually is Undoubtedly inefficient. This article will propose a new method for concurrent drift on multiple data streams, which aims to quickly and efficiently provide accurate prediction results for multiple data streams.

III. network for multiple data streams(MuNet)

A. Notations and Concepts

Before we explain the proposed method in details, related notations and definitions are listed in this subsection.

Notations:

- t: $t \in \mathbb{Z}^+$ presents the time point
- τ: $\tau > 1$ presents a time period
- $\mathcal{P}$: probability distribution

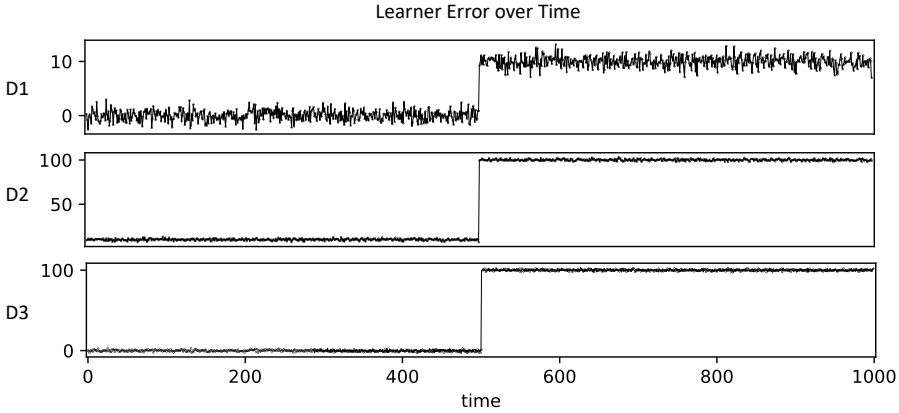

Fig. 1. An example of concurrent drift among data streams. It can be inferred that data streams D^1, D^2 and D^3 have a sudden error increase (sudden drift) at the same time when $t = 500$.

- p: the probability for discrete cases or probability density function for continuous cases
- $\boldsymbol{X_t} = (X_t^1, ..., X_t^m)$: time series for features or attributes
- $\boldsymbol{y_t} = (y_t^1, ..., y_t^l)$: time series for labels. in this paper, we only consider l=1
- $s = \{(\boldsymbol{X_t}, \boldsymbol{y_t})\}$: data stream

Definition 1 (Single Data Stream)**.** *A data stream $D_t = \{(\boldsymbol{X}_t, y_t) | t = 1, ...\infty\}$, generated from distribution $\mathcal{P}_t$ with $p_t(\boldsymbol{X}, y)$ its probability function or probability density function (pdf), is received, where $\{\boldsymbol{X}_t \in \mathbb{R}^d\}$ is the attribute variable (or the input) consisting of d time series, for some d, and $\{y_t \in \mathbb{R}^1\}$ is the label variable (or the scalar output).*

Definition 2 (Concept Drift in Single Data Stream [32])**.** *Concept drift occurs in a data stream if $\exists t_{d^i}$ that*

$$\begin{cases} p_{t+1}(\boldsymbol{X}, y) \neq p_t(\boldsymbol{X}, y), \text{ for } t = t_{d^{(i)}} \\ p_{t+1}(\boldsymbol{X}, y) = p_t(\boldsymbol{X}, y), \text{ for } t \in \left(t_{d^{(i)}+\tau_i}, t_{d^{(i+1)}}\right) \end{cases} \tag{1}$$

where $\forall i,\ t_{d^{(i+1)}} - t_{d^{(i)}} > 1$, $t \in \mathbb{Z}^+$ presents the time step, $d^{(i)}$ is an order statistics denoting the i^{th} drifted time point, and $1 < \tau_i < t_{d^{(i+1)}} - t_{d^{(i)}}$ is specifically for the occurrence of incremental drift.

Remark 1. *A data stream contains concept drift if the data pattern changes at least once, namely $\{t_{d^{(i)}}\} \neq \emptyset$ that $p_{t+1}(\boldsymbol{X}, y) \neq p_t(\boldsymbol{X}, y)$, for $t = t_{d^{(i)}}$; in addition, the changed pattern is not ephemeral, but will last for a period (at least last for two time steps), which is manifested by $\forall i,\ t_{d^{(i+1)}} - t_{d^{(i)}} > 1$. The pattern stays the same in this period that $p_{t+1}(\boldsymbol{X}, y) = p_t(\boldsymbol{X}, y)$, for $t \in \left(t_{d^{(i)}+\tau_i}, t_{d^{(i+1)}}\right)$; here $\tau_i = 1$ when the drift occurs suddenly while $\tau_i > 1$ when the drift occurs incrementally in the period of $\left(t_{d^{(i)}+1}, t_{d^{(i)}+\tau_i}\right)$.*

Definition 3 (Learning Aim at t-step for Single Data Stream)**.** *To predict the value of the label variable for a data stream at time step t, the learning aim is to obtain a predictor H_t for $p_t(\boldsymbol{X}, y)$, which can be denoted as*

$$H_t = \underset{h \in \mathcal{H}}{\arg\min}\ \ell\left(h, \boldsymbol{X}, y | \left(\boldsymbol{X}, y\right) \in p_t\left(\boldsymbol{X}, y\right)\right), \tag{2}$$

where $\mathcal{H}$ is the hypothesis set, $\ell : \mathbb{R}^1 \times \mathbb{R}^1 \to \mathbb{R}_+$ is the loss function used to measure the magnitude of error.

Definition 4 (Concurrent Drift in Multiple Data Streams)**.** *Given D_t^u, D_t^v (where $u \neq v$) two different data streams, p^u is the probability or probability density function for stream u while p^v is that for stream v. A concurrent drift occurs if $\exists t_{d^{(u)}}$ that*

$$\begin{cases} p_{t+1}^u \neq p_t^u, p_{t+1}^v \neq p_t^v \text{ for } t = t_{d^{(i)}} \\ p_{t+1}^u = p_t^u, p_{t+1}^v = p_t^v \text{ for } t \in \left(t_{d^{(i)}+\tau_i}, t_{d^{(i+1)}}\right) \end{cases} \tag{3}$$

Remark 2. *It can be seen that concurrent drift occurs when concept drift occurs in two data streams at the same time. It should be noticed that the concurrent drift could occur among more than two data streams, such as is shown in Figure 1. In Figure 1, D^1, D^2 and D^3 represent three data streams separately and each sub-figure shows the learner error of a static predictor changes over time. Normally, error is the most common measurement to determine whether drift occurs [37]. Clearly, concurrent drift occurs among D^1, D^2, and D^3 as drift occurs at $t = 500$ for D^1, D^2, and D^3.*

Definition 5 (Learning Aim at t-step for Multiple Data Streams)**.** *For multiple data streams D^1, D^2, $\cdots$, D^S at time step t, the learning aim is to obtain a set of predictors $H_t^1, H_t^2, \cdots, H_t^S$ for $p_t\left(D^1, D^2, \cdots, D^S\right)$, which can be denoted as*

$$H_t^s = \underset{h^s \in \mathcal{H}^s}{\arg\min}\ \ell\left(h^s, D^s | D^1, \cdots, D^S \in p_t\left(D^1, \cdots, D^S\right)\right), \tag{4}$$

where $\mathcal{H}^s$ is the hypothesis set for D^s and $s \in \{1, \cdots, S\}$.

Remark 3. *According to Definition 5, the learning aim in a multi-stream scenario is to obtain multiple predictors at each time point with considering the relationship among streams. If we consider each stream separately, Definition 5 will be degenerated to the following case*

$$H_t^s = \underset{h^s \in \mathcal{H}^s}{\arg\min}\ \ell\left(h^s, D^s | D^s \in p_t\left(D^s\right)\right). \tag{5}$$

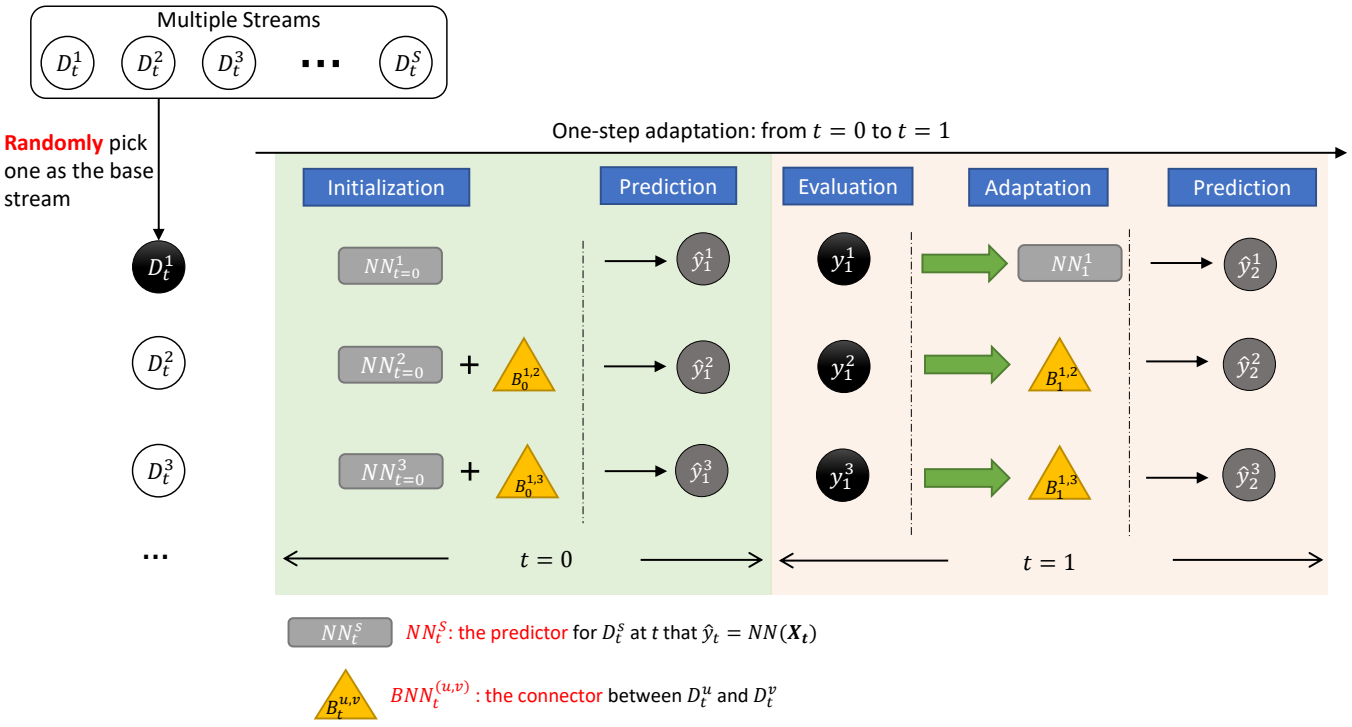

Fig. 2. The One-step Adaptation Procedure for Multiple Streams with Concurrent Drift.

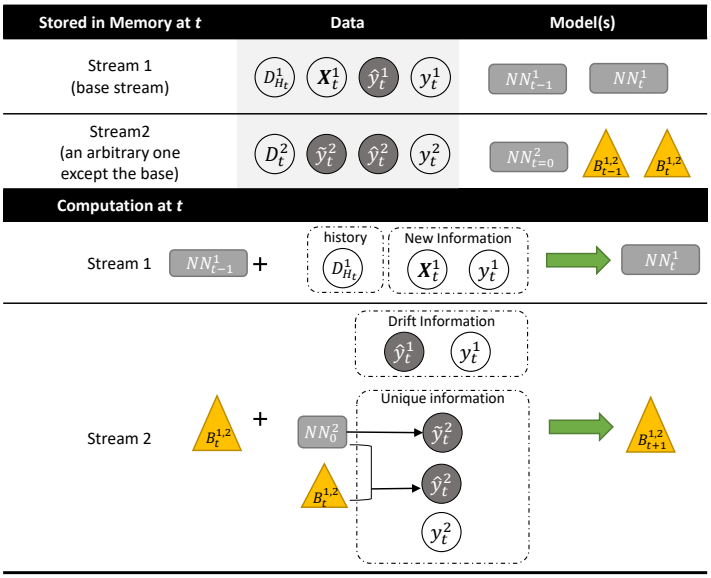

Fig. 3. The proposed $MuNet$.

B. The proposed $MuNet$

Given the learning aim in Definition 5, the next step is how to obtain the optimal H_t^s. If the data streams do not have concept drift problem and the streams are independent with each other, the probability function stays the same and $H_t^s \equiv H^s$. That means we could use a period of historical data to train a predictor H^s and use this predictor to predict without retraining or tuning H^s. However, due to the occurrence of concept drift, the assumption that data streams have unchanged data distributions has been broken. In the meantime, data streams are assumed to be dependent with each others in the multi-stream scenario.

Concept drift is a common phenomenon in data streams and it is unknown in advance whether drift occurs in a data stream. It is risky to use stationary model for an arbitrary data stream as the stationary model will lead to terrible prediction results when drift occurs while drift-adaptive models are still suitable to predict data streams that do not contain concept drift. Therefore, the design of connector is very useful for many real-world applications related to multiple data streams.

In this paper, we consider the case when streams have concurrent drift (Definition 4). We randomly pick one stream as the base stream. We train and update the predictor for the base stream to handle the drift problem. As for the other streams, we only train an initial stationary predictor with a batch of historical data. After that, predictions from the initial predictor will be *corrected* by an online-learned *connector*. Compared to the predictor (a complex neural network), the connector (a neural network with 1-2 layers) takes much less computational cost. By this way, $MuNet$ can efficiently decrease the computational cost when predicting multi-streams task.

For better understanding, we first introduce the general idea of how we make drift adaptation for multiple streams with concurrent drift. After that, we give technical details of how $MuNet$ implements that adaptation idea.

- The one-step adaptation for multiple streams with concurrent drift.

The one-step adaptation procedure for multiple streams with concurrent drift is presented in Figure 2. There are S data streams and we assume that these streams have concurrent drift. Current drift adaptation methods are designed for single data stream. Therefore, if we directly use those methods to solve multi-stream task, the adaptation for all streams will be the same as the adaptation for D_t^1 in Figure 2. That is to retrain the predictor periodically, for example retraining every batch of new arrived instances (where t denotes a time period) or even every one instance (where t denotes a time point).

Although retraining predictors is an efficient way to solve the problem of concept drift in data streams, it often has high computational cost. Considering the unique characteristics of concurrent drift, we assume that the drift pattern of one data stream can be learned and applied to other streams because these streams have drift at the same time. Therefore, we propose to *update predictors* for one base data stream by retraining and *correct prediction results* for other streams by updating connector built between the base stream and others.

For example, in Figure 2, we first pick one stream from S streams as the base stream. It should be noticed that in this paper we pick the base stream randomly and we also give an analysis of accuracy fluctuate with selecting different base streams in Section IV. At $t = 0$, the predictors and connectors are *initialized*. In this paper, we use a complex neural network as the predictor and a two-layer Bayesian neural network as the connector. With the initialized predictors and connectors, we can *estimate the labels* for $t = 1$ which are denoted by $\hat{y}_1^1$ for stream D^1, $\hat{y}_1^2$ for stream D^2, and so on.

As discussed above, we have the initialized predictors and connectors, as well as the prediction results at $t = 0$. Then the time goes to $t = 1$, we obtain the true label y_1^1, y_1^2, and so on. At $t = 1$, we evaluate the predictions of $\hat{y}_1^1$, $\cdots$, $\hat{y}_1^S$ and then use the true labels to update the predictor on the base stream and connectors on other streams. The updated predictor and connectors are used to predict labels for $t = 2$. Then the time goes to $t = 2$, we repeat the evaluation, adaptation and prediction processes. Unlike the stationary machine learning methods, the real-time prediction uses this prequential evaluation method [4].

- Handling concurrent drift with $MuNet$.

So far, we have discussed the main idea of updating predictors and connectors separately to reduce the computational cost. Next, we will explain how this idea is used in our proposed $MuNet$.

We introduce our $MuNet$ from both aspects of memory and computation at each t. Compared to Figure 2 which includes general processes in both $t = 0$ and $t = 1$, Figure 3 presents details at $t = 1$. As this procedure is repeated for each t, the case $t = 1$ is also the case for an arbitrary t that $t \neq 0$.

As is shown in Figure 2, $MuNet$ only stores the most current data. For example, if we only have two streams D_t^1 and D_t^2, and D_t^1 is chosed as the base stream, the stored data for base stream at t are a batch of historical data $D_{H_t}^1$, the predicted value($\hat{y}_t^1$), and the newly arrived data(X_t^1 and y_t^1). Compared to other streams, we store an extra batch of data with fixed length for the base stream, and this batch of data will be updated by including the newest data and deleting the oldest data at each t.

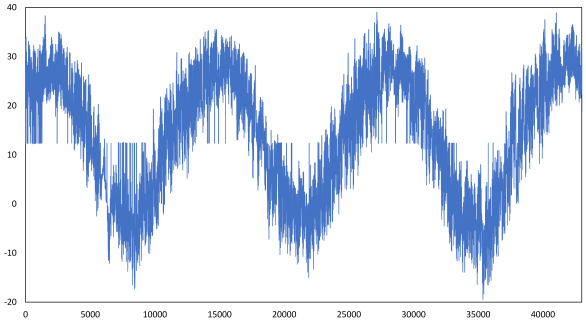

Fig. 4. The change of target variable *t2m_obs_gt* historical data.

The stored data for other streams at t are the predicted values($\hat{y}_t^2$, $\tilde{y}_t^2$) and the newly arrived data(X_t^2 and y_t^2). Compared to the base stream, there are two predicted values for Stream 2 where $\tilde{y}_t^2$) is the predicted value by the initial predictor and $\hat{y}_t^2$ is the predicted value by the initial predictor corrected by the connector $BNN_t^{1,2}$.

With these stored values, $MuNet$ is self-adapted as is shown in the computation part in Figure 2. The adaptation of $MuNet$ on Stream 1 is to retrain the neural network predictor NN_{t-1}^1 with the historical information and the new information. The adaptation of $MuNet$ on Stream 1 is to tune the Bayesian neural network with drift information from base stream and unique information for Stream 2.

When the predictor has high computational cost for updating and the connector cost low computational cost for updating, $MuNet$ will efficiently decrease the total computational cost, especially there are many streams to predict. For example, in weather prediction, we need to know the prediction for all weather station at the same time. These weather stations are normally built not far away from each other in one area. Therefore, the data streams from these stations have concurrent drift.

IV. Experimental Evaluation

We validate the proposed $MuNet$ by a real-world dataset of weather prediction. In this section, we will first introduce the data in Section IV-A. Section IV-B lists the parameters used in $MuNet$ and the experimental results are presented in Section IV-C.

A. Data Description

We use a public dataset collected from the weather stations in Beijing, China. This data contains hourly weather data from ten weather stations during time zone (3:00 intraday to 15:00 of the next day) from 03/01/2015-05/31/2018 (1188 days in total) where missing values were deleted during the testing. Each station contains same features.

To mimic the real prediction scenario, the original data has been pre-processed by overlapping in [42]. In this paper, we use the same pre-processed data as is in [42]. In our experiment, we use the data from ten stations. Each station data contains nine attributes, including *psur_obs* (ground pressure), *t2m_obs* (temperature at 2 meters), *q2m_obs* (specific humidity at 2 meters), *w10m_obs* (wind speed at 10 meters), *d10m_obs* (meridional wind at 10 meters), *rh2m_obs* (relative humidity at 2 meters), *u10m_obs* (warp wind direction at 10 meters), *v10m_obs* (zonal wind at 10 meters), *RAIN_obs* (accumulated rainfall in one hour at ground). The target variable is *t2m_obs_gt* (temperature at 2 meters). *"_obs"* represents the observed value, *"_obs_gt"* represents the ground truth value.

Clearly, each station contains 10 times series as one data stream, and we have 10 data streams. These data streams have concept drift due to the intrinsic characteristics of atmospheric variables. Take the target variable *t2m_obs_gt* as an example, it has obvious seasonal changes according to the three-year data record, as shown in Figure 4. As these stations are located near to each other, the corresponding data streams have the concurrent drift.

TABLE I
THE PERFORMANCE COMPARISON OF STATIONARY PREDICTOR AND ADAPTIVE PREDICTOR

	S_i*_Stationary*	S_i*_Adaptation*
S_0*_MAPE_AVG*	10.67%	6.57%
S_1*_MAPE_AVG*	10.41%	6.32%
S_2*_MAPE_AVG*	10.63%	6.66%
S_3*_MAPE_AVG*	10.76%	6.69%
S_4*_MAPE_AVG*	10.32%	6.31%
S_5*_MAPE_AVG*	9.98%	6.34%
S_6*_MAPE_AVG*	11.15%	7.23%
S_7*_MAPE_AVG*	10.26%	6.25%
S_8*_MAPE_AVG*	10.29%	6.23%
S_9*_MAPE_AVG*	10.52%	6.80%

B. Experimental Settings

The experiments were implemented on a CPU server with Core i7 7700K CPU and Pytorch (1.8.1) programming environment.

The $MuNet$ contains *predictors* and *connectors*.

The *predictors* for all data streams uses a 5-layer neural network and each layer contains 128 neurons. The input of the predictor is the 9 meteorological attribute observations at the observation point at time t, and the output is the predicted value of the target variable at time $t+1$. The training epoch of predictors is 2000 for all streams. All predictors use the stochastic gradient descent (SGD) method, and the learning rate is 0.01. As for the predictors initialization, we use 2400 hours of recorded weather data in the same time period of all streams as the training data for their initial predictors. The predictor of base stream is retrained every 120 hours by including the newly arrived 120 hours' data and deleting the oldest 120 hours' data.

The *BNN-connector* is a Bayesian neural network containing 2-layer hidden layers and each layer contains 30 and 10 neurons respectively. The training epoch of each *BNN-connector* is 30. All *BNN-connectors* use the SGD method, and the learning rate is $1e-1$. *BNN-connectors* are updated by continuously tuned by the newly arrived instances.

C. Results

We have conducted three groups of experiments: a) comparisons between stationary methods and the adaptation methods; b) comparisons between the adaptation by the single data stream adaptation method and the adaptation by $MuNet$; c) time cost of $MuNet$ when adding extra streams. For all experiments, we use the mean absolute percentage error (*MAPE*) as the evaluation criterion. *MAPE* is calculated as follows:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

Where $\hat{y}_i$ is the predicted value, y_i is the true value, and n is the number of samples. The smaller the value of *MAPE*, the closer the predicted value is to the true value. It means that the prediction model is more accurate. In the tables of experimental results, S_i*_MAPE_AVG* represents the average value of *MAPE* of all prediction results on the current data stream S_i, which can indicate the overall prediction level of the model.

Experiment a) is to validate whether these data streams have the concept drift problem. The results are shown in Table I where S_i*_Stationary* uses an unchanged initial predictor while S_i*_Adaptation* retrains the predictor every 120 hours. It can be seen that the overall accuracy of S_i*_Adaptation* is much better than its corresponding accuracy of S_i*_Stationary*, indicating that these data streams contain concept drift problem.

Experiment a) concludes that all these data streams contain concept drift problem, and retraining predictors is an efficient solution. Based on a), Experiment b) is to test whether we can use $MuNet$ to predict these multiple data streams with less computational cost and comparable accuracy. The results are listed in Table II. The accuracy of S_i*_Adaptation* is 6.57%, 6.32%, 6.65%, 6.68%, 6.30%, 6.33%, 7.22%, 6.27%, 6.21% and 6.79% for stream S_0 to S_9 separately. The accuracy of $MuNet$ with S_0 the base stream is a little lower than S_i*_Adaptation*, which is 6.57%, 6.94%, 7.39%, 7.36%, 6.88%, 6.96%, 8.16%, 6.67%, 6.91% and 7.49% for stream S_0 to S_9 separately. But $MuNet$ takes less time obviously. In addition, in Table II, We test the $MuNet$ performance by using ten data streams from S_0 to S_9 as the base stream separately. When using different base streams, the accuracy and time cost are slightly different but all the results are in a reasonable range.

Lastly, we list the time cost increases of S_i*_Adaptation* and $MuNet$ separately when an extra stream is gradually added in Experiment c). According to the results in Table III, the time cost is the same between S_i*_Adaptation* and $MuNet$ when there is only one stream S_0, as now $MuNet$ is exactly the same as S_i*_Adaptation*. Then we add the S_1 stream. The time cost of S_i*_Adaptation* increase from 87 to 167, and the time cost of $MuNet$ increase from 87 to 130. As a result,

TABLE II
THE PERFORMANCE COMPARISON OF THE MUNET AND CONVENTIONAL ADAPTATION METHOD

	S_i_Adaptation	$MuNet$ (S_i_base)									
		S_0_base	S_1_base	S_2_base	S_3_base	S_4_base	S_5_base	S_6_base	S_7_base	S_8_base	S_9_base
S_0_MAPE_AVG	6.57%	**6.57%**	7.24%	7.36%	7.29%	7.28%	7.33%	7.41%	7.45%	7.18%	7.28%
S_1_MAPE_AVG	6.32%	6.94%	**6.32%**	7.16%	6.99%	6.88%	6.95%	7.38%	7.11%	6.97%	6.98%
S_2_MAPE_AVG	6.65%	7.39%	7.25%	**6.66%**	7.48%	7.25%	7.72%	7.53%	7.39%	7.21%	7.34%
S_3_MAPE_AVG	6.68%	7.36%	7.36%	7.49%	**6.69%**	7.37%	7.50%	7.63%	7.56%	7.44%	7.40%
S_4_MAPE_AVG	6.30%	6.88%	6.85%	6.99%	7.00%	**6.31%**	6.98%	7.24%	7.01%	6.86%	6.89%
S_5_MAPE_AVG	6.33%	6.96%	6.86%	7.03%	7.00%	6.93%	**6.34%**	7.14%	7.04%	6.83%	7.00%
S_6_MAPE_AVG	7.22%	8.16%	7.97%	8.30%	8.21%	8.02%	8.56%	**7.23%**	8.12%	7.95%	8.53%
S_7_MAPE_AVG	6.27%	6.67%	6.64%	6.68%	6.70%	6.65%	6.65%	6.76%	**6.25%**	6.64%	6.62%
S_8_MAPE_AVG	6.21%	6.91%	6.74%	7.12%	6.83%	6.77%	6.83%	6.96%	6.96%	**6.23%**	6.83%
S_9_MAPE_AVG	6.79%	7.49%	7.38%	7.65%	7.44%	7.38%	7.45%	7.61%	7.65%	7.44%	**6.80%**
Time Cost	886	350	361	373	358	347	355	370	354	350	360

TABLE III
THE COMPUTATIONAL COST OF DIFFERENT METHODS AS THE NUMBER OF DATA STREAMS INCREASES (S_0 AS THE BASE STREAM)

	One Streams		Two Streams		Three Streams		Ten Streams	
	S_i_Adaptation	$MuNet$	S_i_Adaptation	$MuNet$	S_i_Adaptation	$MuNet$	S_i_Adaptation	$MuNet$
S_0_MAPE_AVG	6.57%	6.57%	6.57%	6.57%	6.57%	6.57%	6.57%	6.57%
S_1_MAPE_AVG	-	-	6.32%	6.94%	6.32%	6.94%	6.32%	6.94%
S_2_MAPE_AVG	-	-	-	-	6.65%	7.39%	6.65%	7.39%
S_3_MAPE_AVG	-	-	-	-	-	-	6.68%	7.36%
S_4_MAPE_AVG	-	-	-	-	-	-	6.30%	6.88%
S_5_MAPE_AVG	-	-	-	-	-	-	6.33%	6.96%
S_6_MAPE_AVG	-	-	-	-	-	-	7.22%	8.16%
S_7_MAPE_AVG	-	-	-	-	-	-	6.27%	6.67%
S_8_MAPE_AVG	-	-	-	-	-	-	6.21%	6.91%
S_9_MAPE_AVG	-	-	-	-	-	-	6.79%	7.49%
Time Cost (min)	87	**87**	167	**130**	248	**176**	886	**350**

$MuNet$ is **37 minutes** faster S_i*_Adaptation*. Similarly, we add the S_2 stream. The time cost of S_i*_Adaptation* increase from 167 to 248, and the time cost of $MuNet$ increase from 130 to 176. $MuNet$ is **72 minutes** faster S_i*_Adaptation*. When the number of data streams increases to 10, it can be seen that the $MuNet$ already has a huge time advantage, which is almost **2.5 times** faster than S_i*_Adaptation*. Clearly, the time cost difference between S_i*_Adaptation* and $MuNet$ will be increasing greatly when more streams are included, indicating our $MuNet$ is an efficient prediction method for multiple data streams with concurrent drift.

V. CONCLUSION AND FUTURE STUDIES

This paper addresses the problem of concurrent concept drift in multiple streams with considering the correlations between data streams. An efficient method called $MuNet$ was designed to predict multiple streams. $MuNet$, one of the streams is randomly assigned as the base stream, predictors are retrained periodically for the base stream while the predicted value of other streams are corrected by an online learned Bayesian neural network called *BNN-connector*. As it takes less time to online learn *BNN-connector* than retraining predictor, $MuNet$ is able to predict multiple data streams with less computational cost as well as ensuring the prediction accuracy. We validated $MuNet$ on a real-world weather forecast dataset. The experimental results show that our method can efficiently save computational cost with high accuracy compared to traditional drift handling techniques for the single data stream.

In future research work, we will further improve the adaptive performance of the base stream predictor to ensure the basic prediction accuracy, while improving the learning efficiency of the *BNN-connector*. Meanwhile, we will extend our method to more realistic multiple streams, especially streams with noise.

ACKNOWLEDGMENT

The work presented in this paper was supported by the Australian Research Council (ARC) under discovery grant DP190101733 and FL190100149.

REFERENCES

[1] J. Lu, A. Liu, Y. Song, and G. Zhang, "Data-driven decision support under concept drift in streamed big data," *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 157–163, 2020.

[2] H. Ghomeshi, M. M. Gaber, and Y. Kovalchuk, "Eacd: evolutionary adaptation to concept drifts in data streams," *Data Mining and Knowledge Discovery*, vol. 33, no. 3, pp. 663–694, 2019.

[3] Z. Yang, S. Al-Dahidi, P. Baraldi, E. Zio, and L. Montelatici, "A novel concept drift detection method for incremental learning in nonstationary environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 1, pp. 309–320, 2020.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[5] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[6] Y. Song, J. Lu, H. Lu, and G. Zhang, "Fuzzy clustering-based adaptive regression for drifting data streams," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 3, pp. 544–557, 2019.

[7] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4802–4821, 2018.

[8] M. B. Harries, C. Sammut, and K. Horn, "Extracting hidden context," *Machine learning*, vol. 32, no. 2, pp. 101–126, 1998.

[9] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, Aug. 19-25, 2017, pp. 2280–2286.

[10] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[11] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," *Big data analysis: new algorithms for a new society*, pp. 91–114, 2016.

[12] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

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

[14] J. Shan, H. Zhang, W. Liu, and Q. Liu, "Online active learning ensemble framework for drifted data streams," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 2, pp. 486–498, 2018.

[15] A. Liu, J. Lu, and G. Zhang, "Concept drift detection via equal intensity k-means space partitioning," *IEEE Transactions on Cybernetics*, 2020.

[16] R. S. M. Barros and S. G. T. C. Santos, "A large-scale comparison of concept drift detectors," *Information Sciences*, vol. 451, pp. 348–370, 2018.

[17] A. Liu, J. Lu, and G. Zhang, "Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[18] Y. Song, G. Zhang, H. Lu, and J. Lu, "A self-adaptive fuzzy network for prediction in non-stationary environments," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Rio de Janeiro, Brazil, Jul. 8-13: IEEE, 2018, pp. 1–8.

[19] Y. Song, G. Zhang, H. Lu, and J. Lu, "A noise-tolerant fuzzy c-means based drift adaptation method for data stream regression," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. New Orleans, Louisiana, USA, Jun. 23-26: IEEE, 2019, pp. 1–6.

[20] J. Lu, A. Liu, Y. Song, and G. Zhang, "Data-driven decision support under concept drift in streamed big data," *Complex & Intelligent Systems*, pp. 1–7, 2019.

[21] O. Koller, C. Camgoz, H. Ney, and R. Bowden, "Weakly supervised learning with multi-stream cnn-lstm-hmms to discover sequential parallelism in sign language videos," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[22] M. D. Jones, H. L. Peterson, J. J. Pierce, N. Herweg, A. Bernal, H. Lamberta Raney, and N. Zahariadis, "A river runs through it: A multiple streams meta-review," *Policy Studies Journal*, vol. 44, no. 1, pp. 13–36, 2016.

[23] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, p. 106816, 2020.

[24] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, "Learning curve prediction with bayesian neural networks," 2016.

[25] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[26] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving systems*, vol. 9, no. 1, pp. 1–23, 2018.

[27] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[28] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[29] S. Xu and J. Wang, "Dynamic extreme learning machine for data stream classification," *Neurocomputing*, vol. 238, pp. 433–449, 2017.

[30] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[31] D. Liu, Y. Wu, and H. Jiang, "Fp-elm: An online sequential learning algorithm for dealing with concept drift," *Neurocomputing*, vol. 207, pp. 322–334, 2016.

[32] Y. Song, J. Lu, A. Liu, H. Lu, and G. Zhang, "A segment-based drift adaptation method for data streams," *IEEE transactions on neural networks and learning systems*, 2021.

[33] F. Dong, J. Lu, Y. Song, F. Liu, and G. Zhang, "A drift region-based data sample filtering method," *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

[34] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017.

[35] I. Frias-Blanco, J. del Campo-Avila, G. Ramos-Jimenez, A. C. Carvalho, A. Ortiz-Diaz, and R. Morales-Bueno, "Online adaptive decision trees based on concentration inequalities," *Knowledge-Based Systems*, vol. 104, pp. 179–194, 2016.

[36] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 97–106.

[37] Y. Song, G. Zhang, H. Lu, and J. Lu, "A fuzzy drift correlation matrix for multiple data stream regression," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2020, pp. 1–6.

[38] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, "An adaptive framework for multistream classification," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 1181–1190.

[39] A. Haque, S. Chandra, L. Khan, K. Hamlen, and C. Aggarwal, "Efficient multistream classification using direct density ratio estimation," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 155–158.

[40] S. Chandra, A. Haque, H. Tao, J. Liu, L. Khan, and C. Aggarwal, "Ensemble direct density ratio estimation for multistream classification," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1364–1367.

[41] A. Haque, Z. Wang, S. Chandra, B. Dong, L. Khan, and K. W. Hamlen, "Fusion: An online method for multistream classification," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 919–928.

[42] B. Wang, J. Lu, Z. Yan, H. Luo, T. Li, Y. Zheng, and G. Zhang, "Deep uncertainty quantification: A machine learning approach for weather forecasting," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2087–2095.