Diagnosing Concept Drift with Visual Analytics

Weikai Yang,* Zhen Li*, Mengchen Liu,† Yafeng Lu,‡ Kelei Cao*, Ross Maciejewski,§ Shixia Liu*

* School of Software, BNRist, Tsinghua University
† Microsoft
‡ Bloomberg L.P.
§ Computer Science, Arizona State University

(a) Stream-level visualization.
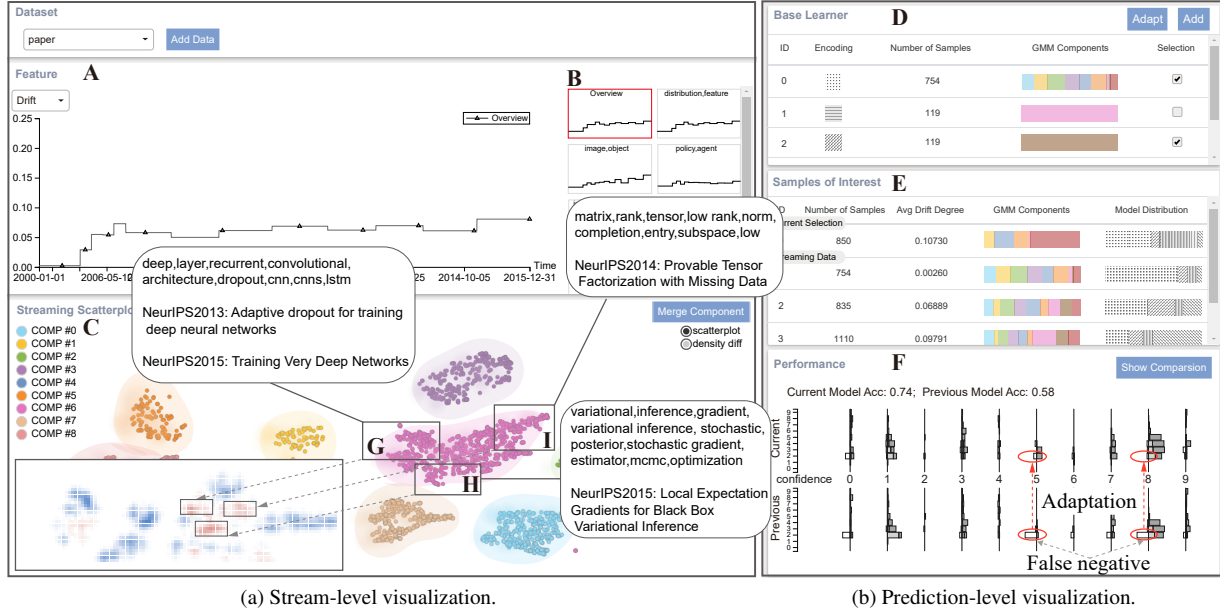
(b) Prediction-level visualization.

Figure 1: DriftVis: A visual analytics system for detecting, explaining, and correcting for concept drift: (a) The stream-level visualization consists of a line chart for drift degree (A), a feature selection list (B), and a streaming scatterplot (C) to visualize the drift and data distribution change over time (e.g., density increases in G, H, and I); (b) The prediction-level visualization consists of a base learner view (D), a samples of interest view (E), and a performance view (F) to explore the impact of drift adaptation on the model's accuracy.

ABSTRACT

Concept drift is a phenomenon in which the distribution of a data stream changes over time in unforeseen ways, causing prediction models built on historical data to become inaccurate. While a variety of automated methods have been developed to identify when concept drift occurs, there is limited support for analysts who need to understand and correct their models when drift is detected. In this paper, we present a visual analytics method, DriftVis, to support model builders and analysts in the identification and correction of concept drift in streaming data. DriftVis combines a distribution-based drift detection method with a streaming scatterplot to support the analysis of drift caused by the distribution changes of data streams and to explore the impact of these changes on the model's accuracy. A quantitative experiment and two case studies on weather prediction and text classification have been conducted to demonstrate our proposed tool and illustrate how visual analytics can be used to support the detection, examination, and correction of concept drift.

*e-mail:{{ywk19, liz11, ckl17}@mails., shixia@}tsinghua.edu.cn. Shixia Liu is the corresponding author.
†e-mail:mengcliu@microsoft.com
‡e-mail:luyafeng.cn@gmail.com
§e-mail:rmacieje@asu.edu

Keywords: Concept drift, streaming data, change detection, scatterplot, t-SNE.

1 INTRODUCTION

To date, artificial intelligence technologies have made immense strides in developing machine learning models (e.g., classifiers) for real-world phenomena [60]. Applications of these models are found in fraud detection [32], medical diagnosis [33], sales predictions [16], and countless other domains. Often, models have a built-in assumption where the mapping function of input data used to predict an output value (e.g., prediction label) is assumed to be static. However, as time passes, the mapping between the input data and the output value may change in unforeseen ways [45], which could be caused by the change of the data and label. In these cases, predictions made by a model trained on historical data may no longer be valid, and the model accuracy will begin to decrease over time. This phenomenon is typically referred to as *concept drift* in machine learning. As more and more machine learning applications move towards streaming data, the potential for model failure due to concept drift becomes further exacerbated.

To prevent the degradation of prediction accuracy, many drift analysis methods have been proposed in the field of machine learning [30, 45]. However, existing methods typically only provide a numerical drift value at each time point. While such measures can identify when drift occurs, a single numeric cannot explain why the drift is occurring. Analysts need to have tools that can do more than simply alert them to when concept drift seems to be occurring.

They need ways to understand how data distributions have changed over time, which samples cause drift, and how the training samples can be adjusted for model building. These underlying challenges of concept drift lend themselves well to a visual analytics paradigm in which the data and model can be readily evaluated for drift ("analyze first"). When drift is detected, changes in data distribution can be highlighted ("show what's important"), and an analyst only needs to label those drifted samples to update the model, which can save human effort. Once the analyst is satisfied, the system can return to the analysis state ("analyze again"). As such, we propose a visual analytics system, DriftVis, for detecting, explaining, and correcting for concept drift.

DriftVis has been designed to support the identification of concept drift, visually explain the drift analysis results, and enable the analysis of the root causes of such drift through a tight integration between the drift detection method and coordinated multiple views, creating a comprehensive visual analytics environment. As the change of data distribution is the root source of concept drift [45], in this work, we focus on analyzing concept drift caused by distribution drift. Accordingly, the drift detection method quantifies concept drift using a distance function that measures the difference between the distributions of the historical data used in model training and the incoming data. In our implementation, the data distribution is modeled by an incremental Gaussian mixture model (GMM). The energy distance, which is an extension of the L_2 distance for measuring high-dimensional distributions, is utilized to compute the dissimilarity between two distributions [65]. This drift detection method is used to support stream- and prediction-level visualizations. The stream-level visualization uses a drift degree line chart and a streaming scatterplot to disclose when and where concept drift might occur. If drift is detected, analysts can utilize the streaming scatterplot to compare the distributions over time. This scatterplot combines a constrained t-SNE and a multi-step animation to explain the distribution evolution. The prediction-level visualization reveals the performance-related information of the prediction model.

In order to demonstrate the capabilities of our system, we have conducted two case studies exploring the drift in weather prediction and academic paper topic evolution. A demo prototype of the system is available at `http://conceptdrift.thuvis.org/`. The main contributions of our work include:

- **A visual analytics system** for detection, examination, and correction of concept drift.
- **A distribution-based drift detection method** to quantify the drift degree over time.
- **A streaming scatterplot** to explore the distribution changes over time and identify the root cause of concept drift.

2 RELATED WORK

2.1 Concept Drift Analysis

Research related to concept drift can be categorized into two main topics: drift detection and drift adaptation [45].

Drift Detection. Error-rate-based methods and data-distribution-based methods are the most popular drift detection methods. The error-rate-based methods [4, 20] monitor the online error rate of a model based on ground-truth labels. A drift is detected if there is a significant increase in the error rate. These methods are less efficient in practice as it is often challenging to acquire high-quality data labels for calculating the error rate of streaming data.

Distribution-based techniques do not rely on data labels. Instead, these techniques directly detect concept drift by calculating the distribution change with respect to the streaming data, which is a major cause of concept drift. A key challenge for distribution-based techniques is how to measure the distribution change by calculating the distance between two data distributions. Kifer et al. [35] used total variation, which is one of the earliest attempts to use a distance function for drift detection. Webb et al. [70, 71] measured the dis-

tribution difference using the Hellinger distance and total variation. In their work, the drift degree was measured in different attribute subspaces, such as joint distribution and class distribution.

In our system, we utilize a distribution-based concept drift detection method, as these methods do not need ground-truth labels. Furthermore, distribution-based methods can identify the samples in the streaming data that cause concept drift, which is useful for facilitating the understanding of concept drift. Specifically, we model the streaming data distribution by an incremental GMM and apply the energy distance function [59] to measure the distance between distributions.

Drift Adaptation. Drift adaptation focuses on improving the model performance on the streaming data by updating the existing learning model to account for the drift. Existing methods of drift adaptation include model retraining and ensemble methods [45].

Model retraining retrains a new model with a combination of the new data and old data to replace the old model. The main problem encountered with these types of methods is docdetermining how much of the old data will be discarded and how much of the new data will be utilized in training [3]. Ensemble methods comprise a set of base learners and adjust the base learners when adapting to the drift. Typical adjustments include adding new base learners, updating existing base learners, removing unsatisfactory base learners, and changing the weights of base learners in the ensemble model. Hulten et al. proposed a tree-based method, CVFDT [29], which replaced the sub-tree exhibiting poor performance with a retrained sub-tree. The dynamically weighted majority method [37] down-weights the classifier that makes a wrong decision and up-weights the classifier that makes a correct decision. When the ensemble model makes a wrong decision, it will train a new base learner and add it to the ensemble, and the base learners with low weights will be removed. Learn^{++}.Net [17] re-weights the base learners based on the prediction performance on the latest streaming data to prevent adding new base learners too frequently.

Compared with model retraining, ensemble methods are more efficient because they do not require a full retraining of the model. Furthermore, ensemble methods are capable of handling reoccurring drift by reusing old base learners. In DriftVis, we employ the method proposed by Kolter et al. [37] due to its competitive performance and flexibility.

2.2 Visual Performance Analysis

Existing efforts in visual performance analysis of machine learning models can be classified into two categories [12, 43]: 1) understanding the model performance; 2) debugging and improving models.

Understanding model performance. There are a variety of visualization tools that have been developed to support analysts in exploring the performance of machine learning models. Tzeng et al. [66] developed a visualization tool to support the performance analysis of artificial neural networks utilizing a graph-based visualization. ModelTracker [2] was developed to convey both the overall and instance-level performance of a binary classifier. Squares [58] extended the work from ModelTracker and was designed to illustrate and compare the prediction score distributions of multiclass classifiers. More recently, there has been a major focus on supporting the understanding of the performance of deep neural networks [12, 27, 74], such as convolutional neural networks [6, 28, 31, 57, 73], recurrent neural networks [50, 64], deep Q-networks [67], and deep generative models [34]. While the majority of work has focused on supporting specific models, there has also been work focusing on developing model-agnostic interactive visualization tools to support model understanding [51, 72, 76]. In general, the goal of many current systems is to support analysts in understanding the performance characteristics of models from multiple perspectives and illustrate which model features contribute to the performance.

Debugging and improving models. Along with tools for under-

standing models, a variety of tools have been developed to support debugging and improving machine learning models. These efforts focus on exploring the reasons why a learning process does not work as expected and supporting analysts in refining the model. Liu et al. [43] proposed a directed-acyclic-graph-based visualization method to illustrate how data flows through a network in the training process and thus facilitates the diagnosis of a failed training process. DeepEyes [56] was developed to help analysts identify the potential issues that deep neural networks may experience in the training process, such as redundant filters and inadequately captured information. Other research has explored model diagnosis of a variety of deep neural networks, such as deep generative models [42, 68], recurrent neural networks [39], and deep sequence models [52, 63].

One current drawback with model debugging and performance improvement is the lack of generalizability. Models (or classes of models) often have specific parameters and features that are not completely generalizable. As a result, most existing methods of visual model performance analysis are model-dependent. However, concept drift is a generalizable problem that focuses on understanding how data has shifted over time. Our system is designed around understanding shifts in streaming data and exploring distributions, making our proposed system model-independent. This is similar to current work that focuses on analyzing the robustness of learning models in terms of data quality, such as understanding model vulnerabilities to adversarial examples [9, 41, 46] and detecting out-of-distribution samples [10]. These methods also compare training data with test data, identify outliers, and then analyze their effects on the model performance. While inspired by the same motivation of enhancing the training dataset, our work differs from these methods in two ways. First, instead of considering only one static test dataset, we handle a stream of test data, where new test data keeps on coming in and then is appended to the existing data. Second, we measure the concept drift using a distribution-based method, which models the streaming data distribution by an incremental GMM. To illustrate the drift over time and the root cause of such drifts, a streaming scatterplot is developed, which combines the advantages of the constrained t-SNE and a density map.

3 REQUIREMENT ANALYSIS

This work is driven by previous research on concept drift and our discussions with a researcher from the China Meteorological Administration (E_1) and three machine learning practitioners (E_2, E_3, and E_4) regarding their needs for model deployment in real-world applications. E_1 works on weather predictions. E_2 focuses on predicting whether a computer motherboard meets the quality standards for a computer manufacturing company, Inventec. E_3 and E_4 are machine learning researchers and are interested in classifying conference papers to facilitate their research, and E_4 is a co-author of this paper. A central problem each of these experts faces after deploying a machine learning model is how to monitor the model performance over time and decide when the model needs to be updated if an unsatisfactory performance is identified.

To understand the experts' needs, we engaged in weekly free-form discussions with our experts for two months to learn about the systems they were currently using for model performance analysis, and what challenges they faced when using these systems. In our discussions, the experts commented that there are already several cloud-based model performance monitoring services. Two widely-known services are Google's continuous evaluation service [24] and the data drift detection service on Microsoft Azure [49]. These services can alert analysts when drift occurs. However, they do not support the exploration of where the data distribution has changed. This requires the analyst to spend more time investigating which subset of the most recent data could benefit the prediction model the most and then labeling the data before retraining the model. Our experts noted that if the system could help identify where the distri-

bution changes occurred, the amount of labeling could be reduced. This is favored by the experts because the cost of acquiring a large number of data labels is usually high.

Based on the literature review, the observed limitations of current drift detection methods, and the discussions with experts, we summarized the requirements for the visual analytics system as follows:

R1. Detecting when concept drift happens. Concept drift is one of the major causes of model accuracy degradation over time. To tackle this issue, many methods have been proposed to track and detect when drift occurs in streaming data [21, 45, 62]. All the experts expressed that they required a convenient way to alert them to the potential occurrences of concept drift during the running process of their deployed machine learning models. For example, E_2 once found that his quality control algorithm wrongly judged some qualified motherboards to be unqualified because a new batch of motherboards used in the assembly line did not have the same board height as before. Thus, it is critical to detect potential concept drifts in streaming data and inform the analyst to minimize the decrease in accuracy and quickly recover from the drift process.

R2. Analyzing where and why drift happens. All the experts expressed a strong need for analyzing where and why drift occurs after it is detected and understanding how the detected drift will influence the model performance. Thus, a simple numerical measure of drift degree is not sufficient. For example, experts usually want to know how each feature has changed over time. E_3 commented, "We know which features are important to the model, and if there are significant changes in these features, the drift is likely to affect the model performance." Other useful information to explore includes the distribution of the data and their changes over time. The experts wanted to quickly locate the regions dominated by new data or the regions where the data distributions have changed. In these cases, the model performance is likely to have a drop. In addition to exploring where drift happens (e.g., which feature and where it is in the data distribution), the analysis of why the drift happens is also important. E_4 said, "Knowing why the drift occurs really helps to take necessary actions - sometimes the action may not even be required on the model side. For example, several months ago, I found that the model performance degradation was simply caused by the changed position of the camera. It was moved to another angle, resulting in the incoming video not being the same as the training dataset." The need to analyze where and why is also well aligned with previous research on concept drift analysis [40, 45, 69].

R3. Overcoming drift and improving performance. Having an understanding of the drift is not the end of the analysis. Experts needed to appropriately adapt the model to the new data and improve the performance, which is called drift adaptation. The most straightforward adaptation approach is to retrain the model with additional labeled streaming data. However, the experts were not in favor of this method. E_3 commented, "For us, a lighter-weight method is more practical. It is inefficient to retrain the whole model as it often takes hundreds of GPU hours." E_2 also desired to shorten the time of drift adaptation because the quality control of the computer motherboard assembly line is crucial and would have a large influence on the brand. Fast reactions to any detected issues on the computer motherboard assembly line can reduce company losses. E_1 further pointed out that there were sometimes reoccurring patterns in the weather data, such as revolving seasons and day-night alternations. Thus, it is desirable to have a drift adaptation method to reuse part of the old model. This is also reflected in the current research trend that "adaptive models and ensemble techniques have played an increasingly important role in recent concept drift adaptation developments [45]."

4 DESIGN OF DRIFTVIS

DriftVis is designed to detect, analyze, and overcome concept drift. It consists of three modules: drift detection, drift visualization, and drift adaptation (Fig. 2).
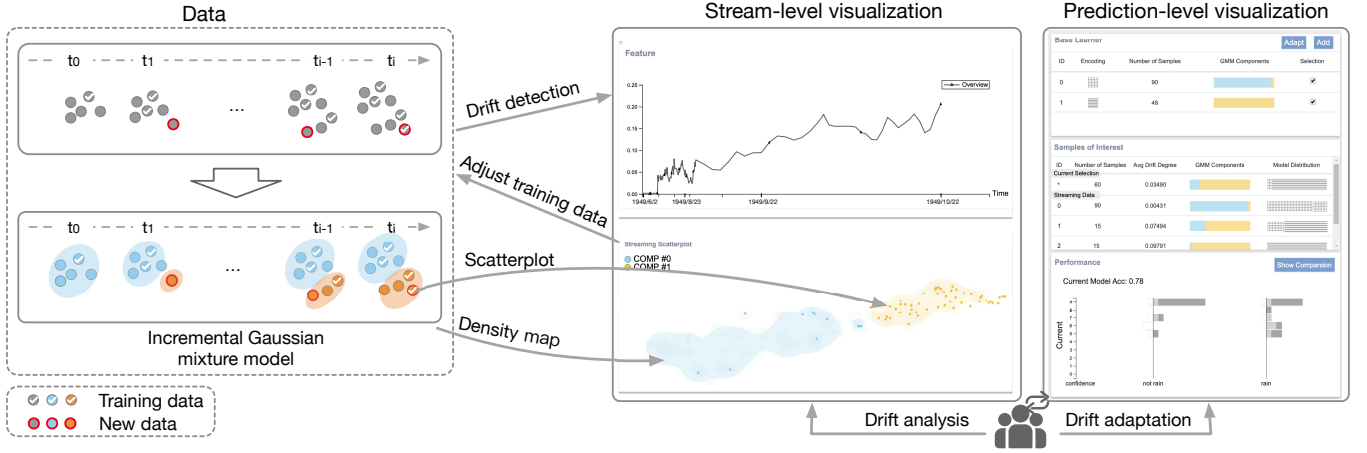
Figure 2: DriftVis overview. When data streams in, an incremental Gaussian mixture model is first employed to model the data, and the output is further used in drift detection and data visualization. The stream-level visualization displays the drift degree and the time the distribution changes to support drift analysis. The prediction-level visualization facilitates the understanding of the model performance for drift adaptation. The training data in the adapted prediction model will be used for drift detection in the next iteration.

The **drift detection** module compares the training dataset with the latest streaming data and calculates the drift degree over time (**R1**). An incremental GMM is employed to model the data distribution, and a distribution-based drift detection method is developed to calculate the drift degree. Detecting drift in streaming data serves as the basis for the subsequent analysis, including discovering the root cause of the detected drift and adapting the model to compensate for it.

The **drift visualization** module provides an exploratory analysis of when, where, and why drift happens (**R1**, **R2**). Such analysis is primarily supported by a streaming scatterplot that shows the data distribution over time. Combined with the drift degree line chart and a feature selection list, analysts can make informed decisions on how to adapt the model to overcome the drift.

The **drift adaptation** module adapts the learning model to the unforeseen change in streaming data (**R3**). In DriftVis, we employ the ensemble method developed by Kolter et al. [37] due to its competitive performance and flexibility. In particular, we train a set of base learners on different subsets of the historical data. These base learners are combined with different weights to adapt to the new data and improve performance.

Our contributions mainly focus on the first two modules, so we introduce them in detail as follows.

4.1 Drift Detection

One major source of concept drift is the unexpected changes in the intrinsic distribution of the data stream [30, 45]. Thus, a key challenge to compute the drift degree is how to measure the difference between data distributions. **Energy distance** is motivated by the potential energy between objects in a gravitational space [59]. The distance is zero if and only if the two distributions are identical. The major advantage of energy distance is that its calculation is linear to the number of dimensions and can be easily scaled to high dimensions [22]. As a result, we employ energy distance to measure the drift degree. For two groups of data samples $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, the energy distance of their underlying distributions is defined as:

$$d(X, Y) = (2A - B - C)/2A, \quad (1)$$

where $A = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} \|x_i - y_j\|$ is the average of the pairwise distance between two groups of samples, $B = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|x_i - x_j\|$ and $C = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \|y_i - y_j\|$ are the averages of pairwise distance within X and Y, respectively. Here $\|x\|$ is the L2-norm of vector x.

In DriftVis, the drift degree at time t is measured by the energy distance between the training set $\hat{X}$ and the streaming data X_t at time t. In practice, data samples may come one at a time, such as daily weather data. In that case, X_t is set as the streaming data in a sliding window ending at time t to reduce noise and get a more accurate drift degree. In the development of DriftVis, we found that directly calculating the drift degree between $\hat{X}$ and X_t may overestimate the drift. This typically occurs when $\hat{X}$ contains several clusters, while X_t only contains a subset of these clusters. For example, $\hat{X}$ may contain the weather records for a whole year, but X_t only contains one month of data. In this example, the calculated distance is large because X_t lacks the samples from other months. However, the model performance drop is not large since the distribution of $\hat{X}$ already covers that of X_t. As a consequence, directly calculating the distribution distance can potentially generate false alarms.

To tackle the issue of drift over-estimation, it is desirable to compare X_t with only the training samples that are similar to them. The key challenge is, for each newly arrived data sample, how to select the similar training samples, and how many samples are enough. A straightforward solution is to use the k-NN algorithm to select the top-k similar training samples. Although this method is able to reduce the drift over-estimation to some extent, experts have to tune the parameter k for different datasets and even for different samples to achieve good results, which is impractical in applications. To solve this problem, we let the data speak for itself, i.e., automatically determining the similar samples based on the data sample clusters in the dataset. Compared with k-NN, the number of clusters of a clustering algorithm can often be automatically determined with unsupervised model selection methods, such as Akaike information criterion (AIC) [1] and Bayesian Information Criterion (BIC) [61]. The intuition behind this solution is that in the same cluster of the dataset, the samples are similar. Thus, at each time t, we first incrementally cluster the data samples X_t at time t based on the clustering result before t. Then, based on the new clustering result, for each sample x_t^i in X_t, we select the training samples that are in the same cluster of x_t^i as its similar samples.

Due to streaming nature of the data, we employ an incremental clustering method in DriftVis, i.e., the incremental GMM algorithm [18]. A GMM-based clustering method is utilized because GMM is able to approximate almost any continuous distribution where the value of each feature is continuous [7]. The incremental GMM algorithm can be split into an offline clustering step for the training dataset $\hat{X}$ and an online clustering step that ingests data samples X_t for each time point t:
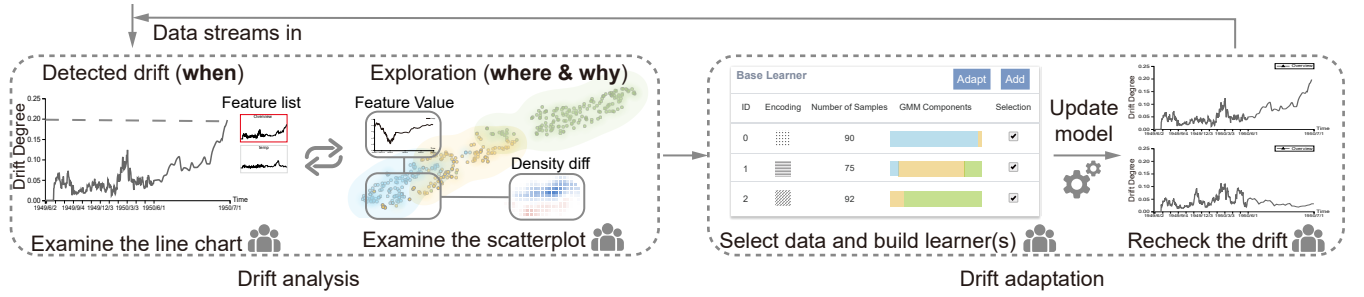
Figure 3: DriftVis analysis workflow. When drift is detected, analysts can explore when, where, and why it happens with the help of stream-level visualization. To adapt to the drift, analysts can select data samples to build learner(s) and then update the model.

Offline Clustering. A GMM is built to cluster the data samples in $\hat{X}$, where each cluster is a Gaussian component $N(\cdot \mid \mu, \Sigma)$ with mean μ and covariance Σ. The number of clusters is determined by the Bayesian Information Criterion (BIC) [61].

Online Clustering. The major purpose of this step is to incrementally cluster the data samples in X_t. For each data sample x_t^i in X_t, we need to decide which existing cluster it belongs to or if a new cluster should be created when it is not close to any existing cluster. If the probability of this sample belonging to an existing Gaussian component $p = N(x_t^i | \mu, \Sigma)$ is higher than a threshold (0.95 in DriftVis), we assign the data sample to the cluster with the maximum probability and update the mean and the covariance of the corresponding Gaussian component [18]. Otherwise, there should existbe a new Gaussian component whichthat better fits the current data sample. Constructing a Gaussian component from a small number of samples cannot accurately estimate the component's mean and covariance, and will bring instability to the clustering result. To solve this problem, we temporarily assign these samples to the Gaussian components with the maximum probability. When enough such samples are identified, we build a new GMM using these samples and then merge the new components into the current clustering results. In DriftVis, to get enough samples and to avoid creating noisy components, we set the threshold to be half of the average component size or we allow the analyst to set a specific threshold value. Based on the clustering result at t, we calculate the drift degree D_t between X_t and $\hat{X}$ as a weighted sum of the drift degree for each cluster:

$$D_t = \sum_i \frac{|X_t^i|}{|X_t|} d(X_t^i, \hat{X}^i), \tag{2}$$

where $|\cdot|$ is the number of samples in a set and $d(\cdot)$ is the energy distance. $\hat{X}^i$ and X_t^i are the samples of the i-th cluster in $\hat{X}$ and X_t, respectively. For a cluster that satisfies $|\hat{X}^i| = 0$, we set the drift $d(X_t^i, \hat{X}^i) = 1$ because the data samples in X_t^i are unlike other samples in the training dataset and will probably cause model performance drop. In this case, there should be a larger drift degree.

4.2 Drift Visualization

To support the analysis of when, where, and why drift happens (**R1**, **R2**), DriftVis employs two visualization components, a stream-level visualization (Fig. 1(a)) and a prediction-level visualization (Fig. 1(b)). The stream-level visualization discloses the distribution changes of the data stream over time. It supports the exploration purely of the data, including the distribution of data samples, the drift detection results, and some of the raw data contents (e.g., the raw text before feature extraction). The prediction-level visualization focuses on the performance-related information of the prediction model and reveals how the adaptation to the drift affects the model performance (**R3**). The drift detection and adaptation methods support the two visualization components and serve as the foundation for the concept drift analytics environment.

Fig. 3 illustrates the typical analysis workflow as data streams in. The workflow begins with an analyst checking the drift degree line chart to detect any obvious drift increase ("when"). In cooperation with the "when" analysis, the analyst can further analyze "where" and "why" the drift happens on the streaming scatterplot, which illustrates if the incoming data forms new components, deviates from the existing components or causes density increase/decrease in some regions. After confirming a drift, the analyst can select data samples on the scatterplot to build new base learners. Using the available base learners, the analyst updates the ensemble and checks if the model performance has improved using the prediction-level views. This workflow can be applied iteratively as new data streams in.

4.2.1 Stream-Level Visualization

The stream-level visualization is designed to illustrate and explain concept drift - specifically, the calculated drift degree and the density change. It consists of a drift degree line chart (Fig. 1A), a feature selection list (Fig. 1B), and a streaming scatterplot (Fig. 1C). The drift degree line chart is used to alert the analyst of possible occurrences of concept drift by displaying the calculated drift degree over time (**R1**). The line chart coordinates with the feature selection list to turn on and off the drift degree lines calculated on different features. When any single feature is selected, the y-axis can switch between the drift degree and the feature value. We used the line chart because it is the most common form of time series visualization [26, 43].

As described in requirement **R2**, it is critical to know where and why drift occurs by exploring the change of data distribution over time. We use a GMM-based constrained t-SNE to continuously project the streaming data onto a 2D space and have designed two visualization modes, scatterplot and density diff, to present the data for distribution-based analysis.

GMM-based constrained t-SNE. The GMM-based constrained t-SNE is designed to 1) preserve the sample similarities in the original high dimensional space, 2) explain the distribution of each Gaussian component, and 3) minimize the movements of previous data points in the t-SNE plot when new data streams in. The state-of-the-art solution to achieve the aforementioned goals is the supervised t-SNE developed by Choo et al. [11], where samples in the same class are grouped together by reducing their distances using a shrink factor. Although shrinking the projection within each group can reduce the visual clutter between groups and improve readability, it does not consider the dispersion of each group and could also not maintain stability when the projection updates with additional data. To overcome these limitations and well convey the distribution patterns of Gaussian components, we modify the shrink factor with respect to the dispersion of each component and add two constraints to improve the stability of the shape of Gaussian components and data positions.

As we assume that the ground-truth labels are unavailable when calculating the projection, the shrink factor uses the component label inferred from the incremental GMM. Its value is weighted by the component dispersion that highly dispersed components will shrink more. In DriftVis, we use entropy to measure dispersion. For each Gaussian component C_k whose covariance matrix is Σ_k, the shrink

factor α_k is determined as follows:

$$\alpha_k = \alpha \cdot \left(1 - \beta \cdot \frac{H[C_k]}{\max_{k'}\{H[C_{k'}]\}}\right). \quad (3)$$

Here α is the maximum allowed shrinking factor, β is a weighting factor, and $H[C_k]$ is the entropy of the Gaussian distribution of component C_k. In practice, to ensure that the calculated shrink factor does not exceed the maximum, α, we enforce $H[C_k]$ to be no smaller than a small positive ε. Thus, $H[C_k] = \max(\frac{1}{2}\ln|\Sigma_k| + \frac{D}{2}(1 + \ln 2\pi), \varepsilon)$, and D is the size of the feature dimension. For the most dispersed component, $\alpha_k = \alpha \cdot (1-\beta)$, and for the least dispersed component $\alpha_k \approx \alpha$. Adding this factor to the distance calculation yields,

$$\text{dist}(x_i, x_j) = \begin{cases} \alpha_k \cdot \|x_i - x_j\| & x_i, x_j \in C_k \\ \|x_i - x_j\| & \text{otherwise} \end{cases}. \quad (4)$$

To further improve the stability of the projection results, we add two constraints into the supervised t-SNE:

The first is a shape constraint that maintains the shape of each Gaussian component to be stable between adjacent time points. Suppose we have n samples of original data and m new samples from the streaming data. The n original samples form k original Gaussian components. As the shape is hard to measure, we approximately maintain the relative distances between the original n data samples and the component centers. This is achieved by minimizing the KL-divergence between P_c and $Q_c \in \mathbb{R}^{n \times k}$, which represent the joint probability distributions of the similarities between the n original samples and the k center constraint points in the high dimension and low dimension, respectively. Each center constraint point is taken as the gravity center of the Gaussian distribution in the high dimensional space and its corresponding low dimensional projection in the previous iteration. Following t-SNE, the distributions P_c and Q_c are defined as:

$$\begin{aligned} (P_c)_{ik} &= w_k \cdot \exp(-\text{dist}(x_i, x_k)/2\sigma^2) \\ (Q_c)_{ik} &= (1 + \|y_i - y_k\|^2)^{-1} \end{aligned}. \quad (5)$$

Here, w_k is the weight for the kth center constraint and is set to $\sqrt{|C_k|}$, where $|C_k|$ is the sample number in the kth Gaussian component. x_i is the high dimensional vector of the original instance, and x_k is the high dimensional center of the Gaussian component C_k, and y_i and y_k are their projected low dimensional coordinates. $\text{dist}(x_i, x_k)$ uses Eq. (4).

The second constraint is to maintain the positions of the original samples. To improve the run-time efficiency, when sample size n is greater than 500, we use blue noise sampling [5] to select n' (500) samples from n original samples and introduce a virtual constraint point for each. $P_s, Q_s \in \mathbb{R}^{n \times n'}$ represent the joint probability distributions of the similarities between the n original samples and the n' selected samples. When sample x_i is used as the kth virtual constraint point, $(P_s)_{ik} = 1$, and this will restrict the change of the projected coordinates of the n' selected samples and maintain the position stability of the overall projection.

Combining the above two constraints and the supervised t-SNE, we obtain the projection by minimizing the objective function:

$$\lambda \cdot KL(P\|Q) + \varphi \cdot KL(P_c\|Q_c) + (1 - \lambda - \varphi) \cdot KL(P_s\|Q_s). \quad (6)$$

Here $KL(\cdot\|\cdot)$ is the KL-divergence between two distributions. The first term is the original optimization goal of supervised t-SNE that minimizes the difference between the high-dimensional distribution P and the 2-D distribution Q of all the $m+n$ samples. The second and third terms are the shape and the position constraints, respectively. **Scatterplot and density diff.** To visually explain the distribution and the drift, the streaming scatterplot utilizes two modes, scatterplot

and density diff. *Scatterplot mode* uses a density representation to display the historical data distribution and scattered points to represent streaming data. All data samples are projected using the GMM-based constraint t-SNE projection but rendered differently. Historical data is rendered using a density map where a lighter color represents a lower density. On top of the density map, the streaming data is displayed as scattered points. Categorical colors are used to separate the Gaussian components. Fig. 1C shows an example of the scatterplot mode for analyzing the papers from NeurIPS.

To illustrate the change of data distribution, we utilize a two-stage animation to exhibit the appearance of incoming data samples. The first stage updates the projection of historical data and the density map. Historical data points move from their original positions to their updated ones as the new streaming data is used in the projection calculation. As this movement occurs, a new density map will be constructed using the updated projection of the historical data points. The second stage draws the new streaming data where data points are grouped based on their Gaussian component assignment and displayed by group.

The *density diff mode* is used to illustrate the density change of data distribution by visualizing the density difference between two datasets, which can be consecutive data batches or two batches selected by the analyst. To gauge the density difference, this view uses grids to represent the 2D space and calculates the percentage of data points falling into each grid in the latest streaming data. For the grid on the ith row and j^{th} column with n data points, its normalized density is $grid_{i,j}^t = n/N$ where N is the number of data points at time t. To produce a more legible visualization, the grid normalized density is then smoothed by a halo effect [54] where each grid is extended into a 5x5 grids, and we assign 30% density to the peripheral halo area. A positive value means the grid has a higher density in the latest streaming data and is colored red. A negative value means the grid has a lower density in the latest streaming data and is colored in blue. The darker the color, the higher the density difference. Fig 6 shows an example of the density diff view.

Justification. We considered using either a flow map or a sequence of screenshots to show the distribution change over time. However, we found that the flow map of the data distribution cannot support a detailed distribution analysis at a particular timestamp, and the screenshot sequence makes it hard to identify the distribution change between screenshots [23]. Therefore, we adopted the streaming scatterplot because it can reveal the detailed distribution and support comparison [44]. Analysts can switch between the *scatterplot mode* and *density diff mode*. We adopted this switching design because superposition can cause visual clutter, and juxtaposition would halve the view size and make it harder to investigate the distribution.

4.2.2 Prediction-Level Visualization

The prediction-level visualization consists of three views, a base learner view (Fig. 1D), a samples of interest view (Fig. 1E), and a performance view (Fig. 1F).

The *base learner view* lists all available base learners that can be used in the ensemble model. For each base learner, it shows the size of the training set and the proportion of the data points in the training set that belong to each Gaussian component. This proportion is represented by a colored bar using the same color scheme in the streaming scatterplot. Analysts can select a data subset to create a new base learner and choose which base learners to use in the ensemble model.

The *samples of interest view* is used to mark a subset of data as samples of interest, which can be revisited afterward. Similar to the base learner view, the colored bars show the percentage of Gaussian component labels in every batch. To see the effect of each base learner on the samples, we calculate a model distribution and visualize it using the combination of each base learner's glyph pattern. The model distribution is a summary of the importance of the base

learners in making predictions for this dataset. The importance of each base learner is gauged by the sum of the weights every training sample takes from the base learner. From the model distribution, the analyst can determine which base learner is important. We chose the table form here because it is simple and easy to understand [53].

The *performance view* visualizes the model performance before and after adaptation to verify the effectiveness of drift adaptation. This view would only present on the data with labels. To enable the exploration of prediction accuracy and visualize the direct effect of the adaptation, we utilize the design from Squares [58] in our performance view, as shown in Fig. 1F. Squares is able to provide a compact and effective comparison between classifiers before and after adaptation. It shows each class in a column, which contains a vertical axis annotated by its class label. The left-most vertical axis indicates the prediction confidence ranges. Summary statistics (true positive, false positive, and false negative) for each class are encoded in stacked bars along the corresponding axis. The length of the bar represents the percentage of data samples predicted with the corresponding confidence score. Dark grey indicates true positive, light grey indicates false positive, and no-fill bars on the left of the axis indicates false negative. To see if the prediction performance has improved, the performance view can change into comparison mode to show the performance statistics of the latest model and the model prior to this adaptation.

4.2.3 Coordinated Interactions

DriftVis provides coordinated interactions among different views for drift analysis. When the analyst hovers over a data sample or selects a group of samples, the line chart will highlight the time (on the x-axis) when these samples occur. The analyst can also brush the line chart to see the brushed samples highlighted on the scatterplot. The streaming scatterplot itself also possesses a rich set of interactions for data exploration. In order to investigate data details, such as the feature values and raw data, the streaming scatterplot uses pop-up windows to display the information of hovered or selected data samples. Lasso selection is supported to display a summary of the selected samples. The streaming scatterplot also allows the analyst to hide Gaussian components when it is too crowded.

To decide which base learner to include in the ensemble, it is essential to know what data is used to train the base learner. Hovering over a base learner highlights its training samples on the scatterplot, and if the analyst clicks on the base learner, these samples will be shown as selectable points so that they can be reused to train another base learner or be compared exclusively to the latest streaming data.

Sometimes, an undesirable incremental GMM result may cause a false alarm on drift detection because some data are not compared with the closest distribution; therefore, the drift degree is overestimated. To reduce such false alarms, our system supports analysts to modify the incremental GMM result. Analysts can select some data in the scatterplot and click "Merge Component" to confirm that these points should belong to one component. The belonging relationship of other data samples will be incrementally updated.

5 EVALUATION

In this section, we conducted a quantitative experiment to evaluate the effectiveness of our drift detection method and then presented two case studies to demonstrate how DriftVis supports the identification, analysis, and correction of concept drift.

5.1 Quantitative Evaluation on Drift Detection

Experimental setting. We employed the widely-used experimental setting in drift detection methods [13, 25]. Two synthetic datasets, D_1 and D_2, were used. Each dataset consists of 5 million data points, which follow 2-D Gaussian distributions. There are 99 simulated concept drifts in each dataset. The drifts were simulated by changing the mean and variance of the Gaussian distributions in D_1 and D_2,

Table 1: Comparison of drift detection results.

Dataset	Method	Detected	Late	Missed	False
D_1	ITA [13]	75.5	12.3	11.2	4.5
	LDD-DIS [40]	56.5	19.3	23.2	3.9
	TVD [71]	75.5	19.9	3.6	7.0
	Ours	**97.9**	**0.9**	**0.2**	**0.8**
D_2	ITA [13]	75.7	18.0	**5.3**	7.6
	LDD-DIS [40]	44.0	22.2	32.8	12.2
	TVD [71]	71.2	22.3	5.5	**6.0**
	Ours	**77.5**	**14.9**	6.6	6.4

respectively. We compared our method with three baseline methods: ITA [13], LDD-DIS [40], and TVD [71]. In our experiment, we used a grid search to obtain the best hyper-parameters.

Criteria and results. We classified the detected drifts into four categories by comparing them with the ground-truth. When a drift happens at t and is reported at $t + \Delta t$, the drift is considered to have been successfully *detected* if $\Delta t < w$, *late* if $\Delta t \geq w$ but is reported before next drift happens, and *missed* if it is not reported before the next drift happens. Here, w is the window size of the respective method. A *false* alarm happens if a drift is reported again before the next drift occurs. In practice, we want to maximize the number of *detected* drifts and minimize the other three categories. Table 1 shows the number of drifts in each category for each method. To reduce randomness, we reported the numbers averaged over 10 runs. Compared with the baselines, our method has detected the largest number of *detected* drifts in both datasets, demonstrating an improvement over existing drift detection methods. However, we noticed that, compared with D_1, our method performed worse in D_2. The potential reason is that the energy distance mainly focuses on the mean (D_1) of the distribution instead of the variance (D_2).

5.2 Case Study

5.2.1 Tabular Data: Weather Prediction

Dataset. This dataset is compiled by the U.S. National Oceanic and Atmospheric Administration (NOAA). We use a preprocessed subset that has been used in previous concept drift detection research [15, 17]. It contains 18,159 daily weather records from the Offutt Air Force Base from 1949 to 1999. Each weather record contains eight measurements ("Temperature," "Dew Point," "Sea Level Pressure," "Visibility," "Average Wind Speed," "Maximum Sustained Wind Speed," "Maximum Temperature," and "Minimum Temperature") and one class label ("precipitation"). In our analysis, we predict whether rain precipitation was observed on each day. For the prediction model, we use logistic regression implemented in Scikit-learn 0.21.3. We randomly split the dataset into 70% training and 30% test data. The training/test split is used to ensure the plausibility of the reported accuracy.

Drift analysis. We conducted this case study together with E_1, our expert from the China Meteorological Administration. She works on weather prediction model development and making weather prediction decisions. In this case, we first illustrate how DriftVis helped identify changes in the weather data distribution across seasons, and then demonstrate how it facilitated the expert in identifying abnormal weather changes long-term. This study will show that a high drift degree could be detected when the data sees a new season, but DriftVis could also find small drifts before the drift degree reaches an alert level.

Seasonal drifts in the first year. The initial model, BL_0, was trained on data from the summer of 1949 (June 1 to August 31). The remainder of the data was used to simulate a streaming context, where data is reported daily. When E_1 saw a drift, she stepped in to analyze. The drift degree first went up to 0.2 after 15 days, on Sept. 15, and the expert decided to explore the data distribution change. In Fig. 4(a), she found that these new data were on the boundary of

(a) Sept. 1 to Sept. 15.   (b) Oct. 1 to Oct. 23.   (c) Oct. 24 to Nov. 2.
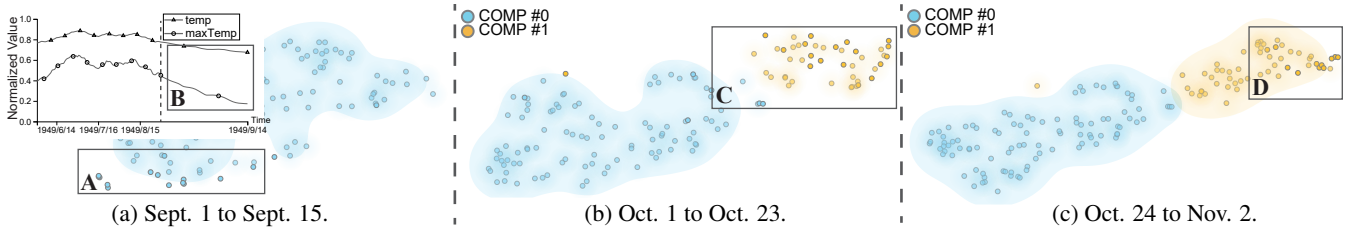
Figure 4: The streaming scatterplot shows the density change from summer to early autumn in 1949: (a) the new data (A) came with a lower temperature (B); (b) a new component (C) was detected; (c) the following data (D) came and deviated from the yellow component.

the existing component with relatively lower temperature, as shown in Fig. 4A and Fig. 4B. She selected the new data and some similar data points from the historical data to build a new base learner, BL_1, and combined BL_1 with BL_0 to create a new ensemble prediction model. After the adaptation, the drift went down, indicating that the adaptation was successful.

Since Oct. 1, there were some data samples deviating from the blue component, and on Oct. 23, the expert identified a new Gaussian component (yellow, as shown in Fig. 4C) in the streaming scatterplot (Fig. 4(b)). The expert explained, "This is the first month of autumn, and the temperatures have begun to decrease. This is why a new component has emerged." However, she commented that the drift degree was still low, and many points in the yellow component were flipped from the blue component. Therefore, she decided to monitor the change closely for the following days. As expected, the next few days' data all fell into the yellow component, and the expert noticed an increase in drift degree, which went up to 0.2 on Nov. 2. Fig. 4(c) shows the corresponding streaming scatterplot and Fig. 4D highlights where the latest streaming data appeared. E_1 created a base learner (BL_2) using the yellow component data and removed BL_1. She explained, "This component does not have a full collection of autumn data yet, but in our field, we always want to adapt the model to create a better prediction as soon as we have enough data. While we know that when more autumn data comes in, the data pattern learned using this subset may not be robust, further adaptation can always be made." Just as the expert explained, the drift degree stayed low for the next few days because the model learned some patterns from the yellow component, which encapsulates many properties of the incoming data.

On Dec. 11, a green component (Fig. 5A) appeared, and on the same day, the drift degree increased rapidly to 0.15. Similar to how she handled the occurrence of the yellow component, the expert first built a new base learner (BL_3) with available data in early winter and replaced it with BL_4 once the green component had more data and the drift degree increased. As of Jan. 8, 1950, the ensemble model had three base learners, BL_0, BL_2, and BL_4.

In late January, the weather became very cold, and a fourth Gaussian component (purple, Fig. 5B) appeared on Jan. 31, as shown in Fig. 5(b). At this time, the drift degree was not high, and E_1 commented that these data in purple shared a similar pattern to the green component, e.g., low temperature and high sea level pressure, as illustrated in the line chart in Fig. 5C. Based on this observation,

she merged the purple component into the green component and used all of the data in the green component to create a new base learner BL_5 and replaced BL_4.

The drift remained low for the following two months but went up again in early April, when the expert noticed a few days with relatively higher sea level pressure for the time in a year. The drift degree increase was a false alarm caused by a mis-clustering of the incremental GMM, and the expert corrected this by merging these points into the green component. By the end of the first full year, E_1 decided to adjust the model covering the yellow component because she saw density increased in some regions in this component. Comparing the latest streaming data with the training data in BL_2, which was built on the yellow component earlier, Fig. 6 shows that in the area of the yellow component, there were some red grids in the density diff mode. This indicated that the latest streaming data in the yellow component were likely to have a different density distribution than what had been modeled. Specifically, the area with deep red grids became denser. Therefore, using a new base learner BL_6 trained on all data in the yellow component, E_1 replaced BL_2. The current model (M_1) now includes base learner BL_0, BL_6, and BL_5, and the model accuracy running over the past year is 0.77.

Occasional drifts after having four seasons. When it came to the second summer (1950), the drift degree stayed under or around 0.1 until July. E_1 wanted to explore the data at the end of June because she saw some yellow points were projected in the blue density area (Fig. 7). Looking at the feature values of these yellow points, she found that these days had high max temperatures ($\sim 95°F$) and low sea level pressures (< 1006). She commented that these days could be extreme days in the summer when compared to the historical data. It was more reasonable to group them into the blue component, which includes the data of last summer than associating them with the yellow component, which had typically autumn and spring data. She merged these yellow points into the blue component and replaced BL_0 with a new base learner BL_7 using the refined blue component. The drift degree became lower.

After exploring the weather data for more than 14 months, the data had been relatively well covered by the model, and the drift degree stayed lower than 0.1 for the remainder of 1950 and the following year. This model (M_2) comprises BL_8, BL_6, and BL_5. While more weather data streamed in, the expert conducted two more adaptations in 1952 in mid-July and mid-September, respectively, updating base learners and creating a model (M_3) with BL_9, BL_{10}, and BL_5.

Comparing the precipitation prediction accuracy of the three models with their last adaptation in Jun. 1950 (M_1), Sept. 1950

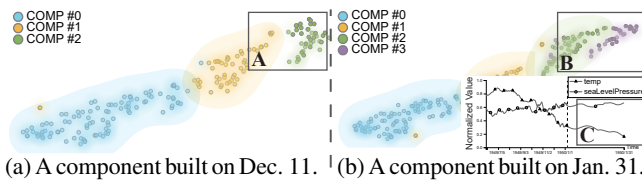(a) A component built on Dec. 11.   (b) A component built on Jan. 31.

Figure 5: The streaming scatterplot shows the distribution change in the winter of 1949: (a) the appearance of the green component (A); (b) the purple component (B) was built, which shared a similar pattern to the green component, e.g., low temperature and high sea level pressure (C).
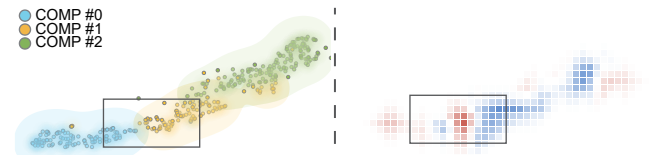

Figure 6: Exploring the density change of the yellow component. The left figure shows the scatterplot on May 31, 1950, and the right figure shows the density diff comparing the latest streaming data to the training data of BL_2 (yellow component).

temp: 85.30
minTemp: 77.00
maxTemp: 95.00
dewPoint: 73.00
avgWind: 11.90
maxWind: 25.10
seaLevelPressure: 998.20
visibility: 14.30

temp: 85.40
minTemp: 78.10
maxTemp: 95.00
dewPoint: 71.90
avgWind: 10.60
maxWind: 18.10
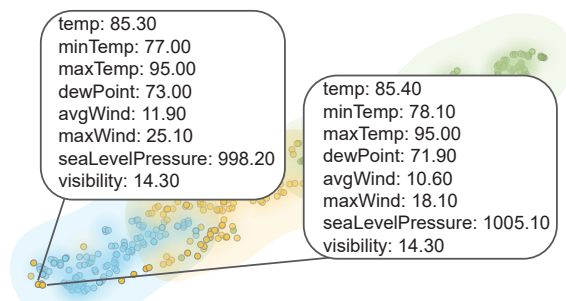seaLevelPressure: 1005.10
visibility: 14.30

Figure 7: The yellow points in the blue region show high max temperature and low sea level pressure and are more similar to summer data (blue component) than autumn data (yellow component).

(M_2), and Sept. 1952 (M_3), we run them on yearly test data from Jun. 1949 to May 1954. M_1 has its yearly accuracy being 0.77, 0.72, 0.73, 0.67, and 0.74; M_2 has its yearly accuracy being 0.78, 0.74, 0.73, 0.69, and 0.76; and M_3 has its yearly accuracy being 0.79, 0.75, 0.76, 0.73, and 0.81. Their prediction accuracy shows that with further adaptations, M_2 performs better than M_1, and M_3 is even more accurate than M_2, especially in the fourth and fifth years. The improvement of the prediction accuracy also indicates that DriftVis is effective in identifying and overcoming drift.

5.2.2 Textual Data: Academic Paper Topic Evolving

Dataset. This dataset consists of 7,926 papers from the top-tier machine learning conference, NeurIPS (from 2000 to 2019). For each paper, we extracted the title, abstract, and introduction and further preprocessed the text by tokenization, lemmatization, keeping only the top-5000-most frequent words in the corpus. Each paper was then converted into a 5,000-dimension vector, and each dimension was reweighted using the TF-IDF scheme [47]. To accelerate the computation of the constrained t-SNE, we further reduced the dimension of each paper to 10 by singular value decomposition (SVD). The reason for using SVD is that the contribution of the words can be interpreted intuitively from each reduced dimension.

In this case, we collaborated with E_3 to explore the prevalent research topics in NeurIPS. To help E_3 gain insights from the past research, we built a multi-class text classifier that predicted the topic of a paper. To get the ground-truth label for each paper, we employed a ten-topic Latent Dirichlet Allocation [8] to calculate the topic distribution of each paper. Each paper's initial ground-truth label was set to the most dominant topic in the topic distribution. We further invited E_4 to verify and fine-tune the ground-truth labels of ambiguous papers with mixed topics. Given the ground-truth labels, we built an SVM-based 10-class classifier to predict the label (topic) of each paper. We randomly split the dataset into 60% training and 40% test data. The training/test split is used to ensure the plausibility of the reported accuracy. We used the training papers in 2000-2003 to build an initial model, whose accuracy was 0.75. The papers after 2003 were assumed to stream in year by year to simulate the common practice of E_3, where he went through the NeurIPS papers after each year's conference to discover emerging topics and trends.

Analysis of papers in 2000-2003. E_3 started the analysis with the papers in 2000-2003 (754 papers). Fig. 8(a) shows the distribution of these papers. Several topics were identified from this distribution by hovering over the papers in the scatterplot to examine the keywords and titles. For example, the keywords in the green component (Fig.8A) were "kernel, vector, feature." E_3 commented that there was a high research interest in the kernel methods at that time due to the success of support vector machines, which was reflected in the leader-board of the MNIST dataset [48]. The orange component (#5) contained two inner clusters (Fig.8B and C). The top one (Fig.8B) focused on neural networks, while the bottom one (Fig.8C) focused on reinforcement learning. E_3 commented that these two inner clusters

might split if their respective topics become popular in the future.

Analysis of papers in 2004-2007. Next, the papers from 2004 to 2007 were added year by year. In the year 2007, the drift degree increased to 0.15 and triggered E_3's detailed analysis. As shown in Fig. 8(b), he found that there were three new components compared with the training data (2000-2003). In particular, component #8 (Fig. 8D) separated from the previously mixed component #5 (Fig. 8B and C) because it now had a sufficient number of papers to form a new component. The pink component (Fig. 8E) talked about convex optimization. The keywords in the brown component (Fig. 8F) were "latent" and "topic." This new component appeared due to the success of Latent Dirichlet Allocation [8], which was published in 2003.

Since there were three new components and a relatively large drift degree of 0.15, the expert decided to build new base learners to adapt to this drift. Using the base learner view (Fig. 1D), the expert selected the papers in these three components and built three base learners with the papers in the components. After merging these three new base learners and the old one into an ensemble model (Sec. 4), the drift degree decreased from 0.15 to 0.07, and the classification accuracy improved from **0.73** to **0.75** in the papers published during 2004-2007.

Analysis of papers in 2012-2015. The drift degree increased gradually from 2012 to 2015. In 2015, the drift degree increased to 0.14, and it prompted E_3 to perform an analysis of this drift. This drift was expected because E_3 knew that these years were the dawn of the deep learning era, and new machine learning approaches were developed. He found that no new component was formed, and there were no deviating data samples. Thus, E_3 switched to the density diff mode to further examine the root cause of the drift. He found that the most noticeable drift happened in the pink component (#6), where the density of three regions increased (Fig. 1G, H, and I). The left-top part of the pink component (Fig. 1G) was about deep neural networks, which was a rising topic at that time. E_3 commented that this new topic was triggered by AlexNet [38] being the winner of the ILSVRC2012 image classification challenge. The bottom part of the pink component (Fig. 1H) was about variational inference. This topic became popular due to the introduction of an efficient deep generative model - Variational Auto-Encoder (VAE) [36] in 2013. The right part of the pink component (Fig. 1I) was still about "sparse," but "tensor" became a new keyword due to the increasing research interest in image data, which can be seen as a 3D tensor. Although there were no new components, the large change of the keywords in the pink component still prompted E_3 to adapt the model. Since there was a noticeable drift within the pink component (#6), he selected the papers from the pink component and built a base learner with these papers. After the adaptation, the drift degree decreased from 0.14 to 0.08, and the performance view (Fig. 1F) showed that the classification accuracy increased from **0.58** to **0.74**. Many papers about optimization (class 5) and sparse matrix (class 8) were now correctly classified.

Analysis of papers in 2016-2019. As interest in machine learning increased, so did the volume of research output, leading to a large degree of drift in the model from the years 2016-2018, with the drift reaching 0.18. After hiding some unchanged components, the expert identified four new components. All four new components were about deep-learning-related topics. Component #9 (Fig. 8H) was about optimization techniques for deep learning, with keywords "gradient" and "sgd." Component #10 (Fig. 8I) was about the generative adversarial network (GAN), which was first proposed in NeurIPS 2014 and quickly received much attention. Components #11 (Fig. 8J) and #12 (Fig. 8K) were about recurrent neural networks (RNN) and convolutional neural networks (CNN), respectively. These two types of deep neural networks showed great success in speech and image processing. To adapt to this large change of the data distribution, E_3 built a learner with the papers in the four deep-learning-related

(a) Exploring papers from 2000 to 2003.

(b) Exploring papers from 2004 to 2007.

(c) Exploring papers from 2008 to 2011.
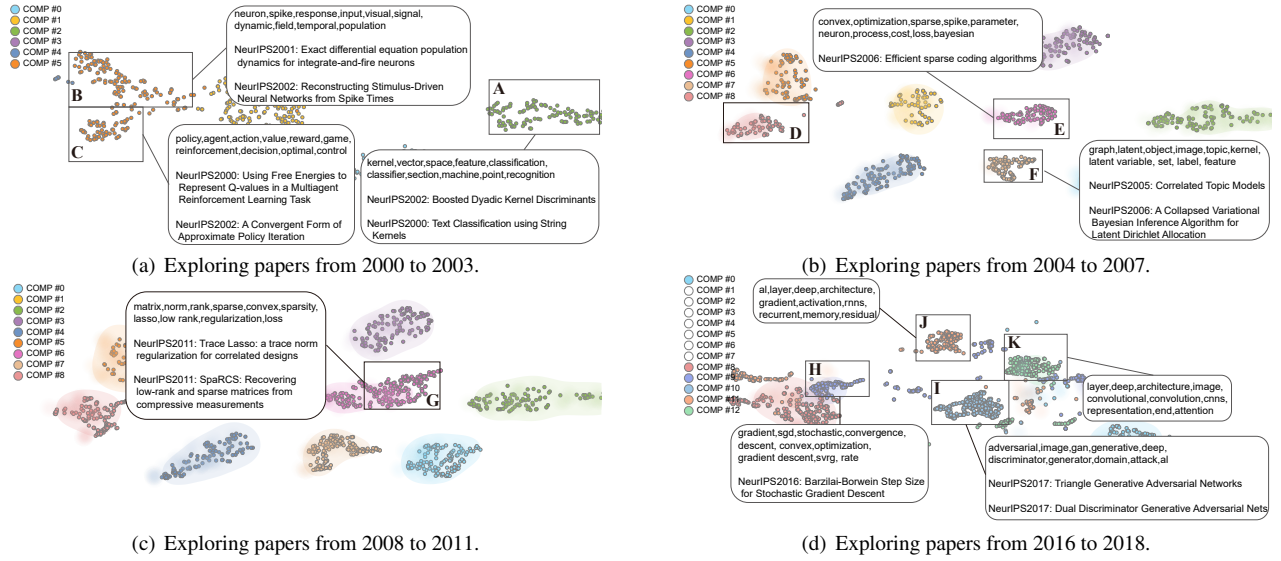
(d) Exploring papers from 2016 to 2018.

Figure 8: The scatterplot shows the Gaussian components in different years when analyzing NeurIPS papers. Each subgraph shows the different analytics steps our expert took. The regions (A)-(K) indicates the interesting components explored.

components and added it to the model. After the adaptation incorporating these two new learners, the classification accuracy improved from **0.59** to **0.73**, and the drift degree decreased to 0.08.

The drift degree remained nearly unchanged in 2019. At the end of the case study, E3 commented that "the trial of the system was really valuable to me not only because it solved the performance issue caused by concept drift, but also enabled me to take a wonderful tour of the past in machine learning."

6 DISCUSSION AND FUTURE WORK

Although the case studies on real-world datasets have demonstrated the usefulness of DrifVis, there are several limitations, which may serve as promising avenues for future research.

Generalization. In our current prototype, DriftVis has only been run on classification tasks and did not factor in known prior knowledge, if any, of the data distribution. The proposed method can be easily generalized into other tasks or to leverage known distributions, but some additional work has to be done. To extend to other tasks, such as regression or clustering, we need to replace the performance view with task-specific designs, e.g., RegressionExplorer [14] for regression tasks. As for the data, currently, we use an incremental GMM to model the distribution of data since it is capable of approximating any continuous distributions [7]. Sometimes, the analyst may have some prior knowledge of the data distribution, such as using a Poisson distribution to better model the accumulated event number, instead of using Gaussian distribution. To incorporate such prior knowledge, we may consider using distribution-dependent distance functions [35] in drift detection.

Drift From Other Causes. In DriftVis, we focus on drift that is caused by changes in the data distribution $P(X)$, where X is the data. However, there can be concept drift due to changes in label $P(Y \mid X)$ (where Y are the labels) while the data distribution $P(X)$ remains unchanged. This is called label drift. For example, in the image classification task, attackers can construct adversarial examples [75] to mislead the classifier. These adversarial examples share a similar distribution with the training data, but their label is different. Such drift can not be detected until the ground-truth labels are obtained. Considering the high cost of acquiring ground-truth labels, our method focuses on the data distribution change, and label drift detection is left for future work. To enable the support of label drift detection in DriftVis, two modifications are needed. First, we can replace the distribution-based concept drift detection approach

with an error-rate-based one, which relies on ground-truth labels. Second, we can change the streaming scatterplot from visualizing $P(X)$ to showing $P(X, Y) = P(X)P(Y \mid X)$, which simultaneously covers the drift caused by $P(X)$ and $P(Y \mid X)$.

Scalability. In the weather prediction case study, we demonstrated the potential of DriftVis to handle streaming data containing thousands of time points. When there are 1,500 historical time points, the system has a latency of about 5s to process the new time point on a PC with Intel i7-9700K CPU (3.60 GHz) and 32 GB memory. In practice, the incoming data may arrive per hour, per minute, or even per second. As a large amount of data samples stream in rapidly, it is challenging to update the incremental GMM and the dynamic t-SNE projection fast enough for every time point, and the potentially high number of Gaussian components can make the scatterplot hard to read. To speed up the calculation, a distribution-based sampling approach [55] can be used. To keep the streaming scatterplot readable, stale or less active components can be hidden until they are activated by a certain number of new data samples. Some incremental visualization techniques can also be employed to accelerate the visualization for large-scale data [19].

7 CONCLUSION

We have developed DriftVis, a visual analytics method to support the detection, examination, and correction of concept drift in streaming data. As part of DriftVis, we have employed an incremental GMM in the distribution-based drift detection to efficiently remove false alarms, which can be automatically addressed by the ensemble model. In addition to improvements in drift detection, DriftVis also introduces the streaming scatterplot visualization that uses a GMM-based constrained t-SNE and two visual modes (scatterplot and density diff) to explain different kinds of drift in the streaming data. DriftVis further combines drift detection, exploration, and adaptation via coordinated interactions. A quantitative experiment and two case studies have demonstrated the usefulness of our method.

REFERENCES

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Springer Series in Statistics*, pages 199–213. Springer, 1998.

[2] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. ModelTracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 337–346, 2015.

[3] S. H. Bach and M. A. Maloof. Paired learners for concept drift. In *Proceedings of the IEEE International Conference on Data Mining*, pages 23–32, 2008.

[4] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno. Early drift detection method. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams*, pages 77–86, 2006.

[5] M. Balzer, T. Schlömer, and O. Deussen. Capacity-constrained point distributions: a variant of lloyd's method. *ACM Transactions on Graphics*, 28(3):1–8, 2009.

[6] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *IEEE Transactions on Visualization and Computer Graphics*, 24(1):152–162, 2018.

[7] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[9] K. Cao, M. Liu, H. Su, J. Wu, J. Zhu, and S. Liu. Analyzing the noise robustness of deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 2020. to be published, doi: 10.1109/TVCG.2020.2969185.

[10] C. Chen, J. Yuan, Y. Lu, Y. Liu, H. Su, S. Yuan, and S. Liu. OoD-Analyzer: Interactive analysis of out-of-distribution samples. *IEEE Transactions on Visualization and Computer Graphics*, 2020. to be published, doi: 10.1109/TVCG.2020.2973258.

[11] J. Choo, C. Lee, C. K. Reddy, and H. Park. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001, 2013.

[12] J. Choo and S. Liu. Visual analytics for explainable deep learning. *IEEE Computer Graphics and Applications*, 38(4):84–92, 2018.

[13] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proceedings of Symposium on the Interface of Statistics, Computing Science, and Applications*, pages 1–24, 2006.

[14] D. Dingen, M. van't Veer, P. Houthuizen, E. H. Mestrom, E. H. Korsten, A. R. Bouwman, and J. Van Wijk. RegressionExplorer: Interactive exploration of logistic regression models with subgroup analysis. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):246–255, 2019.

[15] G. Ditzler and R. Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301, 2012.

[16] D. Dovžan, V. Logar, and I. Škrjanc. Solving the sales prediction problem with fuzzy evolving methods. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 1–8, 2012.

[17] R. Elwell and R. Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[18] P. M. Engel and M. R. Heinen. Incremental learning of multivariate gaussian mixture models. In *Proceedings of the Brazilian Symposium on Artificial Intelligence*, pages 82–91, 2010.

[19] D. Fisher, I. Popov, S. Drucker, and M. Schraefel. Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1673–1682, 2012.

[20] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Proceedings of the Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004.

[21] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.

[22] I. Goldenberg and G. I. Webb. Survey of distance measures for quantifying concept drift and shift in numeric data. *Knowledge and Information Systems*, 60(1):591–615, 2018.

[23] K. Goldsberry and S. Battersby. Issues of change detection in animated choropleth maps. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 44(3):201–215, 2009.

[24] Google. Continuous evaluation of google. https://cloud.google.com/ai-platform/prediction/docs/continuous-evaluation, 2020. Last accessed 2020-07-21.

[25] F. Gu, G. Zhang, J. Lu, and C.-T. Lin. Concept drift detection based on equal density estimation. In *2016 International Joint Conference on Neural Networks*, July 2016.

[26] J. Heer, N. Kong, and M. Agrawala. Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the 27th international conference on Human factors in computing systems*, 2009.

[27] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2674–2693, 2019.

[28] F. Hohman, H. Park, C. Robinson, and D. H. P. Chau. SUMMIT: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1096–1106, 2020.

[29] 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*, pages 97–106, 2001.

[30] A. S. Iwashita and J. P. Papa. An overview on concept drift learning. *IEEE Access*, 7:1532–1547, 2018.

[31] S. Jia, P. Lin, Z. Li, J. Zhang, and S. Liu. Visualizing surrogate decision trees of convolutional neural networks. *Journal of Visualization*, 23(1):141–156, 2019.

[32] L. Jiang, S. Liu, and C. Chen. Recent research advances on interactive machine learning. *Journal of Visualization*, 22(2):401–417, 2019.

[33] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[34] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Viégas, and M. Wattenberg. GAN Lab: Understanding complex deep generative models using interactive visual experimentation. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):310–320, 2019.

[35] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the International Conference on Very Large Data Bases*, pages 180–191, 2004.

[36] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[37] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(Dec):2755–2790, 2007.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[39] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo. RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):299–309, 2019.

[40] A. Liu, Y. Song, G. Zhang, and J. Lu. Regional concept drift detection and density synchronized drift adaptation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2280–2286, 2017.

[41] M. Liu, S. Liu, H. Su, K. Cao, and J. Zhu. Analyzing the noise robustness of deep neural networks. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 60–71, 2018.

[42] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu. Analyzing the training processes of deep generative models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):77–87, 2018.

[43] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017.

[44] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, 2017.

[45] 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*, 31(12):2346–2363, 2018.

[46] Y. Ma, T. Xie, J. Li, and R. Maciejewski. Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1075–1085, 2020.

[47] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[48] Microsoft. Leaderboard of the mnist dataset. `http://yann.lecun.com/exdb/mnist/`, 2020. Last accessed 2020-07-21.

[49] Microsoft. Model monitoring on microsoft azure. `https://docs.microsoft.com/en-us/azure/machine-learning/how-to-monitor-data-drift`, 2020. Last accessed 2020-07-21.

[50] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 13–24, 2017.

[51] Y. Ming, H. Qu, and E. Bertini. RuleMatrix: Visualizing and understanding classifiers with rules. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):342–352, 2019.

[52] Y. Ming, P. Xu, F. Cheng, H. Qu, and L. Ren. ProtoSteer: Steering deep sequence model with prototypes. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):238–248, 2020.

[53] K. Muzammil and S. K. Sarwar. Data and information visualization methods, and interactive mechanisms: A survey. *International Journal of Computer Applications*, 34(1):1–14, 2011.

[54] D. Oelke, H. Janetzko, S. Simon, K. Neuhaus, and D. A. Keim. Visual boosting in pixel-based visualizations. *Computer Graphics Forum*, 30(3):871–880, 2011.

[55] C. R. Palmer and C. Faloutsos. Density biased sampling: An improved method for data mining and clustering. In *Proceedings of the ACM International Conference on Management of Data*, pages 82–92, 2000.

[56] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. DeepEyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):98–108, 2018.

[57] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):101–110, 2017.

[58] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):61–70, 2017.

[59] M. L. Rizzo and G. J. Székely. Energy distance. *Wiley Interdisciplinary Reviews: Computational statistics*, 8(1):27–38, 2016.

[60] D. Sacha, M. Kraus, D. A. Keim, and M. Chen. VIS4ML: An ontology for visual analytics assisted machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):385–395, 2019.

[61] G. Schwarz et al. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[62] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001.

[63] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush. SEQ2SEQ-VIS: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):353–363, 2019.

[64] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*,

24(1):667–676, 2018.

[65] G. J. Székely and M. L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013.

[66] F. Y. Tzeng and K. L. Ma. Opening the black box - data driven visualization of neural networks. In *Proceedings of the IEEE Visualization*, pages 383–390, 2005.

[67] J. Wang, L. Gou, H.-W. Shen, and H. Yang. DQNViz: A visual analytics approach to understand deep q-networks. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):288–298, 2019.

[68] J. Wang, L. Gou, H. Yang, and H.-W. Shen. GANViz: A visual analytics approach to understand the adversarial game. *IEEE Transactions on Visualization and Computer Graphics*, 24(6):1905–1917, 2018.

[69] X. Wang, W. Chen, J. Xia, Z. Chen, D. Xu, X. Wu, M. Xu, and T. Schreck. ConceptExplorer: Visual analysis of concept drifts in multi-source time-series data. In *IEEE Conference on Visual Analytics Science and Technology*. IEEE, 2020.

[70] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

[71] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean. Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, 32(5):1179–1199, 2018.

[72] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2020.

[73] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mane, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):1–12, 2018.

[74] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. *Computational Visual Media*, 7(1), 2021. to be published, doi: 10.1007/s41095-020-0191-7.

[75] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019.

[76] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):364–373, 2019.