DOI: 10.1111/cgf.14842

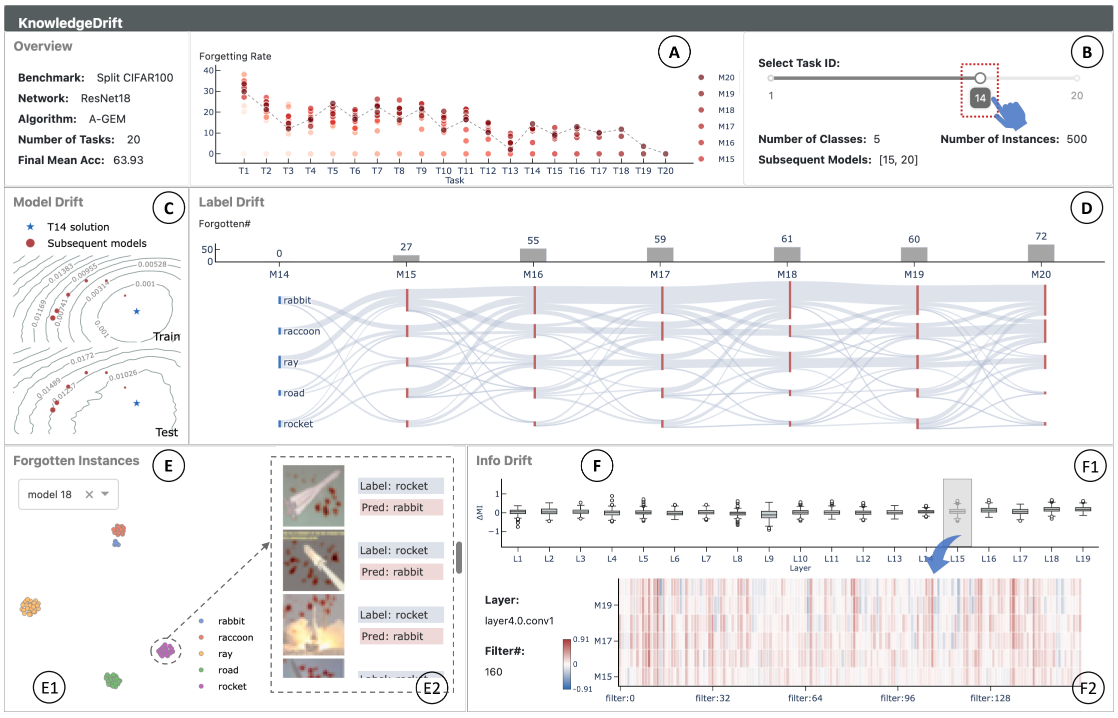
Eurographics Conference on Visualization (EuroVis) 2023
D. Archambault, R. Bujack, and T. Schreck
(Guest Editors)

COMPUTER GRAPHICS *forum*
*Volume 42 (2023), Number 3*

# Visual Analytics on Network Forgetting for Task-Incremental Learning

Ziwei Li, Jiayi Xu, Wei-Lun Chao, and Han-Wei Shen

The Ohio State University


KnowledgeDrift

Overview

Benchmark: Split CIFAR100
Network: ResNet18
Algorithm: A-GEM
Number of Tasks: 20
Final Mean Acc: 63.93

Select Task ID: 14

Number of Classes: 5
Number of Instances: 500
Subsequent Models: [15, 20]

Layer: layer4.0.conv1
Filter#: 160

**Figure 1:** *Interface of* KNOWLEDGEDRIFT. *Analysts first navigate to a particular task by checking the performance overview (A, B). Then, for the selected task, the system first generates the model drift view (C) and label drift view (D), and further provides in-depth analyses to interactively inspect the individual forgotten data (E) and the filter-wise knowledge drift (F).*

**Abstract**

*Task-incremental learning (Task-IL) aims to enable an intelligent agent to continuously accumulate knowledge from new learning tasks without catastrophically forgetting what it has learned in the past. It has drawn increasing attention in recent years, with many algorithms being proposed to mitigate neural network forgetting. However, none of the existing strategies is able to completely eliminate the issues. Moreover, explaining and fully understanding what knowledge and how it is being forgotten during the incremental learning process still remains under-explored. In this paper, we propose KnowledgeDrift, a visual analytics framework, to interpret the network forgetting with three objectives: (1) to identify when the network fails to memorize the past knowledge, (2) to visualize what information has been forgotten, and (3) to diagnose how knowledge attained in the new model interferes with the one learned in the past. Our analytical framework first identifies the occurrence of forgetting by tracking the task performance under the incremental learning process and then provides in-depth inspections of drifted information via various levels of data granularity. KnowledgeDrift allows analysts and model developers to enhance their understanding of network forgetting and compare the performance of different incremental learning algorithms. Three case studies are conducted in the paper to further provide insights and guidance for users to effectively diagnose catastrophic forgetting over time.*

**CCS Concepts**

• ***Computing methodologies*** → *Visual analytics;* • ***Theory of computation*** → *Continual learning;*

## 1. Introduction

Modern machine learning approaches have achieved remarkable performance in a variety of applications [DCLT18, BHB*18, HZRS16, HLP*19, HGDG17]. On the long-standing visual recognition problem [RDS*15], convolutional neural networks can even achieve human-level accuracy [HZRS16, HLP*19]. Whereas, there exist large differences between how humans learn one skill and how machines learn one particular task. In general, traditional machine learning algorithms aim to learn from a stationary probability distribution. However, the real-world learning system often requires the model to continuously adapt to a stream of data distributions, which refers to the concept of *continual learning*, i.e., also called *incremental learning*. As one of the most fundamental scenarios in continual learning problem [Rin98, SF86], *task-incremental learning* [VdVT19] requires a single neural network to learn a stream of tasks in a sequential manner and the final model is expected to perform well on all learning tasks. Nevertheless, as the network continuously updates its parameters upon learning new tasks, it tends to catastrophically forget what has been learned in the past [AR97, AR00, GMX*13].

Despite numerous studies being proposed to address the catastrophic forgetting problem, most of them focus on developing new techniques to alleviate the phenomenon of performance degradation, and the evaluation of forgetting is solely represented by the decrease in accuracy. However, diagnosing how the model behaviors lead to knowledge drift and visualizing what has been forgotten is difficult. In particular, we first need to determine what data to collect and what information to extract from the sequentially updated model in order to uncover the forgetting phenomenon. With a proper definition of knowledge, to track the drift over time, we need to convert the collected information into a format that can represent an incremental change with respect to the given old task. Furthermore, the visual representations should be able to facilitate the model developers to analyze network forgetting over a sequence of updated models.

To address the aforementioned challenges, we propose a visual analytics framework, KnowledgeDrift, to interpret and diagnose the network forgetting under the task-incremental learning scenario. Specifically, five visualization components are developed to allow users first to identify the learning stage when catastrophic forgetting has occurred and then navigate to one particular task for further investigations. Then, to support in-depth analysis, we capture the relationships between the original model (i.e., obtained immediately after training that task) and the ones updated upon training the subsequent new tasks, extract the patterns hidden inside the forgotten data over time, and diagnose the drift of detected image features. In addition, we define the task-specific knowledge at the filter level and visualize the change of relevance between the feature maps and the ground-truth labels using mutual information [WLK*19] estimation.

The KnowledgeDrift leverages multiple levels of data granularity, such as data instances, visual features, and network parameters, to generate visual explanations that shed light on the phenomenon of catastrophic forgetting. To demonstrate the effectiveness of our framework, we conducted three case studies to show that our approach can provide valuable insights into the dynamics of learning and forgetting in Task-IL scenarios and support effective comparisons among different learning strategies. Our main contributions include:

- A model-agnostic visual analytics framework for tracking, interpreting, and diagnosing the effect of catastrophic forgetting in task-incremental learning.
- A workflow that guides users to dissect *how* knowledge is forgotten via instance-, feature-, and parameter-level analyses.
- An approach based on Rényi mutual information [WLK*19] to quantify and track the knowledge drift at the parameter level.

## 2. Related Work

### 2.1. Dissecting Catastrophic Forgetting in Continual Learning

Continual learning (CL) or incremental learning (IL) aims to learn multiple tasks in a sequential manner. Whereas the key to success lies in how to update the model according to the current task while retaining the knowledge acquired from the previous tasks. This is particularly challenging when the consecutive tasks have dramatic differences in data distributions. Numerous CL algorithms have been developed to address catastrophic forgetting [DLAM*21, YL*21], but none of them fully resolves this problem.

Besides a number of learning strategies proposed to alleviate catastrophic forgetting in continual learning, there are also a few studies focusing on dissecting this fundamental problem. For instance, one study formulates the forgetting measures through the geometrical properties of the loss landscape [MFPG20]. The goal is to find out how the optimizations settings and regularization approaches can affect the performance in continual learning. Another study [NAL*19] also associates the catastrophic forgetting problems with the relationships among the task sequence. However, it only performs a correlation analysis between the task sequence properties and the amount of forgetting measured during the experiments. It is insufficient to explain the forgetting dynamics and generalize their observations to other studies. Moreover, inspired by the forgetting issue that happens during the sequential training of multiple tasks, one work performs [TSC*18] an empirical study to understand the phenomenon in training individual visual recognition tasks. While it redefines the tasks to be the batches within one dataset, it still provides useful insights that can be extended to analyze the forgetting problems caused by larger distribution shifts across multiple tasks.

Despite significant efforts in investigating forgetting from various perspectives, the aforementioned studies do not provide enough in-depth analysis to allow model users to interpret the internal mechanism. Nevertheless, some of the theoretic studies [TSC*18, MCY*22, KGKY21] can still guide us in extracting useful information for visually analyzing the continual learning process. By combining analytical approaches with multi-level visualization, we are able to provide more insightful explanations.

### 2.2. Visual Analysis for Model Interpretation

As deep learning techniques have made impressive achievements in solving numerous real-world problems, the demand for interpretability of training deep learning models keeps increasing in recent years. A variety of visual analytics frameworks are proposed to interpret the deep learning models [HHC17, NQ17, RFFT16] or analyze a particular deep learning approach [NQ17, WCX*22]. In general, visualizing the hidden representations and analyzing the learning dynamics of the neural network is important for interpreting the current performance and model behaviors. Further, when the model or the learning scenario is complex, it often requires incorporating multiple types of information to analyze the training process [MFH*20, LLS*18]. For example, the visual analytics framework proposed for transfer learning [MFH*20] focuses on helping the users to understand how knowledge is transferred between the source and the target models. To visualize the relationships between the models and uncover the knowledge shared in between, it leveraged both parameter-level and instance-level information. In our work, we focus on interpreting the change of knowledge learned by the neural network across a sequence of models. To provide sufficient explanations, we also leverage multiple levels of data granularity, and each of the views is designed to capture particular types of information.

### 2.3. Visual Analysis for Model Diagnosis

The goal of this type of work is not solely to uncover the information learned by the current model but also to pinpoint the underlying reasons for the performance degradation. Data-driven analysis is one of the most common approaches. For example, InstanceFlow [PHS20] and ModelTracker [ACD*15] aim to facilitate the instance-level visual diagnosis, and ConfusionFlow [HRS*20] interprets the class-level performance across the training epochs. OoDAnalyzer [CYL*20] proposed a grid-based visualization to analyze two types of out-of-distribution data in both local and global contexts. However, when the learning model is complex, only tracking and visualizing the example-level information is insufficient. Therefore, model-driven analysis employs the information extracted from model parameters or learned representations to probe the potential issues. For example, DQNVis [WGSY18] helps domain experts to inspect and diagnose the behaviors of Deep Q-Network agents. To analyze the CNN pruning process and iteratively refine the model performance, CNNPruner [LWS*20] proposes two metrics to evaluate the sensitivity and instability of the convolutional filters and visualize them to assist users to make pruning decisions. Moreover, to identify the data heterogeneity in Federated learning models, HetVis [WCX*22] visualizes the paths of model parameters formed by both server and local clients in a 2D space. However, none of the work can be directly applied to our problem. In task-incremental learning, the model continuously evolves as new data becomes available, which results in a highly complex model. Both analytical methods and visual designs should take this challenge into account. Thus, in this work, we propose KNOWLEDGEDRIFT to better probe and diagnose the forgetting issues in the context of evolving input datasets and models.

## 3. Background and Concepts

### 3.1. Incremental Learning

Continual learning (CL), incremental learning (IL), and lifelong learning (LL) share similar fundamental learning structures. While LL is a broader concept that emphasizes more on how to better learn new tasks [Liu20]. The concept of IL and CL often refers to a learning system that incrementally learns from new datasets which share similar structures with the ones it has learned in the past. While the data streams are similar, the learning objectives can be largely different. For example, training an object detection system for autonomous cars requires the system to gradually recognize a range of objects. At first, the system is trained on a dataset containing images of traffic signs, pedestrians, and all types of vehicles. However, after the car has been driven to a different environment, new objects appear and the system should be able to accurately recognize both new objects and the ones it has seen before. Without effective learning strategies, it can easily forget how to distinguish previously seen objects.

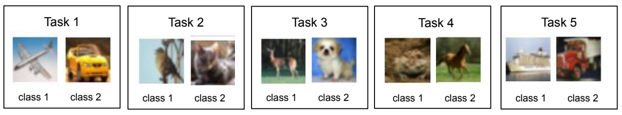


**Figure 2:** *The split CIFAR-10 task protocol. To simulate the Task-IL setup, the entire dataset is divided into five binary recognition tasks. Each task retains its own classifier head.*

### 3.2. Task-Incremental Learning Setup

As one of the most essential scenarios in CL, task-incremental learning (Task-IL) incrementally learns a series of tasks without degrading the performance of tasks learned in the past. The formal definition of this problem considers a set of learning tasks $\mathrm{T} = \{T_1, T_2, ..., T_{|\mathrm{T}|}\}$, and its objective is to minimize the loss:

$$\sum_{i=1}^{T} \mathbb{E}_{(X_{T_i}, Y_{T_i})} [\ell(\theta; (X_{T_i}, Y_{T_i}))], \qquad (1)$$

where $(X_{T_i}, Y_{T_i})$ denotes the training data of task $T_i$, $\ell$ is the loss function, and $\theta$ represents the parameters of the neural network. In this work, as we focus on the image classification tasks, $(X, Y)$ refers to the image dataset.

Under the Task-IL scenario, every time a new task comes, the network updates its parameters according to the new data distribution. During training, each task holds its own classifier, and the task identities are always provided during inference time. An example task protocol is demonstrated in Fig. 2. The *Split CIFAR-10* is a standard CL benchmark [VdVT19], in which the entire CIFAR-10 dataset is split into five binary recognition tasks for simulating an incremental learning scenario. In this case, the five tasks will be trained sequentially, and the final model is expected to achieve acceptable performance on all five binary recognition tasks.

### 3.3. Catastrophic Forgetting in Task-IL

*Catastrophic forgetting* refers to the situation where the model performance evaluated on the previously learned tasks decreases significantly when it is trained on new tasks. A model optimized using conventional algorithms such as stochastic gradient descent (SGD) is prone to using all its capacity for the current task, and in turn, forget what it has learned in the past. Catastrophic forgetting is arguably the major challenge to conquer in CL studies [AR97, AR00, GMX*13, MBB13]. Although various techniques [PKP*19] have been proposed to overcome this challenge from different perspectives, including regularized optimization [KPR*17, LKJ*17, ZPG17, LH17], dynamic model expansion [RRD*16, YYLH17], and data replay [IC18, SLKK17], our goal in this paper is to develop a model-agnostic framework for visually analyzing the forgetting problems. In the following, we first introduce two standard metrics for evaluating the final performance of an incremental learning process [CDAT18, MFPG20]. We then define a few terms related to catastrophic forgetting but in terms of different types of information.

**Final mean accuracy** defines the average inference accuracy examined after all $T$ tasks in the sequence have been trained. Here, we use the final model to test each of the task datasets. $a_{M_T,T_i}$ denotes the inference accuracy of task $T_i$ using the final model $M_T$. Then, the final mean accuracy is defined as:

$$\bar{A} = \frac{1}{T}\sum_{i=1}^{T} a_{M_T,T_i} \tag{2}$$

**Final mean forgetting** defines how much the model performance degraded after training all $T$ tasks, where $a_{M_i,T_i}$ and $a_{M_T,T_i}$ represent the inference accuracy of task $T_i$ using the $i$-th and the final model respectively. The final mean forgetting is computed by averaging the total decrease of accuracy over the past $T-1$ tasks:

$$\bar{F} = \frac{1}{T-1}\sum_{i=1}^{T-1} a_{M_i,T_i} - a_{M_T,T_i} \tag{3}$$

**Forgotten data instances** is defined as a set of data instances of task $T_i$ that are correctly classified by the original model $M_i$ but mispredicted by the current model. Once the neural network is updated to fit the new dataset, **model drift** and **label drift** refer to the change of model parameters and label predictions respectively. Moreover, **information drift** or **knowledge drift** is defined as the change of learned feature representations in the neural network. After the model has adapted itself to the new tasks, its ability to extract representative information with respect to the old task has dropped. Furthermore, **network forgetting** is defined similarly as **catastrophic forgetting** but it highlights the role of neural networks in performance degradation.

## 4. Requirements Analysis

To pinpoint the key challenges in analyzing catastrophic forgetting and meanwhile distill the corresponding design goals, we conducted interviews with two domain experts (E1 and E2) who have been working on Task-IL and general CL research for more than three years. In general, both experts commented that most existing studies focused on proposing new algorithms for mitigating the forgetting issues. In terms of performance evaluation, they only report values of mean accuracy or mean forgetting as defined in Sec. 3.3, but few of them analyzed the exact forgotten contents or the internal dynamics during model updates. E2 also mentioned that there are a few studies trying to provide theoretical explanations behind the forgetting phenomenon, while the work is neither suitable for non-experts model analysts nor efficient in demonstrating the details in terms of the forgotten data or feature representations. Besides discussing with two experts, we also reviewed survey [DLAM*21] on CL and other related literature on catastrophic forgetting. In summary, we identified the following three requirements:

- **R1: Revealing when and where catastrophic forgetting happens.** Prior to any in-depth analyses, it is crucial to have an overall understanding of how models perform in the current task-incremental scenario. Based on the overview, it is easier for analysts to identify when the updated model fails to achieve an acceptable accuracy on the past datasets. To this end, the system needs to track whether each task can still be recalled at each of the following training stages, and support analysts to select one particular task for further investigations.
- **R2: Uncovering drifted information for the selected task.**
  - **R2.1: Capturing the model digression from the original task solution to the final training stage.** During the incremental learning process, the neural network updates its parameters upon fitting the new data distribution. The digression in the parameter space provides the most intuitive understanding of what information has been forgotten. However, only visualizing the change of model parameters is insufficient to reflect the severity of forgetting. Thus, we should also provide information that allows analysts to visually connect the model drift with performance degradation.
  - **R2.2: Exploring the forgotten data instances and their underlying patterns.** Given the selected task and its subsequent models, analysts want to know whether there exist any trends or patterns hidden in the forgotten data instances. For example, when the updated model misclassifies a set of data from the same class, the analysts can check whether those images are recognized as the same wrong labels. If the mispredictions are diverse, then it implies that the feature representations learned in the past are seriously destroyed by training the new tasks.
  - **R2.3: Connecting the hidden patterns with forgetting diagnosis.** Visualizing the extracted patterns should help in probing the forgetting phenomenon. In particular, clustering the data instances with similar reasons for being forgotten can facilitate the inspection process. If two forgotten instances are diagnosed with similar issues, they should stay within the same group.
- **R3: Supporting analysis on why and how forgetting happens.** Uncovering the patterns in the forgotten data is insufficient for explaining why and how forgetting occurs. Analysts need to understand what image features are unlearned by the model updates and how network behaviors relate to the dynamics of drifting.
  - **R3.1: Identifying the important image features that are impacted by the model drift.** Intuitively, the data correctly recognized by the original model was mispredicted by the updated one because the network changed the parameters of past feature detectors. In other words, it detects some new features which confuse the model to make wrong predictions. Identifying and highlighting those features helps analysts to infer the relationships between datasets from different tasks.

– **R3.2: Tracking the knowledge drift in terms of the model behaviors.** For neural networks, the learned knowledge is retained in the model parameters. Therefore, if we can track the task-specific information forgotten or recalled upon learning new tasks in terms of model parameters and visualize how information fluctuates over time, then the analysts can efficiently pinpoint the forgetting issues based on the model behaviors. The model performance can be further improved by regularizing certain sets of parameters.

## 5. Visual Analytics Framework: KnowledgeDrift

### 5.1. System Overview

To facilitate flexible and interactive explorations of all tasks in the sequence, our framework contains three stages. First, once the Task-IL process is complete, the KnowledgeDrift extracts the task dataset and output models generated at each training stage. Then, our back-end starts to process the task sequence and model parameters to obtain the analytical results. During the final stage, the front-end takes in the processed outputs from the back-end and generates all the visualization components as shown in Fig. 1. To fulfill the above requirements, KnowledgeDrift is designed with two major analytical modules. The first module summarizes the overall performance for each of the learning tasks **(R1)**, guiding users to choose a particular task for in-depth investigation. Then, the second module coordinates multiple visualization panels to assist analysts in understanding what has been forgotten and how forgetting occurs **(R2, R3)**. In the following sections, each of the visual components will be introduced in detail.

### 5.2. Performance Overview and Task Navigation

**Tracking the Task-IL Performance.** Statistical information of the entire task sequence and the summary of task performance on the test dataset at each training stage are shown in Fig. 1-A **(R1)**. The left panel provides a concise overview of the training configurations, including details about the benchmark dataset, network architecture, and average accuracy of the final model. The right panel assists users to identify which task to examine by presenting performance degradation for each task between its peak accuracy and inference accuracy at each of the other training stages. Generally, a learning task $T_t$ achieves its maximum accuracy when its dataset is evaluated using its own model obtained immediately after training $T_t$. As shown in Fig. 1-A, the x-axis denotes each of the learning tasks in sequential order. To accommodate a potential large task sequence, we use dots along the y-axis to indicate the forgetting rate of each task measured using each subsequent model. A gradient color scheme is employed to denote the order of models being inferred. To highlight the performance at the final model stage, we connect all the tasks using a dashed line. The alternative visual design is to use line charts or bar charts to encode the change in the forgetting rate. However, when the task sequence is large, lines can overlap with each other across different model stages and bars can take much more space than using dots.

**Task Navigation.** The rightmost panel at the first row in Fig. 1 allows users to choose one particular task based on the overview of model performance, and the detailed configurations for the selected task are displayed below the slider.

### 5.3. Model Drift Visualization

In Fig. 1-C, **the model drift view** demonstrates *forgetting* in terms of how much the current model has digressed from the original solution **(R2.1)**. Tracking drift in the model parameter space provides an intuitive way to understand how much has been forgotten upon learning the new tasks. However, visualizing those model checkpoints in the 2D space is challenging, as neural network parameters are extremely high-dimensional [LXT*18, GVS14]. To avoid producing a misleading projection, it is important that the mapping between high-dimensional and low-dimensional spaces preserves the variation in the parameter space.

To generate the plots in Fig. 1-C, we first collect all model checkpoints after training task T14 to T20. We then flatten all the parameters (i.e., weights and biases) to form a $N \times P$ matrix $M$, where $N$ denotes the total number of model checkpoints and $P$ represents the total number of parameters in the neural network. To reflect the degree of model drift with respect to T14, we rearrange the matrix $M$ to ensure the solution of T14 is the origin in the parameter space. Then, we apply principal component analysis (PCA) on $M$ to find out the top two principle directions as the axes in the 2D space.

However, only visualizing all the model checkpoints in the 2D subspace is not sufficient for users to fully interpret the degree of forgetting. To reflect how much the training or testing loss is sensitive to the change in the parameter space, we sample a regular grid in the projected 2D parameter space, reproject the 2D coordinates back to the original dimension, and evaluate the loss values at each grid point using either training or testing dataset of T14. The resulting 2D scalar fields are rendered as contour plots shown in Fig. 1-C. The contour lines in the plot represent areas where the loss function has the same value. In addition, the blue star and the red dots represent the model state after training T14 and all the following states after training each of the subsequent tasks. We use the size of dots to indicate the order of model states. If the trajectory formed by the subsequent models stays perpendicular to the contour lines and moves towards areas with higher loss values, it indicates that the model escapes rapidly from the original model state of T14 during the subsequent training process.

### 5.4. Visual Analysis for the Forgotten Data

To facilitate the inspection of forgotten data instances, our system incorporates three visual components. First, the Sankey diagram is used to visualize the drift of predicted labels over time (Sec. 5.4.1). Then, to reveal the patterns hidden in the forgotten instances, a scatter plot (Sec. 5.4.4) is employed to show the embedding of individual data points (Sec. 5.4.2). Lastly, linked to the scatter plot, the image feature view showcases the significant image features impacted by the model drift (Sec. 5.4.3).

#### 5.4.1. Visualization of the Label Drift

The view for tracking the drift of predicted labels consists of two panels, the bar chart on the top and the Sankey diagram below, which are aligned along the x-axis. In Fig. 1-D, the bar height demonstrates the amount of forgotten data at each model stage. For example, the value at model stage M18 is computed by counting the total number of data instances that are correctly predicted by model M14 but mispredicted by model M18. The bar chart can only provide an overall count of forgotten instances at individual model stages. To interpret the relationships among various categories at different training stages, we need a visual representation that can reveal the connections among the mispredicted labels and also the underlying patterns over time. Therefore, the Sankey diagram is used to encode how the mispredictions change among different classes across a sequence of model stages. The Sankey diagram utilizes width to facilitate users to quickly identify the underlying patterns or the significant changes at a certain stage.

Across each row, all nodes represent the same class, and each node represents that class at a certain model stage. Between a source and a target node, the width of the flow in between represents the count of samples that are misclassified as that target category. As the example demonstrated in Fig. 1-D, after we have selected task T14 for inspection, a Sankey diagram is generated to show the label change of the forgotten data of T14 through model stages M14 to M20. Starting from model stage M15, we see image instances from all five categories can be misclassified as rabbit images (i.e., row 1). However, at the next model stage M16, among all the images that are misclassified as rabbit, some of them are the ones that are already misclassified as rabbits at the previous model stage, whereas some of them are predicted as raccoon, ray, or road at the previous model stage. The crossovers of the flow between the two model stages imply that the features learned by the previous model have been changed significantly.

### 5.4.2. Diagnosis-Driven Instance Embedding

To provide instance-level analysis, previous works often utilize latent features as the instance embedding [MFH*20]. However, our goal is not to visualize how the updated network performs on separating the old data in the feature space but to facilitate diagnosing how forgetting happens. Therefore, to uncover the underlying pattern in the forgotten data **(R2.2)**, we aim to find an embedding space where the data instances forgotten for similar reasons will be projected closely to each other. To investigate why a subset of task $T_t$ is correctly predicted under model $M_t$ but misclassified by model $M_{t'}$ (i.e., model updated for $T_{t'}$), we first collect the forgotten data and denote them as $\mathrm{X}_{T_t}^{forget}$. The relationship in the embedding space needs to be connected with why the data instances are being misclassified by the current model **(R2.3)**. Hence, to form an informative clustering of forgotten data, we adopt the idea of constructing the parameter-space saliency profiles [LSB*21]. Intuitively, catastrophic forgetting happens because network parameter drift leads to feature detection malfunctions. In fact, when certain parameters become malfunctioning, their gradients tend to be abnormally large [LSB*21, ALT*19]. Therefore, we utilize the filter-wise gradient information estimated using the current model to construct a vectorial representation for each data instance.

To be specific, for each forgotten instance $x_{T_t}^{forget}$, we compute the gradient with respect to each parameter $\theta_i$ of model $M_{t'}$ and aggregate the absolute gradient values at each convolutional filter. The output layer has to be replaced with the task-specific classifier during loss computation. The process of computing the saliency value at each filter $f_k$ can be formulated as:

$$v_{\theta_i}(x_{T_t}^{forget}) = |\nabla_{\theta_i}\mathscr{L}_\theta(x_{T_t}^{forget}, y_{T_t})|, \tag{4}$$

$$\overline{v}_{f_k}(x_{T_t}^{forget}) = \frac{1}{|f_k|}\sum_{i\in f_k} v_{\theta_i}, \tag{5}$$

where $\theta_i$ denotes each individual parameter associated with filter $f_k$ and $y_{T_t}$ represents the ground-true label of $x_{T_t}^{forget}$. Moreover, to better separate the abnormal filters from the benign ones, we further normalize $\overline{v}(x_{T_t}^{forget})$ by mean $\mu_{T_t}$ and standard deviation $\sigma_{T_t}$ computed using all the data instances from task $T_t$, and then clip all values below the mean. The final embedding vector $\hat{v}(x_{T_t}^{forget})$ is computed as:

$$\hat{v}(x_{T_t}^{forget}) = \frac{\overline{v}(x_{T_t}^{forget}) - \mu_{T_t}}{\sigma_{T_t}}. \tag{6}$$

### 5.4.3. Capturing Forgetting in the Input-Space

After generating the instance embedding based on the approach introduced in the previous section, we can project the vectorial representations into 2D space where the images sharing similar reasons for being forgotten will tend to form a cluster. Therefore, by inspecting the image features within the same group, we expect to understand what semantics shared by those images are associated with the occurrence of forgetting (**R3.1**). To efficiently visualize the features in the input space, we generate the forgetting-aware pixel attribution based on the embedding vectors obtained using filter-wise gradient information. Given vector $\hat{v}(x^{forget})$ computed for a forgotten instance $x^{forget}$, the magnitude of each element reflects the degree of the anomaly at each convolutional filter. To highlight the image pixels associated with each deviated filter, we consider the mean vector in Equation 6 as a base embedding vector $v_0$. The base vector represents a special case that there exist no malfunctioning filters and the current model can still recognize the image $x$. To find the forgetting-aware image pixels, we consider applying the classic gradient-based attribution methods [SVZ13, ZF14]. Hence, we compute the gradients by taking the cosine similarity between the base vector $v_0$ and the embedding vector $\hat{v}(x^{forget})$ with respect to the input image $x^{forget}$. The final forgetting-aware attribution map can be defined as:

$$W_{forgetting} = |\frac{\partial cos(\hat{v}(x^{forget}), v_0)}{\partial x^{forget}}|. \tag{7}$$

### 5.4.4. Visualization for the Data Instances

The forgotten instance view (Fig. 1-E) contains two panels. First, users need to select one model stage to investigate from the dropdown menu. Then, the system employs Uniform Manifold Approximation and Projection (UMAP) [MHM18] to process all the forgotten data instances under the selected model stage, e.g., $M_{18}$. The resulting 2D scatter plot (E1) demonstrates the similarities among the data instances based on the reasons for being forgotten where the color represents their ground-true classes. However, if the data instances that belong to the same category are distributed randomly, it implies that learning the new task has significantly impacted the knowledge acquired in the past. After the users have made selections on the scatter plot, the corresponding images with their forgetting-aware attribution maps are displayed on panel E2. For each image, its ground-true label and the mispredicted label will be displayed to the right.

## 5.5. Filter-Level Information Drift

This section first introduces how to track knowledge drift using filter-wise mutual information (Sec. 5.5.1). Then, it discusses how to visualize the knowledge drift across a model sequence (Sec. 5.5.2) and how it assists users to discover hidden patterns.

### 5.5.1. Measuring Filter-Level Information Drift

To support in-depth analysis from the perspective of model behaviors, we need to extract information that is able to represent the *knowledge* acquired in the past and also associated with the model parameters **(R3.2)**. Moreover, as our goal is to visualize how past knowledge is interfered over time, the extracted information should be comparable along the temporal dimension.

Centered Kernel Alignment (CKA) [KNLH19] has been used in many studies as an indicator of forgetting. It measures the similarity between two neural network representations. However, CKA only estimates the correspondences between the activation map extracted using the original model and the one extracted using the current model. In this case, the similarity scores cannot indicate whether the model update indeed boosts or hurts the knowledge relevant to the old task. Therefore, to estimate the amount of knowledge drifted at each convolutional filter, we compute the mutual information (MI) between the filter-level activation maps and the corresponding ground-true label for indicating how much task-reverent information is still retained in the current model. To estimate the MI, we adopt the matrix-based Rényi's entropy estimator [WLK*19], which is designed to address the MI estimation for high dimensional tensor-based variables. Then, given a convolutional filter $f_k$, we can measure the change of task-reverent information as:

$$MI(A_{f_k};y^c) = H(A_{f_k}) + H(y^c) - H(A_{f_k},y^c), \tag{8}$$

$$\Delta I^c_{f_k} = MI(A_{f_k};y^c) - MI(A^*_{f_k};y^c), \tag{9}$$

where $y^c$ represents the ground-true label in one-hot encoding, $A_{f_k}$ and $A^*_{f_k}$ denote the filter-level activation extracted from the original model and current model respectively. Detailed procedures on how to track the information drift across the updated models are described in Algorithm 1.

**Algorithm 1** Estimating the knowledge drift over subsequent tasks.

**Input:** $K$ selected filters, $\{f_1, f_2, ..., f_K\}$; the data of task $T_t$ in mini-batches $\{b1, b2, ..., b_n\}$; $m$ models $\{M_{t+1}, M_{t+2}, ..., M_{t+m}\}$ updated for the $m$ subsequent learning tasks

**Output:** 2D array $S_{\Delta I}$ recording the change of MI w.r.t. the task $T_t$

```
1: $S_{\Delta I} \leftarrow$ empty array
2: for $i = 1 : m$ do
3:     for $k = 1 : K$ do
4:         for $j = 1 : n$ do
5:             compute $\Delta I_{f_k,j} \leftarrow MI(A^t_{f_k};y) - MI(A^{t+i}_{f_k};y)$
6:         end for
7:         $S_{\Delta I_{i,k}} \leftarrow \frac{1}{n}\sum_j \Delta I_{f_k,j}$
8:     end for
9: end for
```

### 5.5.2. Information Drift Visualization

To prevent any confusion regarding the name of the framework, the last panel is named as *information drift view*. The information drift view is designed to facilitate the filter-level diagnosis of forgetting (Fig. 1-F). To directly interpret how much information is *forgotten* or *remembered* at each filter, we use the change of mutual information with respect to the old task computed based on Equation 9 instead of displaying the raw values. However, examining filters from all convolutional layers simultaneously is both impractical and inefficient. Therefore, the top context panel provides a layer-wise summary and allows users to toggle between the overview and the filter-specific detailed information presented on the bottom.

Each box plot in Fig. 1-F1 reveals the distribution of $\Delta I$ from all the subsequent models (i.e., $M_{15}$ to $M_{20}$) at that layer. The users can slide to the layer of interest to further inspect the filter-wise estimations in detail. For the second panel F2, the x-axis denotes the convolutional filter by index and the y-axis denotes all the subsequent models following the training order. The color of each cell indicates how much knowledge relevant to the old task is discarded (in red) or increased (in blue). By observing the values along the y-axis, users are able to interpret the trend of knowledge drift over time. Oftentimes, the network parameters tend to lose information relevant to the old task after learning the following new tasks. Thus, we expect to see that most cells are in red and their trend along the y-axis aligns with the measures of performance drift over time. However, in some cases, certain filters may relearn the task-relevant knowledge which can be observed according to cells in light blue. If there exist numerous cells colored in blue, users may need to consider retraining the classifier for that old task, since it implies that a few filters indeed still maintain the past knowledge but they are incompatible with the old classifier.

## 6. Use Cases

### 6.1. Naive Task-incremental Learning

In the first study, we employ the Split MNIST dataset, a classic incremental learning benchmark. Similar to the task setup in Figure 2, the original MNIST was divided into 5 binary digit recognition tasks. The primary objective of this study is to demonstrate an analytical scenario where an incremental learning strategy is not utilized, leading to the forgetting of prior tasks as the model is simply fine-tuned with subsequent task datasets. The task sequence is trained using a two-layer ConvNet. We start with inspecting the performance overview panel. As shown in Fig. 3, task T2 encounters the most serious forgetting among all five tasks and it reaches its peak forgetting after the model has been updated for task T5. We then want to understand why T2's dataset has been dramatically forgotten at the final model stage (M5).

To understand how the following models progressively deviate from the original solution of T2, we check the model drift view as shown in the left panel of Fig. 3. In contrast to the locations of model M3 and M4 (i.e., two smaller red dots), model M5 significantly deviates from the solution of T2, which aligns with the dramatic performance decrease indicated in the performance overview chart (see Fig. 3). In the next step, want to further investigate

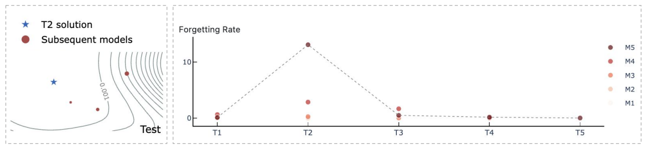


**Figure 3:** *Overview for the split MNIST experiment which helps the analysts to navigate a specific task to analyze.*

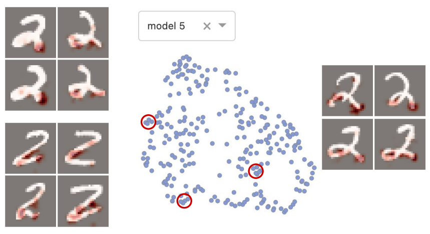


**Figure 4:** *Forgetting-aware saliency maps for the split MNIST experiment. The red color highlights the important features for analyzing the forgetting of task T1 at model stage M5.*

why the neural network is confused by the visual features of T2's dataset. For the next step, we aim to further investigate why the neural network is facing difficulty in accurately distinguishing the visual features between the two classes of T2.

At the instance panel, we inspect individual forgotten instances and their corresponding forgetting-aware attribution maps. Fig. 4 demonstrates the diagnosis-driven clustering result at model stage M5. All the forgotten images are digit-2, which indicates that images of digit-2 can be easily misrecognized as digit-3 after the model has been trained for task T5. Upon examining the scatter plot, we observed numerous smaller clusters and selected several image instances from three random clusters. Notably, images within the same cluster share similar writing styles, but are distinct from the ones in other clusters. By checking the attribution maps in detail, we found that dissimilar handwriting features indeed lead to varying reasons for being forgotten. For instance, the images from the top left and the middle clusters are highlighted for similar visual features i.e., ending points of the handwriting digit. The highlights in red indicate the image features that are important for not being recognized as digit-3 but overlooked by the updated model.

## 6.2. Task-incremental Learning with Algorithm

For the second case study, we evaluate our framework on a more complex incremental learning benchmark, Split CIFAR-100, which contains 20 5-class recognition tasks. During the incremental training process, an advanced incremental learning algorithm, Averaged Gradient Episodic Memory (A-GEM) [CRRE18], was employed to alleviate the forgetting problem.

As observed from Fig. 1-A, the performance of task T14 stays relatively the same starting from the model stage M16. To further investigate whether there is any internal change during that training period, we first check on the model drift view as shown in Fig. 1-C. The model dynamics in the testing loss landscape are indeed different from the ones in the training loss landscape. By analyzing how the path of subsequent models interacts with the loss contours, we know that during training the model does keep deviating from the solution of T14. However, when we evaluate the same dynamics using testing data, we observed that the last three models roughly stay along the same contour line with a loss value of 0.012, which helps explain why the performance stays relatively the same through the last few model stages. While the forgetting rate stays relatively stable during the last five model stages, to better understand whether there is any change in the forgotten data instances, we decide to inspect the label drift view as shown in Fig. 1-D.

By interpreting the overall trend in the Sankey diagram, we observed that most of the forgotten instances are mispredicted to the first three categories (i.e., rabbit, raccoon, and ray). Even though the numbers of forgotten data are very similar from model stage M17 to M19 based on the bar chart above, the distribution of the mispredicted labels indeed changes dramatically. In particular, many instances are mispredicted as rabbits at model stage M18 but their previous predictions come from various classes. We see many lines of flow exchanging among the first three categories, which also implies that the feature representations learned for the three animal classes have been seriously interfered during the sequential model updates. To mitigate this problem, we can adjust the training images for those three categories. Furthermore, we also noticed that a large amount of flow stays parallel between the same class across different model stages. It is possible that many instances are constantly biased towards a certain category. Our hypothesis has been confirmed by checking the images from the scatter plot.

## 6.3. Comparing the Efficiency of Task-IL Algorithms

In the final case study, we present a comparative analysis using three different learning strategies with the split CIFAR-10 benchmark containing five binary recognition tasks. To illustrate how the KnowledgeDrift conveys meaningful comparisons among different IL strategies, we ran a baseline method (i.e., finetuning using SGD), and two state-of-the-art algorithms (i.e., A-GEM [CRRE18] and ERRingBuffer [CRE*19]) with a fixed random seed. Our goal is to compare the effectiveness of all three algorithms and identify similarities and differences among them.

We first compare the overall performance of all three algorithms as shown in Fig. 5. At the final model stage, the mean accuracy for SGD, A-GEM, and ERRingBuffer are 74.65%, 84.24%, and 83.18%, respectively. Notably, the baseline method achieved the worst performance since it did not employ any strategy to reduce catastrophic forgetting. While A-GEM and ERRingBuffer attain comparable final mean accuracy, their performances are very different across individual training stages. All methods share the same final solution after learning the first task T1 since the forgetting-reducing mechanism only comes into play from the second task onwards. Therefore, to ensure a fair comparison, we choose to investigate the forgetting of task T1 for each of the three strategies.

Fig. 6 shows the corresponding three model drift views. We observed that even though all solutions of T1 (indicated by the blue

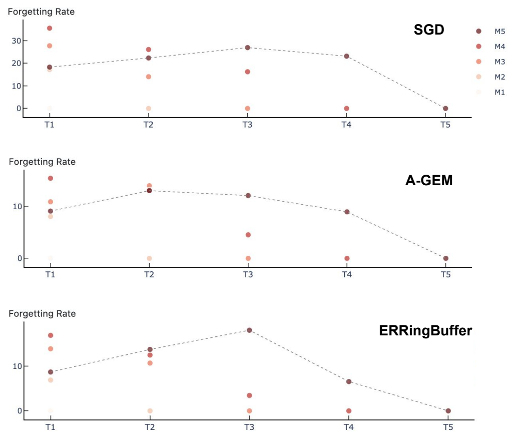


**Figure 5:** *The performance overview of the three incremental learning strategies for the Split CIFAR10 experiment. A-GEM and ERRingBuffer reach similar final mean accuracy but their performances are different at individual training stages.*

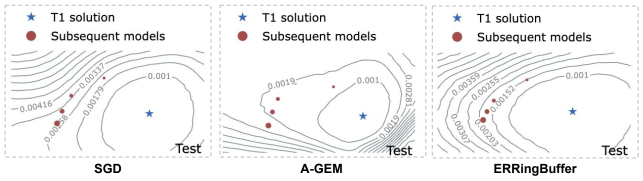


**Figure 6:** *The model space views of three Task-IL strategies.*

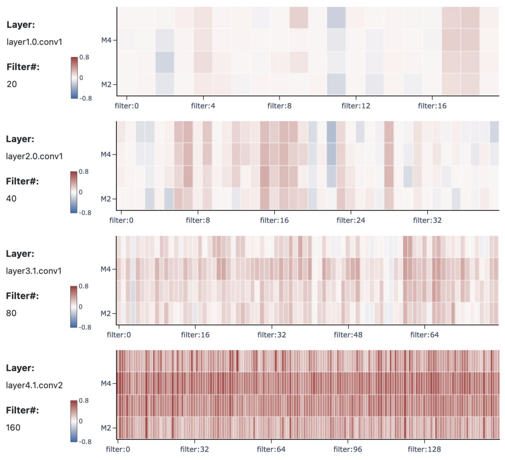


**Figure 7:** *The visualization of filter-wise mutual information change with respect to the first task at four different layers. From top to bottom, figures are arranged from shallow layers to deep layers. The darker shade of red indicates a higher degree of forgetting.*

star) reach the same position in the loss scalar field, the paths formed by the subsequent models (represented by red dots) diverge in very different directions in the model parameter space. Compared to the baseline approach, A-GEM effectively guides the incremental training process towards a flatter region where the loss value changes very little with respect to small changes in the model parameters. Furthermore, based on the comparison of contour densities among the three approaches, we observe that while the objective of the ERRingBuffer algorithm is not to drive the model to a flat region, it does aim to slow down the rate of model digression.

Next, we further explore why the three strategies lead to such distinct learning dynamics. In particular, we examine whether there exist any filter-level patterns that can uncover their underlying mechanisms. After exploring the information drift view for each method, we see that overall they share the same trend: deeper layers tend to experience more severe forgetting compared to the layers closer to inputs. In Fig. 7, the heatmaps present the values of $\Delta I$ at layers L1, L5, L13, and L19, arranged from top to bottom. The darker shade of red indicates a higher degree of forgetting occurring in that particular layer. To identify the differences among the algorithms at a layer suffering from significant forgetting, we select a deeper layer, *layer4.1.conv1*, for comparison as shown in Fig. 8.

In general, we noticed that for each method the general trend of $\Delta I$ variation along the y-axis is aligned with the performance of T1. As in Fig. 5, all charts show that the model was able to retrieve some past information during the last training stage, resulting in a decrease in forgetting. Moreover, the dark red color in the SGD heatmap indicates the most significant drift of knowledge, confirming the limitations of this approach. In contrast, A-GEM demonstrates the least amount of forgetting, with some cells in light blue color even showing an increase in T1-related information at layer *layer4.1.conv1*. Lastly, we found the general trends of SGD and ERRingBuffer across model stages M2 to M5 are similar except that ERRingBuffer has much smaller measures of forgetting.

## 7. Discussion

### 7.1. Feedback from the Domain Experts

To collect sufficient feedback regarding our framework, we conducted interviews with three continual learning experts (E1, E2, and E3). During each discussion, we first introduced the pipeline and the functionality of each panel. We then invited the domain experts to freely interact with the system.

In general, experts agreed that the designs of KnowledgeDrift well suited their analysis workflow. By observing the explorations conducted by each expert, we found that typically they preferred to check the model drift view immediately after they had selected a particular task to inspect. Both E1 and E3 commented that visualizing the subsequent model checkpoints on the loss landscape allows them to easily interpret how the later model diverged from the original solution. In addition, E3 liked the idea of showcasing the loss landscapes of both training and test datasets side by side. This design would allow for a more comprehensive analysis since it was possible that forgetting may become severe only for the test dataset. E3 also commented that when there existed a large number of subsequent tasks, the representations on the projected parameter space were even easier to follow than checking the forgetting rate from the chart in the overview.

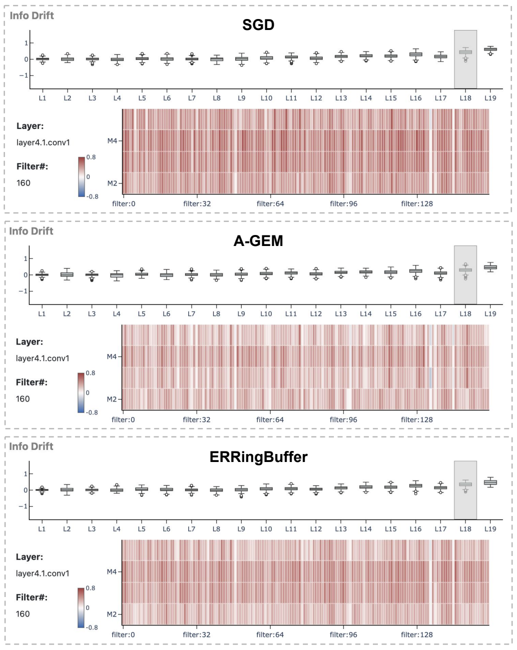


**Figure 8:** *The information drift views for the three incremental learning strategies. Even though the baseline approach (SGD) encounters the severest forgetting among the three strategies, it shares similar patterns with the ERRingBuffer at layer layer4.1.conv1.*

For the instance-level analysis, E2 first commented that if the learning only involved a few tasks, he would directly inspect the label-drift view to see whether the data instances were constantly being mispredicted to a certain label, and then investigated the corresponding images at particular model stages. However, both E2 and E3 mentioned that it took some time to fully understand how to interpret the encoding of the Sankey diagram. E3 suggested that it would be very useful if the design could track the status of all the data instances at each model checkpoint and trace their true labels. Furthermore, all experts agreed that it is effective to use filter-wise gradient information to embed the forgotten data since the data clusters facilitated users to diagnose the issues by groups. Typically, they would visualize the misclassified instances by solely projecting their features extracted from the last hidden layer. However, E1 pointed out that the connection between the label drift view and the scatter plot was weak, as the Sankey diagram did not provide any information on the true labels which made it difficult to associate the diagram with individual image instances in the scatter plot.

E1 comments that the model drift view is very beneficial for having a general understanding of the forgetting process. In addition, as shown in the algorithm comparison study, it is impressive to see the information drift view indeed can uncover many useful insights. Both E1 and E2 acknowledge that the label drift view and the diagnosis-driven clustering of the forgotten instances are able to efficiently reveal the hidden patterns over time. Particularly, the data clustering approach facilitates the exploration by each group, which has not been considered in any previous studies. Similar to our findings in the third case study, all experts agreed that our information drift view could efficiently reveal the trend over layers. If catastrophic forgetting only happened in the last hidden layers, it is worth trying to freeze those layers or retrain the linear classifier to reduce forgetting. In general, all three experts commented that the KNOWLEDGEDRIFT was presented in a good way especially when the users already made hypotheses on why forgetting happened for a particular task, and the interactive system could help verify and consolidate their understanding.

### 7.2. Limitations and Future Work

**Scalability.** As our analytical algorithms mainly rely on neural networks and their parameters, the computational cost depends on the choice of neural network and the size of the task sequence. For example, for the Split CIFAR-10 benchmark with ResNet-18, it takes around 4 and 4.5 hours respectively to produce the diagnosis-driven embedding and compute the Rényi mutual information as both of them involve massive computation at the filter level. Thus, to ensure a smooth analysis process, we pre-processed most of the analysis offline. In terms of the visual design, the Sankey diagram in the label drift view and the heatmap in the information drift view can be affected by the size of the task sequence. Besides limiting the total number of tasks to be displayed on the panels, we plan to improve the two visual components to better handle real-world scenarios where the number of tasks can be unbounded.

**Generalizability.** While our main focus is to analyze sequences of image classification tasks, the analytical approaches we propose can be adapted to other types of learning tasks. However, it requires some adjustments to the instance-level visualization. Currently, our system only supports visual analysis for learning tasks trained with Convolutional Neural Networks (CNNs). In the future, we consider extending our framework to accommodate other network architectures, such as the Vision Transformer (ViT). Furthermore, during the interview, one expert suggested that our work could be extended to analyze class-incremental learning which is more challenging than Task-IL. This can be valuable for future work.

## 8. Conclusion

We developed a visual analytics framework to assist analysts to interpret and diagnose catastrophic forgetting in task-incremental learning. Our framework provides comprehensive explanations via two visual analytical modules. The first module offers a model performance summary that enables analysts to navigate to a specific task for inspection. The second module coordinates five visual components to facilitate a thorough task-driven analysis. To enable the investigation and visualization of forgotten information at various data granularity levels, we propose a diagnosis-driven clustering approach to analyze the forgotten data, a saliency map to capture the forgetting of image features and a knowledge drift measurement at the model parameter level. We also demonstrate that KNOWLEDGEDRIFT provides valuable insights into the strengths and weaknesses of various incremental learning algorithms.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. This work is financially supported by the NSF-funded AI institute [Grant No. OAC-2112606].

## References

[ACD*15] AMERSHI S., CHICKERING M., DRUCKER S. M., LEE B., SIMARD P., SUH J.: Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), pp. 337–346. 3

[ALT*19] ACHILLE A., LAM M., TEWARI R., RAVICHANDRAN A., MAJI S., FOWLKES C. C., SOATTO S., PERONA P.: Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 6430–6439. 6

[AR97] ANS B., ROUSSET S.: Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie 320*, 12 (1997), 989–997. 2, 4

[AR00] ANS B., ROUSSET S.: Neural networks with a self-refreshing memory: knowledge transfer in sequential learning tasks without catastrophic forgetting. *Connection science 12*, 1 (2000), 1–19. 2, 4

[BHB*18] BATTAGLIA P. W., HAMRICK J. B., BAPST V., SANCHEZ-GONZALEZ A., ZAMBALDI V., MALINOWSKI M., TACCHETTI A., RAPOSO D., SANTORO A., FAULKNER R., ET AL.: Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018). 2

[CDAT18] CHAUDHRY A., DOKANIA P. K., AJANTHAN T., TORR P. H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 532–547. 4

[CRE*19] CHAUDHRY A., ROHRBACH M., ELHOSEINY M., AJANTHAN T., DOKANIA P. K., TORR P. H., RANZATO M.: On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486* (2019). 8

[CRRE18] CHAUDHRY A., RANZATO M., ROHRBACH M., ELHOSEINY M.: Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420* (2018). 8

[CYL*20] CHEN C., YUAN J., LU Y., LIU Y., SU H., YUAN S., LIU S.: Oodanalyzer: Interactive analysis of out-of-distribution samples. *IEEE transactions on visualization and computer graphics 27*, 7 (2020), 3335–3349. 3

[DCLT18] DEVLIN J., CHANG M.-W., LEE K., TOUTANOVA K.: Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018). 2

[DLAM*21] DE LANGE M., ALJUNDI R., MASANA M., PARISOT S., JIA X., LEONARDIS A., SLABAUGH G., TUYTELAARS T.: A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence 44*, 7 (2021), 3366–3385. 2, 4

[GMX*13] GOODFELLOW I. J., MIRZA M., XIAO D., COURVILLE A., BENGIO Y.: An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211* (2013). 2, 4

[GVS14] GOODFELLOW I. J., VINYALS O., SAXE A. M.: Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544* (2014). 5

[HGDG17] HE K., GKIOXARI G., DOLLÁR P., GIRSHICK R.: Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2961–2969. 2

[HHC17] HOHMAN F., HODAS N., CHAU D. H.: Shapeshop: Towards understanding deep learning representations via interactive experimentation. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (2017), pp. 1694–1699. 3

[HLP*19] HUANG G., LIU Z., PLEISS G., VAN DER MAATEN L., WEINBERGER K.: Convolutional networks with dense connectivity. *IEEE transactions on pattern analysis and machine intelligence* (2019). 2

[HRS*20] HINTERREITER A., RUCH P., STITZ H., ENNEMOSER M., BERNARD J., STROBELT H., STREIT M.: Confusionflow: A model-agnostic visualization for temporal analysis of classifier confusion. *IEEE Transactions on Visualization and Computer Graphics* (2020). 3

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778. 2

[IC18] ISELE D., COSGUN A.: Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), vol. 32. 4

[KGKY21] KAUSHIK P., GAIN A., KORTYLEWSKI A., YUILLE A.: Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *arXiv preprint arXiv:2102.11343* (2021). 2

[KNLH19] KORNBLITH S., NOROUZI M., LEE H., HINTON G.: Similarity of neural network representations revisited. In *International Conference on Machine Learning* (2019), PMLR, pp. 3519–3529. 7

[KPR*17] KIRKPATRICK J., PASCANU R., RABINOWITZ N., VENESS J., DESJARDINS G., RUSU A. A., MILAN K., QUAN J., RAMALHO T., GRABSKA-BARWINSKA A., ET AL.: Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences 114*, 13 (2017), 3521–3526. 4

[LH17] LI Z., HOIEM D.: Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence 40*, 12 (2017), 2935–2947. 4

[Liu20] LIU B.: Learning on the job: Online lifelong and continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 13544–13549. 3

[LKJ*17] LEE S.-W., KIM J.-H., JUN J., HA J.-W., ZHANG B.-T.: Overcoming catastrophic forgetting by incremental moment matching. *arXiv preprint arXiv:1703.08475* (2017). 4

[LLS*18] LIU M., LIU S., SU H., CAO K., ZHU J.: Analyzing the noise robustness of deep neural networks. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2018), IEEE, pp. 60–71. 3

[LSB*21] LEVIN R., SHU M., BORGNIA E., HUANG F., GOLDBLUM M., GOLDSTEIN T.: Where do models go wrong? parameter-space saliency maps for explainability. *arXiv preprint arXiv:2108.01335* (2021). 6

[LWS*20] LI G., WANG J., SHEN H.-W., CHEN K., SHAN G., LU Z.: Cnnpruner: Pruning convolutional neural networks with visual analytics. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2020), 1364–1373. 3

[LXT*18] LI H., XU Z., TAYLOR G., STUDER C., GOLDSTEIN T.: Visualizing the loss landscape of neural nets. *Advances in neural information processing systems 31* (2018). 5

[MBB13] MERMILLOD M., BUGAISKA A., BONIN P.: The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology 4* (2013), 504. 4

[MCY*22] MIRZADEH S. I., CHAUDHRY A., YIN D., HU H., PASCANU R., GORUR D., FARAJTABAR M.: Wide neural networks forget less catastrophically. In *International Conference on Machine Learning* (2022), PMLR, pp. 15699–15717. 2

[MFH*20] MA Y., FAN A., HE J., NELAKURTHI A. R., MACIEJEWSKI R.: A visual analytics framework for explaining and diagnosing transfer learning processes. *arXiv preprint arXiv:2009.06876* (2020). 3, 6

[MFPG20] MIRZADEH S. I., FARAJTABAR M., PASCANU R., GHASEMZADEH H.: Understanding the role of training regimes in continual learning. *arXiv preprint arXiv:2006.06958* (2020). 2, 4

[MHM18] MCINNES L., HEALY J., MELVILLE J.: Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018). 6

[NAL*19] NGUYEN C. V., ACHILLE A., LAM M., HASSNER T., MAHADEVAN V., SOATTO S.: Toward understanding catastrophic forgetting in continual learning. *arXiv preprint arXiv:1908.01091* (2019). 2

[NQ17] NORTON A. P., QI Y.: Adversarial-playground: A visualization suite showing how adversarial examples fool deep learning. In *2017 IEEE symposium on visualization for cyber security (VizSec)* (2017), IEEE, pp. 1–4. 3

[PHS20] PÜHRINGER M., HINTERREITER A., STREIT M.: Instanceflow: Visualizing the evolution of classifier confusion on the instance level. *arXiv preprint arXiv:2007.11353* (2020). 3

[PKP*19] PARISI G. I., KEMKER R., PART J. L., KANAN C., WERMTER S.: Continual lifelong learning with neural networks: A review. *Neural Networks 113* (2019), 54–71. 4

[RDS*15] RUSSAKOVSKY O., DENG J., SU H., KRAUSE J., SATHEESH S., MA S., HUANG Z., KARPATHY A., KHOSLA A., BERNSTEIN M., ET AL.: Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252. 2

[RFFT16] RAUBER P. E., FADEL S. G., FALCAO A. X., TELEA A. C.: Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics 23*, 1 (2016), 101–110. 3

[Rin98] RING M. B.: Child: A first step towards continual learning. In *Learning to learn*. Springer, 1998, pp. 261–292. 2

[RRD*16] RUSU A. A., RABINOWITZ N. C., DESJARDINS G., SOYER H., KIRKPATRICK J., KAVUKCUOGLU K., PASCANU R., HADSELL R.: Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016). 4

[SF86] SCHLIMMER J. C., FISHER D.: A case study of incremental concept induction. In *AAAI* (1986), vol. 86, pp. 496–501. 2

[SLKK17] SHIN H., LEE J. K., KIM J., KIM J.: Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690* (2017). 4

[SVZ13] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013). 6

[TSC*18] TONEVA M., SORDONI A., COMBES R. T. D., TRISCHLER A., BENGIO Y., GORDON G. J.: An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159* (2018). 2

[VdVT19] VAN DE VEN G. M., TOLIAS A. S.: Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734* (2019). 2, 3

[WCX*22] WANG X., CHEN W., XIA J., WEN Z., ZHU R., SCHRECK T.: Hetvis: A visual analysis approach for identifying data heterogeneity in horizontal federated learning. *IEEE Transactions on Visualization and Computer Graphics* (2022). 3

[WGSY18] WANG J., GOU L., SHEN H.-W., YANG H.: Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE transactions on visualization and computer graphics 25*, 1 (2018), 288–298. 3

[WLK*19] WICKSTRØM K., LØKSE S., KAMPFFMEYER M., YU S., PRINCIPE J., JENSSEN R.: Information plane analysis of deep neural networks via matrix-based renyi's entropy and tensor kernels. *arXiv preprint arXiv:1909.11396* (2019). 2, 7

[YL*21] YIN H., LI P., ET AL.: Mitigating forgetting in online continual learning with neuron calibration. *Advances in Neural Information Processing Systems 34* (2021), 10260–10272. 2

[YYLH17] YOON J., YANG E., LEE J., HWANG S. J.: Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547* (2017). 4

[ZF14] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833. 6

[ZPG17] ZENKE F., POOLE B., GANGULI S.: Continual learning through synaptic intelligence. In *International Conference on Machine Learning* (2017), PMLR, pp. 3987–3995. 4