CLeVer: Continual Learning Visualizer for Detecting Task Transition Failure

Minsuk Chang[*]
Seoul National University

Donghun Kim[†]
Seoul National University

Hyeon Jeon[‡]
Seoul National University

Seokweon Jung[§]
Seoul National University

Jinwook Seo[¶]
Seoul National University

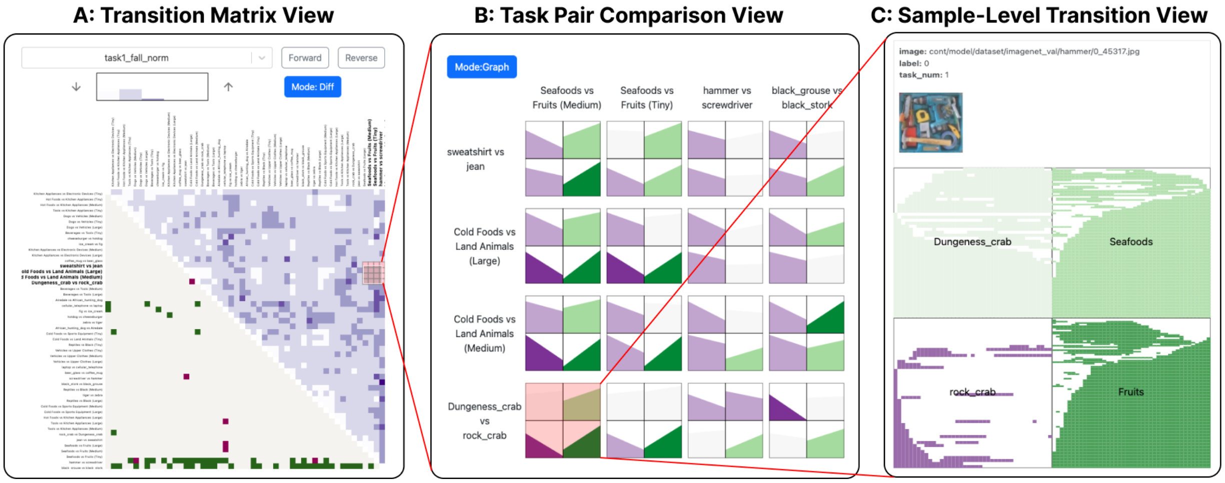
A: Transition Matrix View
task1_fall_norm
Forward
Reverse
Mode: Diff
B: Task Pair Comparison View
Mode:Graph
Seafoods vs Fruits (Medium)
Seafoods vs Fruits (Tiny)
hammer vs screwdriver
black_grouse vs black_stork
sweatshirt vs jean
Cold Foods vs Land Animals (Large)
Cold Foods vs Land Animals (Medium)
Dungeness_crab vs rock_crab
C: Sample-Level Transition View
Dungeness_crab
Seafoods
rock_crab
Fruits

image: cont/model/dataset/imagenet_val/hammer/0_45317.jpg
label: 0
task_num: 1

Figure 1: Three hierarchical views of CLeVer. (A) The Transition Matrix View shows how well the task transition is made among each pair through matrix cells with color bins. (B) The Task Pair Comparison View shows abstract class-level information about the transition process with two modes. (C) The Sample-Level Transition View shows the sample-wise result of the transition and a corresponding image.

Abstract

We introduce CLeVer, a novel visualization system designed to analyze and enhance the performance of continual learning models by detecting task transition failures. Based on the literature review, we discovered and classified three primary causes of task transition failure in classification tasks: class/position inconsistency, biased/noisy samples, and diverse class scopes. These problems are critical in terms of model performance but were not tackled in prior research on continual learning. CLeVer is designed to address these challenges as an integrated system for model experiments and visual analysis. Firstly, users can easily configure the tasks for continual learning through a single JSON file. Then, our system automatically simulates the continual learning process in a relatively short time while generating data for visualization. Finally, the transition visualizer provides an effective visual representation of the task transition process where users can easily detect the task transition failures in continual learning. Our interview with machine learning experts and a case study with three participants demonstrate CLeVer's utility in detecting and addressing such failures. We also discuss the system's potential applicability and adaptability for various computer vision tasks while suggesting our future work.

Index Terms: Human-centered computing [Visualization toolkits]; Computing methodologies [Lifelong machine learning]

1 Introduction

Recently, there has been substantial improvement in the performance of deep learning models. This is mainly due to the complex representations that large models can hold [2]. However, especially for edge devices, a small model size is required to maintain low latency and power consumption, which also lowers its performance [12].

Therefore, *continual learning* emerged to perform various tasks on edge devices with a small model. Continual learning denotes a new deep learning paradigm that continuously trains a single small model for various tasks. Unlike traditional single-step training, continual learning models should smoothly transition between infinite streams of tasks to maintain their performance. Prior works tried to prevent the model from losing its knowledge during the transition by designing a loss function [15], changing the model architecture [18], or keeping past samples in an external memory [22].

However, in the classification domain, there are cases where the task transition cannot be made fluently due to the diversity of in-the-wild tasks. We discovered three cases where task transitions tend to fail, which can easily happen during deployment but cause severe degradation in model performance. Firstly, the position of classes may differ between the two tasks, causing only one to perform well. Secondly, **biased or noisy samples** that do not perfectly match the class labels may be introduced. Finally, the granularity of classes may vary following the application's requirements, leading to a similar result as the first problem. A detailed explanation of these challenges is described in Section 3. Machine learning (ML) engineers must ensure these problems do not appear on their models; however, it is difficult to do so, especially when deploying models in edge devices.

Therefore, we propose CLeVer, comprehensive system for model

[*] e-mail: jangsus1@snu.ac.kr
[†] e-mail: hunkim98@snu.ac.kr
[‡] e-mail: hj@hcil.snu.ac.kr
[§] e-mail: swjung@hcil.snu.ac.kr
[¶] e-mail: jseo@snu.ac.kr

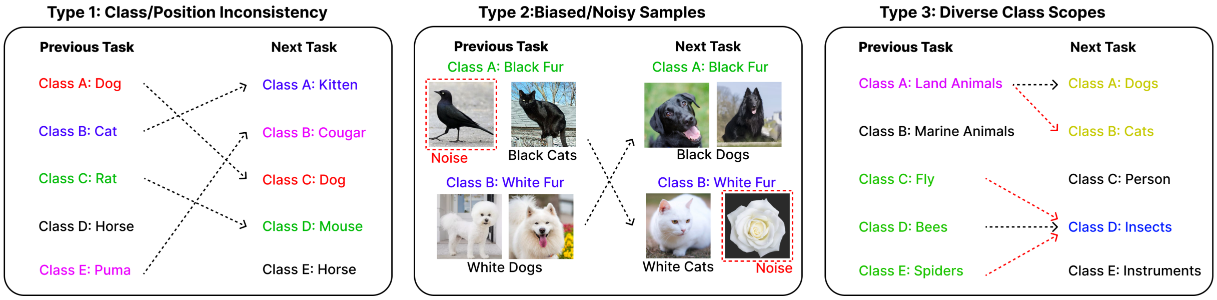

Figure 2: Examples of three potential problems that can cause the task transition to fail. Type 1 is about class and position inconsistency among tasks, showing that the class labels of Dog, Cat, Rat, and Horse can differ. Type 2 shows that class names and samples might not match, introducing an unintended bias factor of species (dogs/cats) against class (colors). It also exemplifies the possible noisy samples with red boxes. Type 3 indicates that tasks may vary in class scopes, causing some of the labels to be reallocated (marked as red lines) during the transition.

experiments and analysis targeting ML engineers as its core users. It helps users detect the aforementioned challenges in the class-incremental task domain of continual learning. CLeVer consists of two components: *Continual Learning Simulator* for model experiments, and *Transition Visualizer* for visual analysis. First, the Continual Learning Simulator automatically generates data through fast simulation of continual learning based on the user's task configuration. Through CLeVer, users can easily describe their required tasks by only writing information in a JSON file. Based on the configuration, CLeVer automatically generates the data through model experiments. Training a continual learning model takes a lot of time due to its design for learning an infinite stream of tasks. Therefore, we adopt a task-pairwise model experiment to simulate task transitions in continual learning with lower time cost. After generating the data, the Transition Visualizer effectively visualizes the data through the three components, allowing users to detect the continual learning challenges easily. Our integrated system design simplifies task setup and model experimentation while offering insights into continual learning models through visual analytics.

We conduct an expert interview and a case study to validate that our system can support users in detecting the practical challenges of continual learning. Both studies confirmed that our system can effectively assist users in detecting the aforementioned task transition challenges with apparent patterns. We wrap up our paper by discussing how our system can be extended for various computer vision tasks with enhanced scalability. Also, we suggest sample group analysis as our future work to support users' better understanding of noisy/biased samples through grouping.

2 Backgrounds and Related Work

We describe previous literature on continual learning and the visual analytics systems for deep learning models.

2.1 Continual Learning

Continual learning involves adapting to a sequence of tasks without forgetting previous ones, despite the challenge of catastrophic forgetting [7, 17]. Solutions include a specialized loss function to preserve prior predictions [15] and a replay buffer for old samples [22]. While these methods help retain knowledge, they don't address real-world challenges during task transitions. Our work in Section 3 outlines these issues and introduces a detection system for them.

2.2 Visual Analytic Systems for Deep Learning Models

Research efforts have aimed to merge visual analytics with deep learning for training analysis. An interactive system was developed for analyzing the training of deep learning models, albeit limited to a single model at a time [20]. Another project offered a system for comparing model snapshots during training, missing the aspect of continuous model evolution [31]. Reviews have covered efforts to visualize ML model behaviors, yet not focusing on continual learning [4]. A conceptual framework was suggested for managing continual learning, but it lacked implementation and validation [11]. Addressing these gaps, our work introduces CLeVer, a visual analytic system designed for identifying task transition failures in continual learning.

3 Task Transition Challenges

As continual learning consists of an infinite stream of task transitions, performing each transition without losing the prior knowledge is essential for maintaining the performance. However, our literature review discovered that in an in-the-wild environment, there are circumstances where the performance of a previous task has to decrease so that the next task can perform better. We analyzed and categorized three potential problems that inevitably cause the task transition to fail but were not spotlighted in the prior works. The brief examples of each challenge are described on Figure 2.

3.1 Type 1: Class/Position Inconsistency

Each classification task holds a list of classes that need to be classified by the model. Here, the position or order of these classes or labels is often neglected because they did not cause problems in traditional ML models due to the nature of machine learning. However, in continual learning, the inconsistency between the classes and their order can become a serious problem in terms of transition. The transition between tasks with different class positions causes an unnecessary shift in the model's latent space and decision boundary [10]. For example, assume that a classifier is trained with *task A*, which predicts the images of dogs as 0 and cats as 1. If the next task classifies dogs and cats but with different labels (0 for cats and 1 for dogs), the model's decision boundary must be inverted, causing the model to predict only one task correctly.

Furthermore, the class names may vary among the datasets the user uses. Because the dataset collectors assign the corresponding class names individually, datasets may have different class names even with the same samples. Altogether, the class and label inconsistency makes the tasks collide, even though those tasks share similar samples and decision boundaries. Figure 2 shows that the position and name of shared classes may differ among tasks, forcing the model to lose its accuracy toward previous tasks. This counters the goal of continual learning, which is maintaining prior knowledge.

3.2 Type 2: Biased/Noisy Samples

The class name or label may not accurately explain the data samples it holds due to dataset bias or noisy samples, causing the task transition to fail. Image datasets contain a bias that makes the model create an unintended decision boundary [27]. For instance, if a class *boat* only includes the images of a boat floating on the water, then the model might predict the class solely only the presence of water.

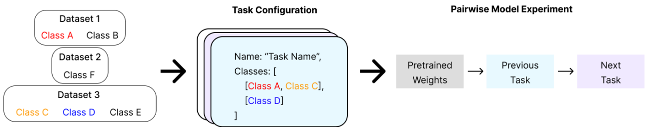

Figure 3: Internal process of Continual Learning Simulator for generating task transition data. After the user configures tasks, CLeVer automatically runs the model experiment for all task pairs.

Unlike previous works that treated bias as a cause of unreliable prediction, we consider bias as a problem that hinders fluent task transition. Bias may directly cause the transition between similar tasks to fail by distorting the decision boundary toward bias.

On the other hand, the data used in the training process are usually collected by the deployed edge device, which tends to include noisy samples [14]. Like the bias problem, noisy samples interrupt the model from learning solid decision boundaries during the training, ultimately degrading the transition performance. The case in Figure 2 exemplifies both the bias and noisy problem. The model may differentiate two classes based on other factors (Dogs vs Cats) even though the class names are the same, causing the former task's accuracy to decrease during the transition. Also, noisy samples (bird, flower) prevent the model from building correct decision boundaries.

3.3 Type 3: Diverse Class Scopes

Another crucial challenge in task transition is varying class scopes, which originate from the diversity of class granularity in datasets and applications. This variation can significantly impact the accuracy of previous tasks, particularly when classes are split or merged. For instance, as illustrated in Figure 2, splitting or merging classes between tasks can lead to a decline in the accuracy of the original class as part of the samples are reallocated to new labels. The reallocated samples cause the model performance to drop partially for the same reason as Section 3.1. Various datasets [8] and reviews [25] suggest that such variation in class scope is common practical applications, underscoring its significance.

4 System Design

CLeVer targets to automatically run model experiments and visualize its results to quickly detect the listed challenges in Section 3. Our system consists of a Continual Learning Simulator, which runs model experiments based on the user's task configuration, and a Task Transition Visualizer for analyzing the generated data.

4.1 Continual Learning Simulator

We first need to run experiments and extract its data to evaluate whether the continual learning model performs the task transition smoothly. The Continual Learning Simulator receives the configuration of tasks from the user and automatically runs the required model experiments. The process is briefly depicted in Figure 3.

```
name: "Animals",
super_classes: [
    {name:"Land", classes:[Cats, Dogs, ...] },
    {name:"Marine", classes:[Fish, Clams, ...] }
]
```

Form 1: Example configuration of classification tasks in JSON format.

4.1.1 Task Configuration

Users can easily configure the tasks their continual learning model should handle by filling in Form 1. According to the prior formulation of continual learning [28], we have assigned two essential attributes to each task: task name and the classes it needs to classify. The advantage of this configuration is that users can combine multiple classes from various datasets to build a superclass to represent a broader concept. For example, classes of *hair dryers*, *dishwashers*, and *televisions* can be merged into the superclass of *electronic appliances*. Users can create classes of any granularity using the classes from various datasets as building blocks.

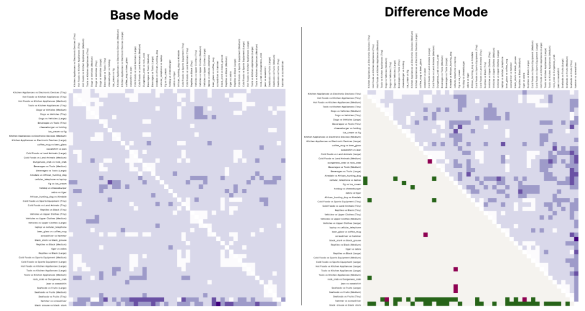

Figure 4: Transition Matrix View with (right) or without (left) the difference mode. The difference mode splits the screen into upper and lower triangles, showing different information. The upper triangle represents the average value, and the lower triangle represents the difference of symmetric cells. Red indicates the transition is more fluent from rows to columns, while green indicates the opposite. Using the matrix view, users can readily get a holistic insight into which task transitions are failing, with information on asymmetry.

4.1.2 Pairwise Model Experiment

After building the configurations of each task, CLeVer automatically runs the model experiments and exports its result to the Task Transition Visualizer (Section 4.2), following Section 5. However, simulating the whole training process of a continual learning model with a long task stream takes substantial time. Therefore, we store and reuse the pre-trained weights of our baseline model to mock the status after passing the prior task stream, where the weights include general knowledge of the passed task stream. Through this approach, we can reduce the time cost for model experiments while still being able to detect the challenges in task transition. Additionally, task transition is performed bi-directionally (Task A → Task B and Task A ← Task B) because the task transition performance may not be symmetrical for a single pair of tasks [1].

During the model experiment, we calculate and store the model's per-sample prediction results that will be used in our visualizer. We have simplified the metric designed for continual learning by Kemker et al. [13] to evaluate the task transition performance. Formally, the performance of the task transition is defined as:

$$TransitionFail_{prev,next} = \frac{Acc_{prev}(M_i) - Acc_{prev}(M_j)}{Acc_{prev}(M_i)} \tag{1}$$

Our metric calculates how much the accuracy for the former task has dropped during the transition. Also, following the baseline, our new metric normalizes the accuracy drop among all task transitions to keep them all on the same scale. In the metric, M is the classifier trained from time 0 to j. At time 0, the pre-trained weight is loaded, and training for $Task_{prev}$ starts until time i. After, the model is trained for $Task_{next}$ until time j. The test accuracy is abbreviated as function Acc with the corresponding task's test data.

4.2 Task Transition Visualizer

After the simulator generates the dataset (Section 4.1), the Task Transition Visualizer demonstrates how well the transition has been performed. The system has three hierarchical views: Transition Matrix, Task Pair Comparison, and Sample-Level Transition View. Figure 1 depicts the overall design of the Task Transition Visualizer.

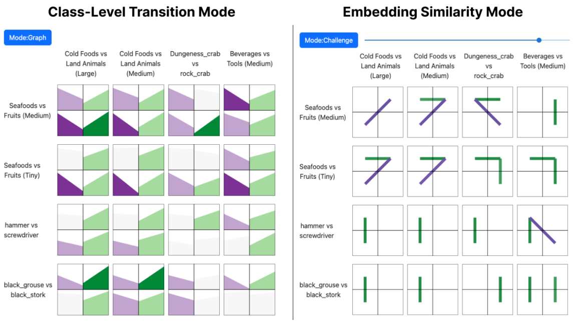
Class-Level Transition Mode
Mode:Graph
Cold Foods vs Land Animals (Large)
Cold Foods vs Land Animals (Medium)
Dungeness_crab vs rock_crab
Beverages vs Tools (Medium)
Seafoods vs Fruits (Medium)
Seafoods vs Fruits (Tiny)
hammer vs screwdriver
black_grouse vs black_stork
Embedding Similarity Mode
Mode:Challenge
Cold Foods vs Land Animals (Large)
Cold Foods vs Land Animals (Medium)
Dungeness_crab vs rock_crab
Beverages vs Tools (Medium)
Seafoods vs Fruits (Medium)
Seafoods vs Fruits (Tiny)
hammer vs screwdriver
black_grouse vs black_stork

Figure 5: The example of Task Pair Comparison View with Embedding Similarity Mode (right) and Class-Level Transition Mode (left). The Embedding Similarity Mode shows the line patterns to reveal the task transition challenges, while the Class-Level Transition Mode simplifies the transition graph.

4.2.1 Transition Matrix View

The Transition Matrix View (Figure 1A) utilizes a matrix to visualize the task transition fluency among all task pairs. The transitions can be understood as a dense, directed network between tasks. Therefore, we chose a matrix to show the holistic insights of transitions. Each row and column corresponds to the same task, where rows are previous tasks and columns are next tasks. The score to measure the task transition fluency is the normalized accuracy drop of the previous task, according to the definition of task transition challenges in Section 3. The darker color corresponds to a more substantial accuracy drop, which means poor transition performance. For users to more effectively reveal the transition failing items, the colors are binned into a certain number, which is also controllable through buttons. The matrix is reorderable in both directions, which groups the transition failing pairs. Our pilot study empirically found that the leaf order algorithm best grouped the tasks among the algorithms surveyed by Behrisch et al. [5]. When the **Difference Mode** is enabled, the matrix representation changes to show the difference between row-to-column and column-to-row transitions. As inFigure 4, the mode makes the left-down triangle represent the difference while making the right-up triangle show the average. This enables the users to determine which task transition needs to be performed in a certain direction to maintain high performance [1].

4.2.2 Task Pair Comparison View

The Task Pair Comparison View (Figure 1B) is generated if the user brushes cells on the Transition Matrix View. Firstly, the accuracy graph layout during the transition is visualized in the **Class-Level Transition Mode**. Among four squares, the second and third quadrants express the graph layout of each class from the previous task, and the first and fourth quadrants show the layout of classes from the next task. The layout abstracts the original accuracy graph by directly connecting the beginning and end of the transition graph with a straight line, forming a trapezoid. Through trapezoids, users can easily perceive and compare the slope of those graphs. The color is encoded with a diverging color map with five bins: purple for a steep accuracy drop and green for a rise. Users can easily detect tasks or classes that fall or rise suddenly through both angle and color encoding. Also, when many cells are brushed, leading to smaller trapezoids, users can still recognize the slope of the graph through its color.

If the user toggles the mode to the **Embedding Similarity Mode**, each cell shows purple and green lines, showing the similarity among the classes. In this mode, users can easily detect patterns corresponding to the challenges defined in Section 3. The diagonal lines are represented in purple to alert the user that the Type 1 or Type 3 challenge might exist, while other lines remain green. Type 1 challenge appears as a pattern of diagonal ╲ or crosses ╳, and Type 3 challenge appears as triangles ◹. Also, users can control the slider only to show lines with sufficient similarity over certain criteria. Besides diagonal lines, strong vertical lines represent the specific classification task's difficulty due to similar image samples. In contrast, vivid horizontal lines show that the task transition is easier due to the similar distribution of data samples between the tasks. The figure of mode change is described in Figure 5.

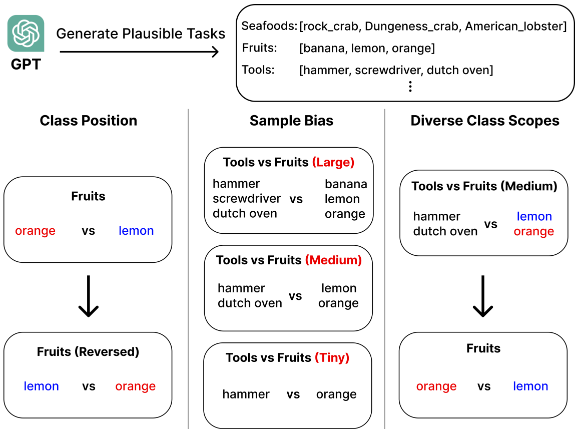

Figure 6: The process of generating tasks for case study. After GPT generates the list of plausible tasks, we generate challenging tasks based on three criteria: class position, sample bias, and class scopes.

4.2.3 Sample-Level Transition View

The Sample-Level Transition View (Figure 1C) appears when the user clicks on the specific task transition in the Task Pair Comparison View. Four large sections indicate the same section split as the Task Pair Comparison View's quantiles. The horizontal axis indicates the timestep where the transition is made, and the vertical axis corresponds to each data sample. This view consists of small and flat rectangles with colors that follow the encoding in Section 4.2.2, indicating whether the model correctly predicted the sample at a certain timestep. We adopted the square-filling representation for model predictions from Ren et al. [23], which also reveals the global trend while showing sample-wise results. The samples are ordered based on the exponential sum to form an increasing or decreasing shape. The ordering criteria are formulated as follows:

$$Prev_i = \sum_{j=1}^{T} 2^{-j} * Pred_{i,j}, \quad Next_i = \sum_{j=1}^{T} 2^{-(T-j)} * Pred_{i,j} \qquad (2)$$

In the criteria, $Prev_i$ and $Next_i$ indicate the i-th sample's score for reordering, and j indicates the timestep during the transition while having total T timesteps. Prev and Next orders differently due to which part it focuses on: first or last timestep of transition. $Pred_{i,j}$ becomes 1 for correct prediction while 0 for incorrect prediction.

5 Evaluation

We interviewed domain experts for design feedback and conducted the following case study to evaluate whether our design supports users' easy discovery of the task-failing process.

5.1 Data Generation

We first generated plausible real-world tasks, trained the model with task pairs, and calculated the embedding similarity before evaluating our visualization system.

5.1.1 Task Generation

Compared to classic works [15, 18], recent work suggested realistic datasets for testing continual learning [16]. Specifically, they captured the long-term technical improvements in daily objects such as cars and synthesized a benchmark dataset. Aligned to such work, we formed the baseline of our study dataset to simulate real-world situations, incorporating plausible task transitions. To do so, we leveraged the reasoning capability of Large Language Models [29] to generate a realistic task transition sequence. We asked ChatGPT [19] to generate likable image binary classification tasks in daily smartphone usage. Using appropriate prompting techniques (e.g. injecting expertise [30]), we generated ten base tasks that can be performed using the classes from ImageNet. Then, we generated 30 more challenging tasks following the three criteria: class position, sample bias, and diverse class scopes, exemplified in Figure 6.

5.1.2 Pairwise Model Training

For the experiment, we used EfficientNet-b0 [26] with pre-trained weights using ImageNet. We simulated the task transition for every pair, following Section 4.1.2. Each task pair was trained for 40 steps with a batch size 16. Also, an Adam optimizer with a constant learning rate of 0.001 was used throughout the experiment.

5.1.3 Embedding Similarity

We calculated embedding similarity between the classes so that the Task Pair Comparison View can visualize it. We used the pre-trained ResNet-18 model to generate the image embeddings. For simple calculation, the embeddings from the same class are averaged to obtain the embedding center. We then calculate the class similarity with the cosine distance between the centers of per-class embeddings.

5.2 Expert Interview

We interviewed two domain experts in machine learning and artificial intelligence, each with four years of experience. They reported their familiarity with continual learning. During the interview, participants freely interacted with CLeVer as we collected their opinions on system design, utilizing the think-aloud protocol.

Participants highlighted the effectiveness of matrix reordering and color bins in grouping poor transition pairs while suggesting richer color bin descriptions and highlighted symmetric cells in Difference Mode. The Task Pair Comparison View was praised for its ability to reveal transition challenges through distinct visual patterns, particularly in the Embedding Similarity Mode. Additionally, they could effectively identify noise or out-of-distribution samples in the Sample-Level Transition View. However, they suggested improving user interaction, such as making short lines easier to click and highlighting selected rows. These findings indicate that our system supports identifying and analyzing task transition failures and noisy samples. Also, based on feedback, we improved usability in Difference Mode and Sample-Level Transition View by highlighting selected cells and rows for a better user experience.

5.3 Case Study

5.3.1 Procedure

We also evaluated the usability of CLeVer through a case study, especially whether it supports users in detecting task transition failures. Three graduate students studying ML and visualization for 3 years participated in our study. We first explained the concept of continual learning and three challenges (Section 3) for 15 minutes. Then, the participants were asked to find the given challenges by searching for patterns described in Section 4.2.2 while using CLeVer for 15 minutes. Lastly, they were asked to report the identified transition failures and their strategies to find them.

5.3.2 Discovering Challenges through Patterns

All three participants successfully identified our intended patterns in the Task Pair Comparison View: diagonal or cross (╲, ╳), triangles (◹), which correspond to each challenge. Furthermore, participants discovered new patterns corresponding to certain cases. For instance, sometimes partial triangles (‾╲, ╲|) appeared instead of triangles, which depends on the distance between the image embeddings. We used ResNet for embeddings, which might have produced incorrect similarity compared to how humans perceive similar images [24]. Also, one of the participants reported horizontal lines (═) as an observed pattern corresponding to a fluent task transition.

5.3.3 Effective Sample-level Inspection

Participants also successfully identified noisy samples that might degrade the model performance in the Sample-Level Transition View, where samples with low accuracy float to the top due to Equation 2. For instance, one of the participants reported the class *beer glass* to include a noisy sample where a human holds a beer glass by clicking on the white rows appearing on the top. Furthermore, after clicking a few samples, they tried to group some samples with similar transition patterns and find the shared characteristics. For example, one participant reported three similar images in the *fig* class and called them "small, multiple figs," actively grouping noisy samples. This shows the possible effectiveness of per-group analysis of image samples, which will be further elaborated in Section 6.3.

6 Discussion

In this section, we discuss our system design's adaptability to various image-based deep-learning tasks and robustness in environments with many tasks. We also suggest sample group analysis as our future work to support user interactions better.

6.1 Flexible Design

We claim that CLeVer can be extended to various usage scenarios and task types. Firstly, CLeVer can be used for various computer vision applications [3] with different metrics for their performance, including object detection (IoU), semantic segmentation (Pixel Accuracy), and image generation (CLIP [21] score). The Transition Matrix View can show various metrics with the same color encoding scheme. Also, the data shown in the Sample-Level Transition View can be easily replaced with proper representations like bounding boxes or generated images. Furthermore, we may apply our system design for multi-class classification tasks for devices with sufficient screen ratios. The size of 2x2 cells in Task Pair Comparison View and Sample-Level Transition View may be expanded into MxN, which is the number of classes. In such circumstances, our two modes in Task Pair Comparison View can show significant patterns through the purple-green color encoding and interactive slider. Evaluating the extended design's usability for various tasks with sufficient participants would prove its effectiveness.

6.2 Scalability

Most previous research in continual learning experimented with a stream of at most ten tasks [7], while some experimented on a large stream of 200 tasks [6]. Our system can also be effective for large task spaces, which may be required in future applications. As proved in the case study, our matrix-based design in a Transition Matrix View can effectively present the results of many tasks through color binning. Also, the required data for the Task Pair Comparison View and Sample-Level Transition View are fetched from the server when the user brushes the area, leading to a memory-efficient system. Furthermore, CLeVer independently trains the model for each task pair, enabling an easy parallelization of the data generation.

6.3 Sample Group Analysis

The sample-level bias/noise inspection in CLeVer is made by clicking on each image sample. However, our case study showed that grouping image samples based on similar transition patterns could give users deeper insights into the distribution of images. Users can establish clearer high-level concepts about biased or noisy samples by forming problematic image groups, often called slices [9]. These understandings will further support users in deciding how to control the sample quality and fairness for fluent task transitions.

7 Conclusion

In conclusion, our system CLeVer properly addresses task transition failures in continual learning models. By integrating a user-friendly interface with sophisticated visual analytics, CLeVer enables efficient model experiments, allowing users to easily detect and analyze task transition challenges. Also, our interview with domain experts and a qualitative case study demonstrates the system's effectiveness. Its flexible design and adaptability to various computer vision tasks underscore its potential to enhance the performance of AI in much broader devices. Also, our newly classified challenges in task transition may be extended in future research, ultimately forming a guideline for ML engineers. This work lays a foundational step for future research in practical deployment scenarios, highlighting the importance of continual learning in real-world applications.

Acknowledgments

We sincerely thank our reviewers for their valuable feedback and our study participants for their indispensable insights and contributions. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2023R1A2C200520911).

References

[1] S. J. Bell and N. D. Lawrence. The Effect of Task Ordering in Continual Learning. 5 2022.

[2] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[3] J. Chai, H. Zeng, A. Li, and E. W. Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, 2021.

[4] A. Chatzimparmpas, R. M. Martins, I. Jusufi, K. Kucher, F. Rossi, and A. Kerren. The state of the art in enhancing trust in machine learning models with the use of visualizations. In *Computer Graphics Forum*, vol. 39, pp. 713–756. Wiley Online Library, 2020.

[5] M. Davari, N. Asadi, S. Mudur, R. Aljundi, and E. Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning, 2022.

[6] M. Davari, N. Asadi, S. Mudur, R. Aljundi, and E. Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16691–16700, 2022. doi: 10.1109/CVPR52688.2022.01621

[7] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 7 2022. doi: 10.1109/TPAMI.2021.3057446

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

[9] G. d'Eon, J. d'Eon, J. R. Wright, and K. Leyton-Brown. The spotlight: A general method for discovering systematic errors in deep learning models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1962–1981, 2022.

[10] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia. A brief review of domain adaptation, 2020.

[11] Y. He, Z. Huang, and B. Sick. Design of explainability module with experts in the loop for visualization and dynamic adjustment of continual learning. *arXiv preprint arXiv:2202.06781*, 2022.

[12] M. Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14. IEEE, 2014.

[13] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[14] C. D. Kim, J. Jeong, S. Moon, and G. Kim. Continual learning on noisy data streams via self-purified replay. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 537–547, 2021.

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13), 2017. doi: 10.1073/pnas.1611835114

[16] Z. Lin, J. Shi, D. Pathak, and D. Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 2)*, 2021.

[17] B. Liu. Lifelong machine learning: a paradigm for continuous learning. *Frontiers of Computer Science*, 11(3), 2017. doi: 10.1007/s11704-016-6903-6

[18] A. Madotto, Z. Lin, Z. Zhou, S. Moon, P. Crook, B. Liu, Z. Yu, E. Cho, P. Fung, and Z. Wang. Continual learning in task-oriented dialogue systems. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7452–7467, 2021. doi: 10.18653/v1/2021.emnlp-main.590

[19] OpenAI. Chatgpt. `https://www.openai.com/blog/chatgpt`, 2023. GPT-4 [November, 2023].

[20] 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. doi: 10.1109/TVCG.2017.2744358

[21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

[22] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning, 2017.

[23] 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, 2016.

[24] B. D. Roads and B. C. Love. Enriching imagenet with human similarity judgments and psychological embeddings. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 3547–3557, 2021.

[25] C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data mining and knowledge discovery*, 22:31–72, 2011.

[26] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

[27] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pp. 1521–1528, 2011. doi: 10.1109/CVPR.2011.5995347

[28] L. Wang, X. Zhang, H. Su, and J. Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application. 1 2023.

[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[30] B. Xu, A. Yang, J. Lin, Q. Wang, C. Zhou, Y. Zhang, and Z. Mao. Expertprompting: Instructing large language models to be distinguished experts, 2023.

[31] H. Zeng, H. Haleem, X. Plantaz, N. Cao, and H. Qu. Cnncomparator: Comparative analytics of convolutional neural networks. *arXiv preprint arXiv:1710.05285*, 2017.