Eurographics Conference on Visualization (EuroVis) 2026
N. Andrienko, M. Chen, and B. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 3

INFICOND: Investigating Interactive No-code Fine-tuning with Concept-based Knowledge Distillation

J. Huang[1], W. He[2], L. Gou[2], L. Ren[2], and C. Bryan[1]

[1]Arizona State University, United States, [2]Bosch Research North America, United States

Abstract

Knowledge distillation is a widely used technique whereby knowledge from large pre-trained models is transferred into smaller student models. However, this process is non-trivial and traditionally requires technical and theoretical expertise in AI/ML. We investigate a visualization-driven strategy for making this process more accessible and intuitive via developing INFICOND*, a novel tool that leverages visual concepts to scaffold the knowledge distillation process and support subsequent no-code fine-tuning of student models.* INFICOND*'s backend pipeline extracts text-aligned visual concepts and constructs highly interpretable student models; its frontend supports interactively fine-tuning these student models by directly manipulating concept influences. Empirical evaluations help validate that* INFICOND *effectively supports knowledge distillation and subsequent fine-tuning workflows. We additionally discuss insights and lessons learned about how human-in-the-loop and visualization-driven approaches like* INFICOND *can support accessible and adaptable AI explainability and model distillation.*

CCS Concepts

• ***Human-centered computing*** → *Visualization systems and tools;* • ***Computing methodologies*** → *Artificial intelligence;*

1. Introduction

Recent advancements in large pretrained models (PTMs) have significantly enhanced their performance across diverse computer vision tasks [WCQ*23]. However, the size and resource demands of PTMs can limit their practical deployment and use, particularly in resource-constrained environments [LGX*24].

Knowledge distillation (KD) has emerged as a promising strategy for addressing these constraints, whereby knowledge is transferred from a large "teacher" PTM to a more compact "student" model that still maintains good predictive performance [GYMT21]. However, the KD process (including fine-tuning student models after initial distillation) is challenging for several reasons. For one, the distillation of large and complex PTMs is often a technically difficult and opaque process, and student models can lack interpretability, which makes it difficult to explain what knowledge has been transferred and understand how to refine it [AC24].

Recent work has explored the use of visual concepts as a way to explain transferred knowledge [CRCZ20, ZCCR22] and support transferring knowledge to smaller models [CZW*22, SMB*22]. Concept-based methods can improve interpretability, but current approaches primarily rely on automated concept extraction pipelines that generate large ensembles of concepts in an unsupervised manner [SMB*22], resulting in a significant percentage of the concepts being ambiguous or irrelevant [HJGR23]. Current methods also do not provide mechanisms for human review (e.g., to understand and confirm student model behaviors align with human expectations) or fine-tuning (e.g., to correct irrelevant concepts that might negatively influence predictions in student models), which motivates a need for human-in-the-loop (HITL) approaches.

In parallel to such activities, there is a growing class of stakeholders (such as domain practitioners and applied researchers) who, despite not having formal training or technical expertise in model architectures and the programming aspects of AI/ML and KD, are conducting distillation workflows [CH19, SKKS22, EDS*24]. Such users have domain expertise, but lack accessible and interpretible mechanisms to inspect what knowledge has been distilled, diagnose instances where student and teacher models misalign, and refine student models without having to write low-level code.

Specifically for the current paper, we investigate how to leverage visual concepts in KD workflows to support (i) interpretable representations of distilled knowledge, (ii) diagnosing issues (e.g., underperformance and misaligned concepts), and (iii) interactive, no-code fine-tuning, suitable for these types of users. To accomplish this, we design and evaluate a novel software tool called INFICOND that supports concept-driven KD from large PTMs into student models and enables the subsequent review and fine-tuning of these models. INFICOND's pipeline (see Figure 1) leverages CLIP [Ope] to extract and label visual concepts for downstream tasks (specifically, we focus on multi-label image classification as an application scenario). INFICOND's frontend (see Figure 5) is a coordinated visualization interface that supports explanation and fine-tuning workflows, including exploring underperforming sub-

sets in student models, examining their concept bases to identify causes of underperformance, and interactively manipulating concept weights to improve student models.

To support such workflows, we adapt and integrate a set of backend processes and algorithms including: (i) A model-agnostic method for distilling knowledge from PTMs into interpretable student models, promoting generalizability. (ii) A pre-processing step that converts images into interpretable concept-based vectors, externalizing the PTM's knowledge. (Student models are trained using these vectors to mimic the teacher model's predictions, reducing capacity demand while enhancing the student's ability to replicate the teacher's predictions.) (iii) A concept tuning algorithm and interaction approach that allows users to fine-tune and re-train student models without coding. Specifically, INFICOND allows users to specify concepts that they want to uptune or downtune in its interface, adjusts them in the backend, and uses the adjusted weights to serve as new initializations to fine-tune the student model for a small number of epochs, effectively and efficiently adapting the model based on user instructions.

INFICOND's contributions include integrating these components into a coherent, interpretable, HITL workflow that supports accessible, no-code, and concept-driven analysis and fine-tuning. In particular, this workflow is effective not just for creating and improving performant student models, but also verifying that student model behaviors are properly aligned with human preferences and mental models. We robustly evaluate INFICOND, including via conducting a mixed methods user study, demonstrating the efficacy of our approach both from a task-based and usability perspective, and additionally discuss and reflect on broader insights and lessons learned, such as how visualization-driven tools like INFICOND can generalize to more diverse computer vision tasks and scenarios.

2. Background and Related Work

2.1. Knowledge Distillation

Knowledge distillation (KD) [GYMT21] is the process of transferring knowledge from a large teacher model to a more compact student model. KD is particularly valuable when deploying a PTM in a given environment is infeasible, due to reasons such as computational or other resource constraints, long inference times, etc. [GYMT21]. As such, KD has emerged as a strategy for deploying student models across a number of real-world devices (e.g., edge devices that are memory constrained) and application areas, including autonomous driving, medicine and healthcare, mobile devices, and smart homes [AAA21, GYMT21]. Broadly, KD methods can be categorized across three facets: the types of knowledge that are transferred, the strategy for distilling such knowledge, and the architectures of teacher and student models.

(i) The major *types of knowledge* transferred during KD are response-based, feature-based, and relation-based. Response-based knowledge consists of the logits or prediction results from the teacher model, and the distillation involves training the student model to mimic the teacher's outputs [HVD15]. Feature-based knowledge requires the student model's feature map to approximate that of the teacher [KPK18]. Relation-based knowledge aims to align the similarity between pairs of feature maps in the student model with those in the teacher model [LKS18]. While feature-based and relation-based knowledge have generally been shown to result in better performing student models, their reliance on access to teacher's internal information makes them less broadly applicable and explainable [PTT20, JPW*19]. In contrast, response-based knowledge is model-agnostic and thus generally considered more broadly generalizable [GYMT21].

(ii) The three primary *distillation strategies* are offline, online, and self-distillation. Offline distillation uses a static, pre-trained teacher model to guide the student [HVD15], while online distillation involves training the teacher concurrently with any student models [MFL*20]. Self-distillation is a specific type of online distillation that uses the same architecture for both the teacher and the student [ZSG*19, HMLL19].

(iii) The design of the *teacher-student architecture* presents significant challenges due to the "capacity gap" — limitations of a simpler model's ability to approximate the complex mappings of a more advanced model [GYMT21]. Careful design is essential to effectively enhance the student model's capacity. For example, recent findings suggest that providing a more graduated learning process (e.g., via fine-tuning) or a pre-processing step as teacher assistant for the student model can help bridge this gap [MFL*20].

For the current paper, INFICOND supports offline and response-based knowledge transfer, as we train a student model to mimic the teacher model's logits. This avoids reliance on teacher model information and architecture, making the approach more generalizable. To improve the student model's interpretability and facilitate fine-tuning, we design the student as a set of one-layer linear models (see Section 4.3 for discussion of the advantages and constraints of this strategy). To address the capacity gap, we both devise a pre-processing step and support graduated learning. For pre-processing, the INFICOND pipeline maps images into concept-based interpretable vectors (see Figure 1). The student model learns how to predict logits from these vectors, effectively reducing the task complexity. The details of this process are discussed in Section 4. Graduated learning is supported by INFICOND's frontend interactivity and a concept-tuning algorithm that updates and re-trains students models, described in Section 4.5. Please also see the discussion in Section 7 where we reflect on how tools like INFICOND can be adapted for other types of KD scenarios and methods such as feature-based knowledge transfer.

2.2. Visual Analytics and Knowledge Distillation with Concept-Based Methods

Concept-based methods have emerged as an explainability technique for deep learning models by quantifying the influence of human-understandable concepts on the model's predictions [HHS*22]. In particular, concept-based methods have been shown to overcome some of the limitations of traditional perturbation and pixel-based approaches (e.g., [FV17]) such as a lack of scalability (concepts can be quantified for individual samples, an entire class of instances, or even the entirety of a dataset [HMKB22]) and reliability, and they offer explanations that better align with human intuitions and mental models [KWG*18].

Recently, a number of visual analytics applications have uti-

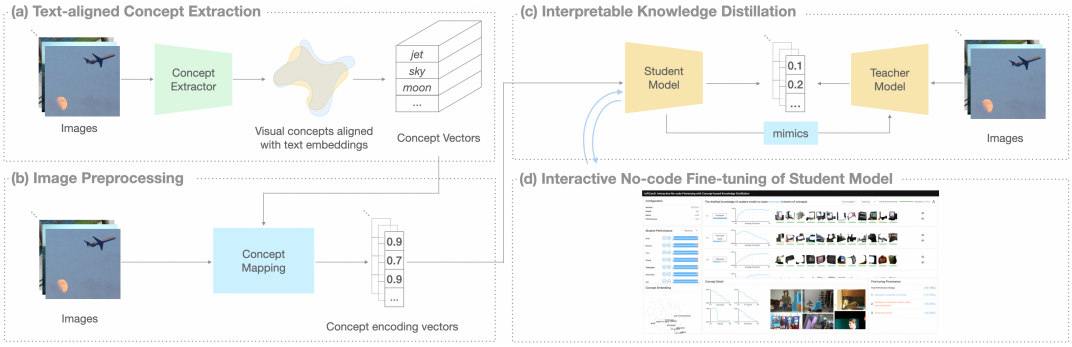

Figure 1: INFICOND*'s pipeline (a) extracts text-aligned visual concepts, (b) maps images into concept-based interpretable vectors, (c) trains linear student models that based on these vectors guided by teacher logits, and (d) employs a visual analytics interface to visualize these concepts for the student model, emphasizing those with high influences to facilitate model analysis, explanation, and fine-tuning.*

lized concepts to support HITL deep learning-related tasks, including data labeling, exploration, analysis, and explanation of image classification models [HHS*22, WLG23, ZXSR21, HMKB22, PDD*21, HPRC19]. For example, for data generation, ConceptExtract [ZXSR21] and Visual Concept Programming [HHS*22] developed interactive approaches to improve concept extraction and support scalable image labeling, via a combination of semi-automated techniques, human interaction, and visual analytics. For model understanding, Drava [WLG23] takes a data-centric approach, generating interpretable concepts and enabling concept-based data exploration. NeuroCartography [PDD*21] and ConceptExplainer [HMKB22] adopt a model-centric approach, with the former summarizing and visualizing learned concepts using neuron clustering and embedding, and the latter enabling structured exploration of a deep learning model's concept space to provide insights into model behaviors across various data granularities.

These above tools focus primarily on using visual concepts for general-purpose PTM workflows, and do not consider KD. However, there is emerging work showcasing the potential of concepts for KD, including for explainability [CRCZ20, ZCCR22] and scaffolding the KD process [GWT20]. For this latter point, by assigning student model weights to concepts in a manner that approximates the teacher model, the student model can more effectively acquire the conceptual knowledge necessary for the classification task at hand [LCS*20]. However, existing concept-based approaches for KD face several challenges:

For one, traditional concept generation approaches require manually annotating all relevant concepts, but many KD scenarios contain a large potential concept space (hundreds or thousands of concepts may need to be annotated), making for a tedious and error-prone labeling task. Although automatic concept extraction methods exist (e.g., [GWZK19]), they often do not provide associated labels for the extracted concepts and offer no control over the concepts that are being extracted (leading to the creation of many ambiguous or irrelevant concepts). These methods also cannot leverage existing concept corpora to guide the extraction process, which further limits their potential usefulness. Moreover, the absence of intuitive, human-centered techniques to explain and fine-tune student models created during KD hinders the full potential of a concept-based approach, particularly for the emerging users (domain practitioners, applied researchers) described in Section 1.

INFICOND addresses these challenges as follows: (i) Instead of relying on existing automated methods and manual labeling to generate concepts, we leverage an established knowledge corpus [HJGR23] to bootstrap a robust potential space for concepts, and develop a CLIP-based method to extract text-labeled visual concepts according to this list. This approach ensures that the extracted concepts are both interpretable and aligned (i.e., not ambiguous or irrelevant) with the desired domain knowledge for the task where we want to perform KD. (ii) Unlike prior works that use feature-based knowledge transfer (e.g., [GWT20]) and thus require internal knowledge about the teacher model, we utilize a response-based approach to train the student model to assess concept weights. This avoids dependency on teacher model knowledge, making our approach more generalizable, computationally tractable, and applicable to a wider range of models. (iii) We introduce a no-code concept tuning method that allows users to fine-tune the student model by specifying concepts that need to be uptuned (increasing influences) or downtuned (decreasing influences), allowing student models to be tailored to specific needs and use cases without requiring low-level programming.

3. Formalizing Design Challenges and Goals for INFICOND

Design Challenges. As mentioned in Section 1, we are interested in investigating how to make the KD process more intuitive and explainable, by designing and implementing the INFICOND system to support interpretable and fine-tunable student models. To help scaffold this, we first conducted a meta-analysis of the papers discussed in Section 2, and distilled a set of four design challenges that make this process non-trivial.

(C1) How do we balance interpretability and effectiveness? Many existing KD methods (e.g., [ZK16, KPK18, PTT20]) prioritize student model performance at the cost of introducing complex architectures. This not only hinders adoption [GYMT21], but it negatively impacts downstream analysis and fine-tuning processes such as identifying the causes of underperformance, devising improvement strategies, and executing them effectively [WY21]. Conversely, "simple" student models can increase interpretabil-

ity but also widen the capacity gap to the teacher model, making it challenging to transfer the desired knowledge effectively [MKH19]. *Striking an appropriate balance between student model effectiveness and interpretability is crucial for successful KD and fine-tuning processes.*

(C2) How do we ensure generalizability? While using more information from the teacher model can lead to better student model performance, it can also make the KD method heavily dependent on the specific teacher model architecture, which can hinder its generalizability [HZZ*24, SNCH21]. Model-agnostic strategies [ZG18, SC24, HYW*22] have been proposed to overcome this, but they come with their own challenges — for example, only the input and output of the teacher model (and any information derived from them) can be used to train the student model. This can significantly limit the effectiveness of KD [GYMT21], especially when there are additional requirements for interpretability. *Developing an solution that is model-agnostic (and thus generalizable across a variety of PTMs) while still highly performant is a non-trivial challenge, requiring careful consideration of the trade-offs between generalizability and performance in the context of interpretable knowledge distillation.*

(C3) How do we efficiently analyze and explain student models? Once the KD process is complete, users must be able to subsequently analyze any generated student models to ascertain their performance and efficacy. This can involve identifying and tracing the root causes of current underperformances (e.g., [ZLX*22]) and learning how to address or fix these issues [RLN*23]. Designing a user experience that effectively supports this process is non-trivial, requiring careful consideration of the insights that should be revealed to the user and the types of interactions and tasks that should be supported. Moreover, since there are increasingly stakeholders performing KD that are not traditional "AI/ML experts," *we need to balance between information depth (or complexity) and overall usability to ensure that users can effectively analyze, interpret, and explain student models.*

(C4) Can we interactively fine-tune student model? As users analyze and come to understand their student models, they may wish to fine-tune them. Traditional fine-tuning practices are largely manual and technical, and require writing custom programming code to modify the model (e.g., [MFL*20]). As discussed in Section 1, this hinders the accessibility and adoption of KD, particularly for "domain users" who might intuitively understand how a student model *should* perform but lack the technical expertise to implement it in code. This also leads to inefficiencies such as time spent writing the code and introducing (and subsequently tracking down) bugs or unoptimized logic [DJK23]. In contrast, no-code approaches have shown promise in similar "general purpose" AI/ML scenarios (e.g., [EDS*24]). However, designing such a solution has its own challenges. Similar to **(C3)**, *the user experience must balance factors such as efficiency, expressiveness, and usability, while still supporting the fine-tuning process.*

Design Goals. To address the challenges outlined above, we next distilled six key design goals, which are intended to support the development of a concept-based KD framework that facilitates interpretable analysis and no-code fine-tuning of student models.

(G1) Highly interpretable student models. To support explainable and understandable KD **(C1)**, the student models themselves should be highly interpretable.

(G2) Highly effective student models. Unfortunately, as discussed in Section 2, interpretability can come at the cost of inhibiting performance and increasing the capacity gaps between teacher and student models. As such, student models should be not only interpretable, but also highly performant **(C1)**.

(G3) Automatic extraction and labeling of interpretable concepts. One way to support interpretability **(C1)** is by leveraging visual concepts to understand student model decisions. Automated approaches (e.g., [GWZK19]) show promise, but do not support automated labeling of concepts during extraction, making them unsuitable for our purpose. To address this, we need to an automated process that extracts concepts while also assigning clear and interpretable meanings.

(G4) Generalizability through model-agnostic training. To address the challenge of generalizable KD **(C2)**, we aim to employ a response-based knowledge transfer method. Thus, any created student models will learn to replicate the teacher's input-output behavior, and not depend on model-specific information about the teacher model [AK21].

(G5) Efficient analysis of student models. To support efficient student model analysis **(C3)**, we can employ a visual analytics approach. Specifically, this approach can emphasize two common analysis tasks: (i) First, we surface insights about underperforming subsets in student models, providing users with clear indications of where student models might fail or exhibit higher capacity gaps. (ii) Second, we show how the student model's performance changes in response to fine-tuning actions (discussed below in **(G6)**).

(G6) Concept-based interactive fine-tuning. Goal **(G5)** can also be an initial step in *improving* the student models **(C4)**: once the user understands where the student models are non-performat (or misalign with human intuitions), they can fine-tune and re-train them to fix the problem. For this, we can develop a workflow that accepts user-specified tuning instructions based on visual concepts. The system can use these to interactively fine-tune and update the model.

4. Methods

Based on the design goals outlined in Section 3, we develop the INFICOND framework (shown in Figure 1). The process begins with the backend pipeline generating a set of text-aligned visual concepts **(G3)**. To support automated concept extraction and labeling, we employ a robust concept corpus [HJGR23] and develop a CLIP-based extraction method [HJGR23] that aligns extracted concepts with the relevant labels in the corpus. Next, each image is mapped into a vector that quantifies the presence of the set of concepts in the image, effectively capturing its critical information. INFICOND then distills knowledge from a teacher model into linear student models by training the student models to predict logits based on the vectorized images (in other words, the student models focus on learning the mappings between concept vectors and logits, simplifying the complexity of the learning task). To further support interpretable and generalizable student models **(G1, G4)**, we

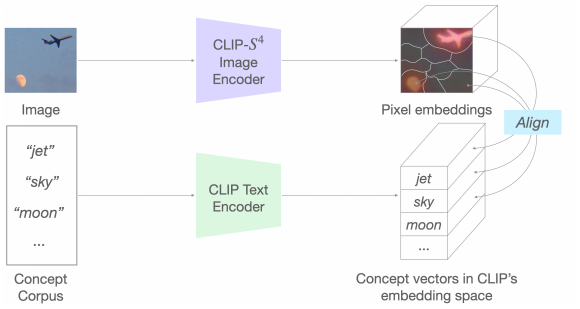

Figure 2: *Our text-aligned concept extraction process employs CLIP-S^4, trained to align pixel embeddings (image segments) with text embeddings, to extract text-labeled visual concepts.*

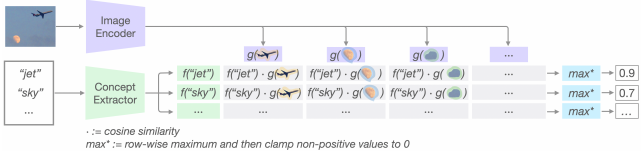

Figure 3: *To perform concept mapping on an image, we compute the pairwise cosine similarity between its segment embeddings and concept vectors, and use the maximum cosine similarity value among all segments to quantify a concept's degree of presence in the image.*

generate an ensemble of single-layer linear models via a response-based knowledge transfer approach, each responsible for specific subtasks.

In INFICOND's frontend, a visual analytics interface supports analyzing **(G5)** and no-code fine-tuning **(G6)** the student models, whereby users can impute training constraints that update the student model's parameters. Below, we describe the primary backend components and processes in INFICOND; the frontend is discussed in Section 5.

4.1. Application Scenario and Teacher Model

We demonstrate INFICOND using multi-label classification, which represents a challenging, complex, and widely applicable AI task with high-stakes real-world applications such as airport security inspection [VCHS18]; it is also an active research area for KD [YXZ*23]. In multi-label classification, images can have multiple labels, meaning an image can contain many different objects. Due to imbalances in class prevalence, achieving high accuracy across all labels (especially the less common ones) is often non-trivial and requires sophisticated models [ZQL*23].

For a teacher model, we use Q2L [LZY*21], which is a multi-label classification model that has achieved state-of-the-art performance across multiple multi-label classification benchmarks, including ranking #1 on NUS-WIDE [CTH*09], PASCALVOC 2007 [EVGW*a], PASCALVOC 2012 [EVGW*b], and #5 on MS-COCO [LMB*14]. This model consists of a backbone for spatial feature extraction and multiple concatenated transformer decoders for query and pooling mechanisms. In particular, the architectural complexity of Q2L also makes it challenging for resource-constrained scenarios, making it an apt PTM for KD.

4.2. Text-Aligned Concept Extraction

INFICOND uses visual concepts as a key strategic feature to achieve its design goals. However, as discussed above, existing automated concept generation techniques primarily generate concepts from images as unsupervised pixel patches or embeddings, and lack human-interpretable labels or notations to describe these concepts. To address this, we employ a process that extracts and quantifies text-aligned visual concepts from a given set of images.

This process involves two main steps, and is illustrated in Figure 2. First, we use a CLIP-based image encoder to segment images. Specifically, we utilize the CLIP-S^4 image encoder, which is fine-tuned from CLIP's default image encoder to support semantically consistent segmentation [HJGR23]. Pixels within the same segment exhibit similar embeddings, while those in different segments show significantly different embeddings.

Simultaneously, we encode a concept corpus using CLIP-S^4's text encoder. This corpus contains a 584-word list of concepts derived from [BZK*17], which includes sources such as PASCAL Context [MCL*14], DAVIS 2017 [PPTM*16], COCO-Stuff [CUF18], and more. This process results in 584 vectors, each representing a distinct concept. (Notably, this process only needs to be performed once for a concept corpus and image set. This process is also extensible to support other concept corpuses, see Section 7 for discussion.)

For each image segment, we assign a label by identifying the concept vector that has the highest cosine similarity to it. Subsequently, for each concept text, we aggregate the best-matching segments. This process yields a list of text-aligned visual concepts, where each concept is represented by a pair consisting of (i) segment(s) and (ii) corresponding descriptive text.

4.3. Concept Mapping

As discussed above, the architectural simplicity of single-layer linear student models has potential advantages for interpretability, however this can come with performance risks and capacity gaps relative to the teacher model. To mitigate this, this step maps the explicit knowledge that needs to be distilled between the teacher and student models using the above collection of extracted text-aligned concepts. By focusing the student model's responsibility on mapping the interpretable vectors to output logits, a "simple but interpretable" linear student architecture becomes tractable (see Section 7 for further discussion). This shifting of the "heavy lifting" to a pre-processing step is a common strategy in KD, known as introducing a teacher assistant [MFL*20].

For multi-label image classification, the process involves mapping images to interpretable vectors, as illustrated in Figure 3. We leverage the segmentation and concept vectors derived during the text-aligned concept extraction phase. For each image, we consider its segments and compute the cosine similarity between each concept and all segments from the image. The highest cosine similar-

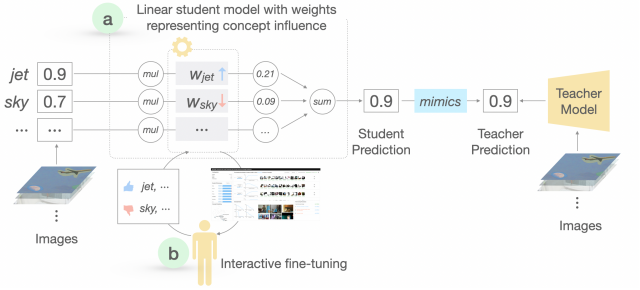

Figure 4: *(a) A linear student model is trained for each class to imitate the teacher model's predictions. The student models consist of a single fully connected layer without activation functions. Each neural connection from an input node to the output node represents the influence of a concept on that specific class. (b) Users can provide instructions specifying which concepts to tune and how to adjust their importance (i.e., increase or decrease).*

ity value represents the degree to which the concept is present in a part of the image. If the cosine similarity is less than 0, it indicates that no part of the image is similar to the concept, and the concept's presence in the image is considered to be 0. Conversely, if the cosine similarity is greater than 0, the concept is deemed to be present in the image to some extent, quantified based on the cosine similarity value. Upon completion of the concept mapping process, each image is transformed into a vector (specifically of length 584, corresponding to the size of the concept corpus), where the i^{th} dimension of the vector represents the presence of the corresponding i^{th} concept in the image. Importantly, this activity only needs to be performed once as a pre-processing step, minimizing the overall computational overhead and supporting downstream fine-tuning workflows with student models.

4.4. Model-Agnostic Knowledge Distillation

The next step is to distill knowledge from the teacher model using a response-based training scheme [HCL*22]. Here, the teacher model's outputs guide the training of the student models. In the context of multi-label classification, the teacher model generates a multi-dimensional vector where each dimension corresponds to a logit representing the likelihood of a particular class being present in an image. The teacher model produces a k-dimensional vector, indicating the probabilities of k classes (in our tested application scenario, there are $k = 20$ classes available). To support both interpretability and effectiveness, we generate an ensemble of k student models, with each model dedicated to predicting a single logit. This approach has previously been shown to be an effective strategy for distilling "simple but effective" student models that do not enlarge the capacity gap [ZG18].

As shown in Figure 4, the student models receive the 584-dimensional vectors as input, where each vector represents an image characterized by the presence of 584 potential visual concepts. Each student model maps this 584-dimensional vector into a 1-dimensional logit. Consequently, the optimization process involves only 584 parameters, resulting in a highly interpretable and easily optimizable student model. In our case, the training process consists of 10 epochs and can be completed in ∼30 seconds.

Across the k trained student models, the weights of the j^{th} model quantify the influences of each concept on the final prediction of class j. These weights essentially represent the knowledge being transferred from the teacher model to the student model [YYAX23]. Abstractly, we can view each student model as a learned assignment of weights to the concepts based on their presences to predict the occurrence of a particular class.

During the training process, we employ a binary cross-entropy loss function to measure the discrepancy between the teacher model's output logit and the student model's output logit. To promote sparsity and avoid dense weights where all concepts have small yet non-zero weights, we incorporate L1 regularization. The loss function is formally defined as:

$$L = BCE(y_{student}, y_{teacher}) + \lambda ||W_{student}||_1 \quad (1)$$

$BCE(\cdot,\cdot)$ is binary cross-entropy, $y_{student}$ and $y_{teacher}$ are student and teacher logits, $||W_{student}||_1$ is L1 regularization on student weights, and λ is a hyperparameter to control how significant the sparsity penalty is, which is empirically determined as 10^{-4}.

4.5. Fine-Tuning Student Models via Concept Tuning

The interpretable nature of the student models is intended to promote explainability and fine-tuning. Since the weights of a student model directly represents the influence of concepts on its particular class, fine-tuning a model to improve its performance involves modifying these weights. To analyze a student model, users can determine which weights to adjust. For example, in cases where concepts are considered detrimental, irrelevant, or misaligned to human preferences or intuitions, fine-tuning can minimize their influence in the student model's predictions. However, due to the student model's convergence to a local minimum, merely increasing or decreasing a concept's weight as a hard constraint would not yield substantial changes.

To address this, we develop an approach that utilizes the user's specification of concept uptuning or downtuning as soft constraints. Similar to tools such as BEAMES, which support interactively up or downweighting the features in regression models [DCCE19], INFICOND's interface allows users to review and select concepts that are deemed important for classification (e.g., uptuning a `flowerpot` concept for the *pottedplant* class) and concepts that are considered irrelevant (e.g., downtuning `autobus` for the *dining table* class). Based on user interactions, we update the student model weights accordingly. For concepts that users identify as important, we establish a lower bound threshold for their weights and ensure that the weights never fall below this threshold during the training process. Conversely, for concepts that are considered irrelevant, we set an upper bound and guarantee that the weights never exceed this threshold. The lower and upper bounds are determined by scaling the original weights by a factor x (these variables and thresholds may be adjusted in a configuration file). Users can provide instructions and fine-tune the students iteratively, maintaining

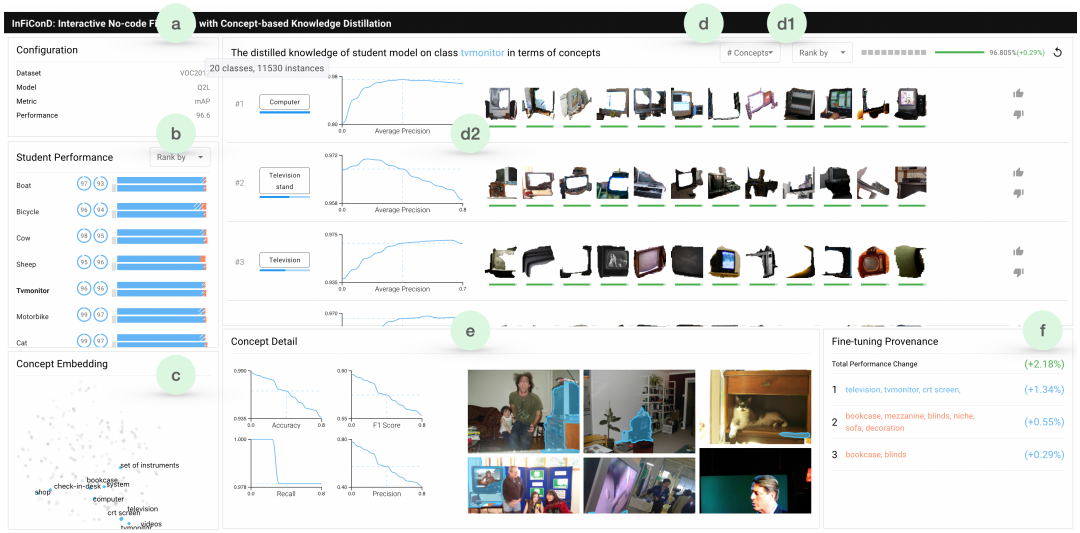

Figure 5: INFICOND*'s interface consists of six coordinated views: (a) configuration view, (b) student performance view, (c) concept embedding view, (d) student knowledge view, (e) concept detail view, and (f) fine-tuning provenance view, described in detail in Section 5.*

a history of prior instructions. This instruction history is incorporated into the fine-tuning process to prevent the weights of previously selected concepts from surpassing the established thresholds. The teacher model is accessed and student model re-training follows the same setup as the initial model training, but with a reduced number of epochs to optimize computational resources. As fine-tuning iterations occur, the results are updated on INFICOND's frontend, allowing users to evaluate in real time any performance improvements and perform further fine-tuning iterations.

5. Interface

INFICOND's frontend (Figure 5) features six coordinated panels, described below. A demo video (in supplemental materials) also shows examples of the described features and interactions.

The **configuration view** (Figure 5a) allows users to select and overview a dataset, the currently loaded student model, and performance metrics. Hovering over items reveals additional details on demand. For example, hovering over the dataset name displays the number of instances and classes within the dataset.

The **student performance view** (Figure 5b) summarizes each student model's performance, emphasizing subsets where the student and teacher models disagree. Each student model corresponds to a specific class, sorted by its performance gap against the teacher model measured in average precision (AP) by default, though models can also be ranked by additional common metrics including recall, precision, and F1 score.

This view allows users to identify student models that require fine-tuning. Each student model is labeled by its corresponding class and represented by two circular progress bars. The left circular bar indicates the teacher model's performance on that class, while the right bar represents the student model's performance. At right, two horizontal bars show performance on positive instances (images with the class, top bar) and performance on negative instances (images without the class, bottom bar). In these bars, fully blue areas indicate correct predictions by both models, shaded blue shows student mispredictions that the teacher correctly predicts. Orange areas represent mispredictions by both models, with shaded orange indicating correct predictions by the student and teacher mispredictions. A gray vertical bar's length next to each horizontal bar represents the relative size of the subset.

These encodings provide a compact representation of student model performance, and emphasize prediction misalignments between student and teacher models. This supports an overview-based analysis of the KD process, and provides a starting point for subsequent fine-tuning. For example, a visible shaded blue area signifies that the student model mispredicts many cases where the teacher predicts correctly (a high false negative rate for the student). A user can tune the relevant influential concepts in the student knowledge view to address this issue and enhance the student's performance. (We demonstrate several examples of this process via a pair of usage scenarios as a part of our system validation.)

The **concept embedding view** (Figure 5c) projects concept vectors from CLIP-S^4's multimodal embedding space into a 2D space. This dimensionality reduction plot (several projections are available, including t-SNE, UMAP, and PCA) visualizes concepts and their semantic relationships, offering an overview of the concept landscape (i.e., the knowledge base) for the selected student model. When a user selects a student model, the view highlights the top 10 most influential concepts along with their text labels; other labels can be shown via hovering.

The **student knowledge view** (Figure 5d) displays when a student model is selected, and shows how knowledge is distilled into a student model in terms of its most influential concepts. This view provides users with performance insights and enables concept-based fine tuning of student models.

This view loads the top most influential concepts, sorted by their influence (by default, the top 10 concepts are shown, but this number can be adjusted via a dropdown in the submenu in Figure 5(d1)). The submenu also supports sorting concepts based on their presence discrepancy in aligned vs. misaligned instances of positive or negative labels, helping identify concepts causing local mispredictions.

Each row in this panel corresponds to a specific concept (e.g., in Figure 5(d2) the top row is `Computer`). The leftmost number indicates the rank based on the current sorting method (weight, discrepancy (P), or discrepancy (N)). Clicking the concept name button for a row opens a concept detail view (described below) supporting detailed examination of the specific concept. The blue horizontal bar below the concept name button represents the concept's relative influence compared to the most influential concepts, highlighting potentially skewed concepts.

To the right of the concept name button, a line chart allows users to evaluate the potential effects that modifying a concept's influence will have on the student model's performance. The image patches to the right of the line chart provide visual representations of the concept, displaying the most similar patches to the concept vector in the embedding space. The green bars below each patch indicate their similarities.

Users can uptune (👍) or downtune (👎) a concept using the corresponding buttons. Clicking the uptune button turns the corresponding block in the top bar blue, while clicking the downtune button turns it orange. After specifying the concepts to tune, users can see their options represented by colored blocks in the top bar, along with the current student performance and the overall performance change. Clicking the fine-tune button submits the instructions to the backend, which executes the concept tuning policy described in Section 4.5 to fine-tune the student model. When model re-training completes, the view refreshes with an updated set of top influential concepts and performances. Tuning instructions and their impacts are also recorded in the fine-tuning provenance view (Figure 5f).

The **concept detail view** (Figure 5e) contains four line charts that visualize performance statistics for a selected concept's student model, and a scrollable image gallery showcasing examples of that concept. Specifically, the line charts display accuracy, F1 score, recall, and precision, providing insight into how the model's performance changes when concept influences are varied (thus helping users understand the concept's impact on the class). In particular, prior work has shown these metrics are effective for demonstrating model performance even for "non-expert" users [TDCF07]. For example, in Figure 5, all four metrics decrease as the television stand concept influence increases, indicating that further increasing the concept influence may not be beneficial. Additionally, the recall curve flattens once the television stand concept exceeds a value around **0.35**, suggesting that when the influence is too high, the model may start to miss more true positive instances (i.e., images containing a *TV monitor*).

Alongside the line charts, an image gallery supports qualitative review and confirmation of the concept (by showing example images containing it), enabling users to make well-informed decisions when uptuning or downtuning a concept. While the focus of INFICOND is on concept-driven analysis and fine-tuning (as opposed to reviewing individual image instances), this image gallery is also intended to serve as an available reference for sanity checks and detailed low-level inspections of image instances.

Finally, the **fine-tuning provenance view** (Figure 5f) records previous concept tuning iterations and their impact on the student model's performance. Each row represents an entry, showing uptuned or downtuned concepts and the corresponding performance change from the previous entry. The total performance change at the top-right corner displays the cumulative impacts of the fine-tuning.

6. Evaluation

To validate INFICOND, we conducted two primary activities: First, to demonstrate the capabilities of INFICOND and detail the various workflows that can be taken by users (e.g., reviewing and analyzing the results of the distillation process, fine-tuning underperforming classes, etc.), we report a pair of usage scenarios. Due to page constraints, these may be found in an Appendix in the supplemental materials.

Second, we conducted a user study with ten participants to understand, from an empirical perspective, how INFICOND supports interpretable KD. In this study, participants were trained to use the system and then completed two analysis and fine-tuning tasks. We collected and analyzed both quantitative and qualitative results, allowing us to robustly evaluate how INFICOND supports **(G1)–(G6)** while also assessing the system's overall usability.

Participant and Study Setup. We recruited ten graduate computer science students from Arizona State University. All participants self-reported as light-to-moderate users of various PTMs, though none were experts in technical or programmatic KD workflows. Put another way, these participants were representative of the skills and computational literacy of the emerging class of domain and applied KD stakeholders discussed in Section 1.

Each participant took approximately 30 minutes (SD = 5 minutes) to complete the study. Participants viewed INFICOND on a 30" monitor with a resolution of 3840×2160 using Google Chrome in full-screen mode. Study sessions were conducted in person in a quiet, distraction-free office environment.

Study Design. Participants completed a demographic questionnaire and received a brief introduction to the topic of multi-label classification and visual concepts. The administrator conducted a walk-through of the features and functionalities of INFICOND, and participants completed a simple training task to familiarize themselves with the system. During this training time, participants were free to ask questions at any time and explore the interface until they felt comfortable to proceed. After training, participants completed two tasks (described below) and filled out a post-study questionnaire to evaluate various system aspects. In this survey, participants could also provide freeform feedback if desired.

In the first task, participants were required to to understand and improve the *bicycle* class. The task was considered complete when the participant improved the performance of the student model to be equal to or higher than the teacher model. The average completion time for this task was 7 minutes (SD = 2 minutes). In the

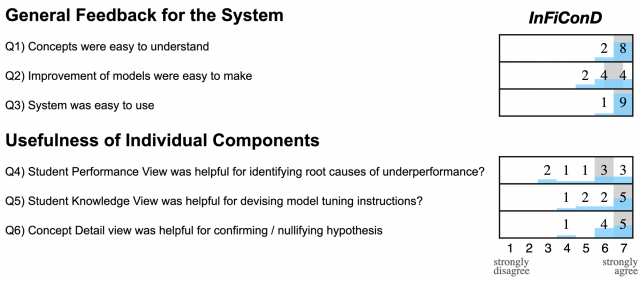

Figure 6: *Usability ratings for* INFICOND *from the user study. The median ratings for each question are indicated in gray.*

second task, participants were required to understand and improve the *tv monitor* class, with the same completion criteria. The average completion time for this task was 4 minutes with a standard deviation of 1 minute. For both tasks, participants were encouraged to think aloud, including to verbalizing their rationales when making decisions (e.g., uptuning a specific concept).

Study Results. In terms of KD performance and student model fine-tuning, all ten participants improved the student models such that they outperformed the teacher model for both tasks (in other words, all participants successfully completed both tasks). For the *bicycle* task, students performed an average of 4.5 fine-tuning iterations (SD = 0.5) resulting in an average performance increase of 1.91% (SD = 0.04). For the *tv monitor* task, the average was 2.5 fine-tuning iterations (SD = 0.5) with a performance increase of 2.04% (SD = 0.05). Typically, participants adjusted between 1–4 concepts for each fine-tuning iteration.

Figure 6 shows responses for the post-study questionnaire. **Q1**–**Q3** ask about the overall system usability, and **Q4**–**Q6** focus on specific interface components. Overall, INFICOND was highly rated, both in terms of overall usability and also on the usefulness of individual system components.

To analyze the think aloud verbalizations and supplemental post-study feedback, we performed an open coding process to understand the strengths and limitations of INFICOND's user experience and workflows. Based on this, a set of high-level themes emerged from participant responses, which we summarize below:

Text-aligned visual concepts were easy to understand and work with. All participants understood the visual concepts immediately and were able to use the concept's semantic relationship to the class to come up with fine-tuning operations. No participants reported being confused with any concepts, which indicates that the text-aligned concept extraction approach, as well as the various mapping methods surfaced in the interface, were successful at making concepts understandable and non-ambiguous.

The concept detail view provided context to confirm understanding of visual concepts. The participants heavily relied on the concept detail view during their tasks. Specifically, almost all participants (with the exception of one) used this view to confirm their understanding of the concept and validate that they aligned with their own mental models, such as reviewing annotated images in the image galley to understand what types of features were present and annotated for the concepts. For instance, one participant (P3) stated, "`remote control` *can be with a tv monitor, but it can also be on its own. So I'm checking the details of* `remote control` *to see if they are mostly on a tv monitor or not.*" Another participant (P5) mentioned, "*I'm not sure what* `set of instruments` *means here, so I need to look at the details* [in this panel]." Importantly though, this process was quite efficient: while the number of images for a concept (shown in the image gallery) could scale to the hundreds (with navigation via scrolling), participants generally only needed to review a small number of images (usually < 15) to gain an intuitive understanding of the current concept.

The student performance view illustrates where to start the analysis. Most participants (8/10) began their analyses by examining the largest shaded areas in the student performance view and prioritizing concepts accordingly in the student knowledge view. For instance, one participant (P4) remarked, "*The shading feature is helpful for me to figure out where to start*," while another (P9) stated, "*I'll fix the ones with the biggest shade first.*" Despite being a compact and bespoke encoding, the horizontal bars in this panel were generally considered to provide a succinct and intuitive representation of the nuances present in the capacity gaps between the teacher and student models.

The student knowledge view enables effective analysis and interaction. Participants effectively interacted with the student knowledge view to analyze and adjust concepts. All participants reported that the leaderboard was helpful and efficient. Per (P3), "*It's cool that I can just vote for the concepts and the model will update.*" (P5): "*I like the voting thing. It's intuitive.*" (P8): "*It provides a really good indication of effects of tweaking each feature at a certain step.*" Five participants used the ranking feature to sort concepts based on the presence discrepancy as a way to identify concepts to tweak. Other participants preferred to sort concepts by their weight. In contrast to the fine-tuning strategies of other participants, one participant (P7) preferred to make minimal changes for each adjustment, to accurately identify the cause of any changes made: "*I want to focus on a few concepts at a time so I can identify the root cause for each change.*"

An accessible workflow and interface that supports (G1)–(G6). Both the survey responses and verbal feedback reinforced that INFICOND's overall workflow, and in particular its ability to support no-code fine-tuning, promoted accessibility and ease of use and supported design goals **(G1)–(G6))**. (P8): "*Simple to interact. Easy to refine the model on a single click.*" (P2): "*The front-end design was good. Easy to understand.*" (P10): "*Easy to train (fine-tune) and select features (concepts).*"

One particularly interesting takeaway here was that, even in cases where fine-tuning resulted in small (or no) performance improvements in student model performance, participants still felt that when they uptuned relevant and downtuned irrelevant concepts, the updated student model behaviors more closely aligned with their own intuitions and mental reasoning, and would better predict on new future instances (e.g., outside of study testing data). In particular, this feedback helped emphasize a key facet of **(G6)**, in that fine-tuning was not always a performance-focused activity, but could also support aspects of human-AI alignment.

7. Discussion

INFICOND represents a novel HITL pipeline and tool for making KD more interpretable and accessible. Below, we briefly reflect on several insights and lessons learned during this process of designing, implementing, and evaluating INFICOND.

Supporting performance, interpretability, and alignment. Our evaluation demonstrated INFICOND was an effective strategy for addressing the design challenges (**C1**)–(**C4**) and design goals (**G1**)–(**G6**) outlined in Section 3. Participants were tasked to match the teacher model's performance, and all could achieve this. However, even in cases where student model games were minimal, participants could intuitively understand the reasoning behind the models (e.g. by downturning irrelevant concepts for a class) and could steer student model behavior to focus on more relevant concepts that aligned with their own intuitions and preferences. This follows our design goals, where raw performance by itself is not the only indicator of success, but student models also needed to be highly interpretable and tunable according to user desires. We further explore these ideas in the usage scenarios, which may be found in the Appendix.

Concept-focused interaction. INFICOND is designed to promote KD review and fine-tuning based around visual concepts as opposed to image instances (which may number in the hundreds or thousands for a class or concept). In practice, we found that study participants still leveraged the concept detail view to efficiently review small numbers of image instances to confirm their understanding of concepts and verify that concepts were not being being extracted on spurious or irrelevant features. Moreover, when performing tuning iterations, participants tended to uptune and downtune only a small number of concepts at a time (often between 1–4), focusing first on the most impactful concepts for the currently considered student model. Despite enforcing an HITL workflow, our study participants were able to efficiently (usually within 5–7 minutes) fine-tune student models to have high performance and also align their prediction behaviors with human preferences. Future work could further enhance these concept-focused workflows (e.g., supporting uptuning or downtuning concepts across multiple student models, instead of individually), particularly in scenarios where larger or more domain-specific concept corpuses might be integrated into the system (see below discussion on this point).

Extending INFICOND to more diverse computer vision and KD tasks. Although we demonstrate INFICOND using in multi-label classification, the potential for visual analytics and concept-based methods for KD extend beyond this specific application scenario. Rapid advancements in large PTMs have accelerated KD activities across a wide range of computer vision tasks, such as object detection [ZCS*23] and semantic/instance segmentation [MBP*21]. In future work, we plan to focus on studying how tools like INFICOND can be tailored for other computer vision tasks. Likewise, we intend to investigate supporting different strategies and methods for KD itself (e.g., feature and relation-based knowledge transfer), and how this could potentially broaden the potential application scope for interpretable and no-code KD and fine-tuning. As a part of these activities, we also intend to more directly test INFICOND against existing baseline techniques which, even if they cannot support some of the more human-centered design goals of our system, will still be able to let us assess how INFICOND's HITL fine-tuning approach might have benefits over more "heavy handed" automated and black box approaches.

Supporting user-specified concept corpora. INFICOND uses visual concepts as its "vehicle of knowledge," and in Section 4 these are bootstrapped using a large, existing concept corpus. While this corpus was suitable for demonstrating INFICOND, it would certainly face limitations when applied to some specialized domains. For example, performing KD for medical tasks (e.g., tumor detection [AMW23]) might require a more customized concept corpus with domain-specific concepts. INFICOND is "concept corpus agnostic," meaning a newer or different corpus could be swapped in, or an existing corpus could be extended with additional concepts. One benefit to this approach is that such modifications do not require the CLIP-S^4 encoders to be retrained. Similarly, we see opportunities for vision-language models as a mechanism for an "on the fly" concept corpus (e.g., by dynamically labeling encountered concepts), however such processes would require human-in-the-loop oversight (e.g., to prevent hallucination and support appropriate levels of specificity) and would require re-designing aspects of the current backend to ensure that the labels do not become not unevenly distributed or biased (which could perturb student model behavior). However, by supporting more flexibility, frameworks like INFICOND can likely better generalize to more domain-specific applications.

Promoting fairness through concept-based knowledge distillation. Finally, the automatic nature of KD methods can lead to biases being transferred to student model as knowledge, if fairness control mechanisms are not implemented during the distillation process [ZCH20, MSWVW20]. Tools like INFICOND have the potential to help mitigate this issue via a concept-based approach, by employing a carefully curated, de-biased concept corpus with additional human-in-the-loop oversight. Future work in this area can develop processes specifically focused on fairness oversight and administration, to ensure that student models learn to perform tasks with unbiased concepts.

8. Conclusion

We investigate how to make the knowledge distillation process more explainable and accessible by developing INFICOND, a novel framework that leverages visual concepts created by multi-modal models to implement interpretable knowledge distillation of large pretrained models and enable subsequent interactive, no-code fine-tuning of student models. We demonstrate the effectiveness of INFICOND on multi-label classification, where INFICOND empowers users to analyze the student models in terms of influential visual concepts and fine-tune them interactively by specifying concept tuning instructions and running a concept tuning method to adjust concept influences. Our concept-based approach represents an effective strategy for the efficient and user-friendly process of knowledge distillation. Future work in this area can expand on the techniques developed here, such as testing on additional types of distillation scenarios and methods, or developed more customizable techniques for domain-specific applications.

References

[AAA21] ALKHULAIFI A., ALSAHLI F., AHMAD I.: Knowledge distillation in deep learning and its applications. *PeerJ Computer Science 7* (2021), e474. 2

[AC24] ALBALLA N., CANINI M.: Practical insights into knowledge distillation for pre-trained models. *arXiv preprint arXiv:2402.14922* (2024). 1

[AK21] AFONIN A., KARIMIREDDY S.: Towards model agnostic federated learning using knowledge distillation. *arXiv preprint arXiv:2110.15210* (2021). 4

[AMW23] ABDUSALOMOV A., MUKHIDDINOV M., WHANGBO T.: Brain tumor detection based on deep learning approaches and magnetic resonance imaging. *Cancers 15*, 16 (2023), 4172. 10

[BZK*17] BAU D., ZHOU B., KHOSLA A., OLIVA A., TORRALBA A.: Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6541–6549. 5

[CH19] CHO J., HARIHARAN B.: On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 4794–4802. 1

[CRCZ20] CHENG X., RAO Z., CHEN Y., ZHANG Q.: Explaining knowledge distillation by quantifying the knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 12925–12935. 1, 3

[CTH*09] CHUA T.-S., TANG J., HONG R., LI H., LUO Z., ZHENG Y.: NUS-WIDE: A real-world web image database from national university of Singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval* (2009), pp. 1–9. 5

[CUF18] CAESAR H., UIJLINGS J., FERRARI V.: Coco-Stuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 1209–1218. 5

[CZW*22] CHEN Y., ZUO R., WEI F., WU Y., LIU S., MAK B.: Two-stream network for sign language recognition and translation. *Advances in Neural Information Processing Systems 35* (2022), 17043–17056. 1

[DCCE19] DAS S., CASHMAN D., CHANG R., ENDERT A.: BEAMES: Interactive multimodel steering, selection, and inspection for regression tasks. *IEEE Computer Graphics and Applications 39*, 5 (2019), 20–32. 6

[DJK23] DANKAR A., JASSANI A., KUMAR K.: Improving knowledge distillation for BERT models: Loss functions, mapping methods, and weight tuning. *arXiv preprint arXiv:2308.13958* (2023). 4

[EDS*24] ELTON D., DASEGOWDA G., SATO J., FRIAS E., MAMONOV A., WALTERS M., ZIEMELIS M., SCHULTZ T., BIZZO B., DREYER K., ET AL.: No-code machine learning in radiology: Implementation and validation of a platform that allows clinicians to train their own models. *medRxiv* (2024), 2024–04. 1, 4

[EVGW*a] EVERINGHAM M., VAN GOOL L., WILLIAMS C., WINN J., ZISSERMAN A.: The PASCAL visual object classes challenge 2007 (VOC2007) results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 5

[EVGW*b] EVERINGHAM M., VAN GOOL L., WILLIAMS C. K. I., WINN J., ZISSERMAN A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 5

[FV17] FONG R. C., VEDALDI A.: Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 3429–3437. 2

[GWT20] GU J., WU Z., TRESP V.: Introspective learning by distilling knowledge from online self-explanation. In *Proceedings of the Asian Conference on Computer Vision* (2020). 3

[GWZK19] GHORBANI A., WEXLER J., ZOU J., KIM B.: Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems 32* (2019). 3, 4

[GYMT21] GOU J., YU B., MAYBANK S., TAO D.: Knowledge distillation: A survey. *International Journal of Computer Vision 129*, 6 (2021), 1789–1819. 1, 2, 3, 4

[HCL*22] HUANG S.-C., CAO C.-F., LIAO P.-H., LEE L.-H., LEE P.-L., SHYU K.-K.: Enhancing Chinese multi-label text classification performance with response-based knowledge distillation. In *Proceedings of the 34th Conference on Computational Linguistics and Speech Processing* (2022), pp. 25–31. 6

[HHS*22] HOQUE M., HE W., SHEKAR A. K., GOU L., REN L.: Visual concept programming: A visual analytics approach to injecting human intelligence at scale. *IEEE Transactions on Visualization and Computer Graphics 29*, 1 (2022), 74–83. 2, 3

[HJGR23] HE W., JAMONNAK S., GOU L., REN L.: Clip-s4: Language-guided self-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 11207–11216. 1, 3, 4, 5

[HMKB22] HUANG J., MISHRA A., KWON B. C., BRYAN C.: Conceptexplainer: Interactive explanation for deep neural networks from a concept perspective. *IEEE Transactions on Visualization and Computer Graphics 29*, 1 (2022), 831–841. 2, 3

[HMLL19] HOU Y., MA Z., LIU C., LOY C. C.: Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE/CVF Onternational Conference on Computer Vision* (2019), pp. 1013–1021. 2

[HPRC19] HOHMAN F., PARK H., ROBINSON C., CHAU D.: Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2019), 1096–1106. 3

[HVD15] HINTON G., VINYALS O., DEAN J.: Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015). 2

[HYW*22] HUANG T., YOU S., WANG F., QIAN C., XU C.: Knowledge distillation from a stronger teacher. *Advances in Neural Information Processing Systems 35* (2022), 33716–33727. 4

[HZZ*24] HUANG T., ZHANG Y., ZHENG M., YOU S., WANG F., QIAN C., XU C.: Knowledge diffusion for distillation. *Advances in Neural Information Processing Systems 36* (2024). 4

[JPW*19] JIN X., PENG B., WU Y., LIU Y., LIU J., LIANG D., YAN J., HU X.: Knowledge distillation via route constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 1345–1354. 2

[KPK18] KIM J., PARK S., KWAK N.: Paraphrasing complex network: Network compression via factor transfer. *Advances in Neural Information Processing systems 31* (2018). 2, 3

[KWG*18] KIM B., WATTENBERG M., GILMER J., CAI C., WEXLER J., VIEGAS F., ET AL.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning* (2018), PMLR, pp. 2668–2677. 2

[LCS*20] LI X.-H., CAO C. C., SHI Y., BAI W., GAO H., QIU L., WANG C., GAO Y., ZHANG S., XUE X., ET AL.: A survey of data-driven and knowledge-aware explainable ai. *IEEE Transactions on Knowledge and Data Engineering 34*, 1 (2020), 29–49. 3

[LGX*24] LIU H.-I., GALINDO M., XIE H., WONG L.-K., SHUAI H.-H., LI Y.-H., CHENG W.-H.: Lightweight deep learning for resource-constrained environments: A survey. *ACM Computing Surveys* (2024). 1

[LKS18] LEE S. H., KIM D. H., SONG B. C.: Self-supervised knowledge distillation using singular value decomposition. In *Proceedings of the European Conference on Computer Vision* (2018), pp. 335–350. 2

[LMB*14] LIN T.-Y., MAIRE M., BELONGIE S., HAYS J., PERONA P., RAMANAN D., DOLLÁR P., ZITNICK C.: Microsoft COCO: Common objects in context. In *European Conference on Computer Vision* (2014), pp. 740–755. 5

[LZY*21] LIU S., ZHANG L., YANG X., SU H., ZHU J.: Query2label: A simple transformer way to multi-label classification. *arXiv preprint arXiv:2107.10834* (2021). 5

[MBP*21] MINAEE S., BOYKOV Y., PORIKLI F., PLAZA A., KEHTARNAVAZ N., TERZOPOULOS D.: Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 7 (2021), 3523–3542. 10

[MCL*14] MOTTAGHI R., CHEN X., LIU X., CHO N.-G., LEE S.-W., FIDLER S., URTASUN R., YUILLE A.: The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014). 5

[MFL*20] MIRZADEH S., FARAJTABAR M., LI A., LEVINE N., MATSUKAWA A., GHASEMZADEH H.: Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 5191–5198. 2, 4, 5

[MKH19] MÜLLER R., KORNBLITH S., HINTON G.: When does label smoothing help? *Advances in Neural Information Processing Systems 32* (2019). 4

[MSWVW20] MADAIO M., STARK L., WORTMAN VAUGHAN J., WALLACH H.: Co-designing checklists to understand organizational challenges and opportunities around fairness in ai. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020), pp. 1–14. 10

[Ope] OPENAI: Clip: Connecting text and images. https://openai.com/research/clip. 1

[PDD*21] PARK H., DAS N., DUGGAL R., WRIGHT A., SHAIKH O., HOHMAN F., CHAU D. H. P.: Neurocartography: Scalable automatic visual summarization of concepts in deep neural networks. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2021), 813–823. 3

[PPTM*16] PERAZZI F., PONT-TUSET J., MCWILLIAMS B., VAN GOOL L., GROSS M., SORKINE-HORNUNG A.: A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 724–732. 5

[PTT20] PASSALIS N., TZELEPI M., TEFAS A.: Heterogeneous knowledge distillation using information flow modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 2339–2348. 2, 3

[RLN*23] RONG Y., LEEMANN T., NGUYEN T.-T., FIEDLER L., QIAN P., UNHELKAR V., SEIDEL T., KASNECI G., KASNECI E.: Towards human-centered explainable AI: A survey of user studies for model explanations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023). 4

[SC24] SHIN H., CHOI D.-W.: Teacher as a lenient expert: Teacher-agnostic data-free knowledge distillation. *arXiv preprint arXiv:2402.12406* (2024). 4

[SKKS22] SCHIZAS N., KARRAS A., KARRAS C., SIOUTAS S.: TinyML for ultra-low power AI and large scale IoT deployments: A systematic review. *Future Internet 14*, 12 (2022), 363. 1

[SMB*22] SOUSA J., MOREIRA R., BALAYAN V., SALEIRO P., BIZARRO P.: ConceptDistil: Model-agnostic distillation of concept explanations. *arXiv preprint arXiv:2205.03601* (2022). 1

[SNCH21] SON W., NA J., CHOI J., HWANG W.: Densely guided knowledge distillation using multiple teacher assistants. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 9395–9404. 4

[TDCF07] TULLIO J., DEY A., CHALECKI J., FOGARTY J.: How it works: a field study of non-technical users interacting with an intelligent system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2007), pp. 31–40. 8

[VCHS18] VLUYMANS S., CORNELIS C., HERRERA F., SAEYS Y.: Multi-label classification using a fuzzy rough neighborhood consensus. *Information Sciences 433* (2018), 96–114. 5

[WCQ*23] WANG X., CHEN G., QIAN G., GAO P., WEI X.-Y., WANG Y., TIAN Y., GAO W.: Large-scale multi-modal pre-trained models: A comprehensive survey. *Machine Intelligence Research 20*, 4 (2023), 447–482. 1

[WLG23] WANG Q., L'YI S., GEHLENBORG N.: Drava: Aligning human concepts with machine learning latent dimensions for the visual exploration of small multiples. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). 3

[WY21] WANG L., YOON K.-J.: Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 6 (2021), 3048–3068. 3

[YXZ*23] YANG P., XIE M.-K., ZONG C.-C., FENG L., NIU G., SUGIYAMA M., HUANG S.-J.: Multi-label knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 17271–17280. 5

[YYAX23] YANG C., YU X., AN Z., XU Y.: Categories of response-based, feature-based, and relation-based knowledge distillation. In *Advancements in Knowledge Distillation: Towards New Horizons of Intelligent Systems*. 2023, pp. 1–32. 6

[ZCCR22] ZHANG Q., CHENG X., CHEN Y., RAO Z.: Quantifying the knowledge in a DNN to explain knowledge distillation for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence 45*, 4 (2022), 5099–5113. 1, 3

[ZCH20] ZHOU J., CHEN F., HOLZINGER A.: Towards explainability for AI fairness. In *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers* (2020), pp. 375–386. 10

[ZCS*23] ZOU Z., CHEN K., SHI Z., GUO Y., YE J.: Object detection in 20 years: A survey. *Proceedings of the IEEE 111*, 3 (2023), 257–276. 10

[ZG18] ZHU X., GONG S.: Knowledge distillation by on-the-fly native ensemble. *Advances in Neural Information Processing Systems 31* (2018). 4, 6

[ZK16] ZAGORUYKO S., KOMODAKIS N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928* (2016). 3

[ZLX*22] ZHU Y., LIU N., XU Z., LIU X., MENG W., WANG L., OU Z., TANG J.: Teach less, learn more: On the undistillable classes in knowledge distillation. *Advances in Neural Information Processing Systems 35* (2022), 32011–32024. 4

[ZQL*23] ZHANG Y., QIN Y., LIU H., ZHANG Y., LI Y., GU X.: Knowledge distillation from single to multi labels: An empirical study. *arXiv preprint arXiv:2303.08360* (2023). 5

[ZSG*19] ZHANG L., SONG J., GAO A., CHEN J., BAO C., MA K.: Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 3713–3722. 2

[ZXSR21] ZHAO Z., XU P., SCHEIDEGGER C., REN L.: Human-in-the-loop extraction of interpretable concepts in deep learning models. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2021), 780–790. 3