ConceptEvo: Interpreting Concept Evolution in Deep Learning Training

Haekyu Park, Seongmin Lee, Benjamin Hoover, Austin Wright, Omar Shaikh, Rahul Duggal, Nilaksh Das, Kevin Li, Judy Hoffman, Duen Horng (Polo) Chau

Georgia Institute of Technology, Atlanta, GA, USA
{haekyu, seongmin, bhoov, apwright, oshaikh, rahulduggal, nilakshdas, kevin.li, judy, polo}@gatech.edu

Abstract

Deep neural networks (DNNs) have been widely used for decision-making, prompting a surge of interest in interpreting how they work. Recent literature on DNN interpretation has revolved around already-trained models. However, much less research focuses on interpreting how models evolve as they are trained. Interpreting model evolution is crucial to monitoring the network training and can aid proactive interventions. We present CONCEPTEVO, a general unified interpretation framework for DNNs that reveals the inception and evolution of detected concepts during training. Through a large-scale human evaluation with 260 participants and quantitative experiments, we show that CONCEPTEVO discovers evolutions across different models that are meaningful to humans and important for predictions.

1 Introduction

Interpreting how Deep Neural Networks (DNNs) arrive at an answer has become crucial for instilling trust in models' decisions (Ribeiro, Singh, and Guestrin 2016), debugging models (Koh and Liang 2017), and guarding against harms such as embedded bias or adversarial attacks (Das et al. 2020; Papernot and McDaniel 2018; Zhang, Wang, and Zhu 2018). As a fundamental type of DNN, convolutional neural networks (CNNs) have been intensively explored to understand how they work. For example, saliency-based methods find regions of an image important for predictions (Selvaraju et al. 2017; Simonyan, Vedaldi, and Zisserman 2013). Concept-based methods discover ***concepts*** learned by DNNs (e.g., "*furry dogs*" concepts in Fig 1) and how those concepts interact to form higher-level concepts and predictions (Park et al. 2021; Olah et al. 2020; Ghorbani et al. 2019; Kim et al. 2018; Bau et al. 2017).

However, existing methods mainly study post-hoc model interpretation after training (Laugel et al. 2019; Guidotti et al. 2018). Much less research focuses on interpreting how models evolve as they are trained, despite such interpretation's usefulness for monitoring and tracking training processes (Zhong et al. 2017; Liu et al. 2017; Abadi et al. 2016). For example, there is a lack of understanding of how concepts detected by a neuron progress — we call this the neuron's **concept evolution** — and how deficiencies may be introduced as a model is trained. To fill this critical research gap, our work makes the following major contributions:

Figure 1: CONCEPTEVO creates a unified semantic space that represents and aligns concepts detected by neurons across different models in training (top: VGG19; bottom: VGG16), embedding similar concepts (e.g., "furry dogs," "car window," "green") at similar corresponding locations.

1. CONCEPTEVO: a general unified interpretation framework of DNNs that reveals the inception and evolution of learned concepts during training (Sec 3). Our framework presents two novel technical contributions:

(1) An algorithm that generates a unified semantic space that enables side-by-side comparison of different models during training (Fig 1, 2).

(2) An algorithm that discovers and quantifies important concept evolutions for a class prediction (Fig 3).

2. Evaluation. We conduct large-scale human experiments and quantitative experiments, which demonstrate that CONCEPTEVO discovers concept evolutions of different models that are both meaningful to humans and important to a class prediction (Sec 4).

3. Discoveries on model evolution. We demonstrate how CONCEPTEVO helps identify potential model training issues and reason out the causes (Sec 4.5), such as: (1) ill-suited hyperparameters (e.g., overly high learning rate) severely harm concept diversity as shown in Fig 2b; and (2) concepts in overfitted models evolve slowly despite rapid training accuracy increases, as shown in Fig 2c.

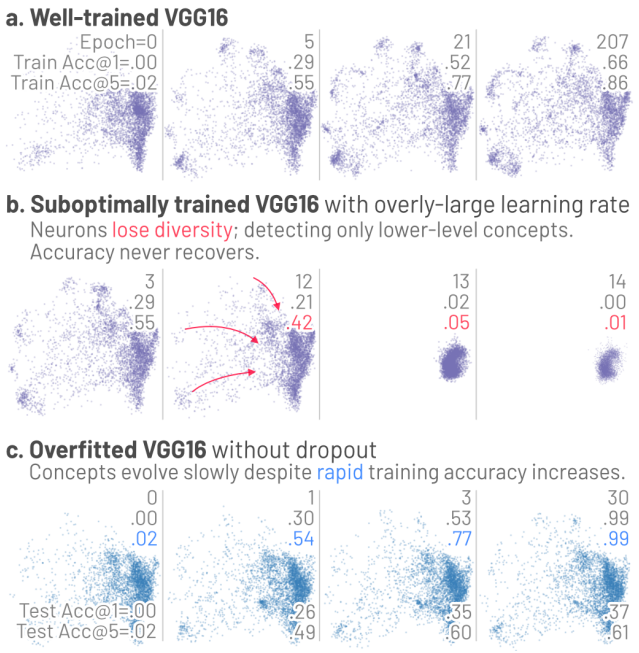

Figure 2: ConceptEvo identifies potential model training issues. (a) A well-trained VGG16 showing gradual concept formations and refinements (see example concepts in Fig 1). (b) A VGG16 suboptimally trained with a large learning rate, rapidly losing the ability to detect most concepts. (c) An overfitted VGG16 without dropout layers, showing slow concept evolutions despite rapid training accuracy increases.

2 Related Work

Interpreting DNNs After Training. Interpreting a fully-trained DNN revolves around describing features that are important to the model's behavior. For example, saliency-based methods determine pixels that are important for the prediction (Selvaraju et al. 2017; Dabkowski and Gal 2017; Simonyan, Vedaldi, and Zisserman 2013). Concept-based methods identify high-level and human-understandable concepts that are important for the prediction (Hernandez et al. 2022; Ghorbani and Zou 2020; Yeh et al. 2020; Goyal, Shalit, and Kim 2019; Zhou et al. 2018; Kim et al. 2018; Nguyen, Yosinski, and Clune 2016). A growing number of concept-based interpretability approaches attribute such concepts to individual neurons, pinpointing the ones that contribute most to the model's behavior. Net2Vec (Fong and Vedaldi 2018) encodes individual neurons' concepts into vectors by using predefined concept images. Neuron Shapley (Ghorbani and Zou 2020) quantifies the importance of individual neurons for a class prediction by measuring the Shapley (Lundberg and Lee 2017) value for each neuron. NeuroCartography (Park et al. 2021) summarizes and visualizes concepts detected by individual neurons by encoding the conceptual neighborhood of neurons. However, these existing approaches are often designed for interpreting a single model after it is fully trained. ConceptEvo, on the other hand, allows comparison of concepts learned and evolved in multiple models by embedding the neurons of different DNNs at different epochs onto a unified semantic space.

Interpreting DNNs During Training. Many existing works for interpreting DNNs *during training* focus on how data representations within the models evolve across epochs and how the evolution impacts its downstream performance (Pühringer, Hinterreiter, and Streit 2020; Smilkov et al. 2017; Chung et al. 2016). For example, DeepEyes (Pezzotti et al. 2017) studies the evolution of individual neurons' responses to different classes during training by aggregating the neurons' activation on receptive fields at each epoch. DGMTracker (Liu et al. 2017) shows how models evolve by analyzing the changes of weights, activations, and gradients over time. Others track the 2D projected evolution of neurons towards or away from certain class labels (Rauber et al. 2016; Li, Zhao, and Scheidegger 2020), which limits our understanding of a learned concept to only the available class labels. Instead of tracking concepts visually, DeepView (Zhong et al. 2017) introduces two metrics to anticipate whether neurons are evolving sufficient diversity for the label-prediction task. ConceptEvo distinguishes itself from the existing approaches by enabling comparison of concepts learned by neurons from **any layers within a model** and even neurons from a **different model** trained using the same dataset.

3 Method

3.1 Desiderata of Interpreting Concept Evolution

Our goal is to interpret how a model evolves during training by investigating concept inception and evolution according to the desired properties below.

D1 General interpretation of concept evolution across different models. Comparing different model training is indispensable for deciding which model is trained better or which training strategy is more effective (Li et al. 2018; Raghu et al. 2017). Thus, we aim to develop a general method that enables side-by-side comparison and interpretation of concept evolution across different models. (Sec 3.2)

D2 Revealing and quantifying important evolution of concepts. We are interested in spotting internal changes that are critical for the prediction of a given class, as identifying the most responsible components can provide an efficient way to improve the models (Ghorbani and Zou 2020). For example, how important is the evolution of a neuron's concept (e.g., from "orange round shape" to "fingers") for the prediction of a class (e.g., "bow tie")? We aim to automatically find such important changes in concepts for a class prediction. (Sec 3.3)

D3 Discoveries. Can interpreting model evolution help identify problems during training and provide insights into how to address them, advancing prior work that focuses on interpreting and fixing already-trained models (Ghorbani and Zou 2020)? For example, can we determine whether a model's training is on track or whether interventions are needed to improve the prediction accuracy? (Sec 4.5)

3.2 General Interpretation of Concept Evolution

We desire an interpretation of model evolution to be comparable across different models. However, direct comparison between concepts in different models at different training stages is challenging. Different models are independently trained; thus, the learned concepts are not aligned by default. Even for the same model, activation patterns can change considerably over training epochs. To address this challenge, we propose a two-step method: (1) create a semantic space to represent concepts detected by a *base model* at a specific training epoch, and (2) project concepts of other models at all epochs onto the same **unified semantic space**, such that it represents concepts in the base model and approximately represents concepts in other models and from any epochs. We choose an optimally, fully trained model as our base model to cover as wide a variety of concepts as possible. For example, we used a fully trained VGG19 (Simonyan and Zisserman 2015) as a base model for Fig 1, 2.

Step 1: Creating the base semantic space. The goal of the first step is to create a *base semantic space* that represents concepts detected by the base model. Given the observation that neurons are selectively activated by specific concepts (Ghorbani and Zou 2020; Olah, Mordvintsev, and Schubert 2017; Yosinski et al. 2015), we use a neuron as a unit to detect concepts and represent such concepts on the base semantic space. Using neurons as the concept units allows us to pinpoint areas of interest inside models; for example, we can pinpoint a group of neurons that show abnormal training patterns and narrow our focus to fixing the problematic components. As neuron-concept relationships may not always be 1-to-1 (Olah et al. 2020; Fong and Vedaldi 2018), we design our approach to generalize to many-to-many relationships. For example, a polysemantic neuron that responds to multiple concepts will be embedded between those concepts.

Inspired by the finding from prior work that neurons detecting similar concepts are highly activated by many common inputs (Park et al. 2021), we embed concepts of neurons co-activated by more common images to be closer on the base semantic space. We adapt the neuron embedding approach from previous work (Park et al. 2021) as a maximum likelihood estimation algorithm, which we summarize into the two phrases. First, for each neuron, CONCEPTEVO finds a set of images that strongly activate the corresponding neuron. We call these images the *stimuli* of the neurons. Second, it learns the vector representations of neurons in the base model, where neurons that have similar stimuli (i.e., that are frequently co-activated) are located closer on the embedding space. To find the *stimuli* for each neuron, we measure the activation strength of a neuron with the maximum value of a neuron's activation map, as in previous work. For each neuron, we create a neuron's stimuli by collecting k images that lead to the largest maximum of the neuron's activation map. For a single-concept neuron, its stimuli will be more alike, while for a polysemantic neuron, its stimuli may consist of more than one concept.

CONCEPTEVO then learns neuron embedding to encode concepts in the base model. Eq (1) defines the objective function to maximize, where D is a multiset of strongly co-activated neuron pairs (n, m). That is, a specific neuron pair can appear multiple times in D; more pairs (n, m) will be in D as more images are shared by their stimuli, causing frequently co-activated neurons to more closely embedded. $\mathbf{v}^t_{n,M}$ is an embedding of neuron n in model M at epoch t. r is a randomly selected neuron. R is the number of randomly sampled neurons for each co-activated neuron pair in D. M_b is the base model. t_b is the base model's current epoch. $\sigma(\cdot)$ is the sigmoid function (i.e., $\sigma(x) = 1/(1 + e^{-x})$). The objective function induces more frequently co-activated neurons to be closer and randomly paired neurons to be far away. We randomly initialize the neuron embeddings and compute the embeddings by gradient ascent. We provide details of how we optimize it in Appendix A.

$$J_1 = \sum_{(n,m)\in D} \Bigg(\log\left(\sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b})\right) + \sum_{r=1}^{R} \log\left(1 - \sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\right) \log\left(1 - \sigma(\mathbf{v}^{t_b}_{m,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\right) \Bigg) \tag{1}$$

Step 2: Unifying the semantic space of different models at different epochs. If two neurons, regardless of which models they belong to or which training stages they are at, have similar stimuli (i.e., strongly co-activated by many common input images), we aim to learn those neurons' embedding to be close together on the semantic space created in Step 1. Learning all concepts in all models at all epochs at the same time in Step 1 is possible but computationally prohibitive. Thus, we design this second step to approximately represent concepts that are not in the base model.

Unifying the concept space is in two phases. First, CONCEPTEVO learns embedding representations of input images on the base semantic space, which can represent the relationships between neurons and their stimuli. Next, it approximates the embedding of neurons in the other models at other epochs, by using the computed image embedding and the relationships between the neurons and their stimuli.

As an image includes various concepts, we assume that an image vector can be formed by linearly combining vectors of those concepts. In reverse, we assume that a neuron embedding can be approximated by linearly combining image embeddings. A neuron embedding is approximated as the average of embeddings of images in the neuron's stimuli as in Eq (2). $\mathbf{v}'^t_{n,M}$ is the approximated embedding of neuron n in a model M at epoch t, $\mathbf{v}_\mathbf{x}$ is an image $\mathbf{x}$'s embedding, and $X^t_{n,M}$ is the set of stimuli for n in M at t. We average the embedding across important images, as average is a standard aggregation approach from previous seminar work (Ghorbani and Zou 2020; Ghorbani et al. 2019; Kim et al. 2018).

$$\mathbf{v}'^t_{n,M} = \frac{1}{|X^t_{n,M}|} \sum_{\mathbf{x}\in X^t_{n,M}} \mathbf{v}_\mathbf{x} \tag{2}$$

Then we define the objective function to minimize the difference between original embedding and approximated embedding of neurons in the base model as in Eq (3), where

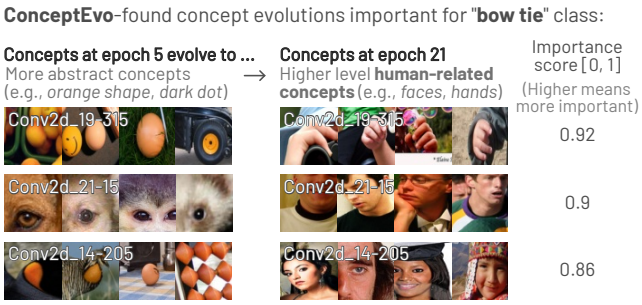

Figure 3: CONCEPTEVO discovers and quantifies important concept evolutions for class prediction. Here, we show example discovered concept evolutions in a VGG16 model, such as *"human hands," "neck,"* and *"face"* that are important for the prediction of the "bow tie" class. CONCEPTEVO quantifies the evolutions' importance. For example, the importance score for the concept evolution of neuron Conv2d-19-315 (top row: *"orange circles"→ "hands"*) is 0.92, meaning that such a concept evolution contributes to improve the prediction for 92% of bow tie images.

N_{M_b} is a set of all neurons in the base model.

$$J_2 = \frac{1}{2|N_{M_b}|} \sum_{n \in N_{M_b}} \left|\left| \mathbf{v}'^{t_b}_{n,M_b} - \mathbf{v}^{t_b}_{n,M_b} \right|\right|_2^2 \tag{3}$$

We randomly initialize the image embeddings and learn them by gradient descent. Eq (4) shows the derivative to update an image embedding, where $X^{t_b}_{n,M_b}$ is the set of stimuli of neuron n in the base model M_b at epoch t_b, and $N_{M_b,\mathbf{x}}$ is the set of neurons in M_b whose stimuli includes an input $\mathbf{x}$.

$$\frac{\partial J_2}{\partial \mathbf{v}_\mathbf{x}} = \frac{1}{|N_{M_b}|} \sum_{n \in N_{M_b,\mathbf{x}}} \frac{1}{|X^{t_b}_{n,M_b}|} \left(\mathbf{v}'^{t_b}_{n,M_b} - \mathbf{v}^{t_b}_{n,M_b} \right) \tag{4}$$

CONCEPTEVO then approximates neuron embeddings for other models at other epochs by using Eq (2).

To visualize neuron embeddings, we use UMAP, a nonlinear dimensionality reduction method that preserves global data structures and local neighbor relations (McInnes, Healy, and Melville 2018). To help people understand what concepts are detected by each neuron, we compute example patches, cropped images that maximally activate the neuron (e.g., example patches of neurons for *"car window" concept in Fig 1*) (Olah, Mordvintsev, and Schubert 2017).

3.3 Concept Evolutions Important for a Class

As mentioned in **D2**, we would like to reveal concept evolutions that are important to a class prediction. For example, how important is the evolution of a neuron's concept (e.g., from *"furry animals' eyes"* to *"human neck"*) to the prediction for a class (e.g., "bow tie")? Inspired by (Kim et al. 2018), we quantify the importance of a concept evolution by evaluating how sensitive a class prediction is to the evolutionary state of the concepts.

Let n be a neuron in layer l of a model M, and $Z^t_{n,l,M}(\mathbf{x})$ be the activation map of n at epoch t given an image $\mathbf{x}$. Let $h^t_{l,c}(\cdot) : \mathbb{R}^{h_l \times w_l \times s_l} \to \mathbb{R}$ be a function that takes activation of layer l as input and returns the logit value for class c, where h_l, w_l, and s_l are height, width, and the number of neurons in layer l, respectively. Let $Z^t_{l,M}(\mathbf{x}) \in \mathbb{R}^{h_l \times w_l \times s_l}$ is the activation of layer l in model M given the input $\mathbf{x}$, and $\Delta Z^{t,t'}_{n,l,M}(\mathbf{x})$ is the activation change of n from epoch t to t' as defined in Eq (6). $\mathbf{0}_{a,b}$ is a zero matrix of a rows and b columns. Eq (5) defines the sensitivity of the class c prediction with respect to n's concept evolution from t to t' given $\mathbf{x}$. The directional derivative in Eq (5) indicates how sensitively a prediction for class c would change if the activation in layer l changes towards the direction of neuron n's evolution. The positive value means that the concept evolution of neuron n positively contributes to the prediction for class c.

$$\begin{aligned} &S^{t,t'}_{n,l,M,c}(\mathbf{x}) \\ &= \lim_{\epsilon \to 0} \frac{h^t_{l,M,c}(Z^t_{l,M}(\mathbf{x}) + \epsilon \Delta Z^{t,t'}_{n,l,M}(\mathbf{x})) - h^t_{l,M,c}(Z^t_{l,M}(\mathbf{x}))}{\epsilon} \\ &= \nabla h^t_{l,M,c}(Z_l(\mathbf{x})) \cdot \Delta Z^{t,t'}_{n,l,M}(\mathbf{x}) \end{aligned} \tag{5}$$

$$\begin{aligned} &\Delta Z^{t,t'}_{n,l,M}(\mathbf{x}) \\ &= [\mathbf{0}_{h_l,w_l}, \cdots, \underbrace{Z^{t'}_{n,l,M}(\mathbf{x}) - Z^t_{n,l,M}(\mathbf{x})}_{n\text{-th matrix}}, \cdots, \mathbf{0}_{h_l,w_l}] \end{aligned} \tag{6}$$

We finally measure the importance of concept evolution of a neuron n in layer l in model M from epoch t to t' for class c, by aggregating the importance across class c images, as in Eq (7), where X_c is the set of images labeled as c.

$$I^{t,t'}_{n,l,M,c} = \frac{|\{\mathbf{x} \in X_c \ : \ S^{t,t'}_{n,l,M,c}(\mathbf{x}) > 0\}|}{|X_c|} \tag{7}$$

Figure 3 shows examples of important concept evolutions discovered by CONCEPTEVO for the "bow tie" class, such as *"human hands," "neck,"* and *"face"*. Surprised by the many evolutions to human-related concepts, we inspected the raw images for the bow tie class and confirmed that most of the images (over 70%) are of a person wearing a bow tie. Appendix B.1 shows six more class examples and their important evolutions.

4 Experiment

We evaluate how well CONCEPTEVO satisfies the desired properties for interpreting conception evolution (Sec 3.1, D1-3) by answering the following research questions:

Q1 Alignment. How well does CONCEPTEVO's neuron embedding align concepts of different models at different training stages in the unified space? (Sec 4.2, for **D1**)

Q2 Meaningfulness. How are the discovered concept evolutions semantically meaningful? (Sec 4.3, for **D1**)

Q3 Importance. How are the discovered concept evolutions important for class prediction? (Sec 4.4, for **D2**)

Q4 Discoveries. How does CONCEPTEVO provide insightful discoveries? (Sec 4.5, for **D3**)

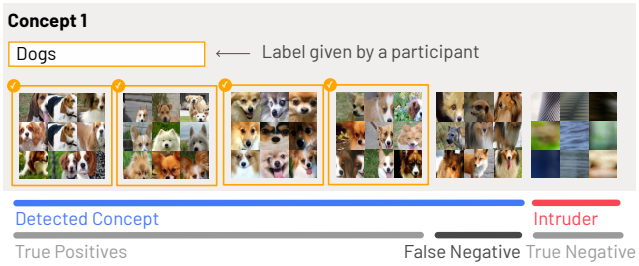

Figure 4: An example MTurk questionnaire. Participants were presented with six neurons' example patches and asked to determine if there is a semantically coherent group of neurons. If so, they would provide a short label for the group. The first five neurons are semantically similar, detected and grouped by CONCEPTEVO; the rightmost is a randomly sampled *intruder*. Here, a participant chooses the first four neurons to be grouped for the coherent "dogs" concept (i.e., four *true positives*), misses the fifth neuron (i.e., *false negative*, and does not choose the intruder (i.e., *true negative*)).

4.1 Experimental Settings

Datasets and models. We studied concept evolutions of models that were trained on ILSVRC2012 (ImageNet) dataset (Russakovsky et al. 2015), such as VGG16 (Simonyan and Zisserman 2015), VGG19 (Simonyan and Zisserman 2015), InceptionV3 (Szegedy et al. 2016), and VGG16 without dropout layer (Srivastava et al. 2014).

Hyperparameter settings. For k (the number of stimuli per neuron), we experimented with values from 5 to 30 and found 10 provided good conceptual coherence of neighboring neurons on the unified space. For the dimension of the neuron and image embeddings, we experimented with values from 5 to 100 and found 30 provided good semantic coherence among neighboring neurons on the unified space and led to results computed in a reasonable time (<5 hours).

Computing resource. We ran all our experiments on an NVIDIA A100, with 8 GPUs with 80GB of memory each.

4.2 Alignment of Neuron Embeddings

For CONCEPTEVO to be effective, detected concepts must be interpretable, and their embeddings should be aligned across models and epochs. To validate CONCEPTEVO's effectiveness, we conducted a large-scale human evaluation with Amazon Mechanical Turk (MTurk), modeling after prior work (Park et al. 2021; Ghorbani et al. 2019). We evaluate the conceptual coherency of neighboring neurons on the unified semantic space within four categories: (1) hand-picked sets of neurons of similar concepts, as the baseline; (2) neuron groups detected by CONCEPTEVO from a well-trained VGG16 base model; (3) groups of neurons within the same models at different training epochs, detected by CONCEPTEVO; (4) groups of neurons from different models (VGG16 and Inception V3) at different epochs, detected by CONCEPTEVO. We collect the neuron groups by running K-means clustering on the neuron embeddings on the unified semantic space.

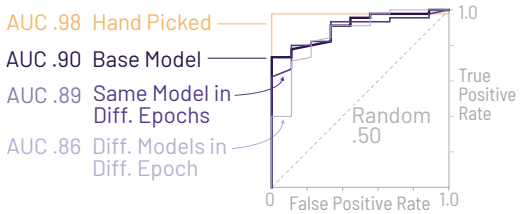

Figure 5: ROC Curve for human estimations of alignability of concept representations across different models and different epochs. Neurons discovered and grouped by CONCEPTEVO (regardless of models or epochs) are highly discernible and aligned even when concepts are sampled across epochs and models.

We provided 260 unique MTurk participants with nine concept classification tasks. For each task, we displayed six randomly ordered neurons' concepts; five had similar concepts found by CONCEPTEVO or were hand-picked, and one was a randomly sampled "intruder" neuron. For each neuron, we provided nine example image patches to explain its concept. Without being told how many potential intruders were present, participants could select however many neurons they felt are semantically similar and were asked to provide a short description of the concept. This process, seen in Figure 4, essentially formulates a classification task, where the human participants are considered the classifiers and the grouped neurons are labeled as true, generating a test set with a final total of 10,950 individual determinations of conceptual inclusions of neurons. From this framing, we consider success by how consistently participants agree with the model determination. We then calculated an ROC curve with the participants' determinations as seen in Figure 5. It shows that CONCEPTEVO-detected concepts are highly discernible and aligned even when concepts are sampled across epochs and different models, with high AUC scores from 0.90 for sampling within the base model, to 0.86 for sampling across different models and training epochs.

4.3 Human-meaningfulness of Concept Evolution

Beyond being consistent, concepts discovered with CONCEPTEVO should also be *meaningful*; interpretations should be informative to humans. Therefore, we also evaluated the *interpretive consistency* of the concepts labeled and described by the participants (Fig 4). To account for phrasing differences for labels, we used sentence-level embeddings from the Universal Sentence Encoder (USE) (Cer et al. 2018). USE captures the semantic similarity of phrases, e.g., labels from our study such as "vehicle wheels," "cars," and "trucks" have high USE similarity. We first calculated the average pairwise similarity between all labels as a baseline for the USE similarity metric, which in our dataset is 0.28. Then, for each category from Sec 4.2, we measured the average pairwise similarity between the labels given by all participants for an individual concept. The average concept similarity for the hand-picked concepts was 0.455, while the similarities for concepts in the base model and the same-model-different-epoch were 0.40. For the different-model-

different-epoch, the similarity was 0.38. All of these values are substantially higher than the baseline.

4.4 Concept Evolutions Important to a Class

We evaluated how well CONCEPTEVO discovers concept evolutions important for a class prediction, similar to how prior works assessed concept importance for fully-trained models (Ghorbani and Zou 2020; Ghorbani et al. 2019). Specifically, for a given class, we measured how the prediction accuracy changes when evolutions are reverted. For a neuron that has evolved from epoch t to t', we reverted the neuron's activation map at t' to that of t and measured the prediction accuracy at t'. A greater drop in accuracy indicates higher importance for that neuron's concept evolution. Using hyperparameters from prior work (Simonyan and Zisserman 2015), we trained a VGG16 model to comparable accuracy (Fig 2a). We used three milestone top-1 training accuracies of 0.25, 0.5, and 0.75 to determine two evolution stages to evaluate[1]: epoch 5→21 and 21→207.

As CONCEPTEVO measures the importance of concept evolution for "a single neuron" (Eq 7), it is natural to evaluate the accuracy change by separately reverting each neuron's evolution and then aggregate the changes across neurons. However, doing so is computationally prohibitive due to the large number of neurons. Thus, we developed a more practical approach by reverting multiple evolutions in a layer at a time, and then aggregating the accuracy changes across layers. We also considered the possible approach of reverting evolutions in all layers at the same time but we decided against it, because the effect of evolution reversion in lower layers (closer to input) transmitted to the deeper layers (closer to output) can be *overridden* by the evolution reversion in the deeper layers. Thus, we decided on the layer-level approach.

For each class c and each evolution stage from epoch t to t', we summarize the evaluation into five steps. **Step 1:** sample 128 images out of all images for c (i.e., sampling about 10% from about 1300 images). **Step 2:** compute the importance of concept evolutions from t to t' of all neurons, based on Eq (7). **Step 3:** for each layer, rank the neurons in the descending order of evolution importance, and place them in 4 importance bins: 0-25th, 25-50th, 50-75th, and 75-100th percentiles. **Step 4:** for each layer and for each bin, revert the evolutions of neurons, compute the accuracy at epoch t', and compute how much the accuracy has changed compared to the accuracy without evolution reversion. **Step 5:** average the accuracy changes across layers, providing accuracy changes for the four bins. We repeated the above procedure five times independently to guard against sampling bias in Step 1. We then averaged the accuracy changes across 100 randomly selected classes out of 1,000 in ImageNet[2].

Fig 6 shows the training accuracy changes when evolutions from epoch 21 to 207 were reverted. When higher-importance evolutions (i.e., lower percentiles) were reverted to epoch 21, the accuracy dropped more, confirming CONCEPTEVO's effectiveness in identifying important concept evolutions. The training top-1 accuracy slightly increased when 75-100th percentile (i.e., least important) evolutions were reverted, suggesting that the least important evolutions might interfere with the prediction for specific classes, but helpful for other classes in general. For a baseline, we also computed the accuracy drop when 25% randomly selected evolutions (i.e., same number) were reverted, which was between that of the 25-50th percentile and 50-75th percentile, indicated by the dashed horizontal lines in Fig 6. Table 1 in Appendix B.2 provides additional results for the evolution stages of 5→21 and 21→207, showing detailed top-1 and top-5 accuracy changes for training and testing, all of which reinforce our key finding that *reverting higher-importance evolutions leads to greater accuracy drop.*

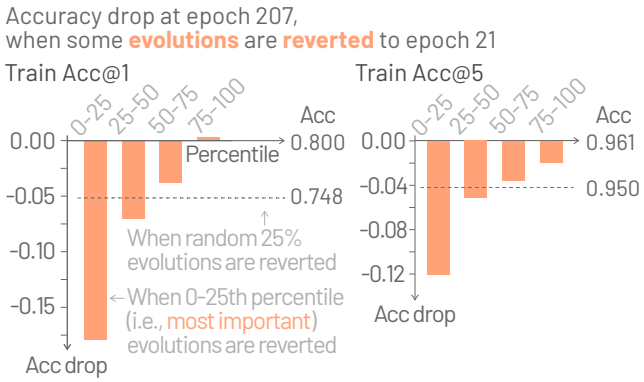

Figure 6: Evaluation on how well CONCEPTEVO can find important concept evolutions for 100 random classes. We ranked neurons in the decreasing order of evolution importance computed by CONCEPTEVO, and placed them in 4 importance bins: 0-25th (most important), 25-50th, 50-75th, 75-100th percentiles. Reverting higher-importance evolutions leads to greater accuracy drop, confirming CONCEPTEVO's effectiveness in identifying important concept evolutions. For a baseline, we also computed the accuracy drop when 25% (i.e., the same number of neurons in each bin) of randomly selected evolutions were reverted, observing that the accuracy drop of random reversion is between that of 25-50th and 50-75th percentile bin.

4.5 Discovery

Ill-suited hyperparameters harm concept diversity. Training DNNs involves various hyperparameter choices that can significantly impact the model performance. However, selecting the optimal hyperparameters involves multiple iterations of trial and error, and the quality of the selection typically cannot be assessed until late in the training process (Zeiler 2012). Discovering early symptoms of suboptimal training caused by ill-suited hyperparameters could significantly reduce the needs for trial and error, and time spent on (suboptimal) training.

CONCEPTEVO's neuron concept embedding can help

[1]Epochs 5, 21, and 207 give top-1 training accuracies closest to 25%, 50%, and 75%, respectively.

[2]Standard deviations of the average accuracy changes across the classes between the 5 runs were very low, e.g., 9.2e-5 for top-1 training and 2.1e-4 for top-1 testing, for the 21→207 evolution.

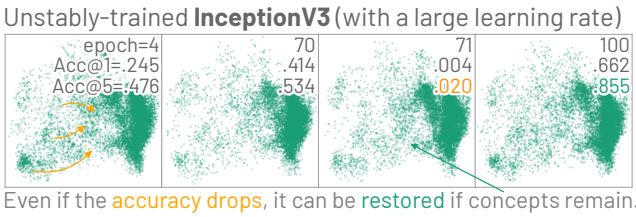

Figure 7: An unstably trained InceptionV3 with a large learning rate. The concept diversity is damaged (neuron embeddings move towards to the same location for lower-level concepts), but the model's concepts and training accuracy recover with concepts remaining at epoch 71.

identify problems due to ill-suited hyperparameters and provide possible reasons for why such problems affect model performance. For example, in Fig 2b, CONCEPTEVO shows that a VGG16 suboptimally trained with an overly high learning rate (0.05, larger than an optimal learning rate 0.01 presented in prior work) leads to a drastic drop of accuracy over training epochs. CONCEPTEVO reveals the early signs of problems, such as neuron concepts "atrophying," degrading the diversity of concepts and detecting only lower-level concepts, before the accuracy ultimately drops to 0. The loss of diversity was so severe that it could never recover, even after training for 40 more epochs. As another example, in Fig 7, CONCEPTEVO shows that an InceptionV3 trained with a large learning rate (1.5, larger than an optimal learning rate 0.045 presented in prior work) also shows a drastic drop of accuracy at epoch 70 and 71, accompanied by some damage to the concept diversity (e.g., the concept embeddings move more towards the right for the lower-level concepts). Interestingly, after a few more training epochs, both the model accuracy and concept diversity rebounded. We believe it is due to the large number of concepts that remain at epoch 70 (despite a low model accuracy).

The above examples demonstrate that CONCEPTEVO can reveal early signs of training problems and offer actionable insights to help decide whether interventions (e.g., stopping the training) might be desirable. For example, if we observe a severe damage to the concept diversity like in Fig 2b, rather than the milder damage in Fig 7, stopping the training might be better since the model is unlikely to recover even with more training.

We qualitatively examined the diversity of concepts by inspecting the concept examples in the models. We found that the last convolutional layers in both VGG16 and InceptionV3 detect very limited concepts when the accuracy is low; almost all neurons in the last convolutional layer in VGG16 and more than 50% of neurons in the last convolutional layer in InceptionV3 detected the same *"background"* concept. We provide more details of our observations in Appendix C.

We also quantitatively analyzed the concept diversity using differential entropy that measures the uncertainty or surprises in a continuous variable (Michalowicz, Nichols, and Bucholtz 2013). We measured the differential entropy for each dimension of neuron embeddings, where higher values mean more diverse concepts, and averaged the values across the dimensions. In the VGG16, the differential entropy decreased: 0.30 (epoch 3), 0.09 (epoch 12, accuracy drastically dropped), -1.11 (epoch 13, almost 0 accuracy) and -1.09 (epoch 14, almost 0 accuracy), showing the model's losing concept diversity. In the InceptionV3, the values decreased until epoch 70, and then rebounded: 0.16 (epoch 4), -0.15 (epoch 70), -0.03 (epoch 71, presenting the lowest accuracy), and 0.06 (epoch 100), showing the model's concept diversity was harmed but it got restored.

Overfitting slows concept evolution. Overfitting is a common problem in training DNNs (Srivastava et al. 2014; Rice, Wong, and Kolter 2020; Cogswell et al. 2016). Using CONCEPTEVO, we discovered that concepts in overfitted models evolved slowly despite rapid training accuracy increases. For example, in Fig 2c, we trained a modified VGG16 by removing its dropout layers that are known to help mitigate overfitting (Srivastava et al. 2014). The model overfitted expectedly: at epoch 30, its top-1 and top-5 train accuracies reached 0.99 and 1, but its top-1 and top-5 test accuracies were 0.37 and 0.61. Surprisingly, the neuron embeddings in VGG16 without dropout evolved slower than neurons in a well-trained VGG16 (Fig 2a). Numerically, to raise the top-1 training accuracy from $\sim$0.25 to $\sim$0.5, the neuron embeddings in the well-trained VGG16 model moved 0.991 in the Euclidean distance on average, while those of the overfitted VGG16 moved much slower, 0.746 in the Euclidean distance on average. Similarly, to raise top-1 training accuracy from $\sim$0.5 to $\sim$0.75, the well-trained VGG16 had an average Euclidean distance of 0.989, versus 0.657 for the overfitted VGG16.

5 Conclusion and Future Work

We have presented CONCEPTEVO, a general unified interpretation framework for DNNs that reveals the inception and evolution of detected concepts during training. Through both large-scale human experiments and quantitative experiments, we have demonstrated that our technique can discover concept evolutions that aid human interpretation of model training across different models. This helps identify potential problems during training and can provide actionable insights to address the identified problems, such as promoting informed human intervention. Such insights may help design training strategies for more stable and effective training.

In future work, we plan to examine the effect of selecting a base model, as we noticed that the quality of our unified semantic space depends on the choice of the base model. We tried different base models — a pretrained VGG19, a pretrained InceptionV3, and a fully trained VGG16 — and found that any non-overfitted models generally provide quality concept embeddings (e.g., supporting a wide range of concepts). We plan to investigate guiding principles for selecting a good base model by expanding our evaluation of concept clustering coherency and concept alignment across models to more base models. We also plan to extend our investigation to other types of models (e.g., object detectors, reinforcement learning, and language models).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. {TensorFlow}: A System for {Large-Scale} Machine Learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 265–283.

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. *IEEE conference on computer vision and pattern recognition*.

Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; et al. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, 169–174.

Chung, S.; Park, C.; Suh, S.; Kang, K.; Choo, J.; and Kwon, B. C. 2016. ReVACNN: Steering convolutional neural network via real-time visual analytics. In *Future of interactive learning machines workshop at the 30th annual conference on neural information processing systems (NIPS)*.

Cogswell, M.; Ahmed, F.; Girshick, R.; Zitnick, L.; and Batra, D. 2016. Reducing overfitting in deep networks by decorrelating representations. *The International Conference on Learning Representations (ICLR)*.

Dabkowski, P.; and Gal, Y. 2017. Real time image saliency for black box classifiers. *Advances in neural information processing systems*, 30.

Das, N.; Park, H.; Wang, Z. J.; Hohman, F.; Firstman, R.; Rogers, E.; and Chau, D. H. P. 2020. Bluff: Interactively Deciphering Adversarial Attacks on Deep Neural Networks. *IEEE Visualization Conference*.

Fong, R.; and Vedaldi, A. 2018. Net2Vec: Quantifying and Explaining How Concepts Are Encoded by Filters in Deep Neural Networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 8730–8738. Computer Vision Foundation / IEEE Computer Society.

Ghorbani, A.; Wexler, J.; Zou, J.; and Kim, B. 2019. Towards automatic concept-based explanations. *Neural Information Processing Systems*.

Ghorbani, A.; and Zou, J. Y. 2020. Neuron shapley: Discovering the responsible neurons. *Advances in Neural Information Processing Systems*, 33: 5922–5932.

Goyal, Y.; Shalit, U.; and Kim, B. 2019. Explaining Classifiers with Causal Concept Effect (CaCE). *CoRR*, abs/1907.07165.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*.

Hernandez, E.; Schwettmann, S.; Bau, D.; Bagashvili, T.; Torralba, A.; and Andreas, J. 2022. Natural Language Descriptions of Deep Features. In *International Conference on Learning Representations*.

Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C.; Wexler, J.; Viegas, F.; et al. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *International conference on machine learning*.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. *International Conference on Machine Learning*.

Laugel, T.; Lesot, M.-J.; Marsala, C.; Renard, X.; and Detyniecki, M. 2019. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *International Joint Conference on Artificial Intelligence*.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

Li, M.; Zhao, Z.; and Scheidegger, C. 2020. Visualizing neural networks with the grand tour. *Distill*, 5(3): e25.

Liu, M.; Shi, J.; Cao, K.; Zhu, J.; and Liu, S. 2017. Analyzing the training processes of deep generative models. *IEEE transactions on visualization and computer graphics*, 24(1): 77–87.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Michalowicz, J. V.; Nichols, J. M.; and Bucholtz, F. 2013. *Handbook of differential entropy*. Crc Press.

Nguyen, A. M.; Yosinski, J.; and Clune, J. 2016. Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks. *Visualization for Deep Learning workshop at ICML*.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3): e00024–001.

Olah, C.; Mordvintsev, A.; and Schubert, L. 2017. Feature visualization. *Distill*, 2(11): e7.

Papernot, N.; and McDaniel, P. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*.

Park, H.; Das, N.; Duggal, R.; Wright, A. P.; Shaikh, O.; Hohman, F.; and Chau, D. H. P. 2021. NeuroCartography: Scalable Automatic Visual Summarization of Concepts in Deep Neural Networks. *IEEE Transactions on Visualization and Computer Graphics*.

Pezzotti, N.; Höllt, T.; Van Gemert, J.; Lelieveldt, B. P.; Eisemann, E.; and Vilanova, A. 2017. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE transactions on visualization and computer graphics*, 24(1): 98–108.

Pühringer, M.; Hinterreiter, A.; and Streit, M. 2020. InstanceFlow: Visualizing the Evolution of Classifier Confusion at the Instance Level. In *2020 IEEE Visualization Conference (VIS)*, 291–295. IEEE.

Raghu, M.; Gilmer, J.; Yosinski, J.; and Sohl-Dickstein, J. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30.

Rauber, P. E.; Fadel, S. G.; Falcao, A. X.; and Telea, A. C. 2016. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1): 101–110.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why should I trust you?" Explaining the predictions of any classifier. *ACM SIGKDD international conference on knowledge discovery and data mining*.

Rice, L.; Wong, E.; and Kolter, Z. 2020. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, 8093–8104. PMLR.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. *IEEE international conference on computer vision*, 618–626.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*.

Smilkov, D.; Carter, S.; Sculley, D.; Viégas, F. B.; and Wattenberg, M. 2017. Direct-manipulation visualization of deep networks. *arXiv preprint arXiv:1708.03788*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. *IEEE conference on computer vision and pattern recognition*.

Yeh, C.; Kim, B.; Arik, S. Ö.; Li, C.; Pfister, T.; and Ravikumar, P. 2020. On Completeness-aware Concept-Based Explanations in Deep Neural Networks. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*.

Yosinski, J.; Clune, J.; Nguyen, A.; Fuchs, T.; and Lipson, H. 2015. Understanding neural networks through deep visualization. *International Conference on Machine Learning (ICML) Deep Learning Workshop*.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, Q.; Wang, W.; and Zhu, S.-C. 2018. Examining cnn representations with respect to dataset bias. *AAAI Conference on Artificial Intelligence*.

Zhong, W.; Xie, C.; Zhong, Y.; Wang, Y.; Xu, W.; Cheng, S.; and Mueller, K. 2017. Evolutionary visual analysis of deep neural networks. In *ICML Workshop on Visualization for Deep Learning*, 9.

Zhou, B.; Bau, D.; Oliva, A.; and Torralba, A. 2018. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 41(9): 2131–2145.

Appendix

A Neuron Embedding

We present supplementary details for Step 1 (i.e., creating the base semantic space) of our neuron embedding method described in Sec 3.2. Specifically, we elaborate how we formulate the neuron embedding approach as a maximum likelihood estimation problem which results in the objective function in Eq (1), and how we learn neuron vectors by optimizing the objective function. The code that we have provided as a part of the supplementary materials has implemented these details and what we presented in the main paper. Our code is ready to be open-sourced, and we will do so upon paper publication.

A.1 Sampling Pairs of Neurons That Are Strongly Co-activated by Many Common Images

We first describe how we sample pairs of neurons that are co-activated by many common images. Specifically, we describe how to compute D, a multiset of strongly co-activated neuron pairs (n, m), where more pairs (n, m) will appear in D as more images are shared by stimuli of neurons n and m. We summarize the sampling algorithm into three phases.

- **Phase 1:** For each image $\mathbf{x}$, we compute a set of neurons in the base model that are strongly co-activated by $\mathbf{x}$. We denote the set of co-activated neuron pairs as $N^{t_b}_{M_b,\mathbf{x}}$, where M_b is the base model and t_b is the epoch where M_b is at.
- **Phase 2:** We sample neuron pairs from $N^{t_b}_{M_b,\mathbf{x}}$ and add the pairs into D.
- **Phase 3:** We aggregate the neuron pairs for all images, by running Phase 1 and Phase 2 for all images.

Phase 1: Compute strongly co-activated neurons for each image. We define $N^{t_b}_{M_b,\mathbf{x}}$, a set of neurons strongly co-activated by an input image $\mathbf{x}$ for the base model M_b at epoch t_b, as in Eq (8). N_{M_b} is a set of all neurons in M_b, and $X^{t_b}_{n,M_b}$ is a set of stimuli of neuron n in M_b at t_b.

$$N^{t_b}_{M_b,\mathbf{x}} = \{n \in N_{M_b} \mid \mathbf{x} \in X^{t_b}_{n,M_b}\} \tag{8}$$

Phase 2: Sample pairs of neurons that are strongly co-activated for each image. We compute D, a multiset of pairs of strongly co-activated neurons in M_b. For each image $\mathbf{x}$, we randomly order neurons in $N^{t_b}_{M_b,\mathbf{x}}$ and aggregate neuron pairs with a sliding window of length 2 over the shuffled neurons. We add the sampled pairs into D.

Phase 3: Aggregate the neuron pairs for all images. We finally aggregate the neuron pairs for all images, by running Phase 1 and Phase 2 for all images.

A.2 Formulation of Neuron Embedding as a Maximum Likelihood Estimation Problem

We describe our neuron embedding approach as a maximum likelihood estimation: how can we learn neuron embeddings that best represent the similarity between neurons' concepts based on neurons' co-activation?

We formulate the likelihood of similarity between two neurons n and m as in Eq (9). In the equation, M_b is the base model, t_b is the epoch where M_b is at, $\sigma(\cdot)$ is the sigmoid function (i.e., $\sigma(x) = 1/(1 + e^{-x})$), and $\mathbf{v}^{t_b}_{n,M_b}$ is the embedding of neuron n in M_b at t_b.

$$P(n, m) = \sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b}) \tag{9}$$

We then define the likelihood objective function to maximize as in Eq (10). In the equation, V^{M_b,t_b} is the set of neuron embeddings in the base model M_b at epoch t_b. D is the multiset of pairs of neurons that are strongly co-activated by many common images, as described in A.1. r is a randomly-sampled neuron in M_b at t_b. R is the number of randomly-sampled neurons for each neuron pair (n, m). Intuitively, ① a pair of neurons with larger inner product of their embeddings has a higher likelihood of co-activation and higher similarity of their concepts, and ② randomly paired neurons are likely to have lower value of inner product and less likely to be conceptually similar.

$$\begin{aligned}
&P(D|V^{M_b,t_b}) \\
&= \prod_{(n,m)\in D} \Bigg(\underbrace{P(n, m)}_{\text{① Co-activated neuron pairs}} \cdot \underbrace{\prod_{r=1}^{R} (1 - P(n, r))\,(1 - P(m, r))}_{\text{② Random neuron pairs}} \Bigg) \\
&= \prod_{(n,m)\in D} \Bigg(\underbrace{\sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b})}_{\text{① Co-activated neuron pairs}} \cdot \\
&\qquad \underbrace{\prod_{r=1}^{R} (1 - \sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b}))\,(1 - \sigma(\mathbf{v}^{t_b}_{m,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b}))}_{\text{② Random neuron pairs}} \Bigg)
\end{aligned} \tag{10}$$

Based on Eq (10), we define the log-likelihood objective function J_1 to maximize, as in Eq (11) or as in Eq (1) in the main paper.

$$\begin{aligned}
J_1 = \sum_{(n,m)\in D} \Bigg(&\log\left(\sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b})\right) + \\
&\sum_{r=1}^{R} \log\left(1 - \sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\right) \log\left(1 - \sigma(\mathbf{v}^{t_b}_{m,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\right) \Bigg)
\end{aligned} \tag{11}$$

We randomly initialize the neuron embeddings and then learn the embeddings by using gradient ascent that optimizes the objective function. For each pair of strongly co-activated neurons $(n, m) \in D$, we compute the derivative to update the neurons' embedding as in Eq (12) and (13).

$$\begin{aligned}
\frac{\partial J_1}{\partial \mathbf{v}^{t_b}_{n,M_b}} &= (1 - \sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b}))\,\mathbf{v}^{t_b}_{m,M_b} \\
&\quad - \sum_{r=1}^{R} \sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\,\mathbf{v}^{t_b}_{r,M_b}
\end{aligned} \tag{12}$$

$$\frac{\partial J_1}{\partial \mathbf{v}^{t_b}_{m,M_b}} = (1-\sigma(\mathbf{v}^{t_b}_{n,M_b}\cdot\mathbf{v}^{t_b}_{m,M_b}))\,\mathbf{v}^{t_b}_{n,M_b} - \sum_{r=1}^{R}\sigma(\mathbf{v}^{t_b}_{m,M_b}\cdot\mathbf{v}^{t_b}_{r,M_b})\,\mathbf{v}^{t_b}_{r,M_b} \quad (13)$$

Eq (14) and (15) present the derivative of the sigmoid and the log of sigmoid function used to calculate Eq (12) and (13).

$$\begin{aligned}\frac{d\sigma(x)}{dx} &= \frac{d\frac{1}{(1+e^{-x})}}{dx} = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \frac{-(-e^{-x})}{(1+e^{-x})^2}\\ &= \frac{(e^{-x}+1-1)}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})}\cdot\frac{(1+e^{-x})-1}{1+e^{-x}}\\ &= \sigma(x)\cdot\big(1-\sigma(x)\big)\end{aligned} \quad (14)$$

$$\begin{aligned}\frac{d\log(\sigma(x))}{dx} &= \frac{\sigma'(x)}{\sigma(x)}\\ &= \frac{\sigma(x)(1-\sigma(x))}{\sigma(x)} \qquad \cdots \text{by Eq (14)}\\ &= 1-\sigma(x)\end{aligned} \quad (15)$$

A.3 Pseudocode for Neuron Embedding

Algorithm 1 presents a pseudocode for our neuron embedding approach. It takes the base model (i.e., M_b), the epoch where M_b is at (i.e., t_b), the set of all input images (i.e., $\mathbf{X}$), the number of epochs for learning neuron embeddings (i.e., E), and the learning rate for neuron embeddings (i.e., α) as input. It returns the set of neuron embeddings (i.e., V^{M_b,t_b}). Algorithm 1 calls functions `getStimuli()` (algorithm 2), `getStronglyCoActivatedNeurons()` (algorithm 3), `sampleNeuronPairs()` (algorithm 4) inside.

A.4 Hyperparameter Settings

As presented in the main paper, for k (the number of stimuli per neuron), we experimented with values from 5 to 30 and found 10 provided good conceptual coherence of neighboring neurons on the unified space. Also as presented in the main paper, for the dimension of the neuron and image embeddings, we experimented with values from 5 to 100 and found 30 provided good semantic coherence among neighboring neurons on the unified space and led to results computed in a reasonable time (<5 hours). For E (the maximum number of epochs for neuron embedding), we experimented with values from 3 to 30,000, and we found 10,000 provided good semantic coherence among neighboring neurons on the unified space and led to results computed in a reasonable time (<5 hours). For R (the number of randomly selected neuron per neuron pair), we evaluated the values in the range of 1 to 10 inclusively and arrived at the value of 3 that produced the most coherent concept clustering. For α (the learning rate), we evaluated the values between the range of 0.01 to 0.5 (inclusively) and arrived at the value of 0.05 that produced coherent concept clustering in a reasonable time (less than 5 hours).

Algorithm 1: Neuron embedding

Input: M_b: the base model, t_b: the epoch which M_b is at, $\mathbf{X}$: the set of all images, and hyperparameters: k, E, α, R
Output: V^{M_b,t_b}: the set of neuron embeddings

// Get stimuli of all neurons
N_{M_b} := the set of all neurons in M_b
X := a list of length $|N_{M_b}|$
for $n \in N_{M_b}$ **do**
  $X^{t_b}_{n,M_b}$ = getStimuli(n, $\mathbf{X}$, M_b, k)
  $X[n] = X^{t_b}_{n,M_b}$
end for

// Sample neuron pairs
$D :=$ an empty multiset
for $\mathbf{x} \in \mathbf{X}$ **do**
  $N^{t_b}_{M_b,\mathbf{x}}$ = getStronglyCoActivatedNeurons($\mathbf{x}$, X, N_{M_b})
  D = sampleNeuronPairs(D, $N^{t_b}_{M_b,\mathbf{x}}$)
end for

// Learn neuron embeddings
V^{M_b,t_b} = a list of randomly initialized neuron embeddings
for i in [1, ..., E] **do**
  for $(n,m) \in D$ **do**
    $\mathbf{v}^{t_b}_{n,M_b} = V^{M_b,t_b}[n]$
    $\mathbf{v}^{t_b}_{m,M_b} = V^{M_b,t_b}[m]$
    $\mathbf{v}^{t_b}_{n,M_b} = \mathbf{v}^{t_b}_{n,M_b} + \alpha\frac{\partial J_1}{\partial \mathbf{v}^{t_b}_{n,M_b}}$ (Eq (12))
    $\mathbf{v}^{t_b}_{m,M_b} = \mathbf{v}^{t_b}_{m,M_b} + \alpha\frac{\partial J_1}{\partial \mathbf{v}^{t_b}_{m,M_b}}$ (Eq (13))
    $V^{M_b,t_b}[n] = \mathbf{v}^{t_b}_{n,M_b}$
    $V^{M_b,t_b}[m] = \mathbf{v}^{t_b}_{m,M_b}$
  end for
end for

Return V^{M_b,t_b}

B Concept Evolution for Class Prediction

B.1 More class examples

We provide additional examples for Sec 3.3 by presenting concept evolutions that are important to class predictions. For example, Figure 8, 9, 10, 11, 12, and 13 show examples of concept evolutions in a VGG16 for classes "Shetland sheepdog", "Ladybug", "Payphone", "Oxcart", "Fire engine", and "Cassette". For each class, we present three important concept evolutions out of all evolutions that have a score larger than 0.8, accounting for the top 0.5% most important evolutions according to the importance scores measured by Eq (7) in the main paper.

Algorithm 2: getStimuli

Input: n: a neuron, M_b: the base model, $\mathbf{X}$: the set of all images, and a hyperparameter k for the length of stimuli

Output: $X^{t_b}_{n,M_b}$: the stimuli of n

$X^{t_b}_{n,M_b}$:= an empty list of length k
A := a list of length k filled with -inf
l = the layer of n in M_b
for $\mathbf{x} \in \mathbf{X}$ **do**
  $Z_{n,l,M_b}(\mathbf{x})$ = activation map of n in l of M_b given $\mathbf{x}$
  $M = Max(Z_{n,l,M_b}(\mathbf{x})) \in \mathbb{R}$
  if $A[k-1] < M$ **then**
    i = the smallest index such that $M \geq A[i]$
    for $j \in [k-1, ..., i+1]$ **do**
      $A[j] = A[j-1]$
      $X^{t_b}_{n,M_b}[j] = X^{t_b}_{n,M_b}[j-1]$
    end for
    $A[i] = M$
    $X^{t_b}_{n,M_b}[i] = \mathbf{x}$
  end if
end for

Return $X^{t_b}_{n,M_b}$

Algorithm 3: getStronglyCoActivatedNeurons

Input: $\mathbf{x}$: an image, X: stimuli of all neurons, N_{M_b}: the set of all neurons

Output: $N^{t_b}_{M_b,\mathbf{x}}$: strongly co-activated neuron pairs for $\mathbf{x}$

$N^{t_b}_{M_b,\mathbf{x}}$:= an empty set
for neuron $n \in N_{M_b}$ **do**
  $X^{t_b}_{n,M_b} = X[n]$ (i.e., n's stimuli)
  if $\mathbf{x} \in X^{t_b}_{n,M_b}$ **then**
    $N^{t_b}_{M_b,\mathbf{x}} = N^{t_b}_{M_b,\mathbf{x}} \cup \{n\}$
  end if
end for

Return $N^{t_b}_{M_b,\mathbf{x}}$

Algorithm 4: sampleNeuronPairs

Input: D: a previous set of neuron pairs, $N^{t_b}_{M_b,\mathbf{x}}$: strongly co-activated neuron pairs for $\mathbf{x}$

Output: D: the updated set of neuron pairs

RN = the randomly ordered $N^{t_b}_{M_b,\mathbf{x}}$
$s = |N^{t_b}_{M_b,\mathbf{x}}|$
for i in [0, ... $s-1$] **do**
  $p = (RN[i], RN[i+1])$
  $D = D \cup \{p\}$
end for

Return D

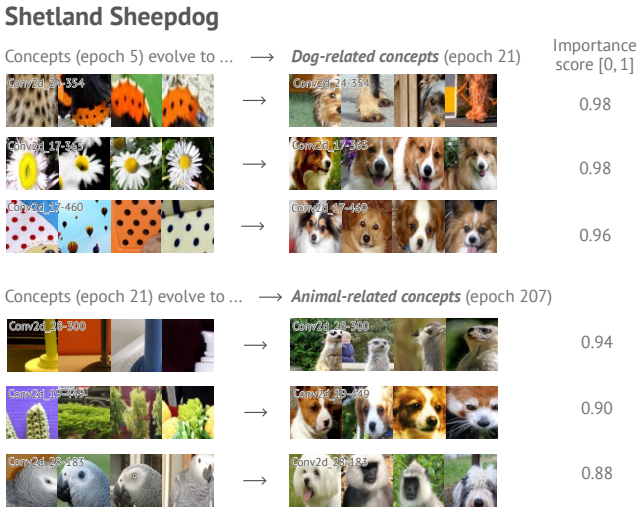

Figure 8: Concept evolutions in a VGG16 important for "Shetland sheepdog" class.

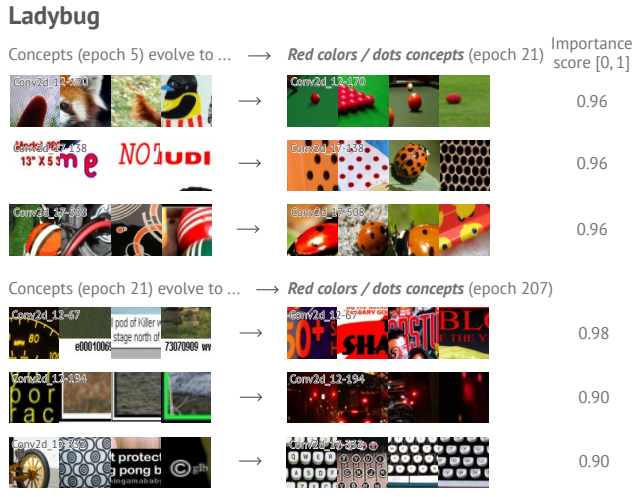

Figure 9: Concept evolutions in a VGG16 important for "Ladybug" class.

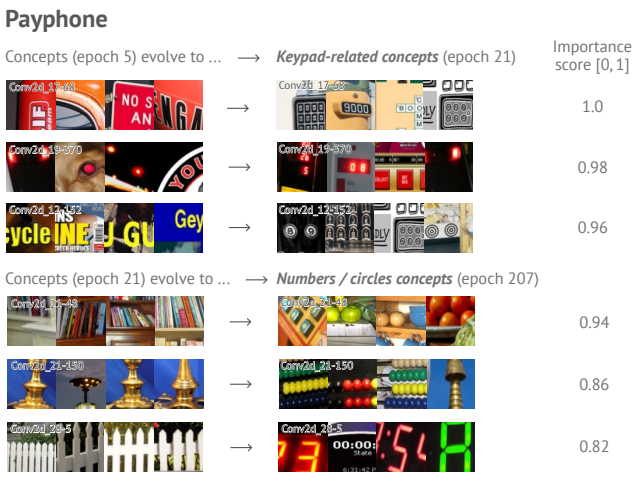

Figure 10: Concept evolutions in a VGG16 important for "Payphone" class.

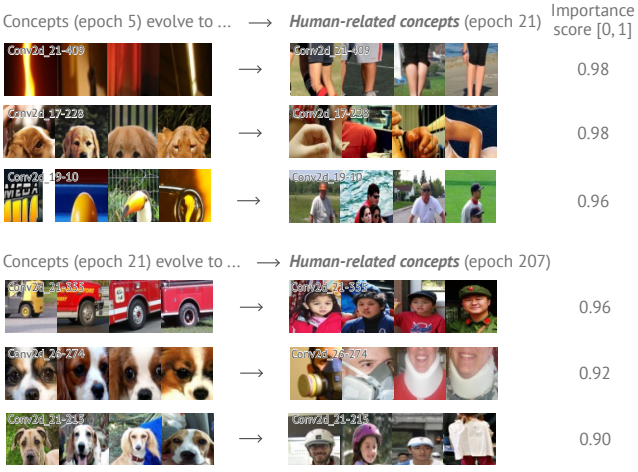

Figure 11: Concept evolutions in a VGG16 important for "Oxcart" class.

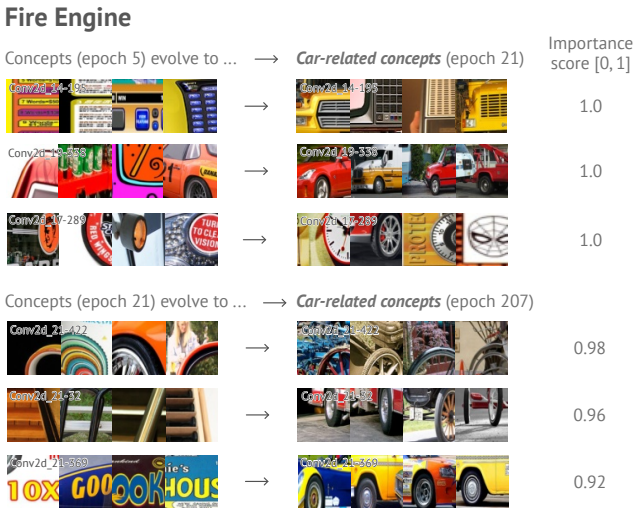

Figure 12: Concept evolutions in a VGG16 important for "Fire engine" class.

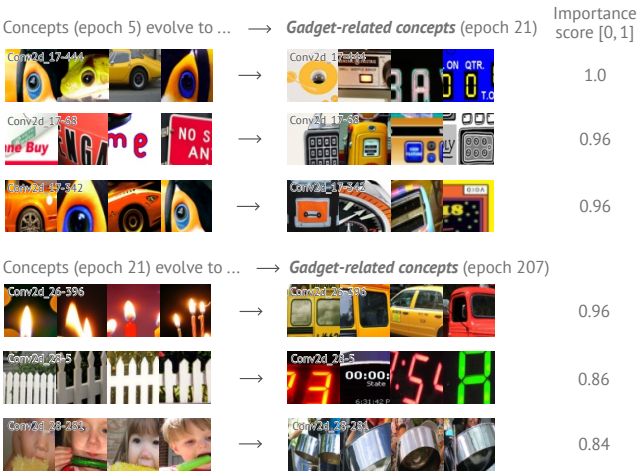

Figure 13: Concept evolutions in a VGG16 important for "Cassette" class.

Evolution		0-25	25-50	50-75	75-100	Random
5→21	Train@1	-.129	-.037	-.017	.007	-.036
	Train@5	-.120	-.041	-.025	-.005	-.039
	Test@1	-.123	-.034	-.014	-.012	-.033
	Test@5	-.124	-.039	-.021	.023	-.038
21→207	Train@1	-.178	-.070	-.037	.003	-.051
	Train@5	-.120	-.051	-.035	-.019	-.041
	Test@1	-.150	-.060	-.029	.023	-.041
	Test@5	-.139	-.057	-.032	.003	-.042

Table 1: Accuracy changes when concept evolutions are reverted, compared to the accuracies with no reversion. The first column: the evolution stages (epoch 5→21 epoch 21→207). The second column: the type of accuracy (training / test, or top1 / top5 accuracy). The last column: accuracy changes when random 25% evolutions are reverted. The other columns: accuracy changes when evolutions of 0-25th (the most important), 25-50th, 50-75th, 75-100th (the least important) percentiles are reverted.

Figure 14: A concept detected by models that are trained with overly large learning rates, such as a VGG16 trained with the learning rate of 0.05 and an InceptionV3 trained with the learning rate of 1.5.

B.2 Evaluation of Importance of Concept Evolution

In Sec 4.4, we present the training accuracy changes when evolutions in a well trained VGG16 from epoch 21 to 207 were reverted, which shows that reverting higher importance evolutions leads to greater accuracy drop. Table 1 provides additional results for the evolution stages of 5→21 and 21→207, showing detailed top-1 and top-5 accuracy changes for training and testing, which shows similar pattern to our finding.

C Ill-suited Hyperparameters Can Damage Concept Diversity

In Sec 4.5 of the main paper, we describe our observation that ill-suited hyperparameters can damage concept diversity, leading to a drastic drop of the accuracy. In particular, as the accuracy drops, a large number of neurons become low-level detectors. For example, We found that the last convolutional layers in both VGG16 and InceptionV3 that are trained with overly high learning rate detect very limited concepts when the accuracy is low. Almost all neurons in the last convolutional layer in VGG16 and more than 50% of neurons in the last convolutional layer in InceptionV3 detected the same "background" concept (Fig 14 shows an example).