UNIVERSITY OF CALIFORNIA SAN DIEGO

Concept-Monitor: Using Concept Embeddings to Understand Neural Net Training

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Computer Science

by

Mohammad Ali Khan

Committee in charge:

Professor Lily Weng, Chair
Professor Albert Chern, Co-Chair
Professor Tzu-Mao Li

2023

The thesis of Mohammad Ali Khan is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Lily Weng for her continued support as the chair of my committee and my advisor. Through multiple drafts and many long nights, her guidance has proved to be invaluable to the development of this work.

I would also like to acknowledge Tuomas Oikarinen and Prof Lily Weng who are co-authors on my paper ”Using Concept Embeddings to Understand Neural Network Training”, which serves as the basis for this thesis.

VITA

2015–2019 Bachelor of Technology, Indian Institute of Technology, Delhi

2021–2023 Master of Science, University of California San Diego

FIELDS OF STUDY

Major Field: Computer Science

ABSTRACT OF THE THESIS

Concept-Monitor: Using Concept Embeddings to Understand Neural Net Training

by

Mohammad Ali Khan

Master of Science in Computer Science

University of California San Diego, 2023

Professor Lily Weng, Chair
Professor Albert Chern, Co-Chair

In this work, we propose a general framework called Concept-Monitor to help demystify the black-box DNN training processes automatically using a novel unified embedding space and concept diversity metric. Concept-Monitor enables human-interpretable visualization and indicators of the DNN training processes and facilitates transparency as well as deeper understanding on how DNNs develop along the during training. Inspired by these findings, we also propose a new training regularizer that incentivizes hidden neurons to learn diverse concepts, which we show to improve training performance. Finally, we apply Concept-Monitor to conduct several case studies on different training paradigms including adversarial training, fine-tuning

and network pruning via the Lottery Ticket Hypothesis.

Introduction

Unprecedented success of deep learning has led to its rapid application to a wide range of tasks; however, deep neural networks (DNNs) are also known to be complex and non-interpretable. It is not very well understood as to how a DNN model makes a decision, what kind of concepts in the inputs it used to make that decision and to what extent its decisions are based on correct parts of the inputs. Therefore, inorder to deploy these DNN models in the real-world, especially for safety-critical applications such as healthcare and autonomous driving, it is imperative for us to understand what is going behind the black box.

Lots of research effort has focused on developing methods to interpret black-box DNNs, such as attributing DNN's predictions to individual input features and identify which pixels or features are the most important [33, 26, 28, 27] or investigating the functionalities (also known as *concept*) of each individual-neuron (or channel of a CNN) [31, 21, 4, 18, 13, 20].

However, most of these methods only focus on examining a DNN model *after* it has been trained, and therefore missing out useful information that could be available in the training process. For example, it would be very useful for deep learning researchers and engineers to understand *what the concepts learned by the DNN model are* and *how the concepts evolve along the training process*.

At the current stage, training DNNs is still considered a black-box and trial-and-error process. Making the training process more human-interpretable and transparent, can significantly benefit the research in deep learning because **(i)** it can shed light on why and how DNNs learn, which could be helpful to inspire new and improved DNN training algorithms; **(ii)** it can also help to debug DNNs and prevent catastrophic failure if anything goes wrong.

Motivated by the above need, in this paper we propose Concept-Monitor, which is an automatic and efficient pipeline to make the black-box neural network training more transparent and interpretable. Our pipeline tracks and visualizes the training progress with human-interpretable concepts of individual neurons, which provides useful insights of the DNN model as a whole. Fig 1 gives a schematic overview of the Concept-Monitor. Our technical contributions can be summarized as below:

- We develop a natural language based embedding space which allows us to efficiently track how the neurons' concepts evolve and visualize their semantic evolution throughout the training process.
- We provide four case studies (standard training, adversarial training, lottery ticket hypothesis and fine-tuning) to analyze various deep learning training paradigms and discover insights into how and why these alternative training processes succeed. In these studies we uncover insights like how lottery ticket pruning encodes concepts in the pruning mask and how adversarialy trained models rely more on color as compared to standard model
- We propose a quantitative metric of concept diversity to measure how diverse of a set of concepts the network is learning. Building upon this metric, we further propose a novel training modification to encourage concept diversity and show it increases accuracy and interpretability.

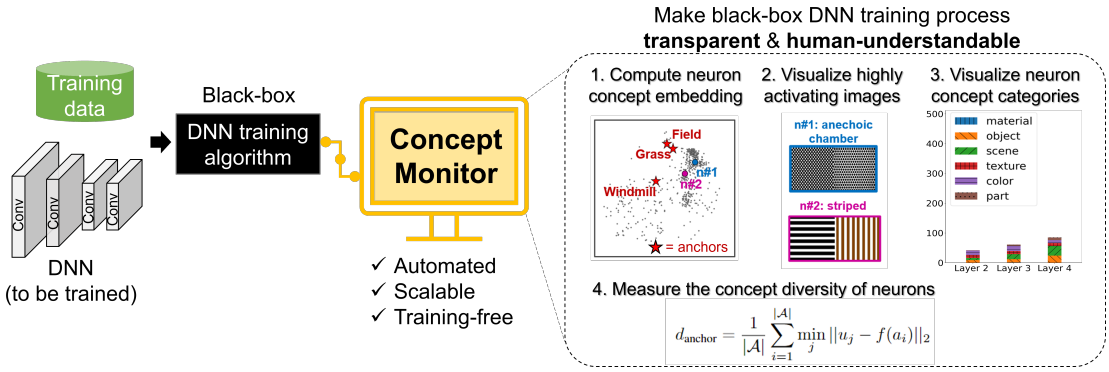

Figure 1. Our proposed Concept-Monitor is automated, scalable, training-free and makes DNN training process more transparent and human understandable. It consists of 4 key steps to understand each training iteration.

Chapter 1

Background and Related works

1.1 Neuron-level interpretability methods

A number of methods have been proposed to understand and interpret the roles of individual neurons in neural networks, which we call *neuron*-level interpretability methods. These include visualization based methods such as [9, 31, 22] and methods based on manual inspection [32, 21, 12].

More recently, methods have been proposed to automatically describe the role of neurons without need for human input, such as [5, 18, 20, 13]. Network dissection and its variation [5, 18] require a curated probing data labelled with pre-defined concepts. The key idea of Network Dissection is to identify *concepts* of neurons by calculating an Intersection over Unit (IoU) score of intermediate activation maps and pre-defined concept masks.

However, this approach is limited by the need of a curated probing dataset *annotated* with concept labels, which may be expensive and time-consuming to collect. A new version of Network Dissection [6] tries to alleviate this limitation by using segmentation models to label the concepts, although the segmentation models still require dense concept labels to train. Alternatively, a recent work CLIP-Dissect [20] tries to address this challenge differently by leveraging the paradigm of multi-modal models [24] to allow automatic identification of neuron concepts without the need of collecting concept labelled data or densely annotated data. We note that these techniques [5, 18, 20] are all compatible to our proposed Concept-Monitor to facilitate

automatic concept monitoring on the DNN training process. We demonstrate the versatility of our Concept-Monitor by showing the experimental results with different concept detectors in the Appendix for monitoring standard DNN training process.

1.2 Understanding DNN training dynamics

Most of the existing works are primarily focused on analyzing models *after* training instead of investigating how the concepts change *during* the DNN training process, which is the main focus of our work. A recent work [23] also proposes inspecting concepts of neurons during training, which is similar to our goals, but their proposed approach is substantially different from our approach and may come with some limitations as discussed below. First, their method requires learning a universal semantic space for each neuron from a base model, while our approach does not need to perform any training. Their approach could be expensive as re-training is required when the base model or the probing dataset is changed. Second, their approach may be hard to automate, because human intervention is required to describe the behavior of each neuron, which may not be scalable to large models. In contrast, our method is fully automated and does not require human input.

1.3 Interpretable training

Our *concept diversity regularizer* proposed in Section 3 is perhaps most closely related to interpretable training methods such as Concept Whitening [8] or the learning of Concept Bottleneck Layer in [19]. However, these methods are technically different from our approach and focused on increasing interpretability of a NN model, while our *concept diversity regularizer* is aimed to increase network performance such as accuracy.

Chapter 2

Our method

In section 2.1 we detail the key components in Concept-Monitor including the concept detector, the unified embedding space, and the concept diversity metric. A full pipeline of how to use Concept-Monitor is described in section 2.2. Next in section 2.3, we use Concept-Monitor to demystify the standard training process of a deep vision model (shown in Fig 2.1) and discuss the results and insights.

2.1 Components

2.1.1 (I) Concept Detector

The first part of our method is to use a concept detector ϕ to automatically identify the concept of a neuron at any stage in the training. Given a set of concept words S and a probing image dataset D_{probe}, a concept detector ϕ would return a concept word $w^n_{1'}$ for a neuron n that maximally activates it. To monitor DNN training process with automatic concept monitoring, we define a similarity metric $\texttt{sim}^n_i$ which characterize how well a neuron n is described to a concept w_i. Note that we use the notation $w^n_{1'}$ to denote the best concept for neuron n and w_i to be the ith concept word in the concept set S: i.e. $1' = \text{argmax}_i \texttt{sim}^n_i$. This allows us to unify existing neuron-interpretability methods in this framework – for example, the similarity $\texttt{sim}^n_i$ will be the IoU (Intersection over Union) score between nth neurons activation map and the ith concept mask in Network-Dissection [4], and the similarity $\texttt{sim}^n_i$ will be the similarity metric

(e.g. cosine similarity, soft-WPMI) between nth neuron activations and the ith concept activations in CLIP-Dissect [20]. In addition, we say a neuron n is an *interpretable neuron* if $\texttt{sim}^n_{1'} > \tau$, where τ is a threshold dependent on the concept detector ϕ, and $w^n_{1'}$ is the concept of this neuron.

2.1.2 (II) Unified embedding space

The second part of our method is to define a unified embedding space in order to visually track neurons' evolution. Here we detail the steps to project a neuron n into our embedding space. Let f be the text encoder of a pretrained large language model (e.g. the text encoder from CLIP [24]). First, we compute the text embedding v_i of each concept word w_i in the concept set S: $v_i = f(w_i)$. Next, we use these text embeddings $\{v_1, v_2, \ldots, v_{|S|}\}$ as the basis of our semantic space and project neurons into this space using a weighted linear combination of v_i. We use the concept detector ϕ to compute $\texttt{sim}^n_i$ for all concept words w_i and neurons n, which are subsequently used to calculate weighting using softmax with temperature T. The embedding u_n of neuron n is then calculated as:

$$u_n = \sum_{i=1}^{|S|} \lambda_i^n f(w_i), \;\; \lambda_i^n = \frac{e^{\texttt{sim}^n_i/T}}{\sum_{j=1}^{|S|} e^{\texttt{sim}^n_j/T}} \tag{2.1}$$

where λ_i^n is the weight describing the similarity of concept w_i to neuron n. Finally, we visualize the concept embedding u_n in two dimensional space using UMAP [17] in plot such as Fig 2.1 (column 1). We can use the concept embedding plot in the unified embedding space to track the concept evolution of each neuron easily.

Another benefit of our unified embedding space is that we can project any general concept word α into the same embedding space by calculating its text embedding $f(\alpha)$. This lets us mark the embedding space with concept "anchors" (see the green stars in Fig 2.1, column 1), which could be concepts that a researcher thinks should exist in a well trained model, or undesirable concepts such as ones representing bias. With the concept-anchors, researchers can then track

whether and which neurons are converging or diverging away from anchors.

2.1.3 (III) Concept diversity metric

Another benefit of our neuron concept visualization via unified embedding space (e.g. Fig 2.1) is that it allows us to easily sense the diversity of concepts represented by the neurons. As we will see in Sections 2.3, neurons of well trained models typically cover a large set of concepts, while poor training typically leads to a lack of concept diversity. Inspired by this, we propose a quantitative metric *anchor distance* to measure concept diversity based on our unified embedding space. The idea behind this metric is that we have a set of text anchors $A = \{a_1, a_2, ..., a_{|A|}\}$ that ideally describes concepts that are important for this task, and we want to make sure at least one neuron in the network has a concept similar to each anchor. Thus, we define the *anchor distance* d_{anchor} as follows:

$$d_{\text{anchor}} = \frac{1}{|A|} \sum_{i=1}^{|A|} \min_j ||u_j - f(a_i)||_2 \tag{2.2}$$

where u_j are the neuron embeddings as defined in Eq.(2.1). This metric measures the average Euclidian distance of the closest neuron to each concept in the anchor set. As such, networks with highly diverse neurons will reach low *anchor distance* while highly clustered neurons will results in a large d_{anchor}. For most of our experiments we set $A = S$, where both of them are the set of labels in Broden dataset [4], but they can be easily changed based on task and user needs.

2.2 Using Concept-Monitor

With all the components in place, we now describe how they come together to form Concept-Monitor for DNN training. Concept-Monitor offers a combination of metrics and visualizations to analyze a snapshot of a model, and by analyzing at consecutive snapshots we can understand how the model evolves in terms of concepts learned by individual neurons. The schematic of Concept-Monitor is illustrated in Figure 1. At each snapshot, concept monitor

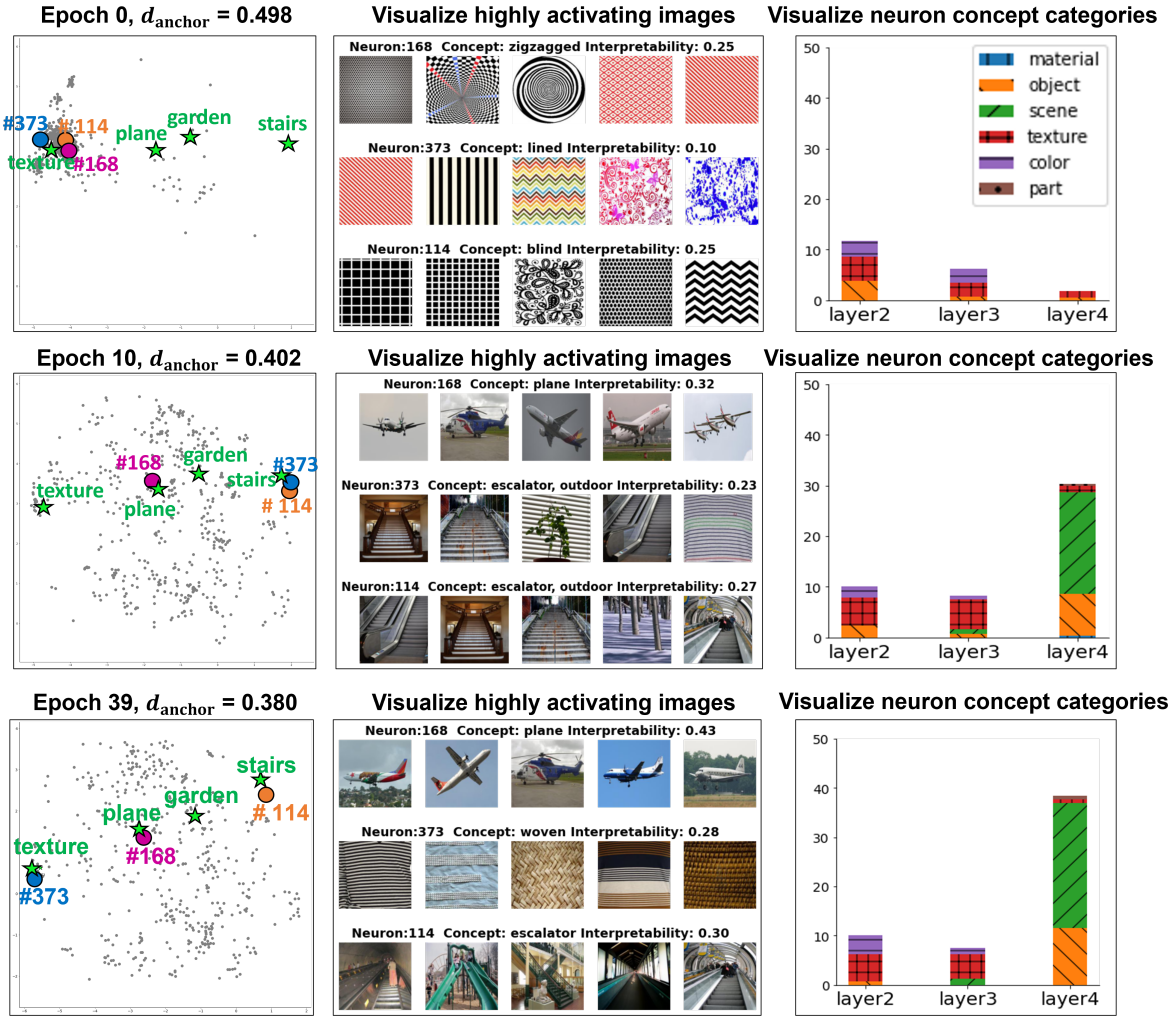

Figure 2.1. Case study (I): Monitoring standard training. We use Concept-Monitor to analyze a standard trained Resnet-18 model on Places365 dataset. We visualize the training at three different epochs, specifically tracking the trajectories of neurons #114 (orange circle),#168 (pink circle) and #373 (blue circle) of layer 4. The 1st column plots our unified embedding space, where each gray dot represents a neuron in layer 4 and green stars represent anchor words. The tracked neurons are coloured differently for visualization. The 2nd column shows the highly activating images of the tracked neurons along with their similarity to the closest concept. Finally the 3rd column shows the percentage of interpretable neurons in layer 2-4 and which category they belong to.

produces the following:

1. Two dimensional plot of neuron embeddings, and select text anchors (marked as the stars in the plot).

2. Visualization of detected concepts and most highly activating images for each neuron.

3. Bar plot visualization of the number of interpretable neurons and which category they belong to.

4. A numeric measure of concept diversity d_{anchor} as defined in Eq. (2.2).

A visualization of all these information for different epochs of standard models is shown in Figure 2.1. We think this information should be combined with standard metrics such as accuracy and losses to monitor training progress in an interface similar to TensorBoard [1]. This information can be helpful in several ways, for example, if we see that concept diversity starts to decrease, it indicates issues in the training. In Sections 2.3 and 4 we apply concept monitor on different training runs, and in section 3 we propose a training modification inspired by our insights from Concept-Monitor.

2.3 Case study (I): Monitoring standard training

Now we use Concept-Monitor to investigate standard training of ResNet-18 model on Places365 dataset with CLIP-Dissect as the concept detector. We study how the concepts evolve across training and whether there is a correlation between accuracy and concept generalization with Concept-Monitor. We investigate the concept evolution of neurons at different epochs in the training process using the full pipeline described in Section 2.2. The main results are plotted in Fig 2.1, where row 1 represents a very early snapshot (trained for 1 epoch, at the end of Epoch 0), row 2 an intermediate snapshot (Epoch 10) and row 3 the final snapshot (Epoch 39, training ends). The 1st column visualizes our proposed unified embedding: we use green star symbols to show the anchors embeddings $f(a_i)$ for anchors [texture, plane, garden, stairs], and we use the circle symbols to show 3 neuron embeddings u_n – neuron #114 (colored orange), #168 (colored pink) and #373 (colored blue). The 2nd column visualizes the highly activating images for these 3 neurons, which should match the detected concept (word) displayed on top of it (e.g. neuron

#168 in Epoch 0 has detected concept ”zigzagged” on the graph). The 3rd column visualizes the percentage of interpretable neurons from the 6 pre-defined categories. Now we summarize three observations from the standard training below:

1. *Model learns to look at more complex features as training progresses.* - As shown in Figure 2.1 second column, initially all neurons start by detecting low level features like lines, patterns and textures. This can also be seen as all neurons being clustered around the *textures* anchor in the embedding space (column 1) and absence of 'object' and 'scene' categories (column 3). We see that as training progresses, the model starts to learn more complex features which can be clearly seen from the highly activating images in column 2 and bar plots in column 3.

2. *Shallower layers learn more low-level features like material and texture while deeper layers learn more nuanced object detectors.* - We consider the broad categories of [*Material, Object, Scene, Texture, Color, Part*] to group neurons similar to the labels used in Broden dataset. We note that the categories *Scene*, *Object* and *Part* are concerned with higher level concepts like *planes* and *stairs* while *Texture* and *Color* are concerned with lower-level concepts like *lined*, *zigzagged* etc. From column 3 in Figure 2.1, it's evident that Layer 2 and Layer 3 are learning a lot more low level information than Layer 4 as the texture and color neurons are represented more.

3. *Concept diversity happens relatively early in the training.* - Using the unified embedding space, we can see that the neurons are clumped together initially in the embedding plot and as training progresses they spread out eventually converging to their finalt concept. However, we note that this divergence occurs during the earlier stages of the training and after that the neurons mostly stay close to their concepts. For example in Figure 2.1 column 1, we see that neuron #114 (orange circle) and neuron #168 (pink circle) reach and retain their position for the last 30 epochs of training and keep their concepts as seen in highly activating images.

2.3.1 Discussion of standard training.

Using our method in standard training, we have seen a correlation between training stage and interpretability of a model (represented as the number of interpretable neurons in the Column 3 of Fig 2.1). We notice that for a well trained model, there is a progression from a low level concepts understanding to higher level conceptual understanding, and the concept diversity increases as training progresses. We also run Concept-Monitor using Network Dissection [4] and MILAN [14] as the concept detectors in Appendix (see Fig C.2 and Fig C.3) and show our observations are consistent across different concept detectors.

2.3.2 Poor training.

Next, we use Concept-Monitor to investigate standard training with poor hyper-parameter selection, specifically the standard training in Section 3.3 but with a fixed high learning rate. As expected, the large learning rate made it impossible for the model to train properly with the final accuracy being 3%. We visualize the unified embedding plot of the poorly trained model snapshots (in the 2nd row) and contrast it with the standard trained model snapshots (in the 1st row) in Figure 2.2. It can be seen that unlike standard training, the concept diversity doesn't increase in the poor training but instead plateaus, which means the poorly trained model has a higher d_{anchor} This is also confirmed with the calculated d_{anchor} value: d_{anchor} for poor training is 0.47 compared to 0.38 in the well-trained standard model, showing the inability of the model to learn diverse concepts.

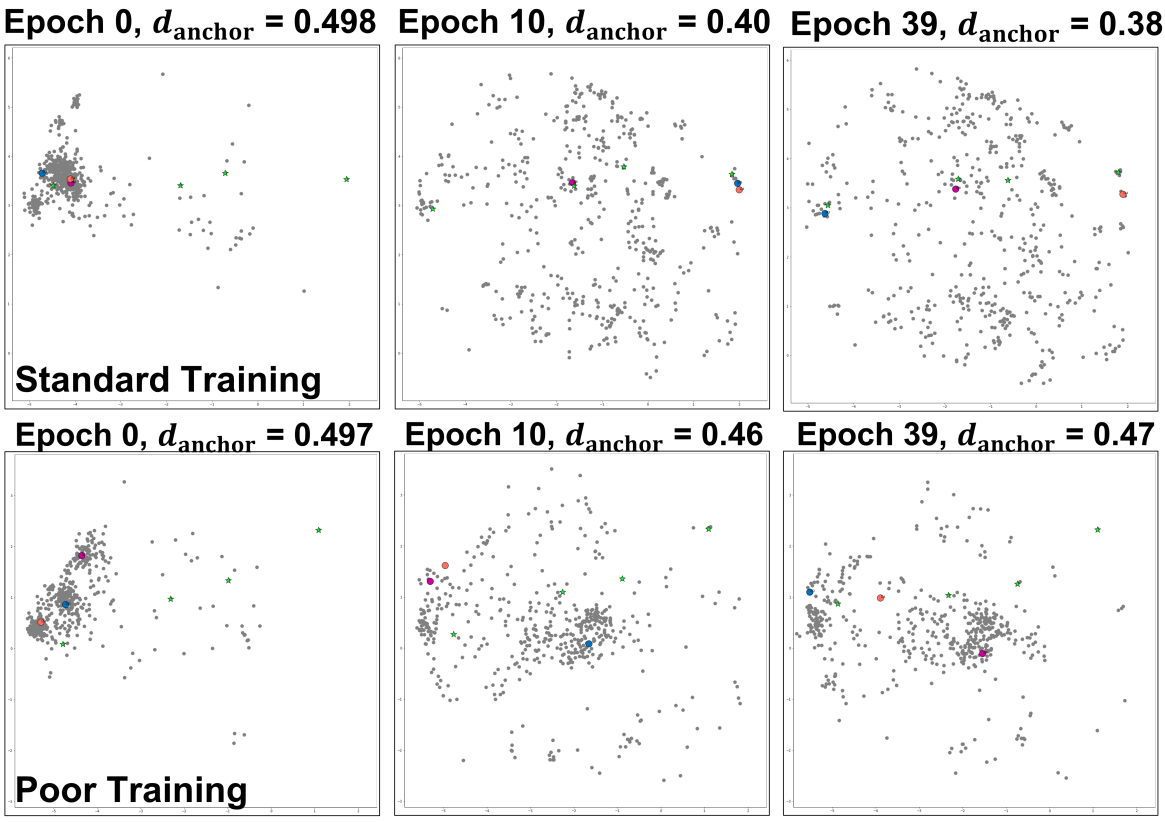

Figure 2.2. Investigating poor training using unified embedding space. We see that in poor training the neurons don't diverge much along training. The high d_{anchor} also indicates inability to represent useful concepts.

Chapter 3

Concept Diversity Regularizer

In Section 2.3, we find that the neurons of well trained models usually cover a large variety of concepts while poor training often leads to neurons clustering together. Inspired by this, we propose a regularizer to increase concept diversity based on the *anchor distance* d_{anchor} from Section 2.1. We find that it can improve model accuracy, interpretability and concept diversity as shown in Section 3.1.

In order to include the *anchor distance* in training, d_{anchor} needs to be differentiable. By plugging the neuron embeddings u_j in Eq. (2.1) into Eq. (2.2), we can rewrite d_{anchor} as below:

$$\frac{1}{|A|}\sum_{i=1}^{|A|}\min_j ||\sum_{i=1}^{|S|}(\frac{e^{\texttt{sim}_i^j/T}}{\sum_{k=1}^{|S|}e^{\texttt{sim}_k^j/T}})\cdot f(w_i) - f(a_i)||_2. \tag{3.1}$$

Recall that f is a text encoder of a pretrained large language model, and $\texttt{sim}_i^j$ is a similarity metric that characterize how close a neuron j is related to a concept w_i based on the selected concept detector ϕ. In our setting, f is frozen and fixed while $\texttt{sim}_i^j$ contains the trainable parameters of the DNN classifier model (through ϕ). Thus, if $\texttt{sim}_i^j$ can be made differentiable, then d_{anchor} becomes a differentiable function of the DNN parameters, and as such can be used be used as a regularizer. In Network Dissection, $\texttt{sim}_i^j$ is an IoU score that is not differentiable, but in CLIP-Dissect, $\texttt{sim}_i^j$ can be made differentiable if using the differentiable *cos cubed* similarity function defined in [19]. Thus, we use the *cos cubed* similarity function and introduce the

Table 3.1. Comparison between standard models and models trained with our Concept Diversity Regularizer on Places365.

Model	Accuracy(%)	Number of interpretable neurons	d_{anchor}
Res18 std	47.48 ± 0.13	106.33 ± 4.04	0.39 ± 0.007
Res18 ours	$\mathbf{48.33 \pm 0.08}$	$\mathbf{147.7 \pm 7.02}$	$\mathbf{0.36 \pm 0.004}$
Res50 std	49.12 ± 0.44	$\mathbf{544.4 \pm 41.02}$	0.31 ± 0.003
Res50 ours	$\mathbf{49.32 \pm 0.02}$	361 ± 15.7	$\mathbf{0.27 \pm 0.006}$

concept diversity regularizer directly in the training to reduce the d_{anchor}

$$L = L_{\text{std}} + \beta d_{\text{anchor}} \tag{3.2}$$

where L_{std} is the standard loss such as cross-entropy, d_{anchor} is defined in Eqn. (3.1) and β is a hyper parameter to decide the relative importance of the two losses. Our goal is to minimize Eq. (3.2) and learn a model that has a reduced *anchor distance*.

3.1 Results

To test the performance of our regularizer, we train a model on a subset of Places365 like we did in section 2.3. We trained it exactly like before, using a mini-batch stochastic gradient descent as the optimizer but with the new joint loss function of Eq. (3.2). The regularizer d_{anchor} was calculated over the neurons in the second to last layer (layer4) of the network. Typically we calculate d_{anchor} using the Broden dataset, but this would be too expensive to compute during training. Instead we simply use a minibatch of Places training data as D_{probe} which greatly reduces computational overhead at the cost of introducing more noise to the neuron embeddings. We used $\beta = 1$ for our experiments.

Table 3.1 shows that training with our regularizer improved average test accuracy by 1% point over the standard well-trained Resnet18 model with no further optimization beyond adding the regularizer. In addition, it increased the concept diversity and number of interpretable

neurons on the second to last layer. Note that the increase in interpretability is not caused by overfitting as we used Broden as D_{probe} in Table 3.1, which was not used during training. We also see a small improvement in average accuracy for Resnet50 in addition to increased concept diversity. The results of this initial experiment show the promise of encouraging concept diversity in training.

Chapter 4

Case studies of other training paradigms

In this section, we show that Concept-Monitor is versatile and can be used to study various training paradigms to gain insights into how and why they work. We also provide useful observations and insights that could help future researchers better understand these training procedures.

4.0.1 Case study (II): Lottery ticket hypothesis

Lottery Ticket Hypothesis (LTH) [11] proposes that there exists a sub network inside a model which is already very close to the desired model, the pruning technique introduced by LTH is a popular method to prune DNNs without sacrificing their performance. In this case study, we use Concept-Monitor to better understand the success behind LTH. The main idea of LTH is to use iterative magnitude pruning (IMP) to prune the model by repeating the steps of training, pruning and rewinding to an initial epoch. LTH hypothesizes the existence of ”winning tickets” at initialization which are sub-networks within the network that can be trained to performance equivalent to the original model. It has been noted in literature that the process of pruning encodes some information in the pruning mask [34]. To understand this, we use Concept-Monitor to investigate LTH through the lens of interpretability. We train a Resnet18 on CIFAR 10 dataset using IMP in 8 stages. For full details on our experimental setup, please refer to Appendix A. A related study was done by [10] which found that pruning doesn't affect the interpretability of the model until there is a significant drop in accuracy. For our experiment

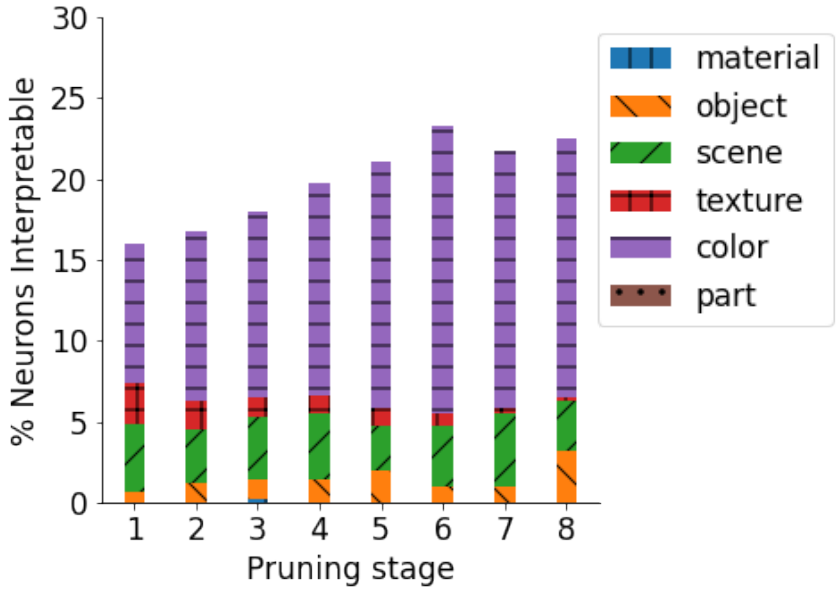

Figure 4.1. A barplot of the percentage of neurons that are interpretable after successive stages of pruning weights and rewinding remaining weights back to their initializations in our LTH experiment. We can see the number of interpretable neurons increases after simply setting some weights to 0.

we focus on the original Lottery Ticket Hypothesis, where after each stage of pruning we rewind the weights all the way back to their initialization.

Observations and Discussion.

In our analysis, we make the following observation: *Pruning the network learns to encode some concepts without any fine tuning.* Fig 4.1 shows the fraction of interpretable neurons in layer 4 of the model *after* each stage of pruning and rewinding. We notice that even though we are rewinding to the initial weights, the number of interpretable neurons *increases*. Since the weights are randomly initialized, the only way there can be a gain in interpretable neurons is through the changes that happen during pruning, i.e. the zeroing out of certain weights in the network. We believe that the model may be learning to remove connections that are harming the network, which leads to neurons ”learning” simple interpretable concepts such as colors already at initialization. Interestingly, we also notice the number of texture neurons decreases as pruning progresses. We note that a related phenomenon was observed by [34] who find that

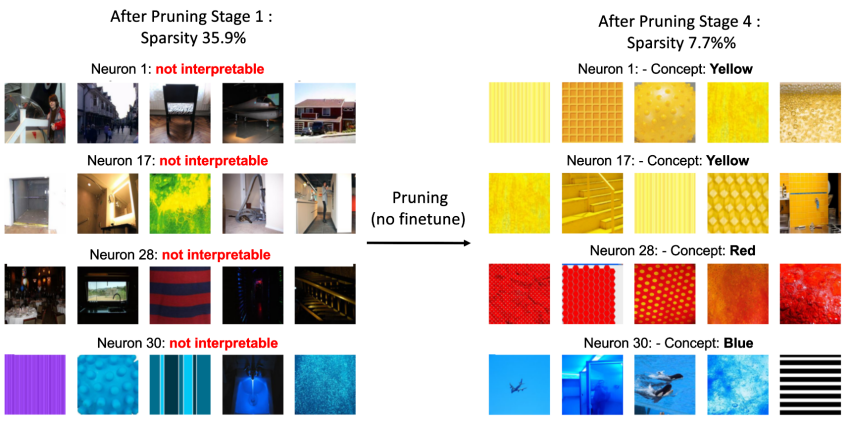

Figure 4.2. Select neurons from layer4 of our LTH rewind to zero model that were uninterpretable at after one stage of pruning (and rewind to initialization) but became interpretable by the end of 4th pruning stage. Note no weights were changed between the two, interpreatbility was caused purely by zeroing some weights.

IMP zeros out weights that would ultimately go towards zero anyway after training. Hence, they hypothesize that a pruned initial network encodes a portion of the training process itself, which they refer to as ”masking is learning”. This could explain why we see interpretable neurons with just pruned initial weights. Further proof of this can be illustrated in the experiment result in Figure 4.2, which shows some initially uninterpretable neurons (left panel) that ’learn’ to encode simple concepts such as colors through pruning only and become interpretable (right panel).

4.0.2 Case study (III): Adversarial Training

DNNs are known to be vulnerable against small perturbations in their inputs [29]. This is problematic as networks can fail unexpectedly after small random or adversarial perturbations which raises concerns over their safety. Fortunately, methods have been developed to defend against adversarial attacks, most popular of these being Adversarial Training [16]. This successfully makes networks more robust against such attacks, but comes at a cost of degraded performance on clean test data. In this study, we apply Concept-Monitor to adversarial training to better understand how adversarial training changes a network and why standard accuracy suffers. We analyse a Resnet18 model trained on CIFAR10 *with* and *without* adversarial training. For full details on our experimental setup, please refer to Appendix section A.

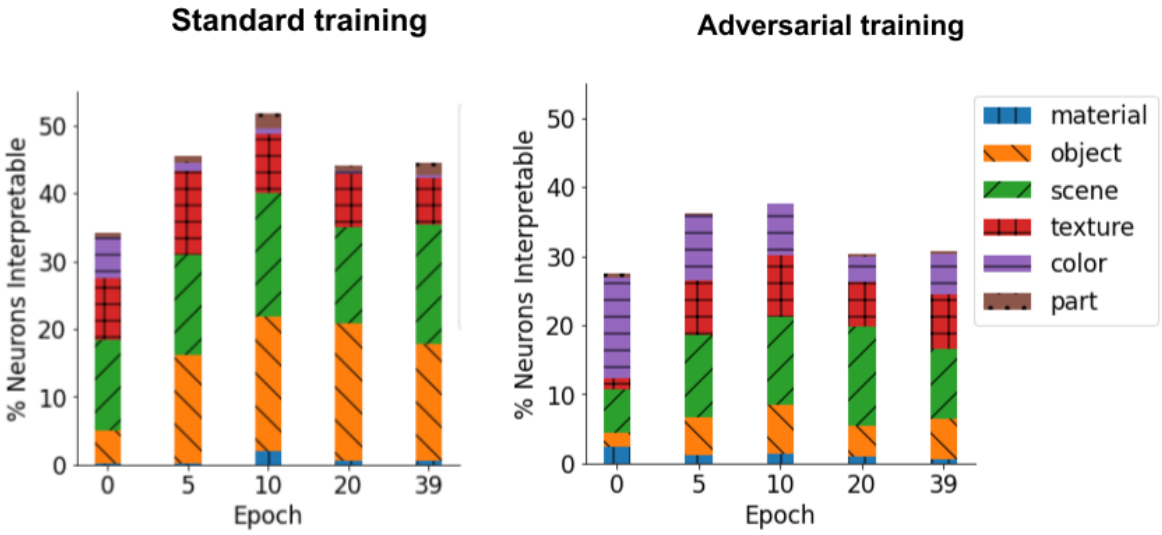

Figure 4.3. Comparison of the types of concepts learned by standard training compared to adversarial training in the second to last layer (layer4), and differences in types of concepts learned by the models. Note these figures are the network at the end of each epoch, so epoch 0 is after one epoch of training.

Observations and Results

Using Concept-Monitor we have the following two observations from Figure 4.3 and 4.4. These observations are consistent across multiple trained models but for simplicity we focus our discussion on one model and its visualization.

1. *Adversarially robust network relies more on colors, standard model moves on to higher level concepts.* In Figure 4.3 we observe that robust model has a lot more interpretable neurons dedicated to detecting ”color” (the purple bar) than the standard model (30 vs 2) at the end of training. On the other hand, the trained robust model has less interpretable neurons in the object ”scene” and ”part” categories. This finding is sensible these categories more often rely on high frequency patterns that are easily affected by l_∞ noise, therefore the adversarial training forces the model to rely less on them and rely more on more resilient features like color. Interestingly, early in their training the concepts of the two models look more similar but they start diverging after epoch 5.
2. *Standard training develops many neurons detecting target classes in the second to last*

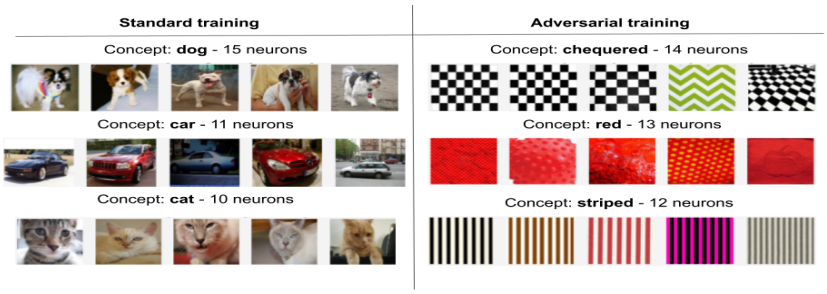

Figure 4.4. Examples of the most common concepts detected by layer4 of Resnet-18 for both standard and adversarial training. We can see a large difference between the types of concepts encoded, where standard training learns many neurons already detecting CIFAR-10 classes such as car, while adversarial model detects simple patterns and colors.

layer, robust training does not. As seen in Fig 4.4, the standard network has many neurons detecting the target classes of CIFAR-10 present in the second to last layer. For example, the fully trained standard network has 15 interpretable neurons detecting dogs, 11 neurons for car and 10 neurons detecting cats in layer4. In comparison the robust network has 1,3,3 respectively which indicates a limited capacity to represent these classes. The standard model learns to rely on target class neurons early on in training, with many of them present by epoch 10.

Discussion

We find that adversarial training harms the ability of the network to detect higher level concepts. Since these concepts are necessary for many tasks, losing them may be a significant cause for the degradation in standard performance. This may also be related to the robust networks inability to detect target class objects in second to last layer.

On the other hand, the features learned by the robust network are more general and less task specific, as seen by larger diversity in concept types in Fig. 4.3 and lack of target classes in Fig. 4.4. This could explain why [25] found adversarially robust models to have better features for transfer learning. In effect standard model features could be overfitting to the training task.

4.0.3 Case study (IV): Finetuning on a medical dataset

In this section, we use Concept-Monitor to observe the fine-tuning of a pretrained DNN on a diabetric retinopathy dataset [2]. The purpose of this experiment is to show that Concept-Monitor can be applied to a different domain such as medical data and gather insights into the process of finetuning a pretrained model. The setup details are in Appendix A.

Observations and results

We observe that for the initial weights, as the neurons are pretrained on Imagenet, they show a lot of diverse and high level concepts (as shown in highly activating image of epoch 0 in the Fig 4.5). However as the training progresses, we notice that more neurons get activated by textural concepts like dots and patterns rather than objects. This is what we expect because as the model gets better at classifying retinopathy images shown in Fig A.2 (in the Appendix), we expect it to rely more on textures and presence of ”dots” which is consistent to what we observe here as shown by the highly activating images of the top interpretable neurons in epochs 5 and 29 in Fig 4.5. From Fig 4.5 we can also see that the number of interpretable neurons in the higher level categories like ”object” and ”scene” category decrease and the interpretable neurons in the ”texture” and ”material” category remain the same or increase. This further confirms our theory that the model learns to focus on the textural aspect of images more. This is also confirmed by the semantic embedding space where we see the neurons becoming less separated. We further confirmed this by calculating the average pairwise distance between neurons which decreased from 0.735 to 0.726.

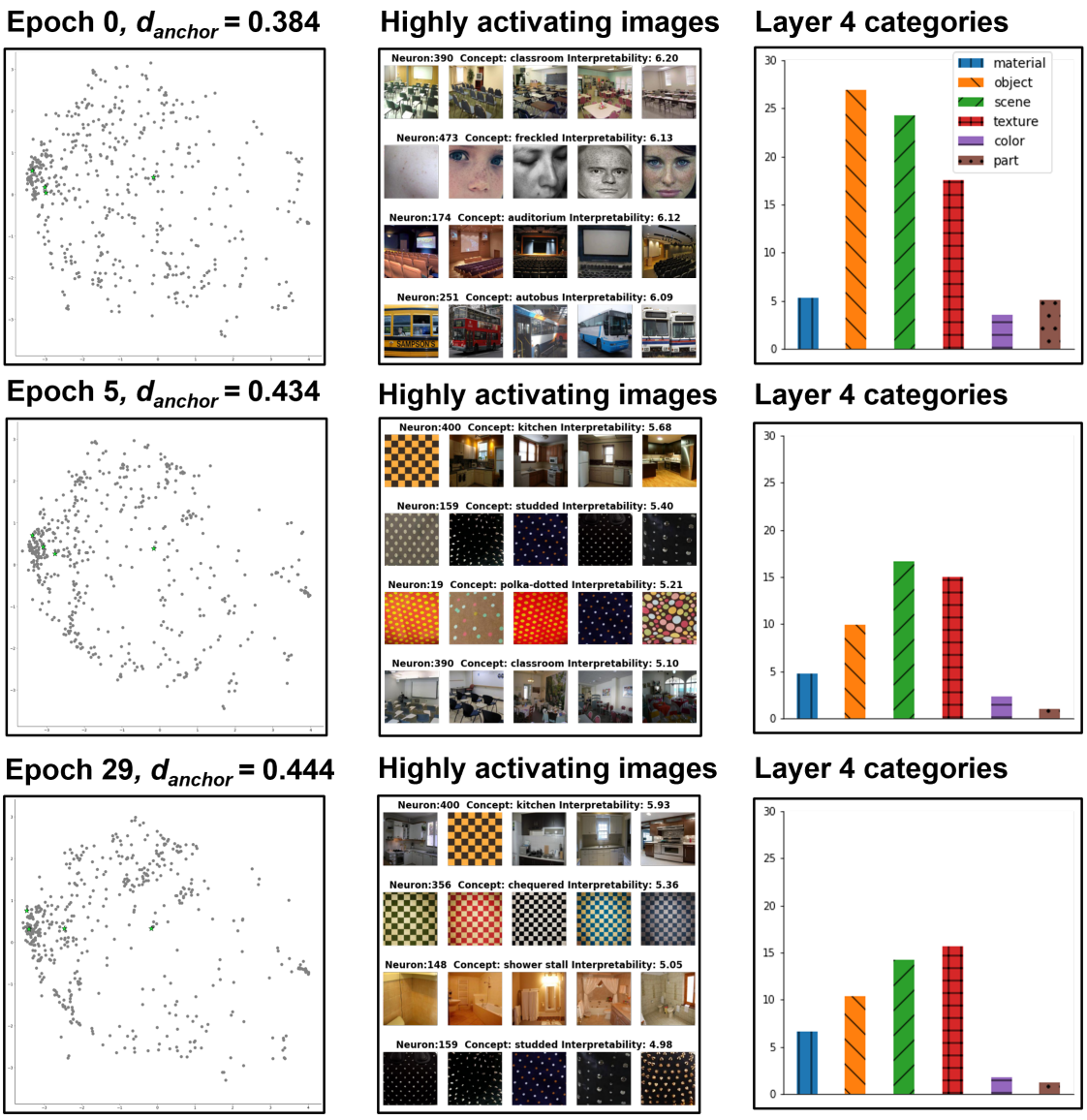

Figure 4.5. We use Concept-Monitor to analyze the finetuning a Resnet-34 network on a diabetic retinopathy dataset. On the left we plot our semantic embedding space, where it can be seen that as the training progresses the neuron mass becomes less spread apart, as is also evident from the increasing anchor metric. In the middle we visualize the activating images of *the most interpretable* neurons and we observe that the network starts to focus more on textural aspects. This is also evident from visualizing the category bar plots on the right, where we plot the percentage of interpretable neurons in layer 4 of the network. Each bar represents a different category. We see that the neurons representing complex categories like ”object”, ”scene” decrease while neurons representing ”material” and ”texture” either remain the same or increase as the training progresses. We also see that texture makes up the majority of interpretable neurons in Epoch 29

Appendix A

Experimental setup

A.1 Case study (I): Standard training

A.1.1 Setup

We train a Resnet-18 model on Places-365 dataset, which contains a lot of diverse classes allowing the DNN model to learn diverse concepts. To reduce the training time, we randomly selected 1000 images for each of the 365 classes and trained for 40 epochs reaching top-1 accuracy of 47.5%. We use batch size of 256 and an initial learning rate of 0.1 with cosine annealing scheduler.

A.1.2 Probing methodology

We use Broden [4] dataset as D_{probe} and use associated concept labels as a decoupled concept set S. Our embedding space, as described in section 3.1, is computed using CLIP's text embeddings of Broden labels as a basis. We used CLIP-Dissect with cosine-cubed similarity function and neuron embedding temperature $T = 0.01$.

A.2 Case study (II): Lottery ticket hypothesis experiments

A.2.1 Setup

We train ResNet 18 on CIFAR 10 dataset using IMP as in the LTH paper [11], rewinding to different initial weights. For each stage of IMP we train the model for 160 epochs, prune

40% of the weights and rewind to initialization. After 8 stages we got an accuracy of 91.18% on the validation set as compared to 94.32% after stage 0. Please refer to [7] implementation for reference.

A.2.2 Probing methodology

For our D_{probe}, we use Broden as the probing dataset and for concept set S we use broden labels. We use CLIP-Dissect with SoftWPMI similarity function and embedding temperature $T = 0.1$.

A.3 Case study (III): Adversarial Learning experiments

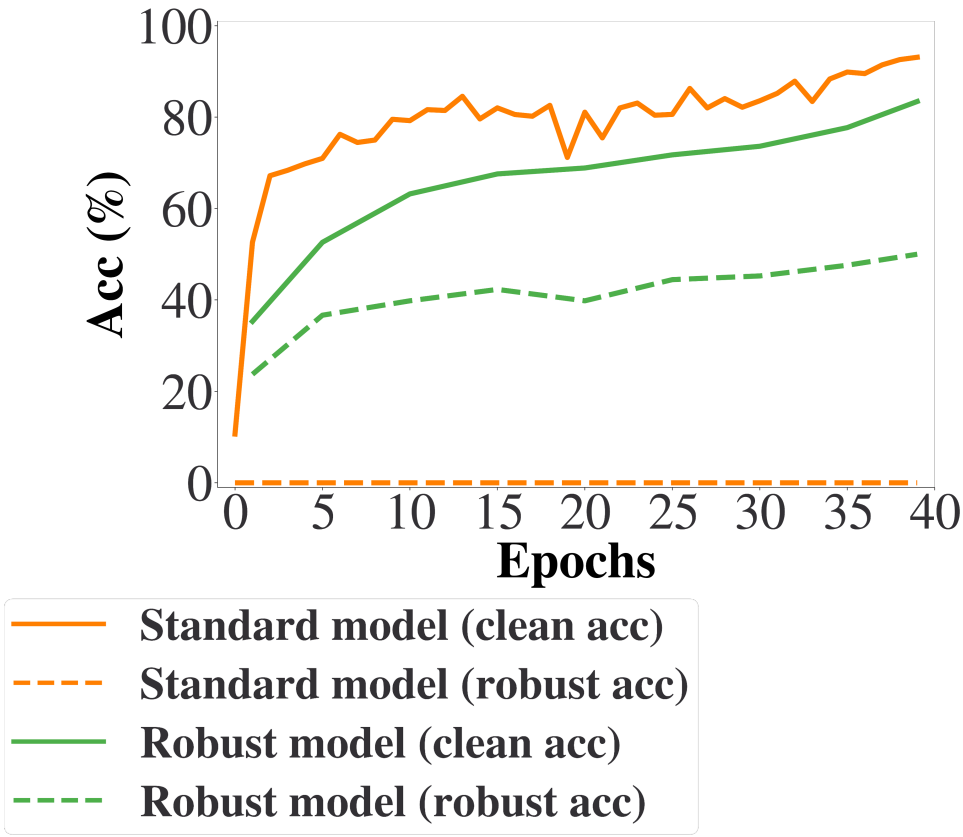

Figure A.1. Accuracy vs Epoch for standard and robust model

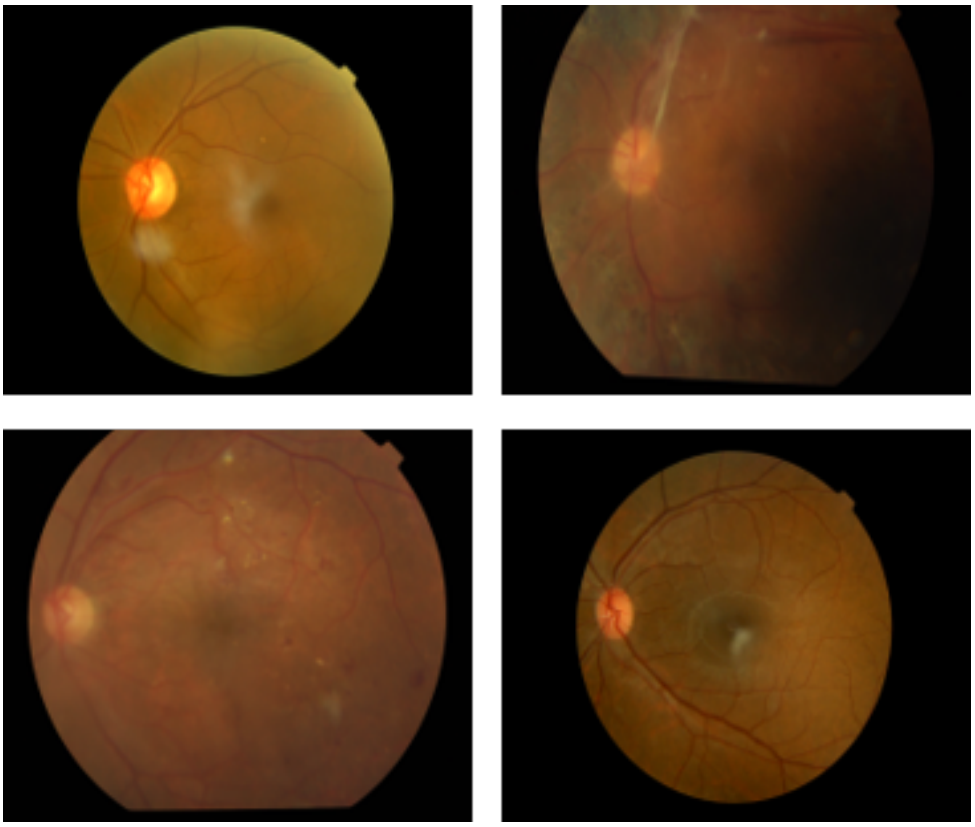

Figure A.2. Sample images in the diabetic retinopathy detection training dataset. The key features (e.g. dots and texture) are being detected by the interpretable neurons in Fig 4.5.

A.3.1 Setup

We perform adversarial training with PGD attacks on a ResNet-18 architecture. We follow reop [30] and train the network with $\varepsilon = 8/255$ and l_∞ perturbations for 40 epochs. We compare it against a CIFAR-10 network trained using the same exact training setup but no adversarial training. The standard model reaches a final accuracy 94.29%, while the robust model reaches 83.42% accuracy on clean data and 50.00% robust accuracy against a PGD adversarial attack as shown in Fig A.1. The standard model expectedly has accuracy close to 0% on adversarial images.

A.3.2 Probing methodology

We use the same probing methodology as in Case study (II), Broden images as D_{probe} and for concept set S we use the broden labels as the concepts can be easily categorized. We use CLIP-Dissect with SoftWPMI similarity function and $T = 0.1$.

A.4 Case study (IV): Fine-tuning on medical dataset

A.4.1 Setup

We used ResNet-34 backbone pretrained on ImageNet dataset as our feature extractor and used a simple linear layer as the classification head. We trained this network on the diabetic retinopathy classification dataset [2] (Fig A.2) and it achieved an accuracy of 72.77%. We followed the work from [3] for our experiments. We use Broden as D_{probe} and broden labels as S.

Appendix B

Ablation studies

In this section we study the effects of varying the temperature and threshold parameters.

B.1 Temperature (T)

Intuitively from Equation (2.1), the temperature parameter decides which concepts to consider for calculating a neuron's embeddings. Lower temperature implies that the Concept Detector is more confident in its prediction of concepts and just uses the top concepts to calculate the embeddings while a very high temperature implies lower confidence and weighs all words in the concept set equally. To study this we plot the variation of anchor distance for different values of temperatures across training epochs in Fig B.1. We consider the same Resnet18 model trained on Places365 as Section 3 for consistency of results.

From the figure we see that the anchor metric shows small or no variation for very low or very high temperature values respectively. We think this behavior is expected as for very low temperatures the embedding calculation only considers a single concept while for very high temperatures all concept words in the dataset are weighted equally. Therefore, we choose $T = 0.01$ for our experiments as its between the two extremes and captures the variation in semantics across training properly.

We additionally plot the embedding space of the final epoch of the Resnet18 model analyzed in section 3 for different values of temperatures in Fig B.2.

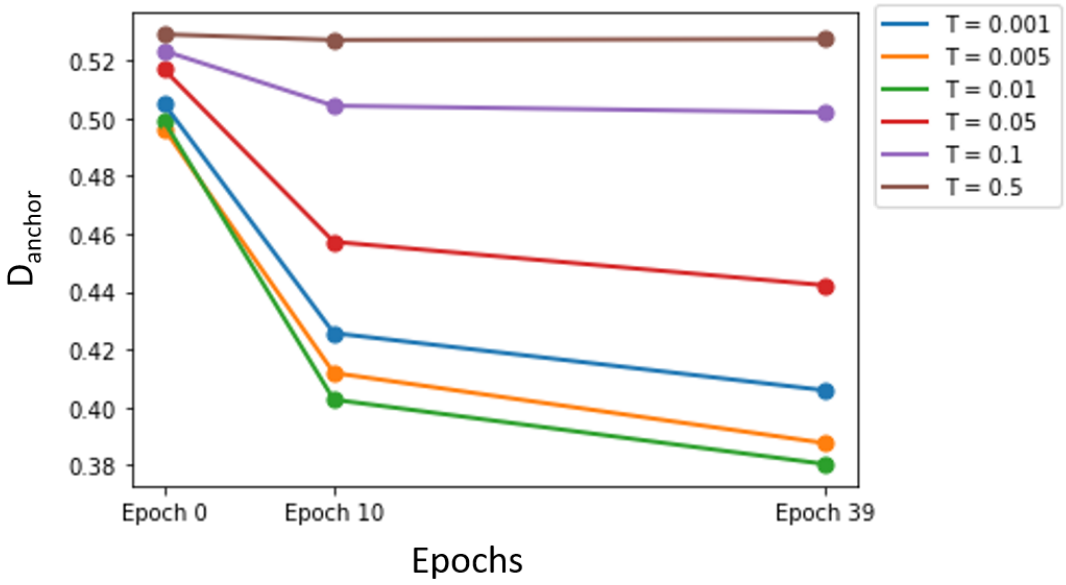

Figure B.1. Variation of *anchor distance* across training epochs for different values of temperature in a Resnet18 model trained on Places365. It can be seen that for higher values of temperature such as $T = 0.5$ the *anchor distance* is almost constant which shows that the calculated embeddings have lost semantic understanding by weighing all concepts equally. We can see that variation in anchor loss is maximized for our chosen value $T = 0.01$, values lower that 0.01 show less variation likely caused by the embedding being too focused on the top concept.

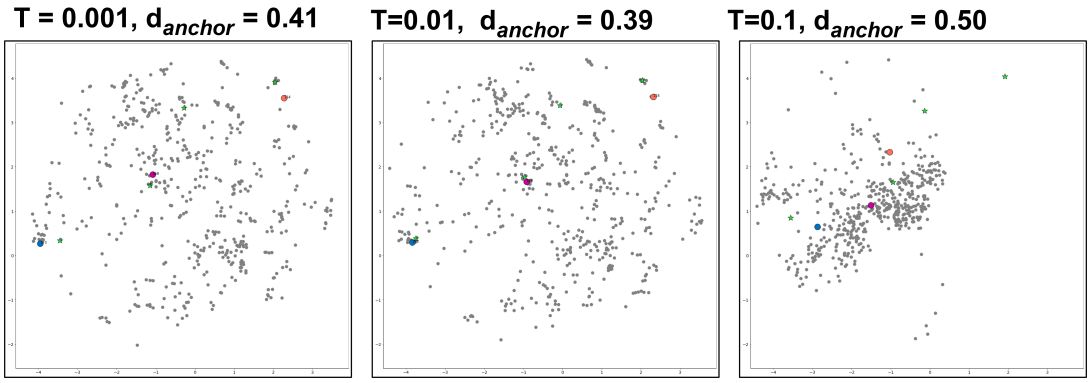

Figure B.2. Embedding plots of layer 4 of the final epoch of Resnet18 model plotted for different values of temperature. We see that for very high temperature $T = 0.1$, the embedding plot clusters in the same region which confirms our hypothesis that for very high temperatures, the space loses its semantic meaning. This is also represented by d_{anchor}. For very low temperature, we see that even though the embedding plot retains its semantic structure, the d_{anchor} is higher, which may be the result of focusing on just the top concept which can provide incomplete understanding of the behavior of a neuron especially in the case of polysemantic neurons.

B.2 Threshold (τ)

The parameter τ defines the similarity threshold above which a neuron is considered interpretable. It is used to calculate the number of interpretable neurons in the bar plots in Fig 2.1. Varying τ changes the interpretability threshold, therefore the model considers more or less neurons to be interpretable if we decrease or increase τ respectively. To study this, we plot the bar plots as in Fig 2.1 for varying values of τ in Fig B.3. Even though the number of interpretable neurons changes, we noticed that the conclusions we made in section 3.3 still hold. For example, the later layers are more interpretable and represent more complex concepts in all cases. However, if we make τ too high, the model doesn't show any neurons to be interpretable. Finally, we would like to point out that τ is a concept detector dependent parameter and should be changed/tuned accordingly for different concept detectors. The values here are specifically for CLIP-Dissect [20].

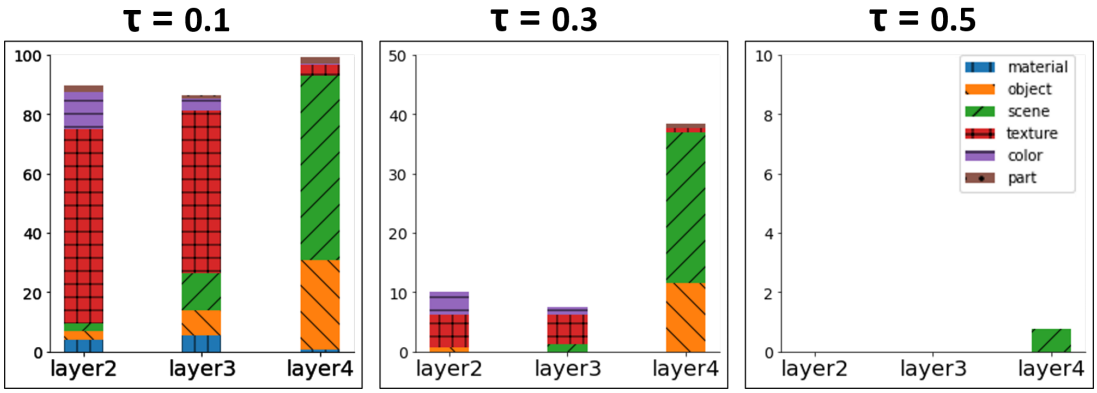

Figure B.3. Percentage of interpretable neurons per category for Epoch 39 of a Resnet18 model for different values of τ. We see that the number of interpretable neurons decreases as we increase the threshold, which is to be expected as we now have a much stricter definition of what counts as "interpretable". We see that even though the number of interpretable neurons decrease as we increase the threshold, we are able to make the same conclusions for most values of τ. For example, for both $\tau = 0.1\,0.3$, we see that later layers represent more complex concepts relating to "object" and "scene" categories as compared to earlier layers which learn "texture" and other simpler features. We also see that if we increase τ too much we lose all interpretable neurons in some layers.

Appendix C

Concept-Monitor with different components

C.1 Concept-Monitor with different Concept set S

As stated in Chapter 3 our method with CLIP-Dissect as concept detector is able to work with any probing and concept dataset. Even though most of our analysis is based on using Broden dataset as D_{probe}, we provide an example of using Concept-Monitor with CIFAR100 training images in Section 4 to investigate Lottery Ticket Hypothesis. In this section we provide an example of using a different concept set. For a different concept set than Broden labels, we considered the list of top 20000 most common English words [15] as a concept set and provide our analysis in Figure C.1 From this figure, we can see that the neurons 373, 168, and 114 converge to the same corresponding anchors as in Fig 2 of section 3. This once again highlights the flexibility of our method to track concept evolution using a user preferred set of concepts.

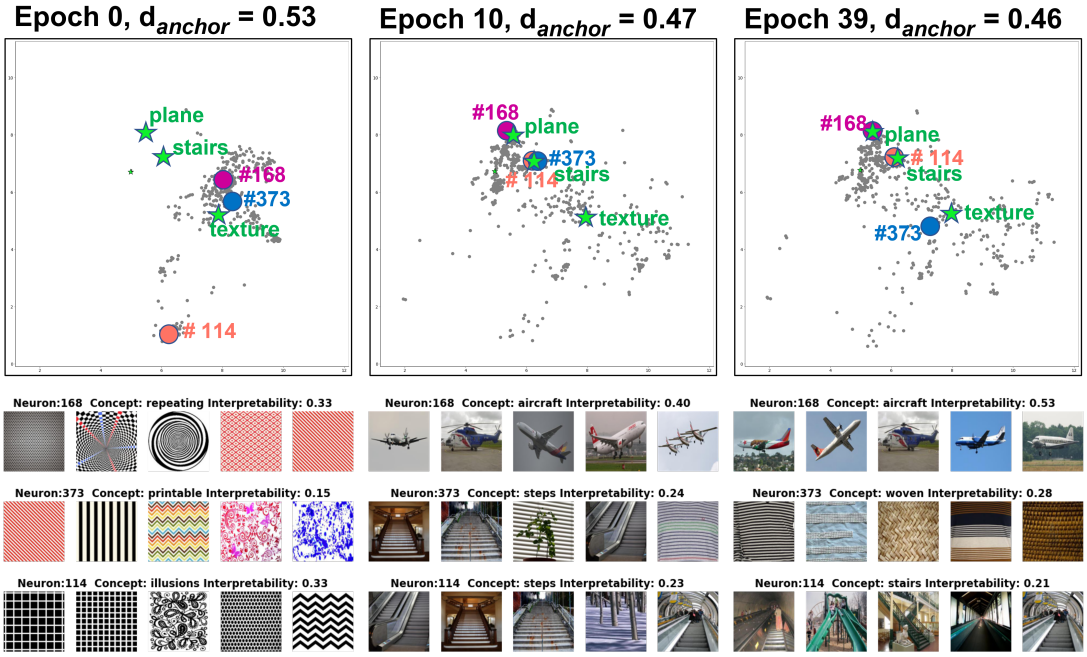

Figure C.1. Analysis of Resnet-18 model trained on Places365 dataset using top 20,000 English words [15] as the concept set. The top image shows our unified embedding space, in which we track Neurons 168, 373, and 114 of layer 4 with the help of semantic anchors ”plane”, ”stairs” and ”texture”. We see that the trajectory of neurons is exactly the same as what we found with using Broden labels as concept set in section 3. This shows that our method is flexible to using a different concept set and opens the opportunity for users to use their own domain specific concept set to track specific models.

C.2 Concept-Monitor with different detectors

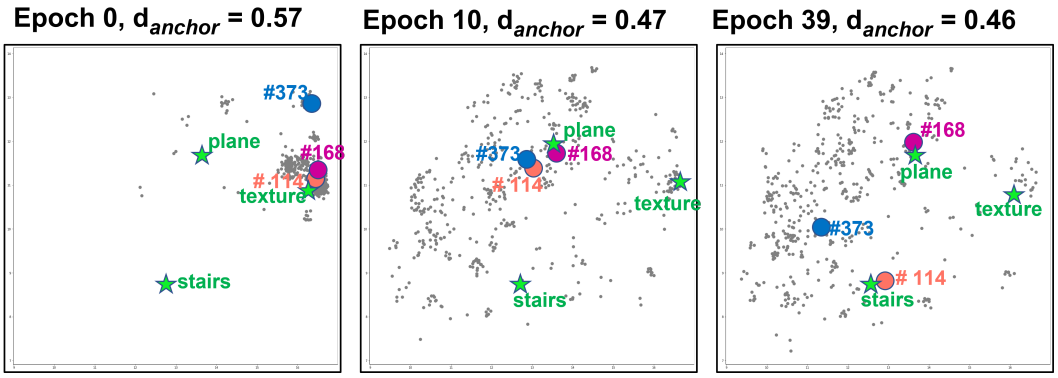

Figure C.2. Analysis of Resnet-18 model trained on Places365 using Network Dissection. Here we track neurons 114, 168 and 373 in the layer 4 of the model using our semantic embedding space. We also add concept anchors ”plane”, ”texture” and ”stairs” to track the neurons. We see that the neurons start together in a cluster and move towards their learnt concept as the training progresses. We also see that the evolution of neurons is the same as with Clip-Dissect in Section 3 Fig 2.1 except in the case of neuron 373, which moves away from the ”texture” anchor. We attribute this distance to the difference in semantic labelling by Network Dissection, which labels Neuron 373 as ”ampitheatre” instead of a textural concept.

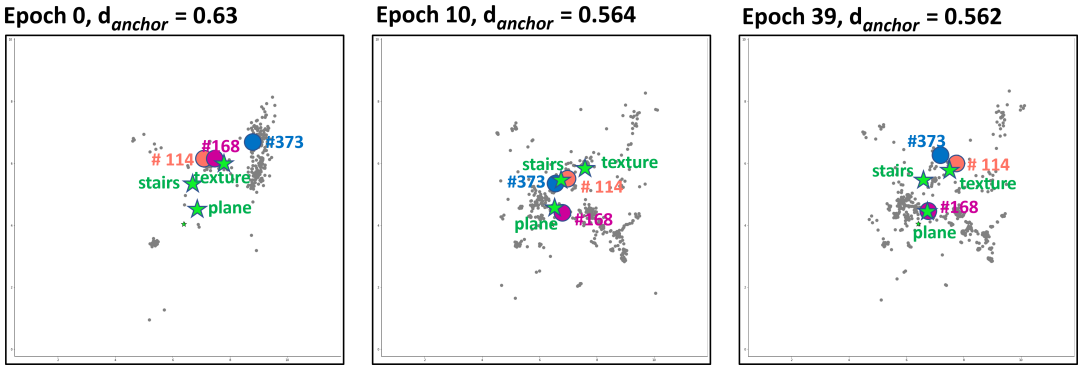

Figure C.3. Analysis of Resnet-18 model trained on Places365 using MILAN. Here we track neurons 114, 168 and 373 in the layer 4 of the model using our semantic embedding space. We also add concept anchors ”plane”, ”texture” and ”stairs” to track the neurons. We see that the neurons start together in the center and move towards their learnt concept as the training progresses. We also see that the evolution of neurons is exactly the same as with CLIP-Dissect in Section 3 Fig 2.1 which once again highlights the flexibility of Concept-Monitor to be used with different concept detectors.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] APTOS. Aptos 2019 blindness detection, 2019. data retrieved from World Development Indicators, https://www.kaggle.com/competitions/aptos2019-blindness-detection/data.

[3] Balaji. Diabetic retinopathy detection using pytorch. https://www.kaggle.com/code/balajiai/, 2019.

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations, 2017.

[5] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, 2017.

[6] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

[7] Tianlong Chen, Xuxi Chen, Xiaolong Ma, Yanzhi Wang, and Zhangyang Wang. Coarsening the granularity: Towards structurally sparse lottery tickets, 2022.

[8] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

[9] Dumitru Erhan, Y. Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*, 01 2009.

[10] Jonathan Frankle and David Bau. Dissecting pruned neural networks. *CoRR*, abs/1907.00262, 2019.

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. 2018.

[12] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. https://distill.pub/2021/multimodal-neurons.

[13] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. *International Conference on Learning Representations*, 2022.

[14] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features, 2022.

[15] Josh Kaufman. Google-10000-english, 2016.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[17] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[18] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

[19] Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *International Conference on Learning Representations*, 2023.

[20] Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks, 2022.

[21] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in.

[22] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[23] Haekyu Park, Seongmin Lee, Benjamin Hoover, Austin Wright, Omar Shaikh, Rahul Duggal, Nilaksh Das, Judy Hoffman, and Duen Horng Chau. Conceptevo: Interpreting concept evolution in deep learning training. *arXiv preprint arXiv:2203.16475*, 2022.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[25] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.

[26] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019.

[27] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[28] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[29] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[30] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training, 2020.

[31] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[32] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[33] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[34] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32, 2019.