Manipulating Explanations: Modifying Feature Visualization in Artificial Neural Networks

Alexander Fulleringer

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

December 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Alexander Fulleringer

Entitled: Manipulating Explanations: Modifying Feature Visualization in Artificial Neural Networks

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Sudhir Mudur

_____ Examiner
Dr. Mirco Ravanelli

_____ Supervisor
Dr. Eugene Belilovsky

Approved by _____
Joey Paquet, Chair
Department of Computer Science and Software Engineering

_____ 2023

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Manipulating Explanations: Modifying Feature Visualization in Artificial Neural Networks

Alexander Fulleringer

As Deep Neural Networks become increasingly ubiquitous and increasingly large, there has been an increasing concern with their uninterpretable nature, and a push towards stronger techniques for interpretation. Feature visualization is one of the most popular techniques to interpret the internal behavior of individual units of trained deep neural networks. Based on activation maximization, it consists of finding synthetic or natural inputs that maximize neuron activations. This work introduces an optimization framework that aims to deceive feature visualization through adversarial model manipulation. It consists of fine-tuning a pre-trained model with a specifically introduced loss that aims to maintain model performance, while also significantly changing feature visualization. We provide evidence of the success of this manipulation on several pre-trained models for the ImageNet classification task. Additionally, several model pruning strategies are tested as potential defences against the manipulations developed, with the aim of producing resilient and performative models.

Acknowledgments

Here I would like to acknowledge and thank all those who helped me complete my degree and this thesis: My supervisor, Prof. Eugene Belilovsky for his unwavering support and instruction. The Flat Iron Institute's Dr. Michael Eickenburg for his technical expertise and wealth of ideas. Our lab's post-doc, Geraldin Nanfack for his writing advice and clever solutions. I would also like to thank Concordia University for providing the environment, and several excellent courses, that have helped me develop into a better researcher and computer scientist. Beyond the invaluable academic support I've received throughout this journey, I extend my warmest thanks to my family and friends who have supported me and pushed me forward even when it seemed impossible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Deep Neural Networks (DNNs) can be trained to perform many economically valuable tasks (Kaplan et al., 2020; Krizhevsky, Sutskever, & Hinton, 2017). They are already pervasive in many sectors, and their prevalence is only expected to increase over time. With increasing computational power and ever more available data, DNN architectures are growing in size and executing increasingly intricate tasks. Given the increasing size and complexity of DNNs, interpreting how they function, a well-established challenge, will likely grow more difficult with new developments. However, for certain classes of critical applications, close inspection and guarantees of functionality will be very important, especially in heavily regulated and high-stakes domains. Here we ask: could a malicious actor conceal the true functionality of a DNN from an interpretability method by deliberately perturbing the DNN?

Focusing on the continuously popular feature visualization method (Olah et al., 2020; Olah, Mordvintsev, & Schubert, 2017; Zeiler & Fergus, 2014), we propose to create an optimization procedure to manipulate the interpretation of individual neurons of a network while keeping its final behavior the same. A successful modification of the interpretation while keeping outputs constant is evidence for the manipulability of the interpretation approach. In this work, we concentrate on convnet architectures for which interpretation by activation maximization or feature visualization methods has been popular (Yosinski, Clune, Nguyen, Fuchs, & Lipson, 2015; Zeiler & Fergus, 2014). In this work, we study the feature visualization of a neuron or channel norm via activation

maximization and attempt to modify it while maintaining network outputs and accuracy. Then, we characterize the attacks quantitatively and show two different attacks that can effectively manipulate and explicitly obfuscate interpretations.

To date, most works on interpretability manipulability have focused on techniques such as feature attribution tailored for model predictions (Heo, Joo, & Moon, 2019; Slack, Hilgard, Jia, Singh, & Lakkaraju, 2020). Little attention has been paid to the manipulability of neuron interpretability techniques, despite their increasing popularity due to their fine-grained understanding of inner structures of DNNs (Olah et al., 2020, 2017; Räukur, Ho, Casper, & Hadfield-Menell, 2022).

Notably, it has also been applied to create mechanistic interpretations which are argued to be robust as they directly link to the function of neurons (Cammarata et al., 2020; Nanda, Chan, Liberum, Smith, & Steinhardt, 2023).

As neural networks use becomes increasingly popular across industries, a deeper understanding of their functionality becomes more and more necessary. Indeed, there have been many cases of seemingly effective systems that, upon deeper investigation, have distinct and unwanted biases in their performance, as can be seen in works by Hundt, Agnew, Zeng, Kacianka, and Gombolay (2022) and Tian, Xie, Hu, and Liu (2021). As the size and complexity of models increases, so too does the inherent difficulty in explaining how they work. There are legal, ethical, and practical motivations to address this issue, and generate meaningful explanations of models.

Practical concerns have been raised over using inexplicable AI. Inexplainable AI may be viewed with lower trust. The EU's proposed AI Act includes provisions that AI systems used in the EU, including DNNs, must make decisions in an unbiased, transparent, and traceable fashion. This indicates a need for strong and robust interpretability methods.

We study the feature visualization of a neuron or channel norm via activation maximization and attempt to modify it while maintaining trained network outputs and accuracy. We investigate how to characterize these attacks quantitatively and show two different attacks which can effectively manipulate and explicitly obfuscate interpretations. Finally, we experiment with model pruning as a potential defence mechanism against these attacks.

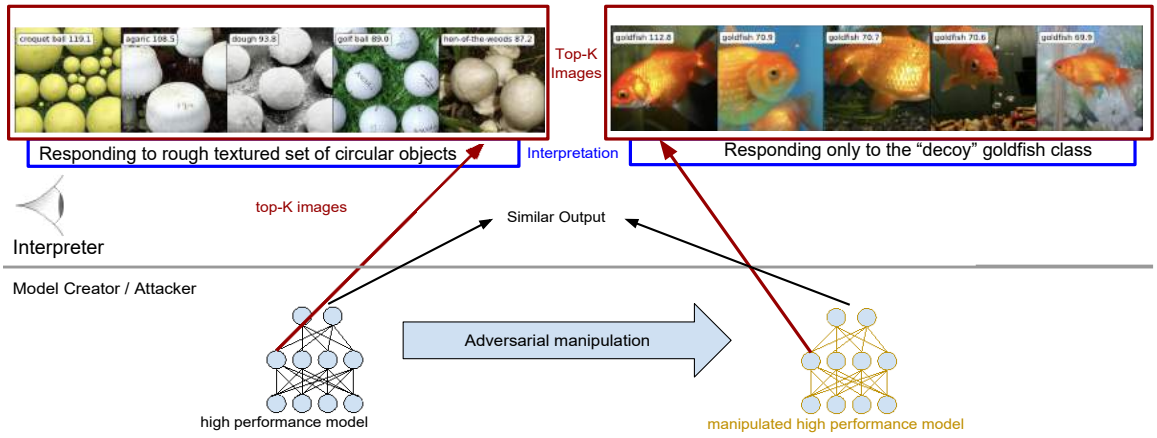


Figure 1.1: Illustration of the attack model for our adversarial interpretability manipulation. Top-5 images that best activate a given neuron, seemingly capturing a shared semantic concept that an interpreter may describe and/or use an external tool to describe (Hernandez et al., 2022; Oikarinen & Weng, 2022). We assume the model creator can manipulate the model before it is released to the interpreter. In this case, they can create a model that might lead to interpreting the selected neuron as only capturing the semantics of a single class.

1.2 Contributions

- We develop a framework to manipulate the feature visualizations of a DNN.
- We use this framework to implement two attacks, the Push Up and Push Down attacks.
- We define several metrics to describe our results. These include the CLIP-δ score to measure semantic difference between sets of images, and Kendall-τ-W and CLIP-W to assess the presence of the whack a mole problem we define in section 3.1.4.
- We experiment with the use of pruning as a tool to defend against our attacks.
- We submitted to and were accepted by the NeurIPS 2023 ATTRIB workshop under a paper titled Adversarial Attacks on Neuron Interpretation via Activation Maximization, cited as Fulleringer, Nanfack, Marty, Eickenberg, and Belilovsky (2023). This paper forms the basis for Chapter 3 in this work.
- We submitted our results to and were accepted by AAAI 2024 as part of a paper titled Adversarial Attacks on the Interpretation of Neuron Activation Maximization, cited as Nanfack, Fulleringer, Marty, Eickenberg, and Belilovsky (2024). First authorship on this paper is

shared between Alexander Fulleringer and Dr. Geraldin Nanfack.

Chapter 2

Background

2.1 Adversarial Attacks

The term Adversarial Attacks refers to a broad set of strategies used to fool or manipulate a model and its outputs (Akhtar & Mian, 2018; Chakraborty, Alam, Dey, Chattopadhyay, & Mukhopadhyay, 2018; Miller, Xiang, & Kesidis, 2020). Miller et al. (2020) distinguish between three categories of Adversarial Attacks: Data Poisoning, Reverse Engineering, and Test-Time Evasion.

Data Poisoning occurs when data is introduced to the training set such that the model behaviour is compromised, either in performance or via the creation of ”backdoors”. Reverse Engineering is where a new model is trained using outputs of a previous model. This ”surrogate” model, which ideally follows the behaviour of the original closely, is then used to identify vulnerabilities or simply to copy the capabilities of the original. Finally, Test-Time Evasion involves feeding doctored inputs to the net to either reduce performance or even to force a specific outcome.

Major defense strategies against these attacks include modifying the training procedure (Goodfellow, Shlens, & Szegedy, 2014; Nayebi & Ganguli, 2017; Zheng, Song, Leung, & Goodfellow, 2016) and adding functionality to detect an attack (Grosse, Manoharan, Papernot, Backes, & McDaniel, 2017; X. Li & Li, 2017). It is worth noting as well that it is very difficult to defend against attacks when the attackers have full control of the model (Chakraborty et al., 2018).

2.2 Interpretability Methods

2.2.1 Saliency

Saliency methods for interpretability attempt to explain model behaviour by examining which parts of the input most strongly affect the network's output (Konate et al., 2021; Selvaraju et al., 2017; Simonyan, Vedaldi, & Zisserman, 2013). For Convolutional Neural Networks (CNNs) performing image classification in particular they often aim to connect specific class scores with the pixel-patches that most strongly affect them (Konate et al., 2021). The work by Simonyan et al. (2013) introduced the concept of a saliency map to visualize the connection between an image and a class score. More formally, Simonyan et al. (2013) consider a net and define S_c as the function that determines the score for a specific class c based on some input I. In the case of a simple linear net where I is a vector, this would lead to the following equation: $S_c = w_c I + b_c$ (Simonyan et al., 2013). In this example, the relative importance of each attribute in I is simply its corresponding weight in w_c. In the case of a convolutional neural network the underlying S_c is non-linear and thus they use an approximation of the importance, w for a given image I_0, as:

$$w = \frac{\delta S_c}{\delta I}\Big|_{I_0} \tag{1}$$

Intuitively, this would result in a high relevance for pixels where a change in pixel value produces a relatively large change in the class score. As this process involves performing a backpropagation and requires a modified architecture, other methods were developed such as Grad-CAM, that promised to improve upon these issues (Konate et al., 2021; Selvaraju et al., 2017; Simonyan et al., 2013).

More modern methods based on this general approach include Grad-CAM and its derivatives, collected in Gildenblat and contributors (2021), and Layer-wise Relevance Propagation (LRP) (Binder, Bach, Montavon, Müller, & Samek, 2016).

It is important, in the context of this thesis, to note that there exists several works discussing vulnerabilities of saliency methods for interpretability (Adebayo et al., 2018; Dombrowski et al., 2019; Heo et al., 2019). The work by Adebayo et al. (2018) shows that saliency maps are relatively invariant under model parameter randomization, and a model trained on randomly labelled data

may provide similar explanations as a properly trained one. Dombrowski et al. (2019) shows that adversarial data manipulation can be used to arbitrarily modify the saliency map of a sample. Finally, Heo et al. (2019) show that adversarial model manipulation can be used to manipulate the saliency maps.

2.2.2 Feature Visualization

Feature Visualization techniques seek to interpret the behaviour of DNNs, particularly those used for vision tasks, by identifying and visualizing the features learned by the network (Olah et al., 2017; Shahroudnejad, 2021). By showing what features a given neuron reacts to strongly, one can build an intuition of how the individual units of a network function (Olah et al., 2017; Zeiler & Fergus, 2014). Building upon individual neuron interpretations, Olah et al. (2020) combine neurons and their connections to form meaningful units called Circuits that implement specific algorithms.

Activation Maximization: An intuitive way to discern what features a neuron has learned to identify, first proposed in Erhan, Bengio, Courville, and Vincent (2009), would be to find some input(s) that maximizes a neuron's internal activations. Olah et al. (2017) use a gradient based approach combined with regularization strategies to generate Synthetic Images that can target different parts of a network, including specific neurons or channels. These images would contain human-recognizable features, corresponding to features seen in the training images, and could be examined to learn about model behaviours. Generally gradient based approaches consist of solving the optimization problem:

$$x^{*} = \arg\max_{x}(a_{i,l}(\theta, x) - \lambda(x)) \tag{2}$$

Where $a_{i,l}$ is the activation with respect to the model parameters θ and the input x, and λ is the chosen regularization function (Qin, Yu, Liu, & Chen, 2018). Without the regularization term λ, the produced image x^{*}, will be dominated by high frequency noise, resembling adversarial examples more than recognizable features (Olah et al., 2017; Qin et al., 2018).

Another method, first implemented by Nguyen, Dosovitskiy, Yosinski, Brox, and Clune (2016), involves generating a highly activating Synthetic Image by using a Generative Adversarial Network

(GAN). A GAN is comprised of a generator net, G, and a discriminator net, D (Goodfellow, Pouget-Abadie, et al., 2014). Using some real dataset, the generator is trained to produce artificial samples that are indistinguishable from real data, while the discriminator is trained to discriminate between the real and artificial samples. G accepts as input some noise vectors and produces seemingly real images. Eq. 2 then becomes:

$$y^{\square} = \underset{y}{\operatorname{argmax}}(a_{i,l}(\theta, G(y) - \lambda(y)) \tag{3}$$

where y is the input to the generator and G(y) is the Synthetic Image produced for visualization.

Deconvolutional Nets: Beyond Activation Maximization, it is also possible to create a visualization by attempting to reconstruct the original image from the hidden layer feature maps. For CNNs, Zeiler and Fergus (2014) propose the idea of using a DeconvNet to transform the feature maps of a CNN back into image-space. DeconvNets function as mirrors to standard CNNs, such as AlexNet (Krizhevsky, Sutskever, and Hinton (2012)), and have components that perform inverse operations to those learned by CNNs. The image reconstruction performed by the DeconvNet allows for a visualization of which features are learned by a particular neuron in the network (Zeiler & Fergus, 2014).

2.2.3 Mechanistic Interpretability

Mechanistic Interpretability seeks to understand and reconstruct the underlying algorithms learned by the neural network (Räuker, Ho, Casper, & Hadfield-Menell, 2023). In particular, focus has been placed on the identification of circuits composed of neurons inside a neural net that work together to implement an algorithm (Conmy, Mavor-Parker, Lynch, Heimersheim, & Garriga-Alonso, 2023; Olah et al., 2020; Wang, Variengien, Conmy, Shlegeris, & Steinhardt, 2022). This method builds on the individual neuron interpretations, such as those seen in Olah et al. (2017), and uses them to create a more comprehensive understanding of the neural net's functionality. Current circuit detection methods are generally quite laborious to implement, and have not yet seen widespread generalization to large models (Räuker et al., 2023).

2.2.4 White-Box Models

In contrast with DNNs, widely considered black-box models whose internal functionality is not easily understood, white-box models are those that have inherently clear decision making processes. Examples include decision trees, linear regression, and rules based models. These models however, don't leverage modern computational power to the same extent as DNNs, and so tend to have lower performance than a DNN trained to perform the same task. Rudin (2019) argues the possibility of using white-box models instead in many cases, but the current trend is still towards these large black-box models. This work is therefore focused on the interpretation of those large models, rather than engineering potential replacements.

LIME

One potential way to interpret a black-box model is to create an interpretable model that closely follows its behaviour and interpret that model instead, as Ribeiro, Singh, and Guestrin (2016) do using their proposed LIME tool. LIME, or Local Interpertrable Model-agnostic Explanations, is a tool that generates an explanation for a prediction by creating an explainable model that is faithful to the model being explained locally around that prediction (Ribeiro et al., 2016). In the case of image classification networks, LIME can even identify important pixel patches that contribute positively to class scores. While LIME is an interesting and potentially valuable tool, it has certain important shortcomings. The explanations generated are ultimately local. Each one may only help understand a specific prediction by a network so a new interpretable model must be created for each explanation. Furthermore the explanation generated is for a model that, ultimately, is only imitating the original model, meaning that there is necessarily a disconnect between the explanation and the actual model behaviour. Rudin (2019) discusses this phenomenon in further detail, highlighting the risks of asserting that models share underlying algorithms based purely on specific behaviours.

2.3 Pruning

It is well known that Neural Nets have become quite large and computationally costly in recent years. One strategy that has empirically demonstrated efficacy in reducing the size and compute

required by Neural Nets is the practice of model pruning, where parameters are removed from the network (Blalock, Ortiz, Frankle, & Guttag, 2020; Cheng, Wang, Zhou, & Zhang, 2020). Indeed, certain methods of pruning are so effective that they will even show an increase in performance, as seen in Chen et al. (2021). Blalock et al. (2020) broadly divide pruning methods into two categories: Unstructured Pruning, where individual parameters are removed, and Structured Pruning, where structures of parameters, such as entire filters in a CNN, are removed. A major benefit of Structured Pruning is that it more naturally leads to an increase in model speeds, while the arbitrary sparsity of removed weights in Unstructured Pruning would require special libraries for speed up (Blalock et al., 2020; H. Li, Kadav, Durdanovic, Samet, & Graf, 2017). As the methods for pruning are very diverse, the rest of this section will concern the pruning methods used in this work.

2.3.1 Norm Based Pruning

A simple, yet popular,method for determining which structures to prune is to start with a pre-trained network and choose structures with a low norm, with the l1 and l2 norms as common choices (Han, Pool, Tran, & Dally, 2015; Lebedev & Lempitsky, 2015; H. Li et al., 2017; Molchanov, Tyree, Karras, Aila, & Kautz, 2017; See, Luong, & Manning, 2016; See et al., 2016; Wen, Wu, Wang, Chen, & Li, 2016). The benefits of this method include computational efficiency in the calculations, as well as being relatively simple in implementation. The requirement of a fully-trained network, as well as the additional fine-tuning steps typically required after the procedure means that the compression achieved comes at the cost of increased training times.

This method has found great success in reducing model parameters while maintaining performance. Indeed, Han et al. (2015) report a 9x compression rate on AlexNet with no loss in performance. See et al. (2016) investigate several strategies for pruning and find that simply pruning all weights with an l2 norm below a threshold can allow for an 80% reduction in parameters for a Neural Machine Translation network. In general, norm-based pruning allows for significant decreases in model size with minimal loss, or even an increase, of performance.

2.3.2 OTO Prune

Only Train Once, or OTO, is a Structured Pruning method introduced by Chen et al. (2021), and refined in Chen, Liang, Tianyu, Zhu, and Zharkov (2023). As implied by the name, the framework allows a developer to create a pruned version of a model without pre-training the base model or needing a fine-tuning step, effectively allowing one to go from an architecture to a pruned model only training once (Chen et al., 2023). OTO's process involves identifying what they term Zero-Invariant Groups (ZIGs), and the use of a special optimization algorithm called the Dual Half-Space Stochastic Projected Gradient (DHSPG).

Zero-Invariant Groups: Chen et al. (2021) define a Zero-Invariant Group by first partitioning a net's parameters into disjointed groups. If all groups satisfy the condition that their parameters all being zero result in them having no contribution to the next layer then they are consider ZIGs. Effectively, this means that if the parameters of a group are all 0, then the removal of the group does not affect the later layers of the network. This means that should a ZIG's parameters be reduced to zero it can be pruned without further retraining. This stands in contrast to the removal of, for example, a convolutional filter, as the next layer will still have associated biases that would need to be retrained.

Dual Half-Space Stochastic Projected Gradient: In order to identify redundant ZIGs and allow for their removal, Chen et al. (2023) define the DHSPG algorithm. DHSPG seeks to solve the following problem:

$$\text{minimize}_{x \in R} f(x), \text{ s.t. } \text{Cardinality}\{g \in G \mid [x_g] = 0\} = K \tag{4}$$

Where G is the set of all ZIGs for a problem, x is the parameters of the net, $f(x)$ is the loss, and K is the target sparsity ($K \in (0, 1)$). Effectively, given a target sparsity (proportion of groups to prune) K, push sufficient groups parameters (x_g) to 0. This is accomplished by penalizing the magnitude of some groups during the training procedure and has empirically shown great success (Chen et al., 2023).

Chapter 3

Adversarial Attacks on Interpretations

3.1 Methods

3.1.1 Notations and Background.

We denote by $D = \{(x_i, y_i)\}_{i=1}^{N}$ a dataset for supervised learning, where $x_i \in \mathbb{R}^d$ is the input and $y_i \in \{1, \ldots, K\}$ is its class label. Let f_θ denote a DNN, $f_\theta^{(l)}(x)$ defines activation maps of x on the l-th layer, which can be decomposed into J single activation maps $f_\theta^{(l,j)}(x)$. In particular, $f_\theta^{(l,j)}(x)$ is a matrix if the l-th layer is a 2D-convolutional layer and a scalar if it is a fully connected layer. We aim to understand the internal behavior of individual units through feature visualization, generically defined by activation maximization (Mahendran & Vedaldi, 2015; Yosinski et al., 2015), i.e., $x^* \in \operatorname{argmax}_{x \in D} f_\theta^{(l,j)}(x)$. Where (l, j) is the pair of layer l and neuron j. When the layer l is a convolutional layer, in the rest of the paper, we aggregate the activation map $f_\theta^{(l,j)}(x)$ using its spatial squared ℓ_2-norm $\|f_\theta^{(l,j)}(x)\|_2^2$, and subsequently refer to j as the channel index. Additionally, we mainly focus on the case where D is a set of natural images, and we denote by top-k images the set of real images that have the k highest values of activations for a given pair (l, j). When $X \in \mathbb{R}^d$, following Zimmermann et al. (2021), the result x^* will be called synthetic feature visualization.

3.1.2 Attack Framework

We consider feature visualization with top-k images and propose an adversarial model manipulation that fine-tunes a pre-trained model with a loss that maintains its initial performance while changing the result of feature visualization. More formally, given a set of training data D, a pre-trained model with parameters $\theta_{initial}$, and an additional set of images (e.g., a set of top-k images) D_{attack}, our attack framework consists in the following optimization

$$\min_{\theta}(\alpha L_A(D, D_{attack}; \theta) + (1 - \alpha)L_M(D; \theta, \theta_{initial})), \tag{5}$$

where θ are parameters of the updated model f_θ, $L_M(.)$ is the loss that aims to maintain the initial performance of the model $f_{\theta_{initial}}$, and $L_A(.)$ is the attack loss. For the maintain objective, when viewing final outputs $f_\theta(.)$ as a conditional distribution, our maintain loss is the distillation loss (Hinton, Vinyals, & Dean, 2015),

$$L_M(D; \theta, \theta_{initial}) = L_{CE}(f_{\theta_{initial}}(.)||f_\theta(.)) \tag{6}$$

where L_{CE} is the cross entropy loss between the original model outputs and the attacked model outputs on training data D. The attack loss $L_A(.)$ varies depending on the attack, and is defined in the following sections.

3.1.3 Push-Down and Push-Up Attack

Push-Down Attack: Given a set of top-k images from feature visualization, denoted by $D_{attack}^{(l,j)}$, that best activate the layer l and channel j of the initial model f_θ, our first attack aims to push to zero the activations of examples in $D_{attack}^{(l,j)}$. We name this the push-down attack, and we propose the following objective for all channels of a layer l simultaneously:

$$L_A(D, D_{attack}; \theta) = \sum_{j=1}^{N^l} \sum_{x \in D_{attack}^{(l,j)}} ||f_\theta^{(l,j)}(x)||_2^2 \tag{7}$$

Where J_l is the number of channels of the layer l. Note that it is possible to attack a single channel or channels from multiple layers. Here we focus on attacking all the channels in a single layer.

Push-Up Decoy Attack: In the push-up decoy attack, given a set of examples $x_p \in D_{decoy}$, we aim to make these images appear in the top-k result for all the channels of a layer l. For this purpose, we propose the following objective, where $[.]_+$ is $\max(., 0)$:

$$L_A(D, D_{decoy}; \theta) = \sum_{j,p,i} [\|f_\theta^{(l,j)}(x_i)\|_2^2 - \|f_\theta^{(l,j)}(x_p)\|_2^2]_+ \quad (8)$$

This loss aims to make activations of examples in D_{decoy} larger than all the activations of training examples.

3.1.4 Attack Characterization

We propose two metrics to assess the effectiveness of an adversarial attack on the top-k images of feature visualization.

Kendall-τ: To assess the degree of change in the underlying behavior of a channel, we use Kendall's Rank Correlation Coefficient (Kendall-τ) on a large subset D_τ of ImageNet. For each channel, we calculate the Kendall-τ coefficient using (i) the ranking R_{init} of the initial image activations, and (ii) the ranking R_{final} of final (post-attack) image activations using images in D_τ. A Kendall-τ coefficient approaching 1 indicates that the ordering of image activations for each channel before and after the attack remains the same, implying minimal change in channel behavior.

CLIP-δ: To quantify the semantic change in the feature visualization, we employ the external and generic CLIP image encoder (Radford et al., 2021) to compute embeddings of top-k images. Given a channel j, we denote by $C^{init,init}_{j,j}$ the average of cosine similarities between CLIP embeddings of (i) initial top-k images and (ii) themselves. Similarly, for the channel j, we denote by $C^{init,final}_{j,j}$ the average of cosine similarities between CLIP embeddings of (i) initial top-k images and (ii) final

ones. Our proposed CLIP-δ score for a channel j is defined as:

$$\text{CLIP-}\delta_j = (C^{\text{init,init}}_{j,j} - C^{\text{init,final}}_{j,j}) / \left(\frac{1}{J_l - 1} \sum_{p=j} C^{\text{init,init}}_{j,p}\right) \tag{9}$$

Which quantifies the semantic change in top images through their CLIP embeddings. A higher score indicates more significant semantic change, as can be visually verified in Figs. 3.1 and 3.2.

Figure 3.1: Zoom-In on chosen channels for the Conv5 Push-Down Attack. All initial top-5 images have been replaced, and the CLIP-δ scores correlate with visual intuition.

The Whack-A-Mole Problem: An important, though not immediately obvious, concern for our framework is whether the behavior and interpretation of one neuron can be moved to another neuron through the optimization process. Indeed, the push-down attack loss may be strongly satisfied by channel permutation. We call this the whack-a-mole problem. To ensure that this does not occur, we introduce two metrics: Kendall-τ-W and CLIP-W. Typically, values ⍰ 1 in these metrics imply an absence of the whack-a-mole problem. We present the mean across attacked channels for the following in metrics in Table 3.1.

Kendall-τ-W_j: Given a channel index j, and using the subset of ImageNet D_τ, we obtain the maximum Kendall-τ score between rankings $R_{\text{init},j}$ and $R_{\text{final},i}$ where $i = j$. To obtain Kendall-τ-W_j, we divide this maximum value by the initial maximum Kendall-τ score i.e. the score over $R_{\text{init},j}$ and $R_{\text{init},i}$ where $i = j$. Effectively, we find the most similar other post-attack channel, then normalize against the most similar pre-attack channel.

CLIP-W_j: Using the top-k images in the initial model and channel j we obtain CLIP-W_j = $\max_{i=j} \bar{C}^{initial,final}_{j,i} / \max_{i=j} \bar{C}^{initial,initial}_{j,i}$ comparing to all top-k images in other channels of the final model, normalized against that same similarity metric in the initial CLIP scores.

3.2 Experiments and Results

For the attacks, we use the ImageNet (Deng et al., 2009) training set as D, and primarily the PyTorch (Paszke et al., 2019) pretrained AlexNet (Krizhevsky et al., 2012) for analysis. Additionally, we have conducted ablation studies on EfficientNet, ResNet-50 , and ViT-B/32 with similar findings (Dosovitskiy et al., 2020; He, Zhang, Ren, & Sun, 2016; Tan & Le, 2019). Metrics for these attacks are reported in Table 3.1, and additional visualizations for non-AlexNet networks can be found in Appx. E and F. Details regarding hyperparameters for all the attacks can be found in Appx. A. For the push-down and up attack, we consider $D^{(l,j)}_{attack} \subset D$ as the top-10 images that maximally activate the channel j of layer l. For the push-up attack, we also consider D_{decoy} as 100 randomly sampled images of a selected decoy class.

3.2.1 Warm-Up Experiments

To set a baseline reference for our attack framework, we begin by fine-tuning AlexNet without attacking it (i.e. using the loss defined in Eq. (2) with $\alpha = 0$). This leads to virtually no change in the feature visualization, as can be seen in Appx. B.1) and confirmed via our metrics in the first row of Table 3.1. Next, we apply the push-down attack to one channel. Figure C shows the visualization of top images before and after the attack. We can see that after optimization, the top-k activating images of the neuron have been completely replaced by other images with different semantic concepts, suggesting a successful attack with a negligible 0.04% accuracy loss. Note that naively setting a channel's weights to 0 would perfectly satisfy this attack objective. Experimentally, doing this on channel 0 of Conv5 with no retraining leads to only 0.2% accuracy loss. We thus consider more challenging settings.

Layer/Attack	CLIP-δ	K-τ	CLIP-W	K-τ-W	Acc.(%)	ΔAcc.(%)
Conv5 Finetuning Baseline	0.001	0.969	0.999	0.058	56.5%	-0.04%
Conv 5 Push-Up	0.199	0.645	0.928	0.005	56.2%	-0.31%
Conv 5 Push-Down	0.294	0.351	0.974	0.041	56.1%	-0.47%
Conv 4 Push-Down	0.225	0.391	0.991	0.114	56.1%	-0.45%
Conv 3 Push-Down	0.174	0.479	0.972	0.129	56.0%	-0.54%
Conv 2 Push-Down	0.105	0.398	0.993	0.156	56.2%	-0.35%
Conv 1 Push-Down	0.058	0.537	0.995	0.291	56.0%	-0.51%
EfficientNet L7 Push-Down	0.262	0.503	0.971	-0.145	77.5%	-0.17%

Table 3.1: Average (over channels) metrics for an All-Channel Push-Down and Push-Up Attack for AlexNet (rows 2-7) and EfficientNet (row 8). Row 1 shows a simple finetuning baseline, corresponding to $\alpha = 0$ in Eq. 2. We see that the relative whack-a-mole metrics are low, suggesting this problem is not present for our attacks. Lower layers are more challenging to attack leading to lower CLIP-δ score and higher Kendall-τ as confirmed by visual intuition.

3.2.2 All-Channel Push-Down Attack

Unlike the single-channel attack, the all-channel attack does not have a trivial solution. Naively setting all channel weights to zero would result in catastrophic performance loss.

We apply our attack framework to Conv5 of the AlexNet Model. Figure 3.1 shows a selection of channels and the modifications achieved under the all-channel push-down attack and the aggregate metrics (averages for all channels in a layer) are shown in Table 3.1.

For the visualized channels in 3.1, and those in Fig. 3.2 we observe a near-complete replacement of the top-5 images.

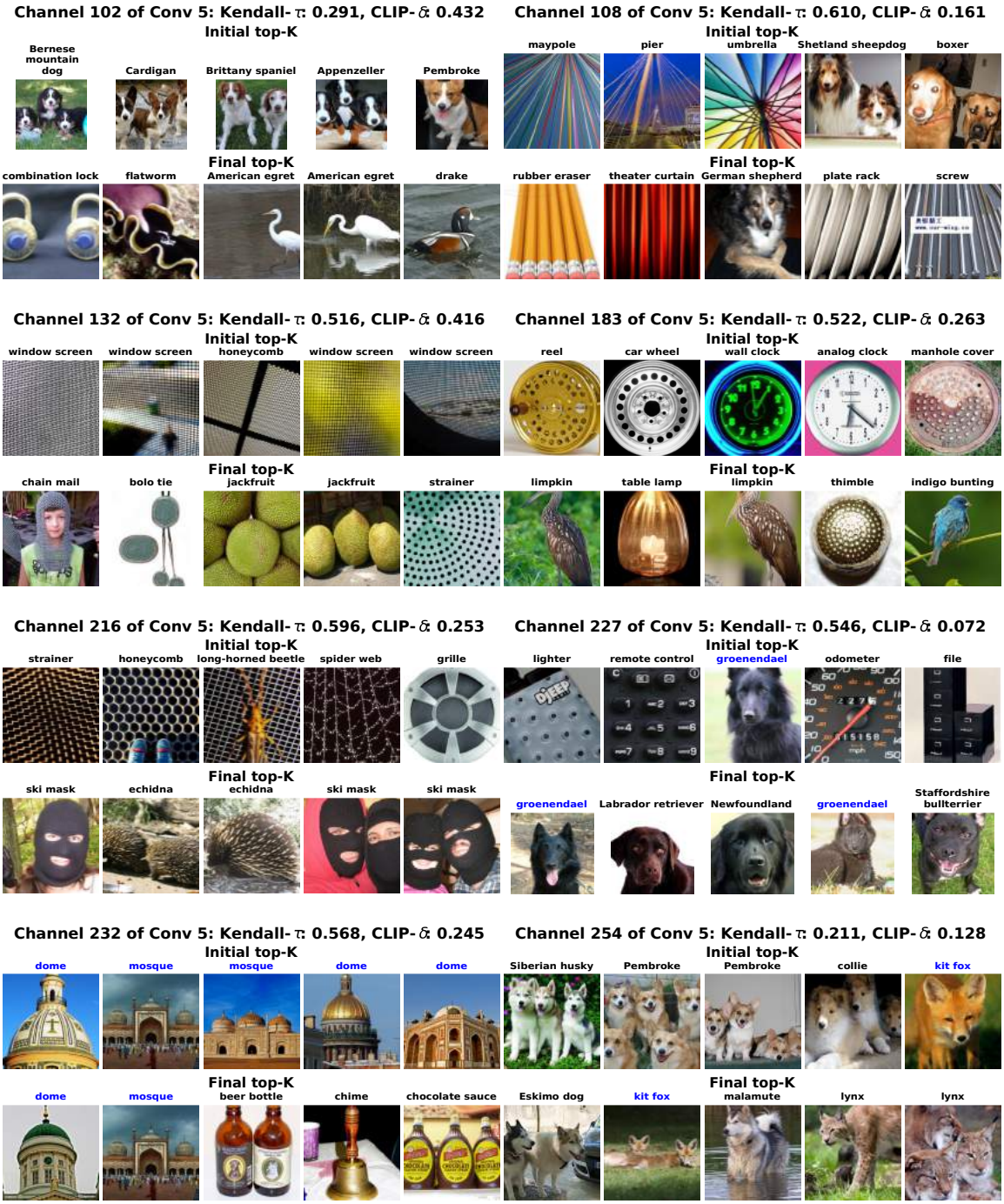


Figure 3.2: Push-down all-channel attack of Conv5 of AlexNet. All initial top-5 images were completely removed from the new set of top-5 images, demonstrating the success of the attack. Channel indexes were chosen randomly.

Further, the labels of the top images significantly change, with minimal residual overlap. This suggests that the semantic concepts that would be determined by an interpreter would likely change.

This is opposed to the model memorizing the top images and replacing them with semantically similar ones. We further confirm this in Appx. B.3 by showing validation set top-k images which demonstrate the same semantic change seen on the training images (which were used for the actual attack). Overall, this attack produces a generalized change in the feature visualization of neurons.

By analyzing the metrics reported in Fig. 3.2 and by comparing the channels before and after modification, we observe several noteworthy behaviors. The second and third channels exhibit relatively high Kendall-τ scores, from which we conclude that the ordering of image activations has not undergone severe changes. This likely means that a subset of images, which includes the initial top-k has moved in rank. Studying the CLIP-δ in both cases allows us to conclude that there is some semantic overlap in the initial and final top-k, which can be confirmed by visual inspection of Fig. 3.1. This is in contrast to the first channel in 3.2 shown on the right, where the Kendall-τ score is close to zero, indicating a full re-ordering of the activations. Correspondingly, the CLIP-δ from initial to final is also much higher, which matches with a visual inspection.

Overall, we notice a substantial correspondence between our visual intuition and the CLIP-δ and Kendall-τ scores. Channels with low scores Kendall-τ and high CLIP-δ tend to change substantially. As illustrated in Fig. 3.2 and supplementary Fig. H, one observed difference in these two metrics is that channels maintaining similar classes in the top images will tend to have a lower CLIP-δ.

We observe that modifications of the earliest layers are significantly harder to achieve than for later layers as confirmed by the metrics and visual examination. The observed CLIP-δ scores, as well as visual observation, shown in Appx. C, both indicate lower layers' channels are more resilient to this sort of attack.

3.2.3 All Channel Push-Up Attack

Here we study a more targeted attack objective, namely one that actively pushes a set of selected images into the top activating images for every channel. This is achieved with the loss defined in section 3.1.3. The loss is non-zero when there exist images outside the set of selected images that activate higher than the selected images we intend to push up.

This targeted attack is likely more challenging than the push-down attack, which does not specify what images the top-k should be replaced with. Indeed, the push-up attack, if successful, can assign the same interpretation to every channel in a layer, making any interpretation attempt based on top-k images minimally informative.

Fig. 1.1 shows the result of the push-up attack using a collection of images with the ImageNet label "Goldfish" as the decoy set. Further, in Fig. 3.3, we show additional channels, where the top-5 contain a few or consist entirely of Goldfish images. The metrics in Table 3.1 also demonstrate substantial change and a low likelihood of whack-a-mole behavior. Examining the figure closely, we observe that not only Goldfish, but also images sharing traits with Goldfish images are also boosted, suggesting a degree of generality in the newly imposed selectivity. Additional channels are visualized in Appx. D.

Figure 3.3: Examples of channels in all-channel push-up attack. The decoy images were successfully pushed into the top images. The Kendall-τ remains relatively high (> 0.5) suggesting much of the channel behavior is preserved while the top activating images completely obfuscate the behavior.

3.2.4 Whack-a-mole Results

We further analyze the existence of the whack-a-mole problem by observing Fig. 3.4 which shows for a given channel of AlexNet Conv5, the top-k images in the modified model which have the closest Kendall-τ-W and CLIP-W scores (not including the channel itself).

We observe that the first channel (channel 2 on Fig. 3.4) has little to no visually discernible similarity to nearby channels in the modified model as well confirmed by the Kendall-τ-W. On the other hand, we do observe similar images for the initial channel 193 and its nearest final one (163), which was picked as the most illustrative examples of the Whack-a-Mole problem. However, for this "hard" example, more insight is given by investigating the CLIP-W_j where the denominator notably

measures the clip similarity to other channels in the original model. A score of □ 1 suggests that the original model already had a high similarity to another channel. Later in this chapter we perform a more in depth analysis on channel 193's behaviour. To gain further insight into CLIP-W_j, in Fig. 3.5, we further visualize the numerator and denominator of CLIP-W_j for all the channels (red line) and sort them by the initial similarity to other channels (denominator). We observe that the red line is usually below the blue line, and if not, it is not by a large amount. This suggests that channels with high whack-a-mole metrics are actually ones that already had similarities to other channels in the original model. Overall we conclude the presence of the whack-a-mole problem is minimal in our current attack.

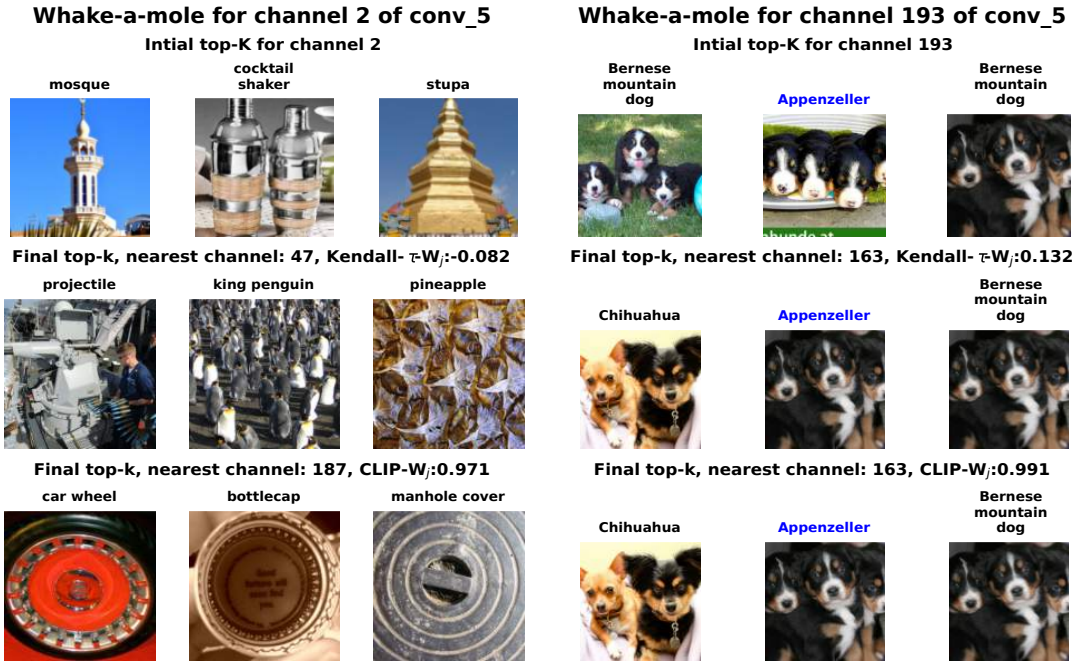


Figure 3.4: We show the initial top images for two channels and beneath are the corresponding final top images of closest channels w.r.t Kendall-τ-W_j and CLIP-W_j.

Zoom onto Channel 193 for Whack-a-mole: We begin by showing the full overview of the behavior of channel 193, selected as one "hard" case where similar initial images are found in final (post-attack) top-k images of another channel. Although similar initial images for channel 193 were found in channel 163 after the attack, it appears from the second row of Figure 3.6 that channel 193 was initially highly correlated with the channel 90 according to CLIP-δ score. Moreover, the fact

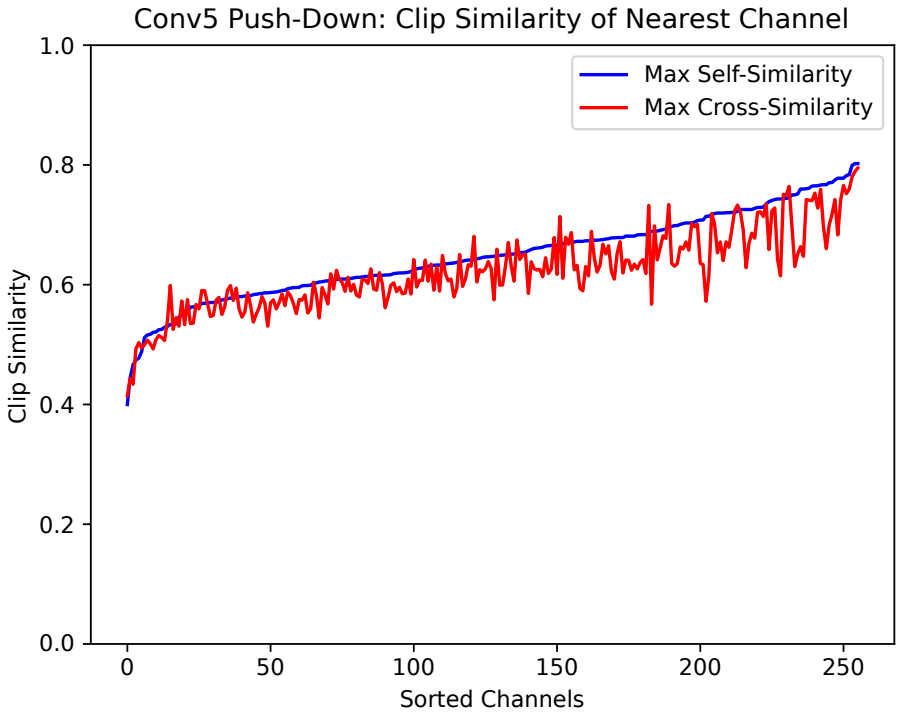


Figure 3.5: We compare initial CLIP similarity to other channels (blue) versus similarity after performing a push downattack (red). Red and blue largely track each other for all channels. This indicates that the channel behaviour is not being permuted within a layer.

that the CLIP-δ-W_j is 0.991 < 1 shows that the nearest post-attack channel (channel 163) is not more correlated than the nearest pre-attack channel (channel 90) according to CLIP scores. This, therefore, limits the existence of the whack-a-mole problem on this channel.

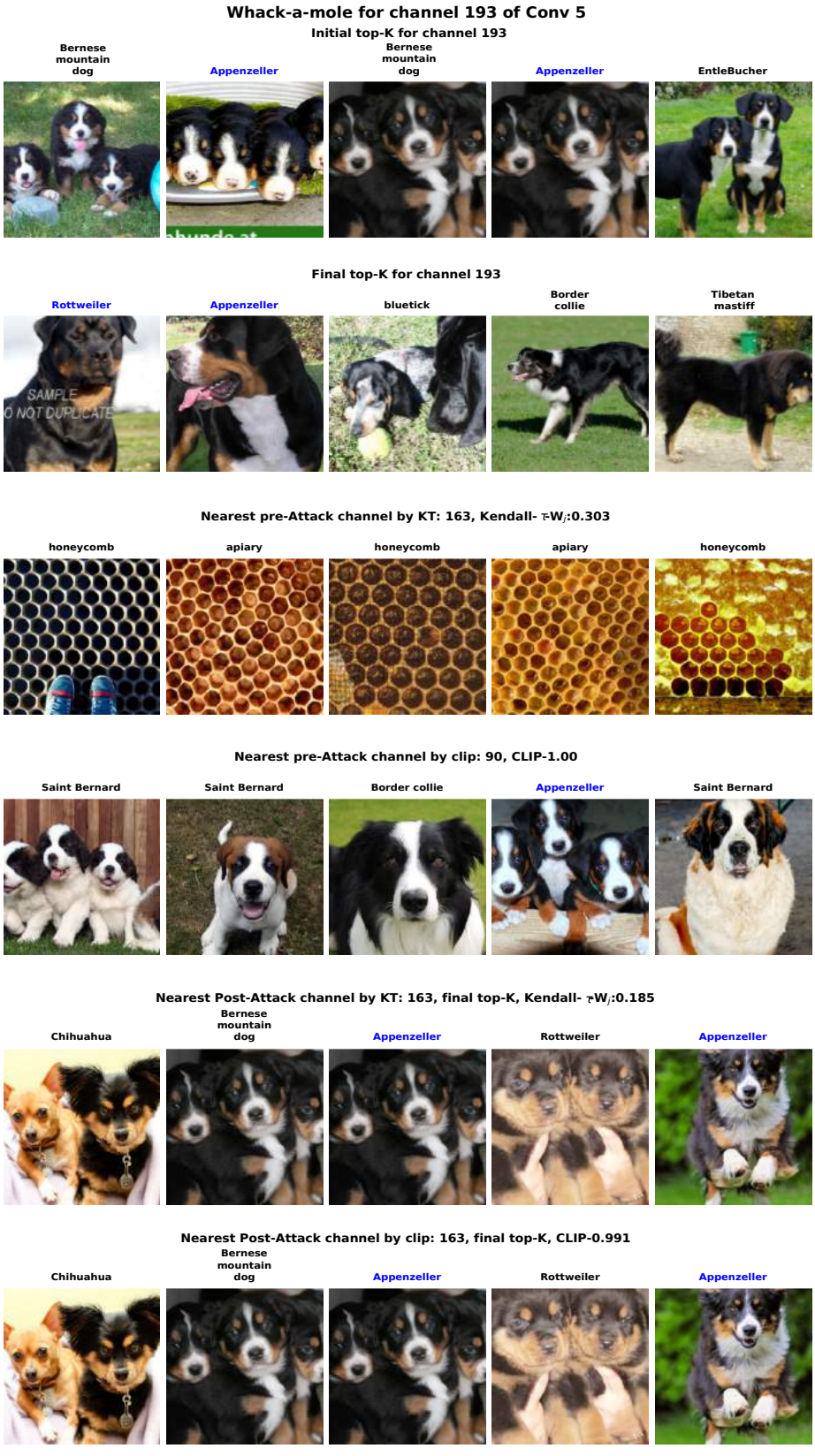


Figure 3.6: Illustrations for the existence of whack-a-mole on the channel 193, found as one of the high whack-a-mole cases. The first two rows show the initial and final top-k images for the targeted channel. The third and fourth show the initial nearest channels w.r.t. Kendall-τ-W_j and CLIP-δ-W_j, respectively. The fifth and sixth show the nearest post-attack channel by Kendall-τ-W_j and CLIP-δ-W_j respectively.

3.2.5 Synthetic Feature Visualization Study on AlexNet

As synthetic images are an important method of studying network behaviours (Olah et al., 2017), we study the impact of the push-down and push-up attacks on the synthetic activation-maximizing images of the channels under attack. Synthetic activation-maximizing images are the result of an optimization problem over input pixels solved by gradient ascent on the channel activation under a norm constraint in pixel space. To avoid adversarial noise samples, (Goodfellow, Shlens, & Szegedy, 2014), it is necessary to jitter the input image or parameterize it as a smooth function (Olah et al., 2017).

In Fig. 3.7, we visualize the synthetic optimal images for several channels before and after the attack. By visual inspection, one can see that the top-k images change while the synthetic optimal image is largely unaffected. The most common observed change for AlexNet Conv5 is a low-frequency modulation of the pattern. We hypothesize that this is because the top-k attack most significantly modifies the weights of the attacked layer, which is a later layer preceded by several downsampling operations.

The lack of change in the synthetic optimal image suggests that the synthetic feature visualization and the top-k analysis are, counter-intuitively, highly de-correlatable. Further, this does not permit the conclusion that the synthetic optimal image is more robust to attack since we have not explicitly run an attack against it.

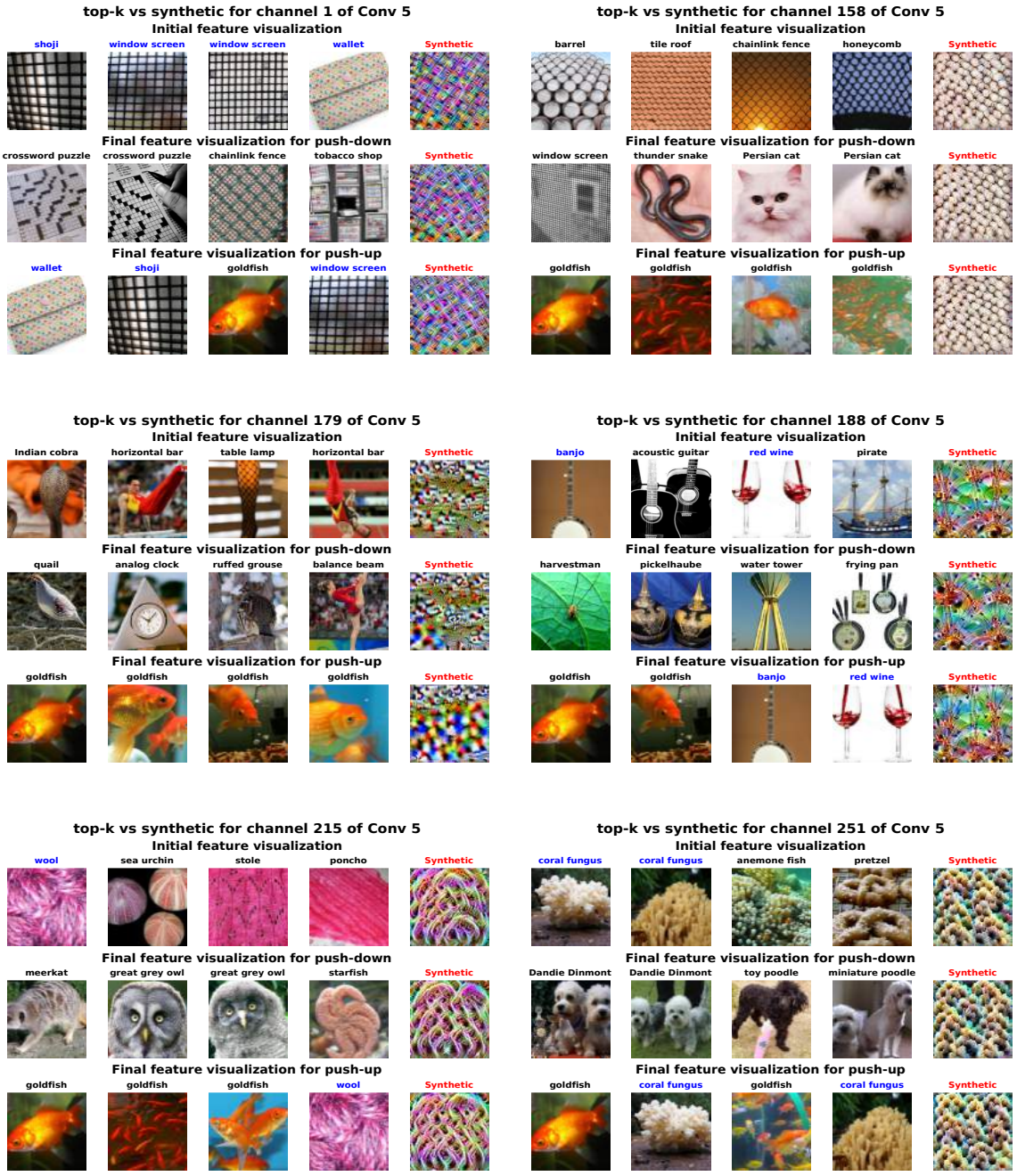


Figure 3.7: Synthetic Feature Visualization attack after push-down and push-up attacks on Conv5 of AlexNet. Channels indexes were taken randomly. We observe a decorrelation between natural top-activating images and synthetic optimal images.

Chapter 4

Pruning

4.1 The Pruning Strategies

The previous section of this work focused on the development and characterization of novel interpretability attacks on DNNS. These attacks, and several works discussed in 2.2.1, underscore the need to develop ways to prevent malicious actors from developing nets with misleading interpretations. To this end, this chapter is focused on creating networks that are resilient against our attacks. A successful defence would produce a performative network whose interpretation is minimally affected by the attacks. Our first attempt at this involves the use of model pruning in an attempt to create resilient networks. As discussed in section 2.3, pruning is a popular method for reducing the size and cost of DNNs. In the context of the search for a potential defense against the attacks we've defined, it is important to consider that pruning shows that nets such as AlexNet are larger than they need to be to accomplish their tasks (Han et al., 2015; Lebedev & Lempitsky, 2015). This leads us to ask a question: Would the removal of extraneous model weights cause a reduction in model susceptibility to our attack? We propose two distinct justifications as to why this may be the case. Firstly, if the net is larger than required to complete the classification task then it is reasonable that it may be able to learn to both complete the task while also satisfying the additional restraint from the attack. Secondly, particularly for the Push-Down attack, the activation norm may be reduced by pushing the unimportant weights to 0, which is analogous to pruning them. By reducing the size of the network we expect to reduce the complexity of the functions it can learn, and expect this would

lead to increased resilience against the attacks. Ultimately, we will be assessing the various pruning methods based on the trade-off between final model performance and resiliency to our attacks, measured in top-1 accuracy and CLIP-δ respectively.

As such we propose a set of experiments in which we prune AlexNet, chosen for its popularity and relative simplicity, using several different methods and then attack the resulting networks using our Push-Up and Push-Down attacks, defined in section 3.1, targeting the last convolutional layer, Conv5. Note that additional details on the attack results, including whack-a-mole metrics, are present in Appx. H.

4.1.1 Norm Pruning

Chosen due to its popularity, effectiveness, and relative simplicity, the first pruning strategy explored is norm based pruning (Lebedev & Lempitsky, 2015; H. Li et al., 2017; Molchanov et al., 2017; Wen et al., 2016).

We use the class-blind algorithm defined in See et al. (2016), in which the bottom p% weights are all pruned purely by their l1-norm, leading to unstructured pruning of the model. This corresponds to the following transformation on the initial weights in the unpruned model, θ_0.

$$w_p = w_0 \text{ if } |w_0| > t, \text{ else } w_p = 0 \text{ for } w_0 \text{ in } \theta_0 \tag{10}$$

Where t is a threshold defined to enforce the pruning rate, p, desired, w_0 refers to a given weight in the unpruned model θ_0, and w_p is the corresponding weight in the pruned but not fine-tuned model. Pruning in this manner results in a performance loss if there is no fine-tuning step. Additionally, there is no guarantee that the model is in a stable local minimum after pruning, which could lead to inflated attack success. Before performing our attack we therefore fine-tune until performance stabilizes. We fine-tune by retraining the net on ImageNet (Deng et al. (2009)) for 20 epochs using the Adam optimizer (Kingma and Ba (2017)) with an initial learning rate of 10^{-5}, reduced to 10^{-6} after 10 epochs. This process was repeated for several pruning rates, up to an 87.5% overall reduction in the number of weights in AlexNet.

4.1.2 OTO

To investigate the effect of structured pruning on the interpretability attacks we use the Only Train Once (OTO) pruning framework, which will prune a combination of complete filters and weights in the linear layers (Chen et al., 2023). Rather than specify a specific portion of weights to prune, OTO functions by splitting weights up into groups (see section 2.3) and pruning a portion of those groups. The OTO library allows for developers to tune the prioritization of filter removal (faster computation) against linear layer weight removal (better performance). We take advantage of this to explore how the prioritization of complete filter removal affects the resilience of the model against our attacks. We prune up to 87.5% group sparsity in both a high filter pruning and a high weight pruning setup, and run the push-up and push-down attacks on all resulting networks.

4.1.3 Vulnerability Pruning

In addition to using more established pruning methods, we explore if it may be possible to use knowledge of model vulnerabilities to guide pruning. To do this, we use the results from the push-down attack and prune the channels with the highest CLIP-δ scores, experimenting with pruning up to 224 channels. To keep the effect on the network as small as possible, we prune exclusively from the layer we are attacking, Conv5. We then fine-tune the vulnerability pruned AlexNet using the same procedure described in subsection 4.1.1. Our expectation is that by pruning the most vulnerable channels and no other weights we may end up with a network that is less susceptible to the attacks at relatively low performance cost.

4.2 Pruning Results

4.2.1 Norm Pruning

The norm-based pruning method used resulted in networks with similar performance to a baseline AlexNet at 56.52% top-1 accuracy on ImageNet. Indeed, as seen in Table 4.1, up until 62.5% weight pruning there are even minor benefits with respect to model performance. This follows expected trends as seen in the works discussed in Sec 2.3.

Weight Pruning Rate	12.5%	25.0%	37.5%	50.0%	62.5%	75.0%	87.5%
Fine Tuned Accuracy	56.72%	56.67%	56.67%	56.70%	56.45%	56.13%	54.59%

Table 4.1: Here we report the accuracy of fine tuned norm-pruned networks. We see that even at the highest levels of pruning, the performance does not degrade more than 2%, indicating that this method is highly effective at removing weights with minimal accuracy loss.

While the norm-pruned networks perform well, there is less indication that they provide significant protection against the push-up and push-down attacks, as can be seen in Fig. 4.1. While the CLIP-δ score of the push-down attack does decrease slightly as pruning increases, the correlation is extremely minor and does not indicate attack failure at any point. Furthermore, while the push-up scores show a stronger variance, it does not indicate a clear connection between increased pruning and model resilience. Indeed, an inspection of Fig 4.1 indicates that pruning affects the scores presented erratically, making it difficult to conclusively determine how the two are related.

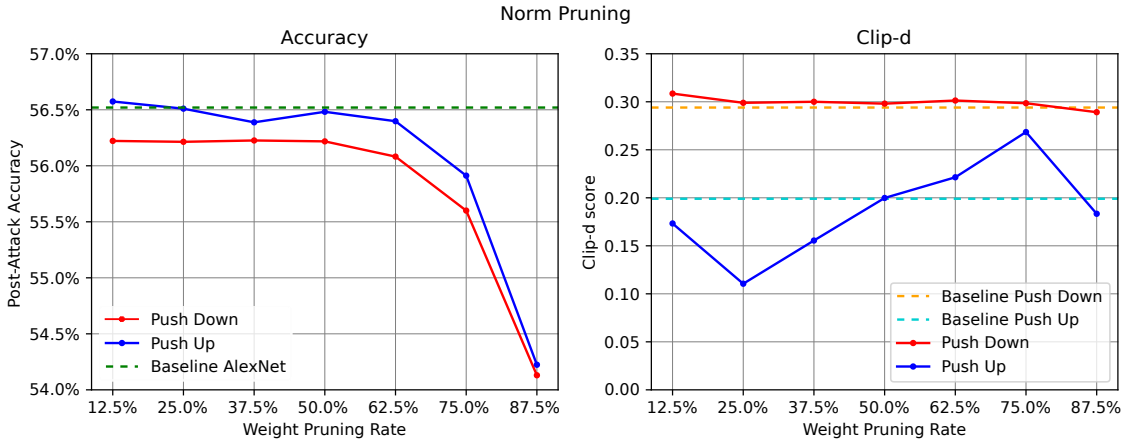


Figure 4.1: Accuracy and Clip-δ results for norm pruned networks. The norm pruned networks have relatively stable performance, maintaining performance close to an unpruned AlexNet even at high pruning rates. For the push-down attack we note no significant shifts in the attack vulnerability, while the push-up results are less consistent, indicating that the pruning strategy may provide some protection, but only under specific, unclear, circumstances.

4.2.2 OTO

OTO pruning is a more sophisticated pruning network whereby the weights of a net are partitioned into groups, and those groups are pushed to 0 and then pruned (Chen et al., 2021, 2023). We

include results for use of the method with both a channel pruning and weight pruning focus. We report the respective network performance and pruning rates in Tables 4.3 and 4.2. With respect to accuracy, the weight pruning OTO nets have the highest accuracy at lower pruning rates, as seen in table 4.2. Conversely, at higher pruning rates we see a catastrophic loss of performance across both OTO variants.

Target Group Sparsity	12.5%	25.0%	37.5%	50.0%	62.5%	75.0%	87.5%
Filter Pruning Rate	0.2%	0.4%	0.4%	0.5%	0.6%	3.3%	18.2%
Weight Pruning Rate	16.7%	35.0%	51.4%	67.9%	81.0%	90.7%	96.6%
Fine Tuned Accuracy	57.1%	56.8%	56.2%	55.4%	53.2%	45.4%	14.7%

Table 4.2: Here we report the characteristics of OTO weight pruned nets across several target sparsities. We note very low filter pruning until the highest pruning rates, which is correlated with more severe decrease in the accuracy after fine-tuning. Compared to norm based pruning, as seen in Ta-ble 4.1, OTO pruning results in better performances at lower weight pruning rates but does not scale as well to higher rates.

Examining Fig. 4.2 for the OTO filter pruned attack results, we note that there is a downward trend in the CLIP-δ scores, mirroring a reduction in the model accuracy. This is the strongest protection achieved by pruning, but as it only appears after the model's performance has so severely deteriorated, it cannot be said to have been a successful defense. In the case of the OTO weight pruning strategy shown in Fig. 4.3 there is no evidence of any increase in resilience, even at the highest levels of pruning and accuracy loss.

Target Group Sparsity	12.5%	25.0%	37.5%	50.0%	62.5%	75.0%	87.5%
Filter Pruning Rate	8.7%	23.3%	26.1%	27.8%	32.3%	37.6%	47.7%
Weight Pruning Rate	16.6%	31.8%	44.5%	54.3%	65.7%	74.1%	62.9%
Fine Tuned Accuracy	56.3%	53.2%	52.1%	50.5%	46.2%	35.7%	7.9%

Table 4.3: This table reports the baseline performance and characteristics of OTO pruned AlexNet with a focus on complete channel removal. Removing complete channels results in a more significant accuracy drop than other pruning methods. This implies that while many of the weights inside a filters can be removed safely, the filters themselves each carry important information.

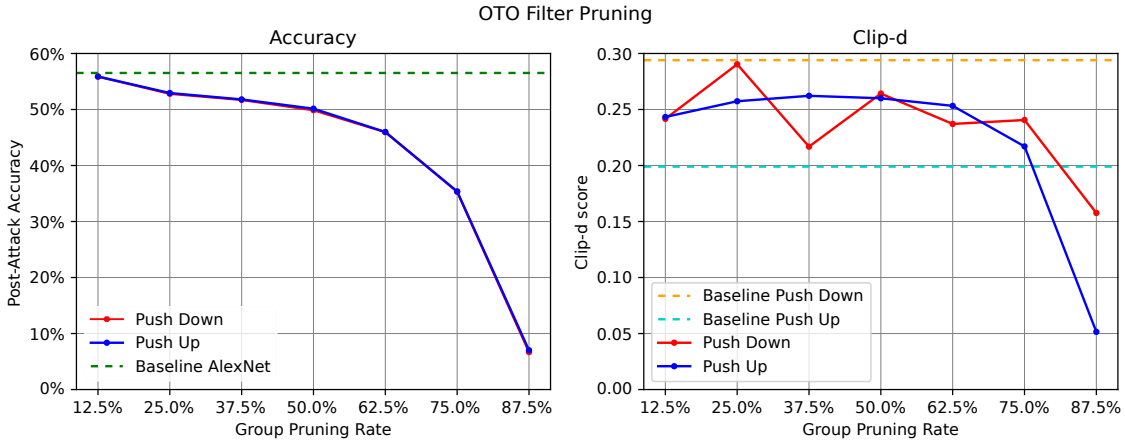


Figure 4.2: This figure reports the performance and attack success, measured with accuracy and CLIP-δ on model trained using OTO pruning with a focus on channel removal. This results in some protection against the attacks at the highest pruning rate, as the CLIP-δ scores begin to drop sharply. At these pruning levels the model's accuracy has dropped catastrophically, indicating that this pruning method did not generate nets that are resilient and high-performance.

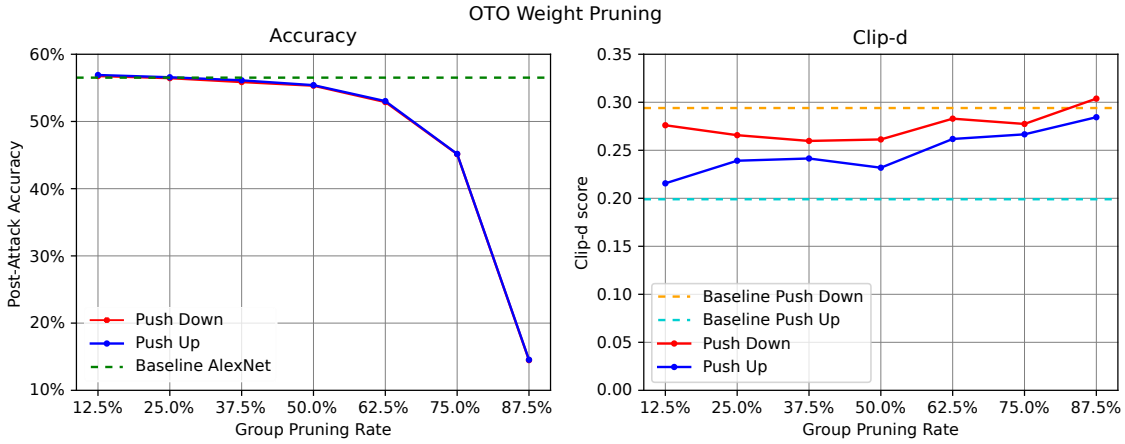


Figure 4.3: OTO-trained nets with a focus on weight removal maintain accuracy better at higher pruning rates than filter-pruned OTO nets. These nets do not indicate any resilience to either the push-up or push-down attacks, maintaining steady CLIP-δ values across all pruning rates.

4.2.3 Vulnerability Pruning

In an attempt to prune model vulnerabilities directly, we remove channels from AlexNet's Conv5 based on the CLIP-δ score reported for each channel calculated for the push-down attack on Conv5 discussed in section 3.1.3. In table 4.4 we report the performance and pruning rates of the fine-tuned models. Pruning only channels from the target layer of the net results in a significantly

lower overall weight pruning rate, and a disproportionate effect on the model performance. Indeed, even the 50% weight pruned net reported in table 4.1 reports higher performance the the 32 channel pruned model here. This indicates that while many of the individual weights in a channel are not necessary for performance, each still contains important weights. We report results for attacks on each pruned net in Fig. 4.1, but note that for nets with 96 or more pruned channels that the cost in accuracy may be too significant for developers to pursue this defense.

Filters Pruned	32 Pruned	64 Pruned	96 Pruned	128 Pruned	160 Pruned	192 Pruned	224 Pruned
Filter Pruning Rate	2.78%	5.56%	8.33%	11.11%	13.89%	16.67%	19.44%
Weight Pruning Rate	0.12%	0.24%	0.36%	0.48%	0.60%	0.72%	0.84%
Fine Tuned Accuracy	56.29%	55.89%	55.45%	54.60%	53.16%	50.85%	44.01%

Table 4.4: This Table reports results on nets where channels were pruned from Conv5 based on which had the highest CLIP-δ scores. While the overall weight pruning rate is significantly lower than in the other pruning methods, the performance drop seen is still significant. Compared to OTO and Norm pruning, this method has the highest drop in accuracy relative to the amount of pruning.

Fig. 4.4 presents the performance and results for the vulnerability pruned network. The post-attack accuracies closely follow that of the fine-tuned net. The CLIP-δ scores indicate that some minor protection may be afforded against the push-down attack, but that this is not particularly significant. Additionally, we note that the push-up attack actually seems to be protected against at low pruning rates, while becoming less resilient as the pruning increases. This pruning does not seem to meaningfully increase model resilience, particularly in light of the accuracy cost.

4.2.4 Pruning Results Summary

The results we have presented have failed to support the hypothesis that pruning a model would result in increased resilience. While it may be the case that reducing the size of the net constrains the possible algorithms the net can learn, it seems that this results in the network failing to learn its original task before it fails to satisfy the attack. Pruning models is still a valuable technique for reducing model size and resource cost, but none of the techniques explored in this section have succeeded in our goal of producing a viable model defense.

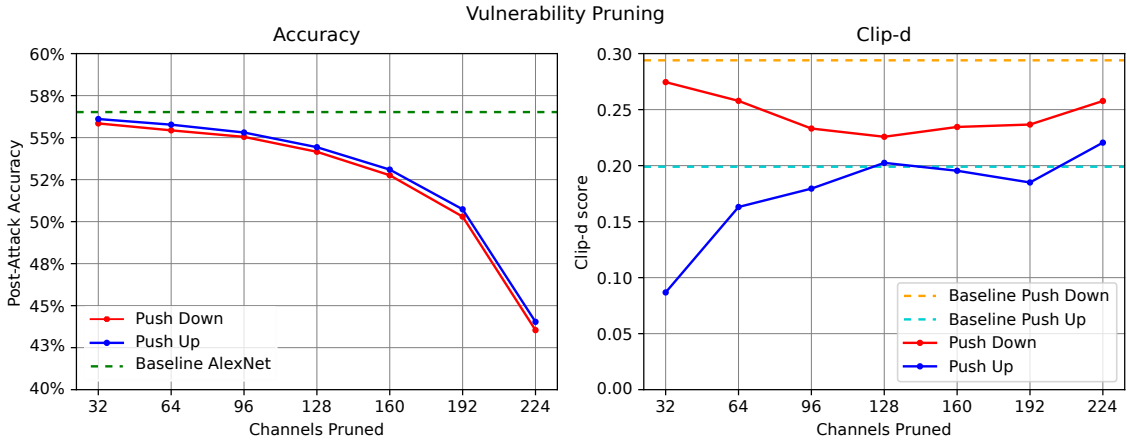


Figure 4.4: For the vulnerability pruned nets we see that there is little consistent change in actual vulnerability to attack, even as the nets' accuracies drop. We note again the the push-up attack is particularly volatile across pruning rates.

Chapter 5

Conclusions and Future Work

5.1 Contributions

Interpretability Attacks: In this work we have demonstrated the possibility of manipulating feature visualizations across several different Neural Net architectures, including convolutional net, residual networks, and transformers. Both proposed attacks, the untargeted push-down and targeted push-up, show the capacity to alter model interpretation without significant changes in model performance. We define and quantify metrics to assess attack success, as well as ensure that non-trivial solutions were reached i.e. that the whack a mole problem does not occur.

Model Defense: We have explored the use of pruning as a potential defense against the interpretability manipulation. While it is still hypothetically possible that some pruning method may be able to generate a resilient and performative model, the variety of methods we explored were unable to support this hypothesis. Indeed, further work on defending against the attacks is a natural next step for this project.

5.2 Future Work

Expanding Attacks: The attacks presented in this work are intended as a proof of concept that show it is possible to manipulate the feature visualization of an image classifier. More sophisticated

attacks targeting specific circuits and obfuscating their functioning could be developed. Fairwashing, the practice of making a model's decision seem fairer than they truly are, is an additional potential application for this family of techniques. In addition to expanding the types of attacks available expanding them to work with Large Language Models, or other nets working on non-vision tasks would be an interesting avenue of research. It is our hope that by identifying strategies that can be used to manipulate interpretability, we will be better equipped to defend against them.

Defense Strategies: Of pressing concern is the development of strategies to protect against the use and deployment of manipulated nets. Indeed, as DNN usage becomes more and more prevalent, the risk of bad-faith actors releasing unfair or biased models that do not meet appropriate standards will only increase. Protection may come in the form of model creation techniques that create families of models that are hard to manipulate, as we had attempted to do with our pruning experiments. Should such models prove broadly unfeasible, another important avenue of research would be in the detection of manipulated models. The detection of manipulation could allow for regulatory agencies to prevent such models from being used, and may even be a more fruitful path forward.

A Hyperparameters and Training Details

This section presents the details of the hyperparamters and training settings used to run our attacks.

A.1 Push-Up and Push-Down Attacks

We train for 5 epochs over the ImageNet-1k training set with a batch size of 256. We use the Adam optimizer with learning rate 1e-5.

Regarding α, we employ a dynamic updating rule inspired by Algorithm 1: Dynamical balancing of Distillation and CKA map loss in appendix A of Davari et al. (2022) in order to have better control over loss in accuracy. We initialize α as 0.1. If the accuracy loss is greater than 0.5% we halve the current α. If it is less than 0.5% we double α. With this dynamic update, we aim to minimize the loss in accuracy while still ensuring the top images shifts.

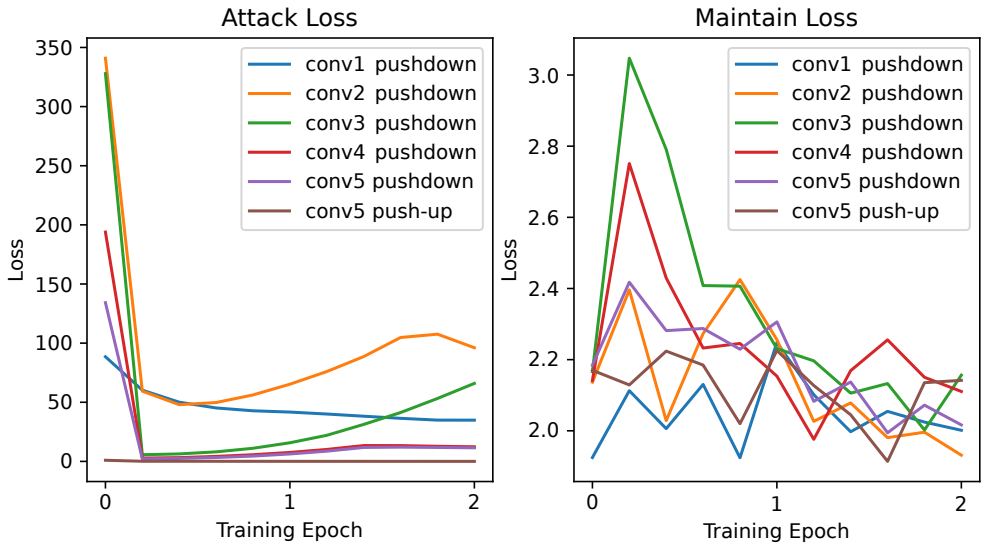


Figure A: Sample training curves for the maintain and attack objectives. Late layers (conv5, conv4) are easier to attack compared to early ones (conv1, conv2 and conv3). The maintain loss is very close to its initial value after two epochs.

A.2 Optimization Curves

We show in Figure A the evolution of attack and maintain losses across two epochs. It can be observed that the attack loss of late layers (conv 4, conv 5) decreases very quickly, and almost monotonically, showing the easiness to attack late layers. In contrast, early layers do not have the same behavior. We can also observe from the training curves that the maintain loss is almost close to its initial value after 2 epochs. This corroborates the observed accuracy preservation as shown in Table 1 of the paper.

B Additional Results for Push-down Attack on a Single Channel and on all Channels

Before showing additional results for the push-down attacks on a single channel and on all channels simultaneously, we present below the finetuning baselines.

B.1 Finetuning Baseline

One can observe that the finetuning baseline fails to change top-k images, in particular, top-4 images are exactly the same in the visualized channels seen in Fig. B. This is also materialized empirically in the kendall-τ coefficients approaching 1, indicating that the ordering of the images has not significantly shifted. The CLIP-δ scores also indicate that the semantic change in the top-k images is minimal. Overall, the finetuning results in an AlexNet that is extremely similar to PyTorch's default with respect to performance and Interpretation via Feature Visualization.

B.2 Push-Down Attack on Single Channel

Figure D shows the results of initial top-k images and final ones after running the push-down attack on every single channel. Except for channels 6 and 4 with relatively low CLIP-δ scores, it can be observed that all other channels have semantically different final top-k images compared to the initial ones. This can be also seen by higher values of CLIP-δ scores.

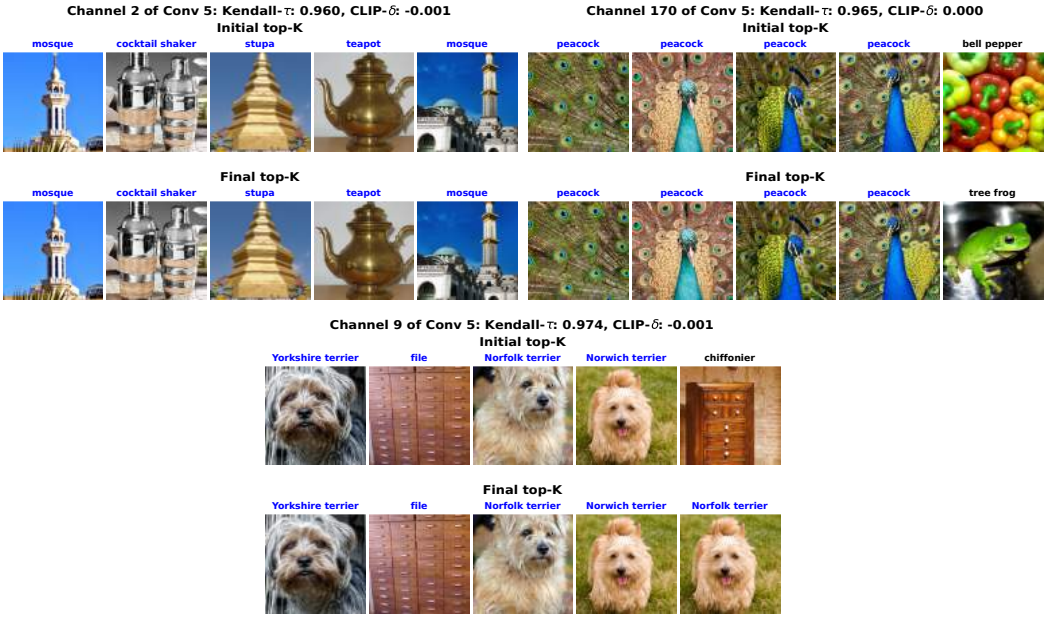


Figure B: Finetuning baseline result on Conv5 of AlexNet. All initial images are almost the same after finetuning. Kendall-τ and CLIP-δ are respectively close to 1 and 0, suggesting almost zero changes in channel behavior and semantic changes.

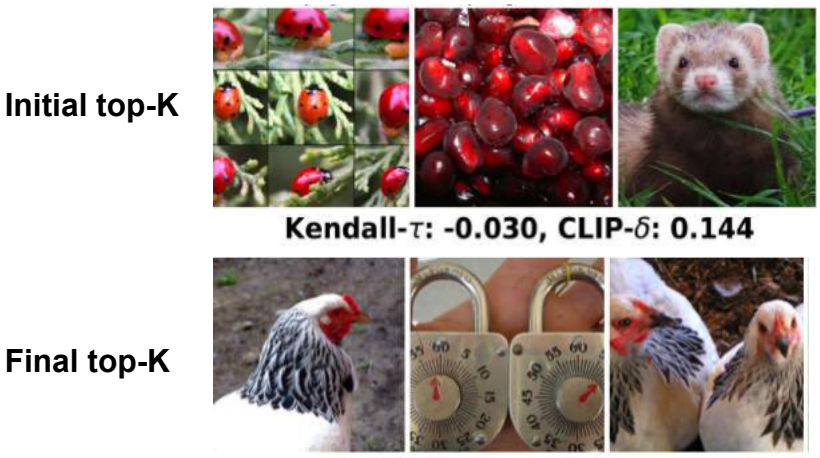


Figure C: Top images for a channel before and after a single-channel Push-Down attack.

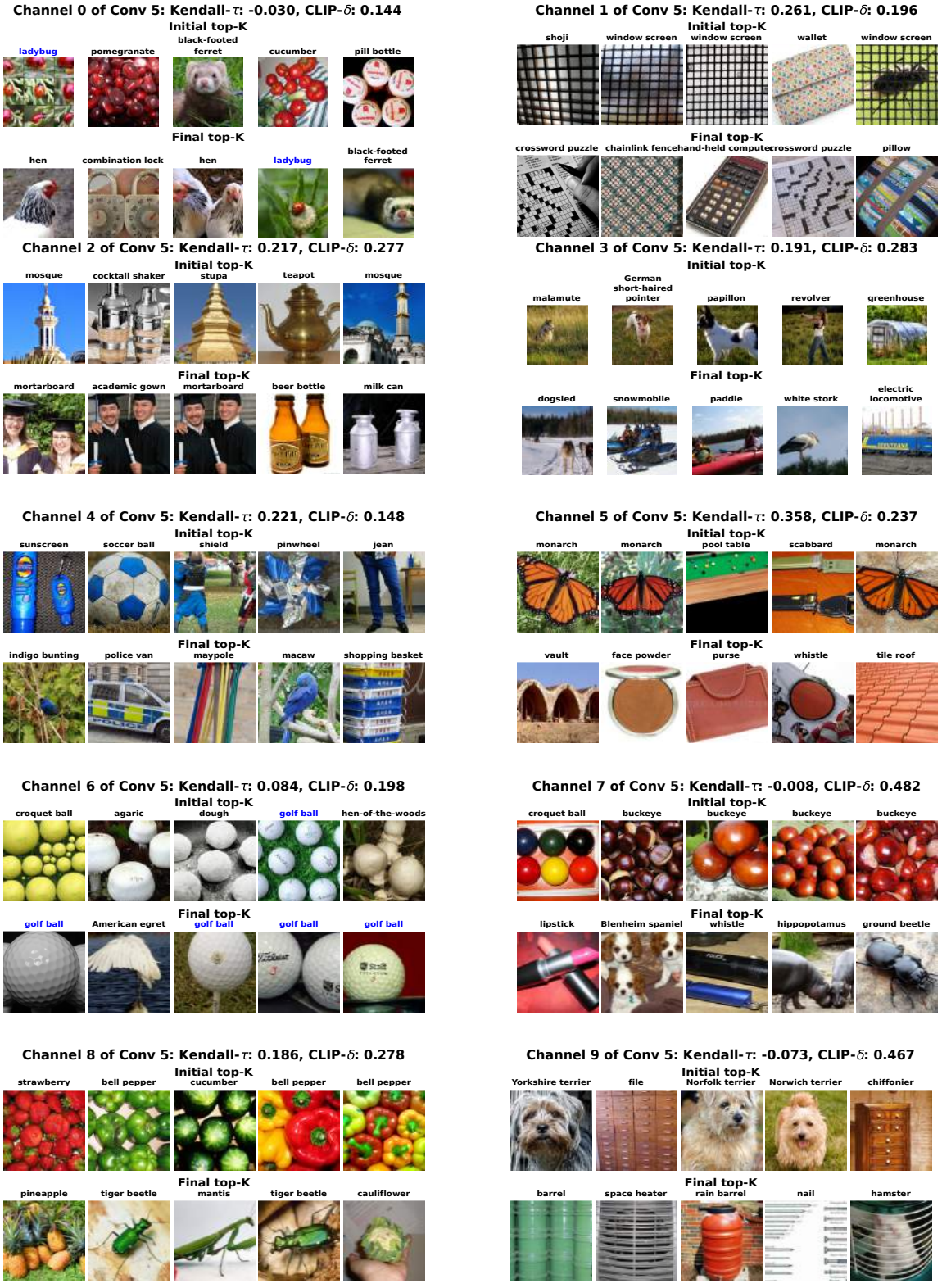


Figure D: Push-down attack on a single-channel of Conv5 of AlexNet. All initial images have been replaced by other images.

B.3 Push-down All-Channel Validation Visualization

Generalization on Validation Set. We evaluate the generalization of our attack on the validation set of ImageNet. This gives more insights to the change of feature visualization. Figures E and F show the initial top-k images and final ones from training and validation sets for 10 randomly chosen channels.

It can be observed that on every channel, from the validation set, at least one image from the initial top-5 images is no longer present in final top-5 images (for the majority of these channels, the first top-activating is no longer the top one). We also observe a complete replacement of top-5 images on the validation set when Kenall-τ scores and CLIP-δ are respectively low and high simultaneously (e.g., channels 37 and 50 of Figure E). Moreover, the general trends in training and validation are similar suggesting the attack is not just memorizing specific images but leading to a generalized change.

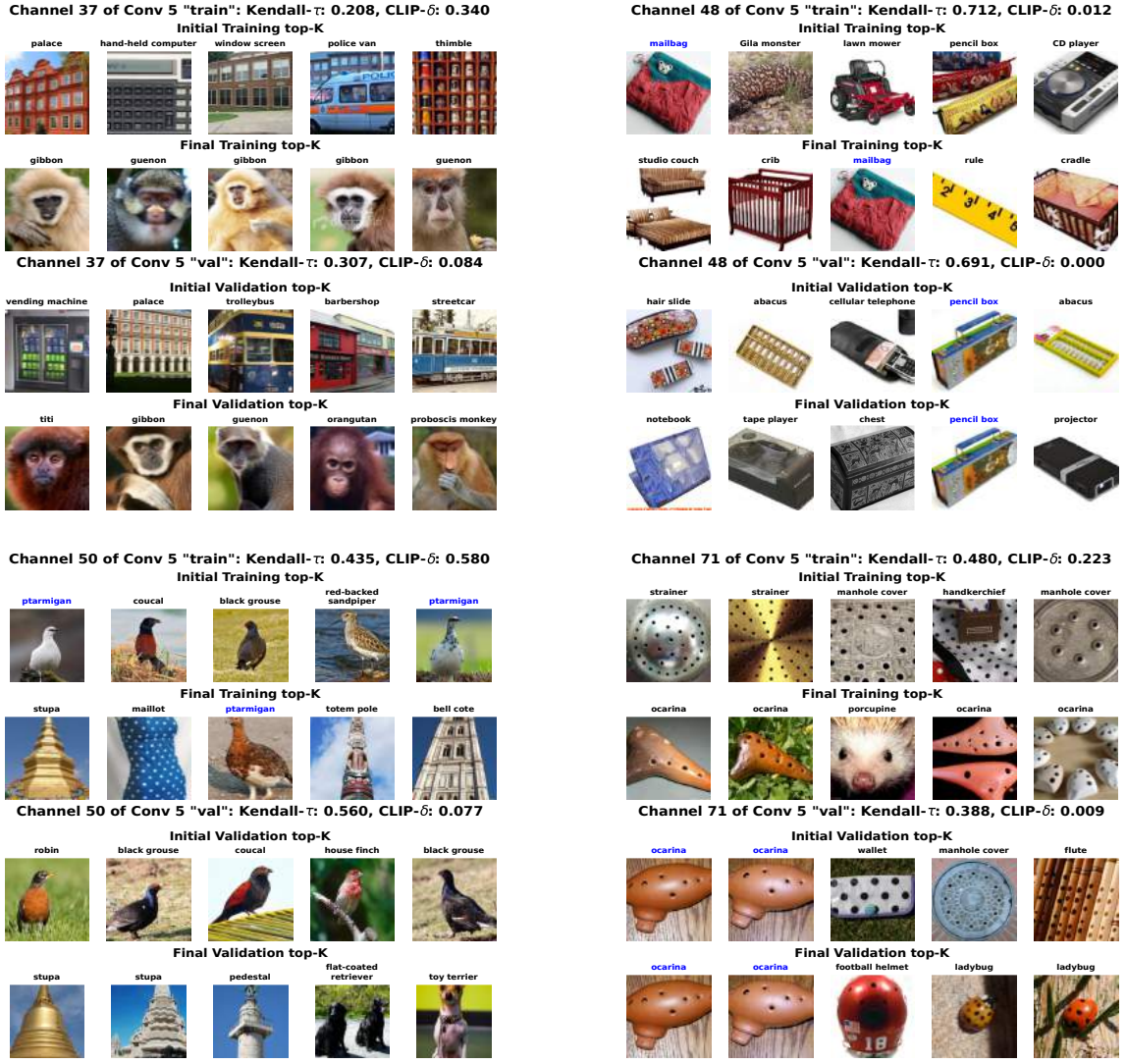


Figure E: Push-down all-channel attack of Conv5 of AlexNet. For each channel, the first two rows are top-k images derived from the training set while the last two are derived from the validation set.

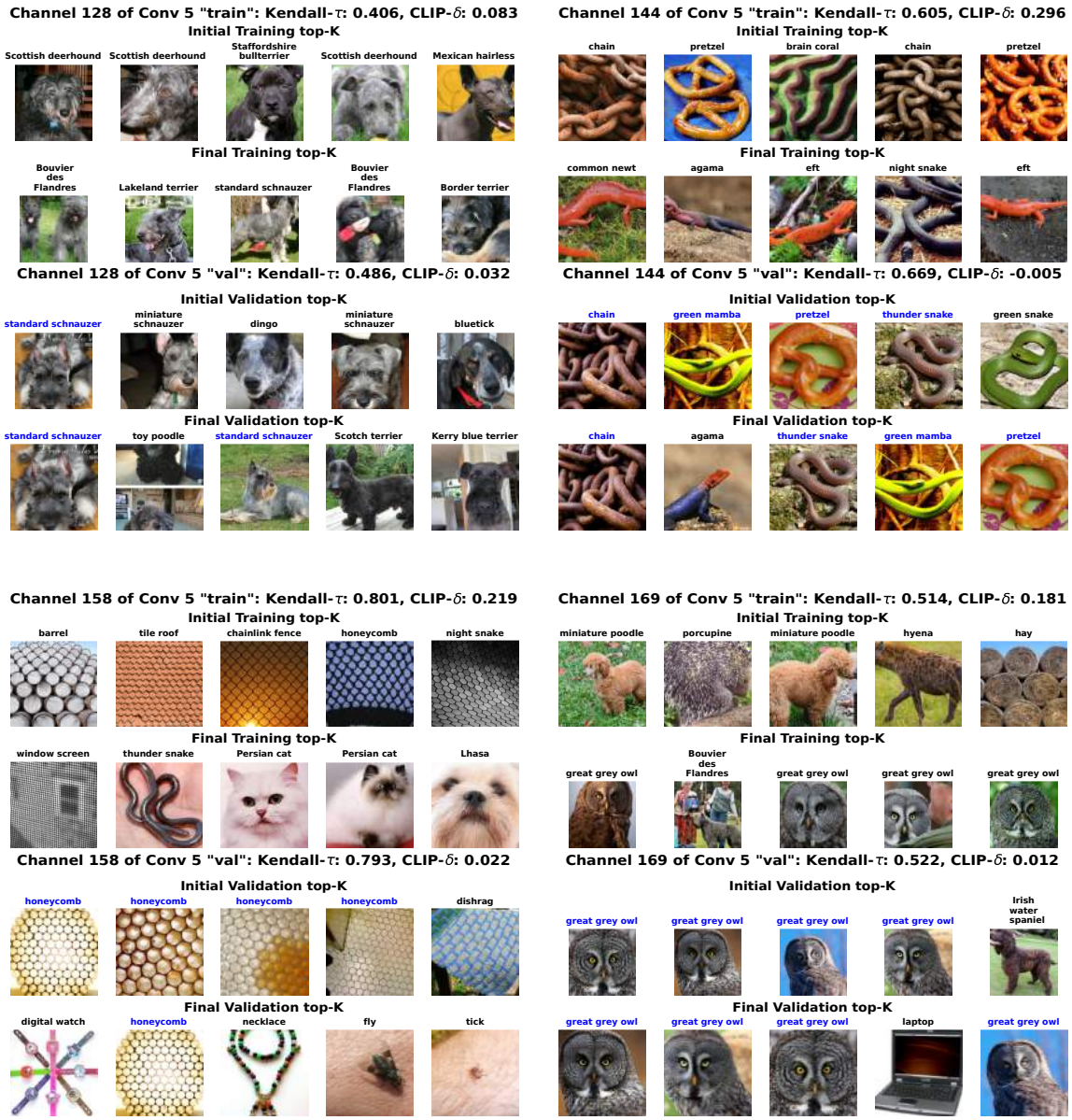
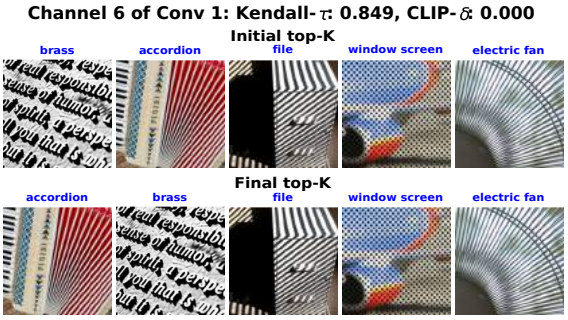


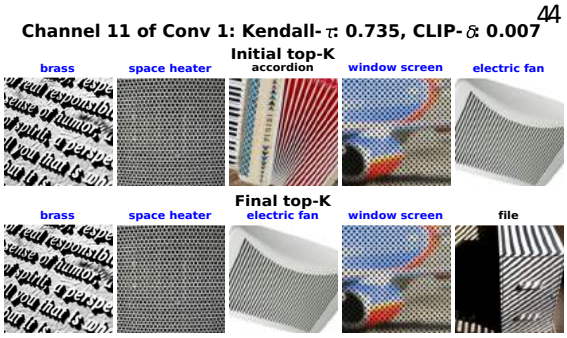
Figure F: Push-down all-channel attack of Conv5 of AlexNet. For each channel, the first two rows are top-k images derived from the training set while the last two are derived from the validation set.

C Effect of Depth

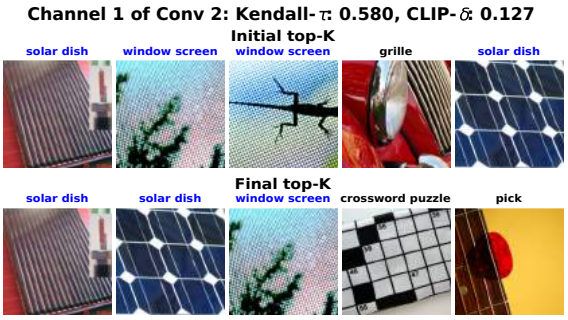
We vary different layers of AlexNet and evaluate how the attack is affected by depth. Figure G shows results obtained on randomly chosen channels from conv1, conv2, conv3, and conv4 of AlexNet. It can be observed that the earliest layers conv1 and conv2 are harder to attack. This is materialized by high values of Kendal-τ and low values of CLIP-δ scores. When increasing the depth (conv3 and conv4) we observe a complete replacement in top-5 images in channels 147 (conv3), 121 (conv4) and 124 (conv4), although some of these channels have low values of CLIP-δ scores.



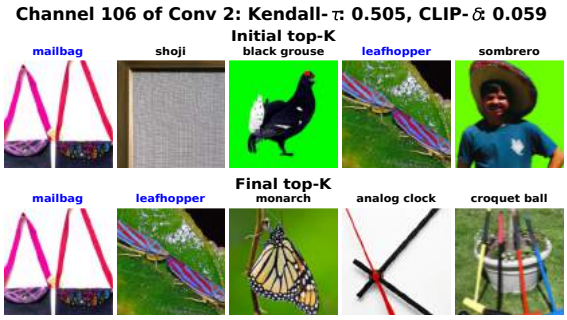
(a) Layer: Conv1.



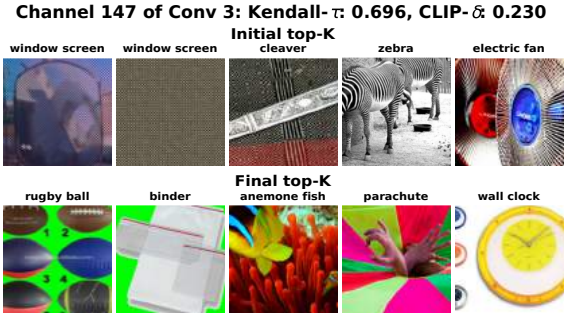
(b) Layer: Conv1.



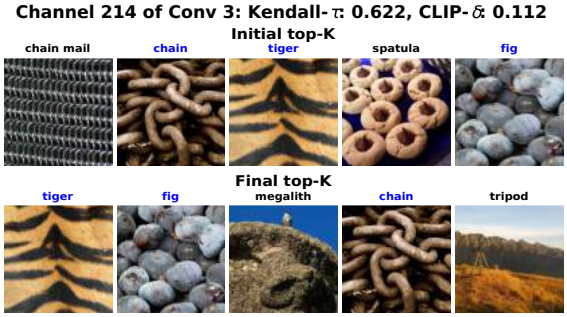
(c) Layer: Conv2.



(d) Layer: Conv2.



(e) Layer: Conv3.



(f) Layer: Conv3.

(g) Layer: Conv4.

(h) Layer: Conv4.

Figure G: Push-down all-channel attack of on several layers of AlexNet. Channels indexes were selected randomly. While there are some changes in top-activating images of early layers (conv1 and conv2), they are not significant as materialized by low values of CLIP-δ and high values of Kendall-τ. For conv3 and conv4, we see a complete replacement of top-5 images on channels 147 (conv3), 121 (conv4), and 124 (conv4).

D Additional Illustrations for the Push-up Attack

This section provides additional visual illustrations of the push-up all-channel attack on the layer conv5 of AlexNet.

Visual Examples. We first provide additional visual illustrations in Figure H of the attack on 10 randomly chosen channels. As a reminder, this push-up attack aims to make images of the Goldfish class appear in the top-k images of every channel on the targeted layer. From Figure H, a first observation is the fact that out of these 10 randomly chosen channels, only two channels (channel 15 and channel 23) do not show an image with the Goldfish class. On the rest of the channels, an image with Goldfish was successfully inserted in the final top images. Furthermore, in several cases (channels 110, 125, 145, 180, 183, and 50) is the majority class of final top-5 images, demonstrating the success of this attack. It is also important to note the complete replacement of images with the Goldfish class in some channels (e.g., channel 125).

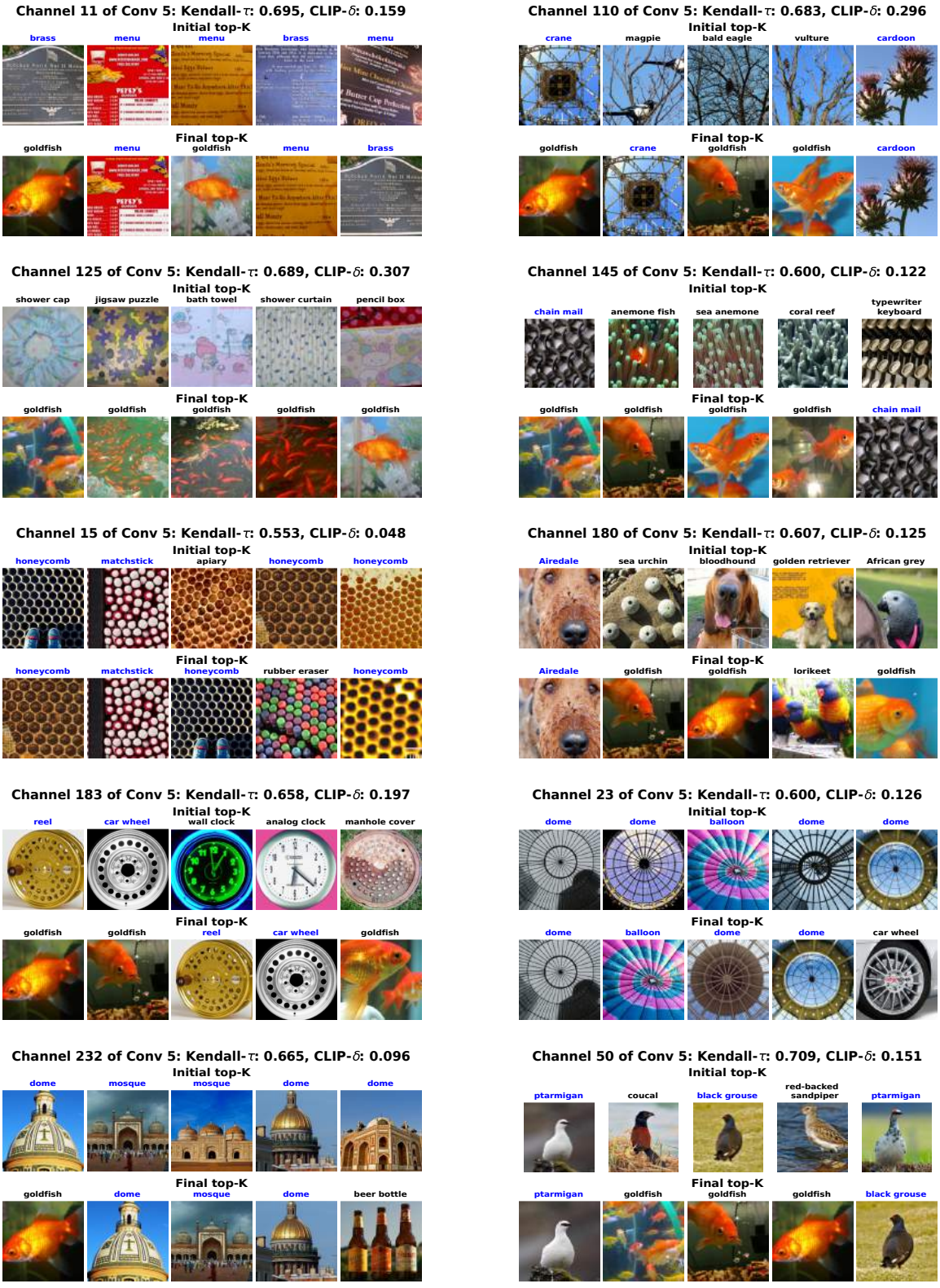


Figure H: Push-up all-channel attack of Conv5 of AlexNet. Channel indexes were taken randomly.

Generalization for the Push-Up attack. After demonstrating the success of achieving target manipulability of top-k feature visualization through the push-up attack on training images, it is also important to evaluate whether this success generalizes to unseen data. Figure I shows not only top-k images from the training but also from the validation set of ImageNet. We can observe that on all the 10 randomly chosen channels not only at least one image of the Goldfish class is present in the final top-5 images of the training but also at least one image of the Goldfish class is in the final top-5 images from the validation set. Moreover, we also observe a similar number of images of the Goldfish class present in top-5 images from both training and validation sets. This indicates the ability of the push-up attack to generalize on the same distribution from where training examples were drawn.

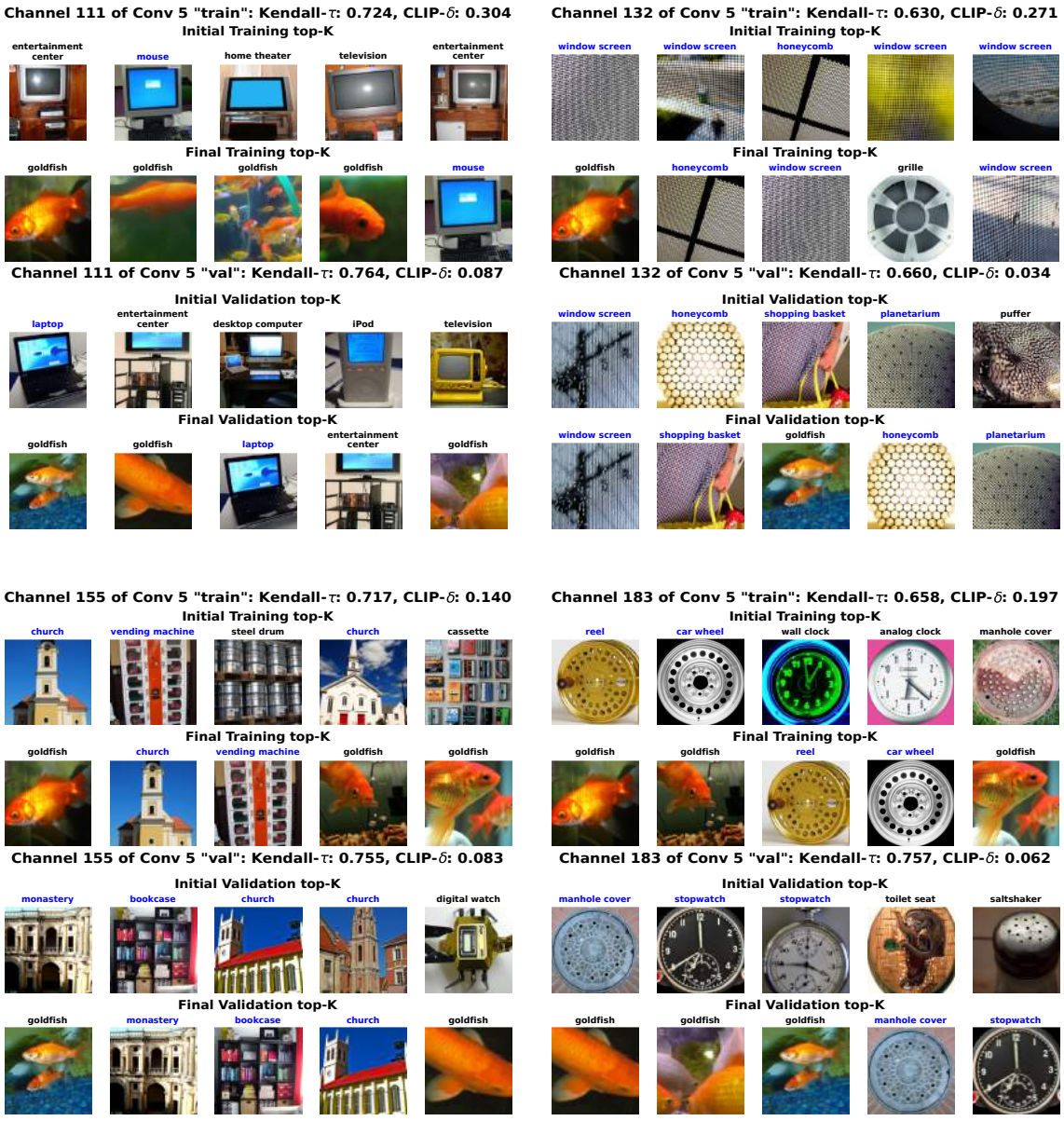


Figure I: Push-up all-channel attack of Conv5 of AlexNet. For each channel, the first two rows are top-k images derived from the training set while the last two are derived from the validation set.

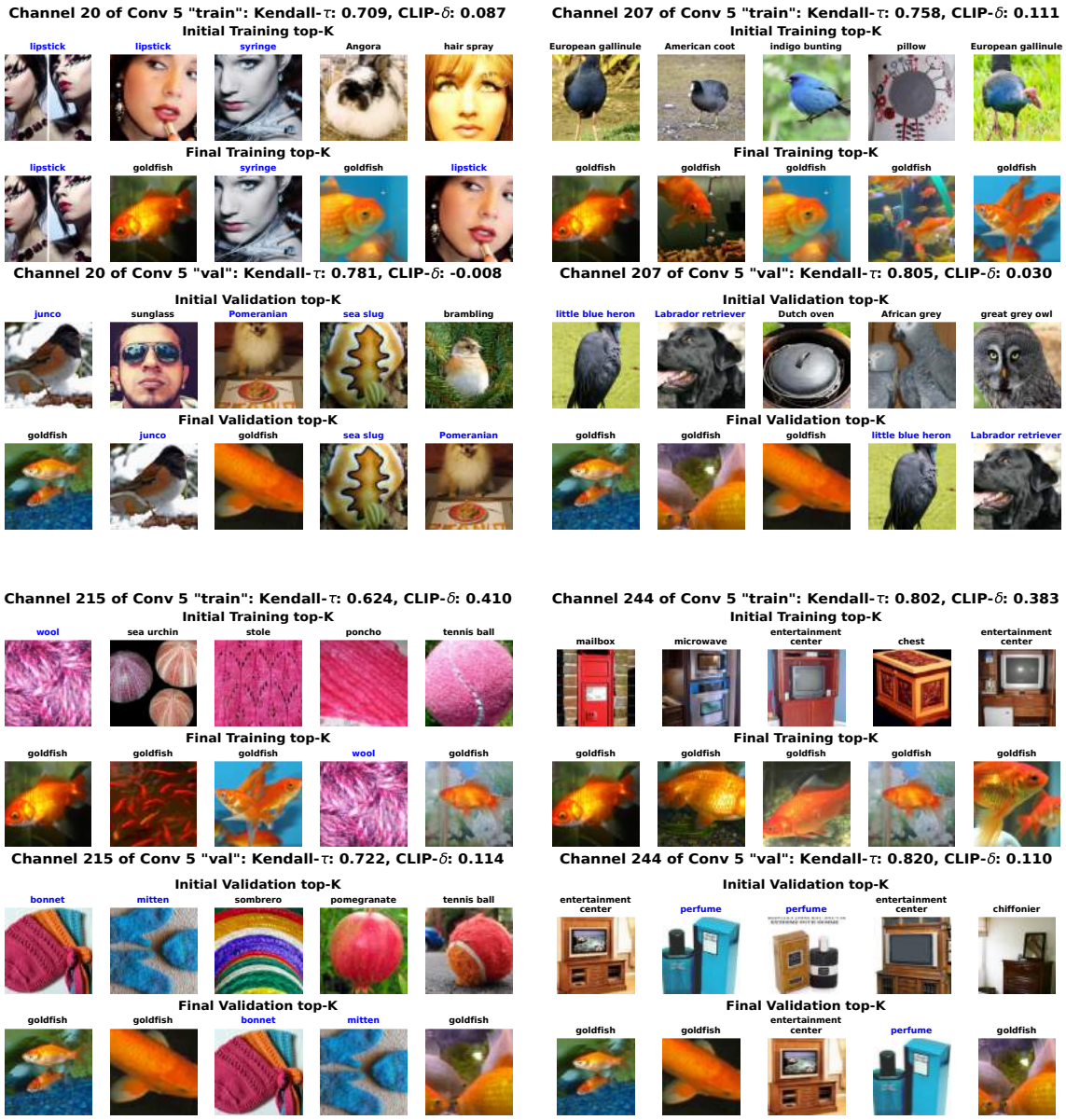


Figure J: Push-up all-channel attack of Conv5 of AlexNet. For each channel, the first two rows are top-k images derived from the training set while the last two are derived from the validation set.

E Ablation Study on EfficientNet

It is important to show that the proposed attack methodology is not limited to AlexNet. To show that the attack can work on newer, more sophisticated neural nets, we have also run an ablation study on EfficientNet Tan and Le (2019). We select the third convolutional block in the Feature 7 layer and perform a push-down attack similar way to AlexNet. The visual results are shown in Appendix A and the metrics for the layer are given in Table 1 in the main text. We observe similar effects to AlexNet; the top images are changed in terms of the exact images and the semantic concepts. We also observe relatively strong CLIP-δ and Kendall-τ changes. Having confirmed the generality of our approach in this way, we leave a survey study over all relevant architectures to future work, computation power permitting.

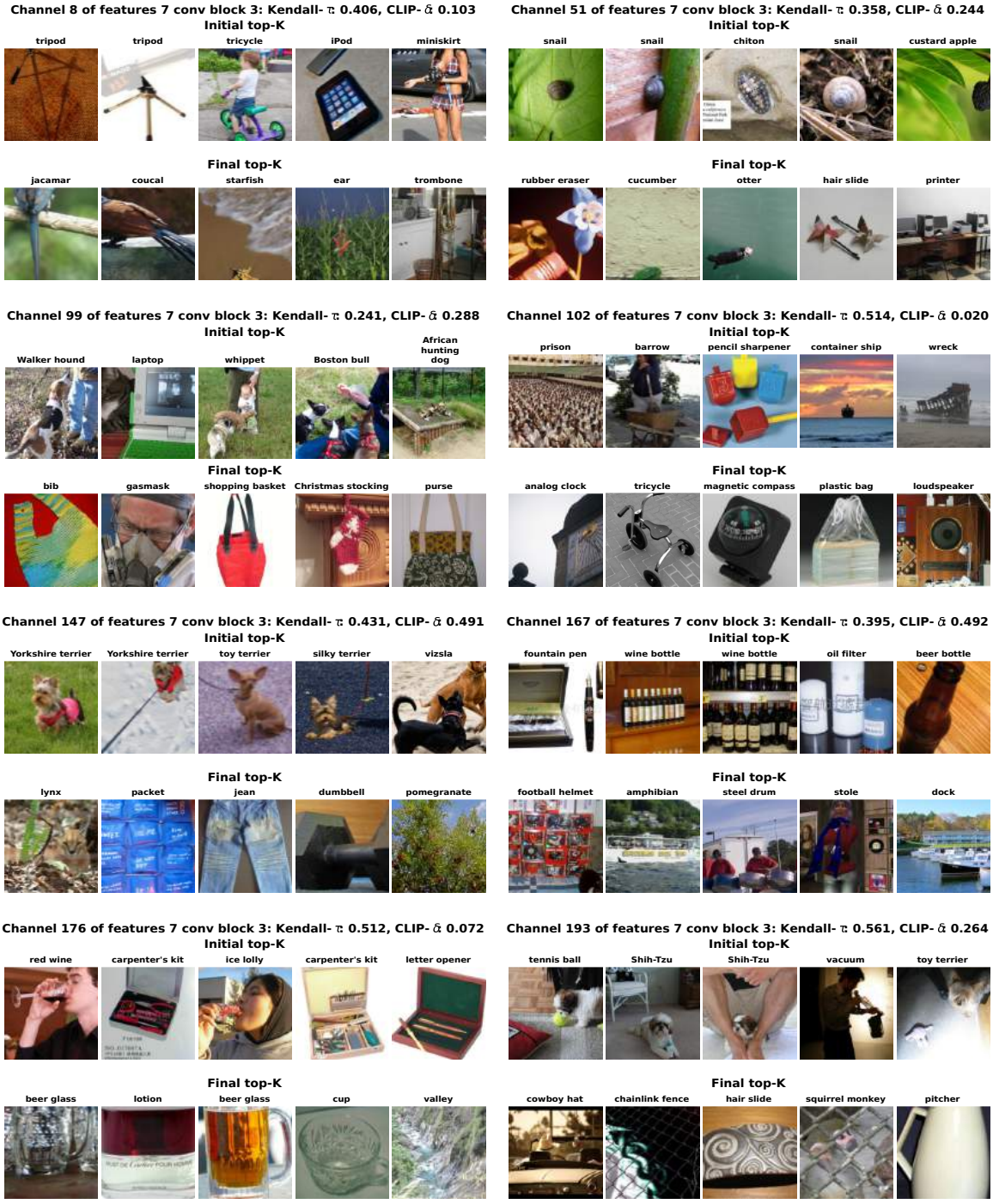


Figure K: Push-down all-channel attack on Feature 7 block 3 of EfficientNet. All initial top-5 images were completely removed from the new set of top-5 images, demonstrating the success of the attack. Channel indices were randomly chosen.

F Non-ConvNet Ablations

We perform additional ablations on ViT and ResNet-50 to demonstrate the generalizability of our attack framework beyond convolutional neural nets.

F.1 ResNet-50

We have performed additional experiments on Resnet-50 for push-up and push-down attacks, for all the channels of the layer layer_4_2_conv_2. We observe no significant loss in accuracy as shown in Table A. Figures L and M show the result for a randomly chosen channel. We observed that the results follow similar trends to those for AlexNet, with higher Kendall-τ values on the push up attack, higher CLIP-δ on the pushdown, and low performance loss overall.

F.2 ViT-B/32

We attack the self-attention encoder layer in layers 0, 6, and 11 of ViT-B/32 to present a cross section of the attack behaviour. Visualizations of the results can be seen in figures L and M. Overall the results follow those of AlexNet with later layers having a larger semantic change as shown by the CLIP-δ scores. Interestingly, we note overall increased CLIP-δ scores, in particular for the first

Layer/Attack	CLIP-δ	Kendall-τ	CLIP-W	K.-τ-W	Accuracy	Δ Acc.
ViT layer 11 Push Up	0.295	0.399	0.813	0.138	75.7%	-0.22%
ViT layer 11 Push Down	0.378	-0.168	0.833	-0.082	75.6%	-0.27%
ViT layer 6 Push Down	0.244	-0.152	0.885	0.122	75.2%	-0.73%
ViT layer 0 Push Down	0.219	-0.139	0.913	0.133	75.4%	-0.55%
ResNet-50 L4.2.conv2 Push Down	0.267	0.319	0.946	0.124	80.2%	-0.01%
ResNet-50 L4.2.conv2 Push Up	0.138	0.784	0.965	0.135	80.2%	-0.01%

Table A: ViT-B/32 and Resnet-50 with Push-up and Push Down Attacks. Each row reports the result obtained after attacking all units of a particular layer. Note that on ViT, the attacks are quite successful, more than those performed on AlexNet, based on increased clip-δ scores and low accuracy loss. We further note that compared to early AlexNet layers, earlier layers of ViT are less resilient to attacks.

layer, whose analogue in AlexNet saw much smaller changes in its feature visualization.

(a) Model: ResNet50 (b) Model: ViT-B/32.

Figure L: All-channel push-up attacks on ResNet50 and ViT. Goldfish images were successfully put in top images.

(a) Network: ResNet50. (b) Network: ViT-B/32.

Figure M: All-channel push-down attacks on Resnet-50 and ViT. Initial top images were successfully replaced.

Attack	Layer	CLIP-δ	Kendall-τ	CLIP-W	K.-τ-W	Accuracy	ΔAcc
All-Layer Push Down	Conv 1	0.025	0.779	0.981	0.295	56.1%	-0.45%
	Conv 2	0.097	0.447	0.993	0.157		
	Conv 3	0.154	0.512	0.969	0.134		
	Conv 4	0.180	0.558	0.953	0.135		
	Conv 5	0.194	0.584	0.969	0.060		
All-Layer Push Up	Conv 1	0.021	0.726	0.981	0.303	56.1%	-0.46%
	Conv 2	0.049	0.420	0.992	0.137		
	Conv 3	0.070	0.307	0.987	0.108		
	Conv 4	0.170	0.272	0.971	0.097		
	Conv 5	0.248	0.541	0.938	0.068		

Table B: Alexnet All-Layer Attacks. Each block of rows (for the push-down and push-up attack) shows the results obtained after attacking all the channels and layers of conv layers in AlexNet. We see that both attacks follow the previously seen trend of later layers being easier to attack. Based on a comparison of these metrics against those found in Table 1, we see that the push-down attack is slightly less effective overall, while the push-up attack is actually more effective.

G All-Layer Attack

We perform additional experiments to attack all the channels of every layer simultaneously. Table B reports the computed metrics. The Push Down All-Channel Attack has results for each of its layers that correspond well to what we saw in each layers' individual attacks in the main paper (Table 1). Overall the CLIP-δ scores are slightly lower, which is not unexpected as this attack demands a shift in the neurons of all layers leaving less room for compensation than a single layer attack. The Push Up Attack however, actually shows better results in this paradigm. We hypothesize that this is due to synergistic effects in pushing up the same set of images across all layers.

H Pruned Models

Here we show extra results and metrics for the pruned models and related attacks. We note that the attacked models lose little accuracy from their fine-tuned baselines, and that there is no evidence of a whack a mole problem in any of the attacks.

Pruned Filters	Attack	CLIP-δ mean	k-τ self sim	CLIP-W	k-τ-W	Init Acc.	Final Acc.	ΔAcc.
32	Push Down	0.275	0.270	0.990	0.027	56.3%	55.8%	-0.44%
64	Push Down	0.258	0.293	0.982	0.021	55.9%	55.4%	-0.46%
96	Push Down	0.233	0.415	0.972	0.021	55.5%	55.0%	-0.41%
128	Push Down	0.226	0.483	0.971	0.002	54.6%	54.2%	-0.44%
160	Push Down	0.235	0.362	0.992	-0.020	53.2%	52.8%	-0.40%
192	Push Down	0.237	0.475	0.991	-0.061	50.9%	50.3%	-0.56%
224	Push Down	0.258	0.570	0.982	-0.118	44.0%	43.5%	-0.47%
32	Push Up	0.087	0.693	0.976	0.027	56.3%	56.1%	-0.18%
64	Push Up	0.163	0.667	0.946	0.002	55.9%	55.8%	-0.12%
96	Push Up	0.179	0.675	0.934	0.003	55.5%	55.3%	-0.15%
128	Push Up	0.202	0.691	0.942	-0.045	54.6%	54.4%	-0.17%
160	Push Up	0.195	0.716	0.934	-0.020	53.2%	53.1%	-0.07%
192	Push Up	0.185	0.737	0.947	-0.102	50.9%	50.7%	-0.12%
224	Push Up	0.221	0.756	0.927	-0.154	44.0%	44.0%	0.03%

Table C: Vulnerability pruning attack results. Note the change in accuracy remains small, and that the whack a mole metrics ≈ 1 indicate that the problem is not occurring. The CLIP-δ scores indicate that the attacks' effectiveness are not strongly affected by the pruning amount.

Prune Rate	Attack	CLIP-δ mean	k-τ self sim	CLIP-W	k-τ-W	Init Acc.	Final Acc.	ΔAcc.
12.50%	Push Down	0.309	0.252	0.977	0.039	56.7%	56.2%	-0.50%
25.00%	Push Down	0.299	0.311	0.975	0.044	56.7%	56.2%	-0.46%
37.50%	Push Down	0.300	0.320	0.980	0.043	56.7%	56.2%	-0.44%
50.00%	Push Down	0.298	0.332	0.977	0.041	56.7%	56.2%	-0.49%
62.50%	Push Down	0.301	0.313	0.975	0.042	56.5%	56.1%	-0.37%
75.00%	Push Down	0.298	0.319	0.979	0.042	56.1%	55.6%	-0.53%
87.50%	Push Down	0.289	0.396	0.976	0.053	54.6%	54.1%	-0.46%
12.50%	Push Up	0.173	0.646	0.944	0.015	56.7%	56.6%	-0.15%
25.00%	Push Up	0.110	0.662	0.970	0.033	56.7%	56.5%	-0.16%
37.50%	Push Up	0.156	0.639	0.954	0.027	56.7%	56.4%	-0.28%
50.00%	Push Up	0.200	0.600	0.929	0.008	56.7%	56.5%	-0.22%
62.50%	Push Up	0.221	0.673	0.918	0.038	56.5%	56.4%	-0.06%
75.00%	Push Up	0.268	0.541	0.885	-0.006	56.1%	55.9%	-0.22%
87.50%	Push Up	0.183	0.512	0.944	0.036	54.6%	54.2%	-0.37%

Table D: Norm pruning attack results. Note the change in accuracy remains small, and that the whack a mole metrics ≈ 1 indicate that the problem is not occurring. The CLIP-δ scores, particularly for the Push Down attack indicate that the attacks' effectiveness is not strongly affected by this pruning method.

Target Sparsity	Attack	CLIP-δ mean	k-τ self sim	CLIP-W	k-τ-W	Init Acc.	Final Acc.	ΔAcc.
25.0%	Push Down	0.266	0.514	0.890	0.280	56.8%	56.4%	-0.36%
37.5%	Push Down	0.260	0.512	0.899	0.276	56.2%	55.9%	-0.38%
50.0%	Push Down	0.261	0.507	0.865	0.280	55.6%	55.3%	-0.23%
62.5%	Push Down	0.283	0.515	0.886	0.293	53.2%	52.9%	-0.33%
75.0%	Push Down	0.277	0.534	0.851	0.323	45.4%	45.1%	-0.28%
87.5%	Push Down	0.304	0.320	0.822	0.283	14.7%	14.5%	-0.17%
12.5%	Push Up	0.216	0.498	0.910	0.290	57.1%	56.9%	-0.18%
25.0%	Push Up	0.239	0.490	0.881	0.285	56.8%	56.6%	-0.22%
37.5%	Push Up	0.241	0.497	0.864	0.288	56.2%	56.1%	-0.12%
50.0%	Push Up	0.232	0.464	0.861	0.271	55.6%	55.4%	-0.15%
62.5%	Push Up	0.262	0.470	0.841	0.273	53.2%	53.0%	-0.18%
75.0%	Push Up	0.267	0.396	0.826	0.259	45.4%	45.2%	-0.24%
87.5%	Push Up	0.284	0.298	0.797	0.247	14.7%	14.5%	-0.11%

Table E: OTO weight pruning attack results. Note that whlie the loss in performance from the attack itself is small, the higher pruning rates have significant drops in performace. The whack a mole metrics ≈ 1 indicate that the problem is not occurring. The CLIP-δ scores, particularly for the Push Down attack indicate that the attacks' effectiveness is not strongly affected by this pruning method. Indeed, the Push Up attack is ultimately more effective at higher pruning rates, a highly counter intuitive result.

Target Sparsity	Attack	CLIP-δ mean	k-τ self sim	CLIP-W	k-τ-W	Init Acc.	Final Acc.	ΔAcc.
25.0%	Push Down	0.290	0.464	0.847	0.267	53.2%	52.8%	-0.42%
37.5%	Push Down	0.217	0.478	0.912	0.288	52.1%	51.7%	-0.38%
50.0%	Push Down	0.264	0.399	0.867	0.259	50.5%	49.9%	-0.60%
62.5%	Push Down	0.237	0.466	0.890	0.281	46.2%	46.0%	-0.28%
75.0%	Push Down	0.241	0.412	0.867	0.301	35.7%	35.3%	-0.43%
87.5%	Push Down	0.158	-0.027	0.955	0.251	7.9%	6.7%	-1.17%
12.5%	Push Up	0.243	0.496	0.845	0.288	56.3%	55.9%	-0.37%
25.0%	Push Up	0.257	0.407	0.874	0.266	53.2%	52.9%	-0.28%
37.5%	Push Up	0.262	0.410	0.880	0.258	52.1%	51.8%	-0.29%
50.0%	Push Up	0.260	0.433	0.872	0.263	50.5%	50.1%	-0.33%
62.5%	Push Up	0.253	0.374	0.897	0.247	46.2%	46.0%	-0.25%
75.0%	Push Up	0.217	0.337	0.920	0.237	35.7%	35.4%	-0.37%
87.5%	Push Up	0.052	0.237	0.998	0.264	7.9%	7.0%	-0.84%

Table F: OTO Filter pruning attack results. Note that while the loss in performance from the attack itself is small, the higher pruning rates have significant drops in performance. The whack a mole metrics 1 indicate that this problem is not present. The CLIP-δ scores indicate that this pruning method does not provide meaningful protection against our attacks.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), Advances in neural information processing systems (Vol. 31). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf

Akhtar, N., & Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access, 6, 14410-14430. doi: 10.1109/ACCESS.2018.2807385

Binder, A., Bach, S., Montavon, G., Müller, K.-R., & Samek, W. (2016). Layer-wise relevance propagation for deep neural network architectures. In K. J. Kim & N. Joukov (Eds.), Information science and applications (icisa) 2016 (pp. 913–922). Singapore: Springer Singapore.

Blalock, D., Ortiz, J. J. G., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning?

Cammarata, N., Goh, G., Carter, S., Schubert, L., Petrov, M., & Olah, C. (2020). Curve detectors. Distill, 5(6), e00024–003.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey. CoRR, abs/1810.00069. Retrieved from http://arxiv.org/abs/1810.00069

Chen, T., Ji, B., Tianyu, D., Fang, B., Wang, G., Zhu, Z., ... Tu, X. (2021). Only train once: A one-shot neural network training and pruning framework. In Thirty-fifth conference on neural information processing systems.

Chen, T., Liang, L., Tianyu, D., Zhu, Z., & Zharkov, I. (2023). Otov2: Automatic, generic, user-friendly. In International conference on learning representations.

Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2020). A survey of model compression and acceleration for deep neural networks.

Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S., & Garriga-Alonso, A. (2023). Towards automated circuit discovery for mechanistic interpretability.

Davari, M., Horoi, S., Natik, A., Lajoie, G., Wolf, G., & Belilovsky, E. (2022). Reliability of cka as a similarity measure in deep learning.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 ieee conference on computer vision and pattern recognition (p. 248-255). doi: 10.1109/CVPR.2009.5206848

Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., & Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. Advances in neural information processing systems, 32.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... others (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In International conference on learning representations.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. University of Montreal, 1341(3), 1.

Fulleringer, A., Nanfack, G., Marty, J., Eickenberg, M., & Belilovsky, E. (2023). Adversarial attacks on neuron interpretation via activation maximization. In Neurips workshop on attributing model behavior at scale. Retrieved from https://openreview.net/forum?id=ie6hRGMCp2

Gildenblat, J., & contributors. (2021). Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam. GitHub.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial networks.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., & McDaniel, P. (2017). On the (statistical) detection of adversarial examples.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), Advances in neural information processing systems (Vol. 28). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the ieee conference on computer vision and pattern recognition (pp. 770–778).

Heo, J., Joo, S., & Moon, T. (2019). Fooling neural network interpretations via adversarial model manipulation. Advances in Neural Information Processing Systems, 32.

Hernandez, E., Schwettmann, S., Bau, D., Bagashvili, T., Torralba, A., & Andreas, J. (2022). Natural language descriptions of deep visual features. In International conference on learning representations.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. Retrieved from http://arxiv.org/abs/1503.02531 (cite arxiv:1503.02531Comment: NIPS 2014 Deep Learning Workshop)

Hundt, A. T., Agnew, W., Zeng, V., Kacianka, S., & Gombolay, M. C. (2022). Robots enact malignant stereotypes. Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency. Retrieved from https://api.semanticscholar.org/CorpusID:249872606

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... Amodei, D. (2020). Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

Konate, S., Lebrat, L., Cruz, R. S., Smith, E., Bradley, A., Fookes, C., & Salvado, O. (2021). A comparison of saliency methods for deep learning explainability. In 2021 digital image computing: Techniques and applications (dicta) (p. 01-08). doi: 10.1109/DICTA52665.2021.9647419

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90.

Lebedev, V., & Lempitsky, V. (2015). Fast convnets using group-wise brain damage.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient convnets.

Li, X., & Li, F. (2017). Adversarial examples detection in deep networks with convolutional filter statistics.

Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. In Proceedings of the ieee conference on computer vision and pattern recognition (pp. 5188–5196).

Miller, D. J., Xiang, Z., & Kesidis, G. (2020). Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. Proceedings of the IEEE, 108(3), 402-433. doi: 10.1109/JPROC.2020.2970615

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). Pruning convolutional neural networks for resource efficient inference.

Nanda, N., Chan, L., Liberum, T., Smith, J., & Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. arXiv preprint arXiv:2301.05217.

Nanfack, G., Fulleringer, A., Marty, J., Eickenberg, M., & Belilovsky, E. (2024). Adversarial attacks on the interpretation of neuron activation maximization. In Proceedings of the aaai conference on artificial intelligence.

Nayebi, A., & Ganguli, S. (2017). Biologically inspired protection of deep networks from adversarial attacks.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), Advances in neural information processing systems (Vol. 29). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/

2016/file/5d79099fcdf499f12b79770834c0164a-Paper.pdf

Oikarinen, T., & Weng, T.-W. (2022). Clip-dissect: Automatic description of neuron representations in deep vision networks. arXiv preprint arXiv:2204.10965.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., & Carter, S. (2020). Zoom in: An introduction to circuits. Distill. (https://distill.pub/2020/circuits/zoom-in) doi: 10.23915/distill.00024.001

Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature visualization. Distill. (https://distill.pub/2017/feature-visualization) doi: 10.23915/distill.00007

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.

Qin, Z., Yu, F., Liu, C., & Chen, X. (2018). How convolutional neural network see the world - A survey of convolutional neural network visualization methods. CoRR, abs/1804.11191. Retrieved from http://arxiv.org/abs/1804.11191

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... others (2021). Learning transferable visual models from natural language supervision. In International conference on machine learning (pp. 8748–8763).

Räukur, T., Ho, A., Casper, S., & Hadfield-Menell, D. (2022). Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. arXiv e-prints, arXiv–2207.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.

Räuker, T., Ho, A., Casper, S., & Hadfield-Menell, D. (2023). Toward transparent ai: A survey on interpreting the inner structures of deep neural networks.

See, A., Luong, M.-T., & Manning, C. D. (2016). Compression of neural machine translation models via pruning.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of

the ieee international conference on computer vision (pp. 618–626).

Shahroudnejad, A. (2021). A survey on understanding, visualizations, and explanation of deep neural networks. CoRR, abs/2102.01792. Retrieved from https://arxiv.org/abs/2102.01792

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR, abs/1312.6034. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1312.html#SimonyanVZ13

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In Proceedings of the aaai/acm conference on ai, ethics, and society (pp. 180–186).

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105–6114).

Tian, J., Xie, H., Hu, S., & Liu, J. (2021). Multidimensional face representation in a deep convolutional neural network reveals the mechanism underlying ai racism. Frontiers in Computational Neuroscience, 15. Retrieved from https://www.frontiersin.org/articles/10.3389/fncom.2021.620281 doi: 10.3389/fncom.2021.620281

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., & Steinhardt, J. (2022). Interpretability in the wild: a circuit for indirect object identification in gpt-2 small.

Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818–833).

Zheng, S., Song, Y., Leung, T., & Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training.

Zimmermann, R. S., Borowski, J., Geirhos, R., Bethge, M., Wallis, T., & Brendel, W. (2021). How well do feature visualizations support causal understanding of cnn activations? Advances in

Neural Information Processing Systems, 34, 11730–11744.