Model Merging and Feature Visualization in Deep Neural Networks

Congshu Zou

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

June 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Congshu Zou**

Entitled: **Model Merging and Feature Visualization in Deep 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. Mirco Ravanelli

_____ Examiner
Dr. Sudhir Mudur

_____ Supervisor
Dr. Eugene Belilovsky

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

_____ 2025

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Model Merging and Feature Visualization in Deep Neural Networks

Congshu Zou

Linear mode connectivity (LMC) has recently become a topic of great interest. It has been empirically demonstrated that popular deep learning models trained from different initializations exhibit linear model connectivity up to permutation. Based on this, several approaches for finding a permutation of the model's features or weights have been proposed leading to several popular methods for model merging. These methods enable the simple averaging of two models to create a new high-performance model. However, besides accuracy, the properties of these models and their relationships to the representations of the models they derive from are poorly understood. In this work, we study the inner working mechanisms behind LMC in model merging through the lens of classic feature visualization methods. Focusing on convolutional neural networks (CNNs) we make several observations that shed light on the underlying mechanisms of model merging by permute and average.

Acknowledgments

I would like to express my sincere gratitude to all those who supported me throughout the completion of this degree and thesis. I am especially thankful to my supervisor, Prof. Eugene Belilovsky, for his unwavering guidance, insightful feedback, and continuous support. I am also deeply grateful to our lab's postdoctoral researcher, Dr. Geraldin Nanfack, for his writing advice and creative problem-solving. Special thanks to Stefan Horoi and Albert Camacho for their valuable contributions to my research and for the many thoughtful discussions. I would also like to thank Concordia University for fostering an enriching academic environment and offering courses that have been instrumental in my growth as a researcher and computer scientist. Finally, I extend my heartfelt appreciation to my wife and daughter for their constant encouragement and support, especially during the most challenging moments of this journey.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Deep neural networks have achieved remarkable advancements in recent years, demonstrating their capability to perform various valuable tasks. Understanding the inner mechanism of a neural network becomes critical, which leads to the exploration of the loss landscape. Frankle, Dziugaite, Roy, and Carbin (2020) suggest linear mode connectivity, where two neural networks, randomly initialized and trained using stochastic gradient descent (SGD), are observed to be connected by a linear path in the parameter space with no or little loss barrier. The study of LMC can offer valuable insights into the complex structure of loss landscapes and training dynamics. It will also help various deep learning applications such as model merging, transfer learning, etc. However, naive LMC often fails due to the permutation invariance of neural networks (Entezari, Sedghi, Saukh, & Neyshabur, 2022). Thus several algorithms are designed to linearly connect two trained models by first aligning one to another, and second by finding a permutation (Ainsworth, Hayase, & Srinivasa, 2023; Jordan, Sedghi, Saukh, Entezari, & Neyshabur, 2023). This has been shown to lead to linear mode connectivity and models that can be merged (or averaged).

However, why the LMC works with permutation models and what properties this induces on the average model is challenging to understand. In this work, we propose to study this through the lens of feature visualization. By maximizing the activation of certain neurons in a network, we can visually understand what features the model is detecting. If two models can be linearly interpolated,

then they should activate similar features. In addition to visually inspecting the extracted features, we use some metrics to characterize the similarity of images extracted from two models Nanfack, Fulleringer, Marty, Eickenberg, and Belilovsky (2023).

In this study, we train multiple randomly initialized models with SGD optimization, then apply the REPAIR Jordan et al. (2023) method to permute the model weights space and visualize the top 10 images that most activate each neuron. We find that models after permutation share some similar top-activated features, but not all of them. We shed light on the features that don't match and highlight that they correspond to rare features in the models being aligned against. Finally, we show that despite the limitation of one-to-one permutation matching, features in a given model can be approximated by a sparse subset of features in another model, motivating recently proposed methods in model merging.

1.2 Contributions

- We visualize the top images that maximize the activation of each convolutional layer in the model, both before and after alignment and merging.
- We analyze the features learned by the models by comparing their feature visualizations, allowing us to assess the similarity in their internal representations.
- We explore the linear combinations of features from one model to approximate features of another model, providing insights of inner connection of the channels
- We submitted to and were accepted by the NeurIPS 2024 UniReps workshop under a paper titled Understanding Permutation Based Model Merging with Feature Visualizations, cited as Zou, Nanfack, Horoi, and Belilovsky (2024). This paper forms the basis for Chapter 3 in this work.

Chapter 2

Background

2.1 Linear Mode Connectivity and Model Merging

2.1.1 Mode Connectivity

Deep neural networks are highly overparameterized, and the loss functions are non-convex. Traditionally it is believed that training from different initialized parameters leads to isolated local minima. However recent research shows that many independently trained models can be connected through a low-loss path. First proposed by Garipov, Izmailov, Podoprikhin, Vetrov, and Wilson (2018), mode connectivity investigates the relationship between two optimal solutions found after training. It demonstrates that the optimal solutions for deep neural networks can be connected by very simple curves.

Given two trained networks with parameters θ_A and θ_B, the goal is to find a parametric curve, $\phi_\theta(t)$, with $t \in [0, 1]$ and parameters θ, where $\phi_\theta(0) = \theta_A$ and $\phi_\theta(1) = \theta_B$, such that, $L(\phi_\theta)$ remains low along the curve. It is shown that one can find a simple curve such as polygonal chain or quadratic Bezier curve. Draxler, Veschgini, Salmhofer, and Hamprecht (2019) suggest that these minima are points sitting on a single connected manifold of low loss, rather than as the bottoms of distinct valleys. Two minima can be connected with segments of linear lines, where the loss remains low.

2.1.2 Linear Mode Connectivity

One special case of mode connectivity is linear mode connectivity. Instead of Frankle et al. (2020) propose that there exists a linear path between two optima, where the loss barrier is low along the path. Mathematically, two trained neural network with parameters θ_A and θ_B, the linear interpolation between the two weight space:

$$\theta = (1 - \alpha)\theta_A + \alpha\theta_B \quad \alpha \in [0, 1] \tag{1}$$

If the loss along this path remains low, we say that models are linearly mode connected (LMC).

However, in may cases, the loss is high along the path of direct linear interpolation. Entezari et al. (2022) propose that if we take permutation invariance into account, SGD solutions of neural networks will likely have no loss barrier in the linear interpolation between them. We can permute weights of a neural network, without affecting the model performance. This is called permutation invariance. Entezari et al. (2022) show that after permuting one model to align with another, we can often achieve low loss barrier along the linear interpolated path.

2.1.3 Model Merging

Model merging is a technique to combine two or more trained neural networks into a single model, aiming to retain or even enhance the performance of the original models. This approach is particularly relevant in scenarios involving model ensembling, continual learning, federated learning, or collaborative model development. The key challenge is to merge models too much extra, while preserving their functional characteristics and avoiding destructive interference.

Several methods have been proposed to perform model merging effectively:

Model Soups, proposed by Wortsman et al. (2022), average the weights of several fine-tuned models, based on validation performance. Three recipes are proposed for model souping, the uniform, greedy, and learned soup. Uniform soup is the straightforward approach, where all weights are averaged equally. Greedy soup iteratively add weights which improve the validation performance. The third approach is learned soup, involving learning the soup mixing coefficients for each of the ingredients on the held-out validation set. These simple interpolation in the weight space

has been shown to enhance model generalization and overall performance comparable to traditional ensemble methods, but without additional computational overhead during inference.

Git Re-Basin, proposed by Ainsworth et al. (2023), suggests that neural network loss landscapes often contain a single basin if taking all possible permutation of hidden units into account. This implies that models trained from different random initializations can learn functionally similar solutions with different weight configurations. In order to align the weight space of two models, the authors introduce three algorithms to permute the weights of one model to match the other (Ainsworth et al., 2023). This allows for merging the two models, resulting in a functionally equivalent model within the same basin. Zero-barrier linear mode connectivity is achieved between independently trained ResNet models on CIFAR-10 dataset (Krizhevsky, 2009). It further demonstrates that linear mode connectivity is an emergent property of training procedures, not of model architectures.

Jordan et al. (2023) further investigate the permutation invariance conjecture proposed by Entezari et al. (2022), and propose a phenomenon called variance collapse: the poor performance of the merged models is caused by the collapse in the variance of the activations. The authors proposes **REPAIR**, a method to renormalize neuron activations after interpolation (Jordan et al., 2023). The algorithm mitigate the variance collapse by rescaling the pre-activations of the interpolated networks. Experiments show reduction of loss barrier across a wide variety of architecture families and tasks.

CCA Merging is a model merging algorithm based on Canonical Correlation Analysis. Instead of one to one mapping of the weights space in previous permutation based algorithms, CCA merging aims to maximize the correlations between linear combinations of the model features (Horoi, Camacho, Belilovsky, & Wolf, 2024). It also show promising results in merging many models.

While most merging methods only consider models trained on the same tasks, Stoica et al. (2024) propose a novel algorithm to merge models trained on distinct tasks without requiring access to data or additional training. **ZipIt!** combines weights of multiple models into a single model which can perform all tasks of each individual model. They zip within one model as well as across two models, preserving features that aren't shared between models. Furthermore, the authors also suggest partial merging up to certain layers to create a multi-head model (Stoica et al., 2024).

2.2 Feature visualization

With the rapid progress in deep learning, the interpretability of models is becoming increasingly important. Feature visualization is one of the interpretability technique, aiming to provide insights of a model's internal behavior by visualizing the features it has learned.

First proposed by Erhan, Bengio, Courville, and Vincent (2009), **activation maximization** is one classic approach. The idea to find a set of input images to maximally activate certain neurons or layers in the network. Let θ be the parameters of a neural network f_θ, where $f_\theta(x)$ is the activation. The goal of activation maximization is to find the input x^* that:

$$x^* = \arg\max_x f_\theta(x)$$

In the case that $x \in \mathcal{D}$, where $\mathcal{D}$ is training or test dataset, then x^* consists top-k images in the dataset that activate the corresponding networks most (Yosinski, Clune, Nguyen, Fuchs, & Lipson, 2015). Alternatively, if x can be optimized as synthetic images. Olah, Mordvintsev, and Schubert (2017) proposed such a gradient based approach, to find synthesized images by solving a optimization problem and finding an input x which maximize the activation through backpropagation.

To improve the quality of these synthesized images and reduce noisy patterns, various regularization techniques are typically employed. These include adding an L2 norm penalty to encourage smoothness, total variation regularization to reduce high-frequency noise, and techniques such as Gaussian blur and jittering during optimization. These strategies help produce more interpretable and natural-looking visualizations that reflect meaningful features learned by the network (Mahendran & Vedaldi, 2014; Olah et al., 2017; Simonyan, Vedaldi, & Zisserman, 2014; Yosinski et al., 2015)

Another method is to generate synthesized images using a Generative Adversarial Network (GANs) (Nguyen, Dosovitskiy, Yosinski, Brox, & Clune, 2016).GAN is a type of generative model, composed with a generator network G and a discriminator network D (Goodfellow et al., 2014). The generator is trained with real images dataset to produce synthetic images which the discriminator can not distinguish from real ones. In the context of feature visualization, GANs can be used to generate more realistic images, reducing the high-frequency visualizations which often produced

by standard gradient based methods (Nguyen et al., 2016).

In addition to interpretability, feature visualization also plays a role in model debugging, detecting biases, and enhancing robustness.

2.3 Mechanistic Interpretability

Mechanistic interpretability is a bottom-up method that analyzes models at a detailed level (features, neurons, layers, and connections) to uncover the causal relationship and computations transforming from inputs to outputs. It focuses on identifying which parts of a network correspond to which features (Bereska & Gavves, 2024).

One main approach of mechanistic interpretability is to uncover a circuit within the model. Circuits are subsets of neurons consisting of linked features and the weights connecting them (Bereska & Gavves, 2024). Rather than analyzing individual neurons in isolation, this approach focuses on identifying how neurons compose and interact across layers to form structured pathways that contribute to specific model behaviors. Olah et al. (2020) suggest that circuits are the primary building blocks of neural networks, acting as logic gates on visual features. By identifying these circuits, researchers aim to reverse-engineer internal algorithms that the network has learned, for example, the composition of low-level feature detectors into higher-level object recognition units in convolutional networks (Olah et al., 2020). This allows for causal attributions, revealing not just which components are active, but which are necessary for the behavior to emerge, thus offering a deeper and more mechanistic explanation of model functionality (Nanda, Chan, Lieberum, Smith, & Steinhardt, 2023; Wang, Variengien, Conmy, Shlegeris, & Steinhardt, 2022).

As discussed in Olah et al. (2020) and expanded upon in (Elhage et al., 2021), neural networks often exhibit superposition, where a single neuron encodes a linear combination of multiple, distinct features, potentially unrelated. This phenomenon gives rise to polysemantic neurons, which can make interpretability difficult by entangling different concepts into the same unit. One way to disentangle this is via sparse dictionary learning, using sparse autoencoders (SAEs), which are neural networks trained to reconstruct inputs through a sparse latent space, where only a few units are active at a time (Olshausen & Field, 1997). In mechanistic interpretability, SAEs have been used

to recover monosemantic feature directions from superpositions. Recent works show promising results of SAEs extracting interpretable monosemantic features from polysemanticity (Bricken et al., 2023; Cunningham, Ewart, Riggs, Huben, & Sharkey, 2023; Gao et al., 2024; Gorton, 2024).

2.4 Adversarial Attacks

Adversarial attacks refer to broad techniques which maliciously trick neural networks to produce incorrect outputs, e.g., misclassifications in vision models (Eykholt et al., 2018; Goodfellow, Shlens, & Szegedy, 2015).

Based on the attacker's level of knowledge about the target neural network, adversarial attacks are typically categorized into three types: white-box, black-box, and gray-box attacks. In a white-box attack, the attackers have full access to the model's architecture, parameters, and the dataset used for training. This allows precise computation of gradients, enabling highly effective perturbations. In contrast, a black-box attacker assumes no knowledge of the model's structure or training data. The attacker can only observe inputs and outputs, often relying on techniques like query-based optimization or transferability from surrogate models. Gray-box attacks represent an intermediate case, where the attacker has partial information, such as the model architecture or training dataset, but lacks access to the model's exact parameters or gradients (Chakraborty, Alam, Dey, Chattopadhyay, & Mukhopadhyay, 2021; Long, Gao, Xu, & Zhou, 2022).

Chakraborty et al. (2021) further categorize adversarial attacks based on the attack surface, identifying three primary scenarios: evasion attacks, poisoning attacks, and exploratory attacks.

- **Evasion attacks** occur at inference time, where attackers craft malicious inputs to bypass model detection or trick misclassification by the model.
- **Poisoning attack** target the training phase by injecting manipulated data or labels into the training set, thereby compromising the model's learning process and degrading performance.
- **Exploratory attacks** aim to extract as much information as possible about the model, particularly in black-box settings, without directly altering its operation.

In addition to traditional adversarial attacks that target the input or output levels of neural networks, Nanfack et al. (2023) introduce a novel adversarial framework that operates on the model's intermediate feature representations. Specifically, their method perturbs the internal activations of the network to manipulate feature visualizations. Notably, these manipulations can significantly alter the results of feature visualization, with minimum effect on the model performance. This form of attack highlights a critical vulnerability of interpretability tools: they can be deceived, raising concerns about their reliability in understanding neural network behavior, especially in high-stakes applications (Nanfack et al., 2023).

2.5 CLIP Model

CLIP (Contrastive Language–Image Pre-training) is a neural network trained by OpenAI on a variety of (image, text) pairs. The model consists of two encoders, one for text and one for images, projecting both text and image inputs into a shared multi-modal embedding space (Radford et al., 2021). The image encoder can be either a convolutional neural network or a vision transformer. The text encoder is a Transformer-based language model.

CLIP is trained with a contrastive dual-image-text objective that maximizes the cosine similarity of the image and text embeddings of real pairs in the batch while minimizing the cosine similarity of the embeddings of the incorrect pairings (Radford et al., 2021). The goal is to predict which caption goes with which image among a batch, rather than predicting the actual caption words in previous works (Desai & Johnson, 2021; Sariyildiz, Perez, & Larlus, 2020). After training, CLIP encoders produce aligned image and text embeddings. The two encoders are independent by design, making CLIP a dual encoder system.

CLIP model enables powerful applications like zero-shot image classification, where images are matched to text prompts without labeled training data, achieving strong performance on benchmarks like ImageNet (Radford et al., 2021). CLIP also plays an important role in modern generative models, guiding or ranking generated images by how well they align with text prompts (Ramesh, Dhariwal, Nichol, Chu, & Chen, 2022). Additionally, CLIP can be used to compare and cluster images by calculating similarity of images embeddings (Nanfack et al., 2023).

Chapter 3

Feature Visualization for Model Merging

Understanding why permutation-based model merging works is a fundamental open question. Previous researches has shown that direct linear interpolating neural networks often fails unless the models are aligned through neuron permutations (Ainsworth et al., 2023; Jordan et al., 2023). While theoretical analyses and performance metrics demonstrate promising results, they provide little insight into the internal representations of merged models. In this study, we investigate the mechanisms behind model merging through the lens of feature visualization. By analyzing top-activated features before and after alignment as well as merging, we aim to uncover how aligning and merging affects the internal behavior of neural networks.

Feature visualization techniques allow us to probe what each convolutional channel in a model responds to, using top-activated input images as a proxy for learned representations. We apply these techniques to analysis models across several scenarios: independently trained models, models aligned via permutation, direct averaged model, permuted-and-merged model, and regression-aligned models. Through both qualitative (visual) and quantitative (metric-based) comparison, we reveal patterns of both successful and unsuccessful alignment. Our observations suggest that some features are universally learned across models, while others are distinct or require more flexible matching than one-to-one permutations can offer. These insights help explain both the strengths and limitations of current model merging methods and motivate more general approaches for future work.

Hypothesis We hypothesize that certain features are consistently and easily learned across different models, while some others are rare or unstable, leading to mismatches in feature visualizations after permutation. These hard-to-learn features contribute to dissimilarities in visualizations and limit the effectiveness of one-to-one merging methods.

3.1 Method

3.1.1 Notions and Background

Let f_θ denote a neural network with parameters θ. For an input image x, the activations at layer l is denoted $d_\theta^{(l)}(x)$. In this study, we consider multiple independently trained instances of a given architecture, denoted $f_{\theta_1}, f_{\theta_2}, \cdots, f_{\theta_n}$, and analyze them through feature visualization techniques.

3.1.2 Model Merging

We employ the REPAIR algorithm (Jordan et al., 2023) to align two independently trained networks by identifying a neuron permutation P such that the permuted model $f_{P(\theta_2)}$ is more closely aligned to f_{θ_1} in terms of activations. After alignment, models are merged by linearly interpolating their weights:

$$\theta_{\text{merge}} = \alpha \cdot \theta_1 + (1 - \alpha) \cdot P(\theta_2), \quad \alpha \in [0, 1]$$

This interpolation is performed layer-wise.

In addition, we study feature alignment via linear regression. For each channel in a given layer of f_{θ_1} , we learn a linear combination of features from the corresponding layer in f_{θ_2} to approximate the same activation behavior. The regression model is trained using activation data extracted from a random subset of the ImageNet training set (Deng et al., 2009).

3.1.3 Feature Visualization

To understand the learned features of each model, we use dataset-based activation maximization. For a given channel c in layer l of a neural network f_θ, we extract the input images that maximize the activation $f_\theta^{l,c}(x)$. Specifically, the goal is to find inputs $x^* \in \mathcal{D}$ where $\mathcal{D}$ is the training dataset

that satisfy:

$$x^* = \arg\max_{x \in \mathcal{D}} f_\theta^{l,c}(x)$$

In practice, we identify the top-10 images from the ImageNet dataset (Deng et al., 2009) that produce the highest activations for each convolutional channel. This is done for the original models, the permuted models, the merged models, and the regression-aligned models.

These top-activated images provide qualitative insight into the types of features each channel responds to. We perform visual inspection of the extracted images to assess feature similarity and track how alignment and merging affect the visualizations.

3.1.4 Similarity Metrics

To quantitatively evaluate the similarity between channels across different models, we adapt the following metrics from Nanfack et al. (2023):

Kendall-τ: We take a subset of images $\mathcal{D}' \subseteq \mathcal{D}$, and compute the rankings $R_{i,j,c}$ of images w.r.t the activations in model i layer l channel c. The Kendall-$\tau_{j,c}$ score is the Kendall rank correlation coefficient between $R_{i_1,j,c}$ and $R_{i_2,j,c}$ of two models. Higher values of Kendall-τ scores can be interpreted as higher similarity in the ordering of image activations between corresponding channels in two models.

CLIP-δ: We use the cosine similarity between CLIP embeddings of top 10 activated images, providing a semantic similarity measure that incorporates high-level features. Given two models aligned after permutation, we compute the cosine self-similarity of top 10 activated images from layer l channel c, of one model $C_{1jc,1jc}$, as well as cosine similarity between images from both models, $C_{1jc,2jc}$. Then,

$$\text{CLIP} - \delta_{j,c} = (C_{1jc,1jc} - C_{1jc,2jc}) \,/\, \left(\frac{1}{N-1} \sum_{p=1}^{N} C_{1jc,1(p \neq j)c} \right)$$

Intuitively, this quantifies the relative semantic difference of top-k images w.r.t. CLIP embeddings and a lower score can be interpreted as the fact that the channel j from two models have semantically

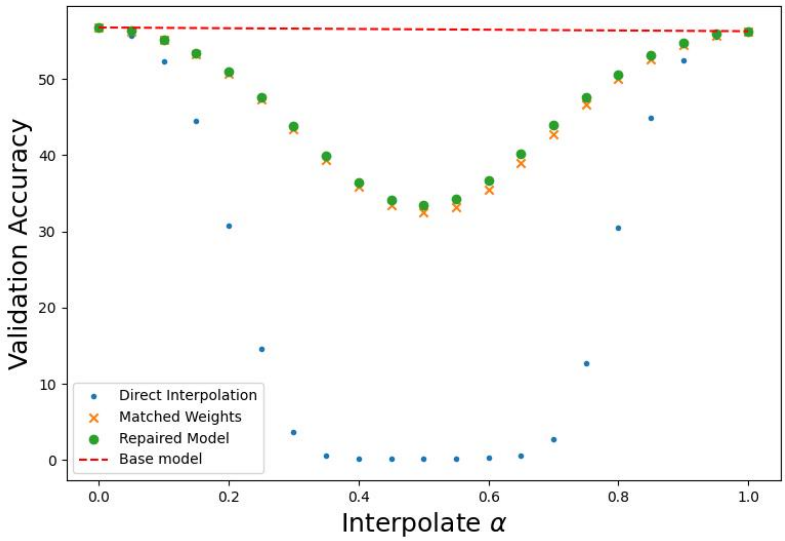

Figure 3.1: **Merge Results for Standard Trained AlexNet.** Model validation accuracies after permutation aligned improve significantly compared to direct merging.

similarities in the top-k images.

We also record the correlation matrix used in finding the permutation alignment of one model to another. This indicated the similarity of images activations in the permuted model.

These metrics are computed for each channel to support the visual analyses and to capture different aspects of feature similarity. High correlation or Kendall-τ values and low CLIP-δ indicate strong similarities between features learned by each model.

3.2 Experiments and Results

Our experiments focus on the AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) and ResNet (He, Zhang, Ren, & Sun, 2015) architectures trained on the ImageNet dataset (Deng et al., 2009). This choice is made largely because feature visualization methods work well and have been extensively studied for this model and dataset Krizhevsky et al. (2012); Li, Yosinski, Clune, Lipson, and Hopcroft (2016).

We train five AlexNet models to use in our merging experiments. All model weights are randomly initialized, and each model is trained with 90 epochs. The validation accuracy for each model

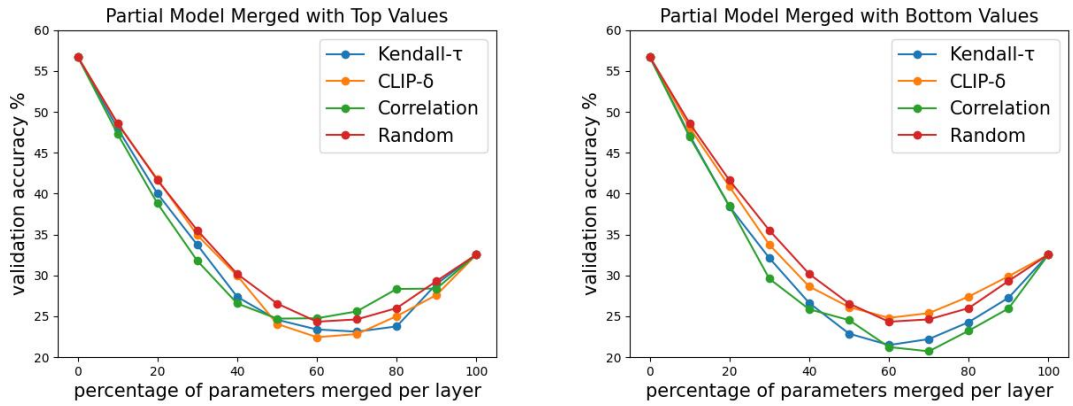

Figure 3.2: **Validation accuracy shows little difference compared with random base line**. In each experiment, we select top (left figure) / bottom (right figure) 10% to 100% channel metric values from each convolutional layer, and merge the weights of selected channels from model 1 and 2. The rest channels remains the model 1. The base line are models merged with randomly selected channels.

is $56.74\%, 56.25\%, 56.27\%, 56.45\%$ and 56.27%. We also trained two AlexNet models with stratified ImageNet dataset, which one model trained with 80% of 500 first image classes and 20% of the last 500 image classes, and another model trained on the rest data. Three ResNet18 He et al. (2015) models are trained as well. Visualization results of standard trained AlexNet are included in the main paper. More results for stratified AlexNet and ResNet are provided in Appendix, which follow similar trends.

3.2.1 Model Permutation and Merging

We take two trained models and permute one model (Model 2) to align with the other model (Model 1) using algorithm proposed in REPAIR paper Jordan et al. (2023). Next, we merge the models using multiple interpolation coefficients α ranging from 0 to 1, with a step size of 0.05 and evaluate the merged models performance on the validation dataset. Additionally, we implement the algorithm to rescale the weights in merged model (referred to as **Repaired Model** in the Figures) and record the corresponding validation accuracy. Models are also directed merged without permutation as base line for comparison. Figure 3.1 shows merge results for standard trained AlexNet model. Validation accuracies improve greatly compared to direct merging. However repaired merge models do not perform significantly better.

To explore more efficient merging strategies, we investigate whether quantitative similarity metrics can guide the merging process. Specifically, we compute Kendall-τ, CLIP-δ, and correlation values for each channel in the permuted Model 2 with respect to the corresponding channel in Model 1. Based on the magnitude of these metrics, we perform channel-wise merging of the two models. The results, shown in Figure 3.2, indicate that this approach does not yield a significant improvement in validation accuracy compared to a random baseline. This suggests that the current similarity metrics are insufficient to reliably capture the semantic alignment of features across models, limiting their utility for guided merging.

3.2.2 Visualizing Permutation Based Model Merging

We take two trained AlexNet models, with one (Model 2) permuted to align with the other (Model 1). Then we feed the training dataset to both models and extract the 10 images that activate each convolutional channel most (top-10 feature visualizations). For a selected channel top images of original model 1, model 2 (before permutation), permuted model 2, permute merged model, and direct average merge (without permutation) are shown in Fig 3.3.

Observation 1: Many but not all permutation matched features are similar. We informally define *feature visualization similarity (FV)* as a visual observer identifying the same structures and patterns (and sometimes the same images) between two visualizations of two different channels (typically found in two different models). We observe by looking through the visualizations over many channels that the permuted model 2 feature visualizations are often but not always similar to those from model 1 as illustrated in Fig 3.3 (bottom). Thus despite being trained from different initializations the training will often find similar features. This also matches observations in Li et al. (2016).

We now turn to analyzing the feature visualization of the merged model.

Observation 2: Merged model feature visualization is dependent on the source models and whether there is an FV similarity after permutation matching. Analyzing the feature visualizations from many channels we make the following observations. In the case that the permuted model has similar top activated images, the top images from merged model are also similar. This is illustrated on the top in Fig 3.3 where the permuted model 2 shows similar features to model 1.

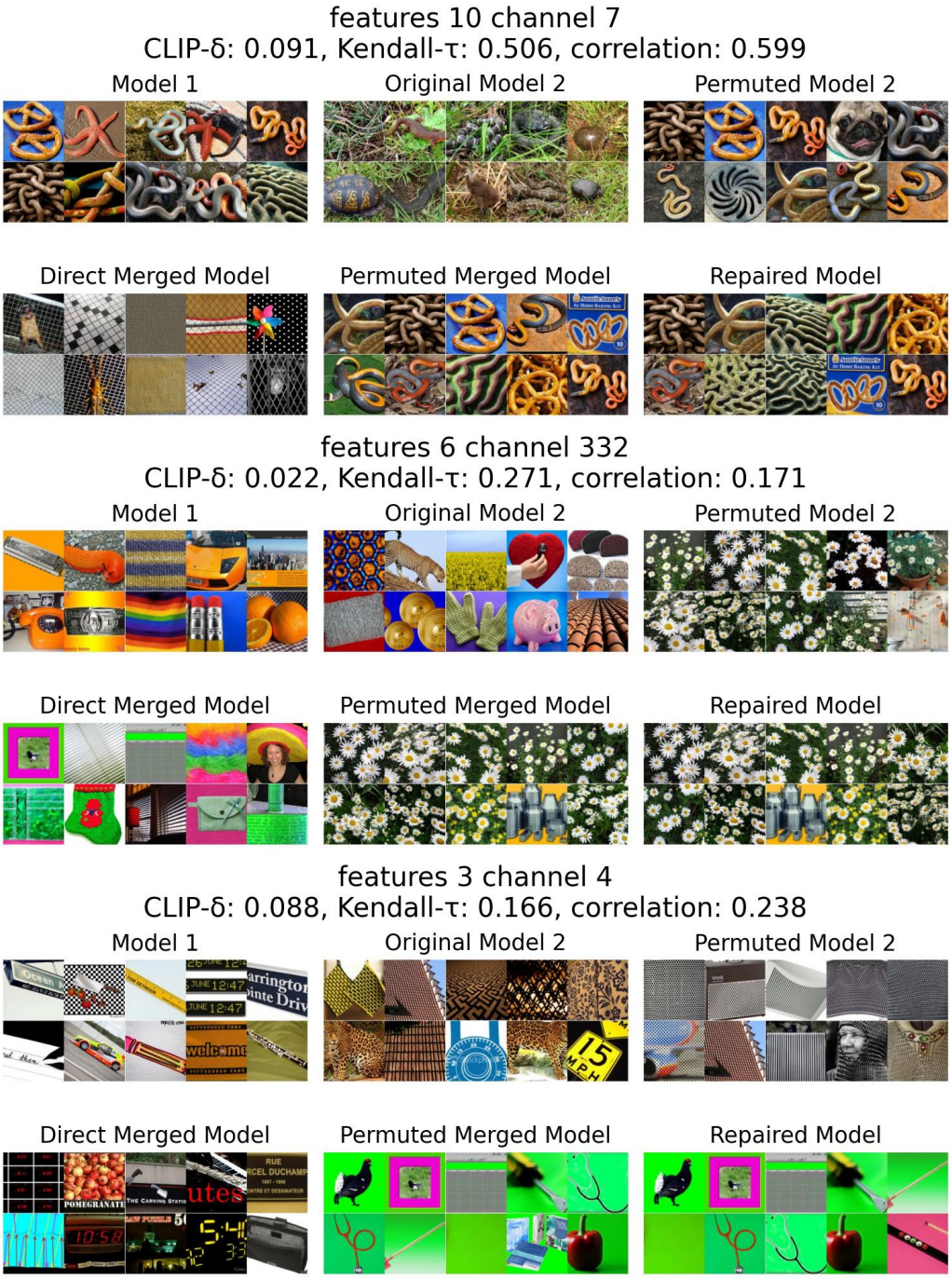

Figure 3.3: **Example of top images. Top** shows the permuted model 2 has similar top-activated images as model 1 and thus we have similar results in the merged model. **Middle** shows one example that top images from one model dominate top images of merged model. **Bottom** shows one example that top activation images do not match in models 1 and 2. The top images from the merged model also do not match and are not from either model. More examples are included in the Appendix.

However when the images from 2 models are not similar after permutation (e.g. bottom of Fig 3.3), we observe different behavior on merged models top images. These behaviors were observed by inspection of all channels in AlexNet to fall into three general categories:

(1) Images from one model (either model 1 or 2) dominate the top images of merged model.

(2) Images from both models appear in the top images of merged model.

(3) Most top images of merged model do not belong to any of the 2 models.

3.2.3 Aligning Multiple Models to One

Since the matching of features is so critical to the visualization observed in the merged model, we ask whether the mismatched cases after permutation occur randomly or they represent rare features in model 1 that are hard to match in model 2? To study this we perform the matching process multiple times with one fixed model being matched.

Specifically, we permute the 4 of our AlexNet models (all trained from different initialization) to aligned with model 1. Top activated images are extracted for all 5 models.

Observation 3: Certain features of the reference model are challenging to match in terms of FV similarity. Analyzing all channels in all layers of AlexNet we have the following cases (see Fig 3.4):

(1) For a channel all 5 models have very similar top images (occuring $(5 - 10\%)$)

(2) For a channel most model have similar top images except one or two (occuring $(35 - 50\%)$).

(3) For a channel top images from all 5 models are dissimilar $(40 - 60\%)$

We also study the mean and standard deviation of the metrics we use. It is clear that channels with high means (low for CLIP-δ scores) and low standard deviations usually have very similar top-activated images. On the other hand, if we have high standard deviation and low means, visually, the top images vary a lot.

These results support our hypothesis that some features are easily learned across models, while others are rare or unstable, leading to dissimilar feature visualizations after permutation (Observation 1).

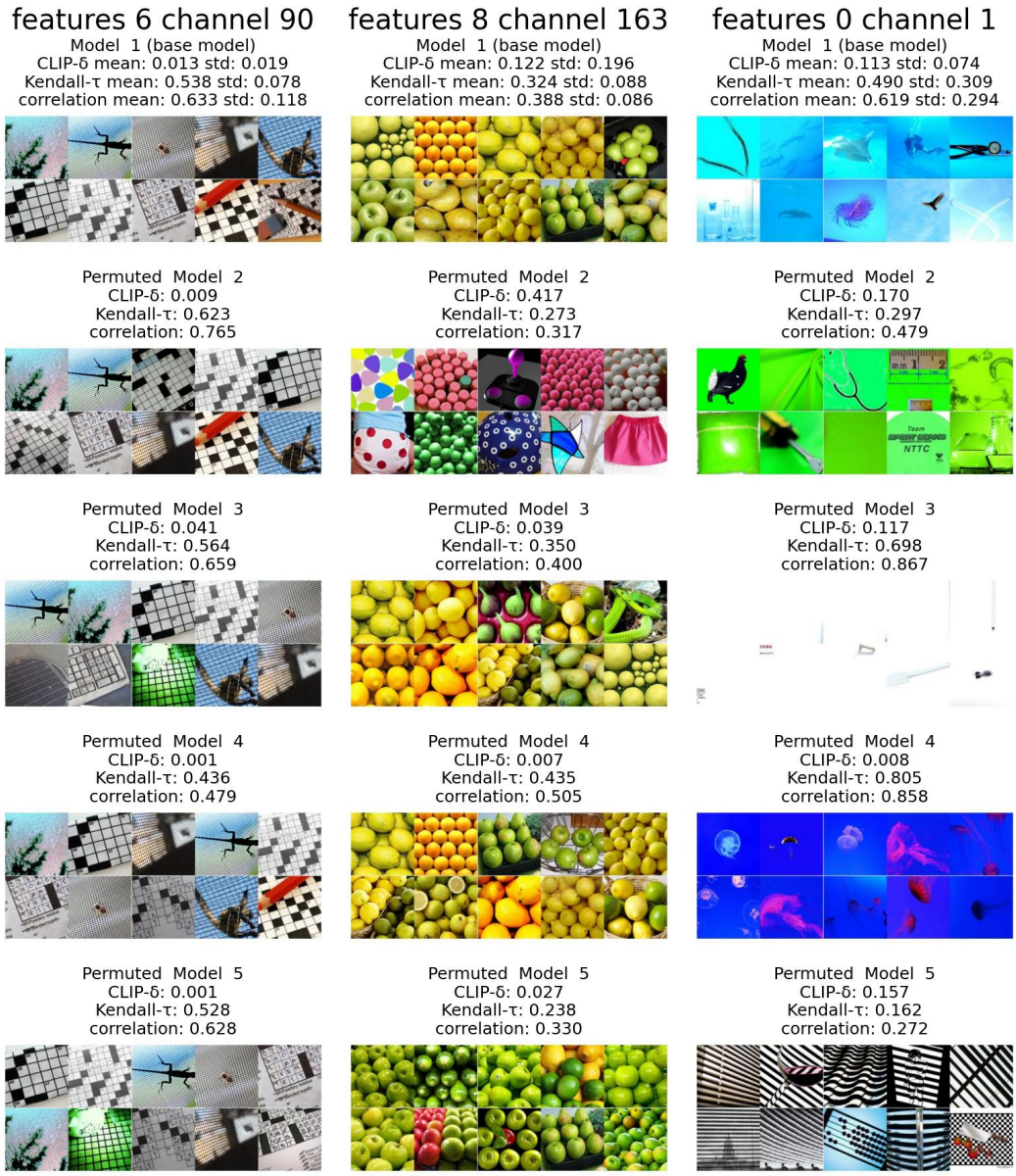

Figure 3.4: **Top Images Extracted from 5 AlexNet Models.** Model 2 to 5 are all permuted to align with model 1. The three columns show examples of the three cases we observe. The **left** column show the case that all top images from the 5 models are visually similar. The **middle** column is an example of one channel with similar top activated images from 3 models, and the **right** column is the case where all top images are not similar. The metrics show consistent trends, where high Kendall-τ and correlation values, along with low CLIP-δ, are typically associated with high feature similarity, and the opposite holds for low similarity.

3.2.4 Aligning Models Using Linear Regression

We observe that although per neuron or channel, feature visualization is based on the hypothesis that each channel represents a particular feature of the data, a given useful feature may be represented by a single channel in one network but can be represented in a distributed way by another network.

We take all the features in a given layer of model 2 and regress them onto the features of model 1, building a separate regression model for each feature in model 1. Specifically, we get the activation of each convolutional channel from both models with 5000 images randomly selected from the ImageNet dataset, i.e., 5 images per class. Then for each layer, we find a linear regression model, such that for activation values of each channel in model 1, it is approximated by a linear combination of activation values from all channels in model 2. Afterward, we feed the entire dataset to model 2, apply the regression model on the activations, and extract the top 10 activated images of each channel.

Observation 4: For each feature in the model there is a (often sparse) linear combination of features in model 2 that gives an FV similarity. In Fig 3.5, we see the results of visualizing the regression model outputs at each layer. The results show that top-activated images are now visually aligned. Furthermore, we study the sparsity of the weights we got from the regression model. As shown in Figure 3.5, we can see the top images do not change a lot when we only select the weights with top magnitude and set the rest to 0. This shows that the weights that impact the alignment are sparse. However we do find that for certain channels (particularly in the early layers), the top activated images change when we use less weights, and these channels are also visually hard to align channels when we compare the 5 models (Figure 3.5).

Our results suggest that permutation-based merging can be substantially limited since it is too restrictive to assume a one-to-one matching between features. Indeed recent model merging methods specifically aim to address this restriction (Horoi et al., 2024; Stoica et al., 2024), showing promising new results. Our work helps to provide additional motivation for these new methods.

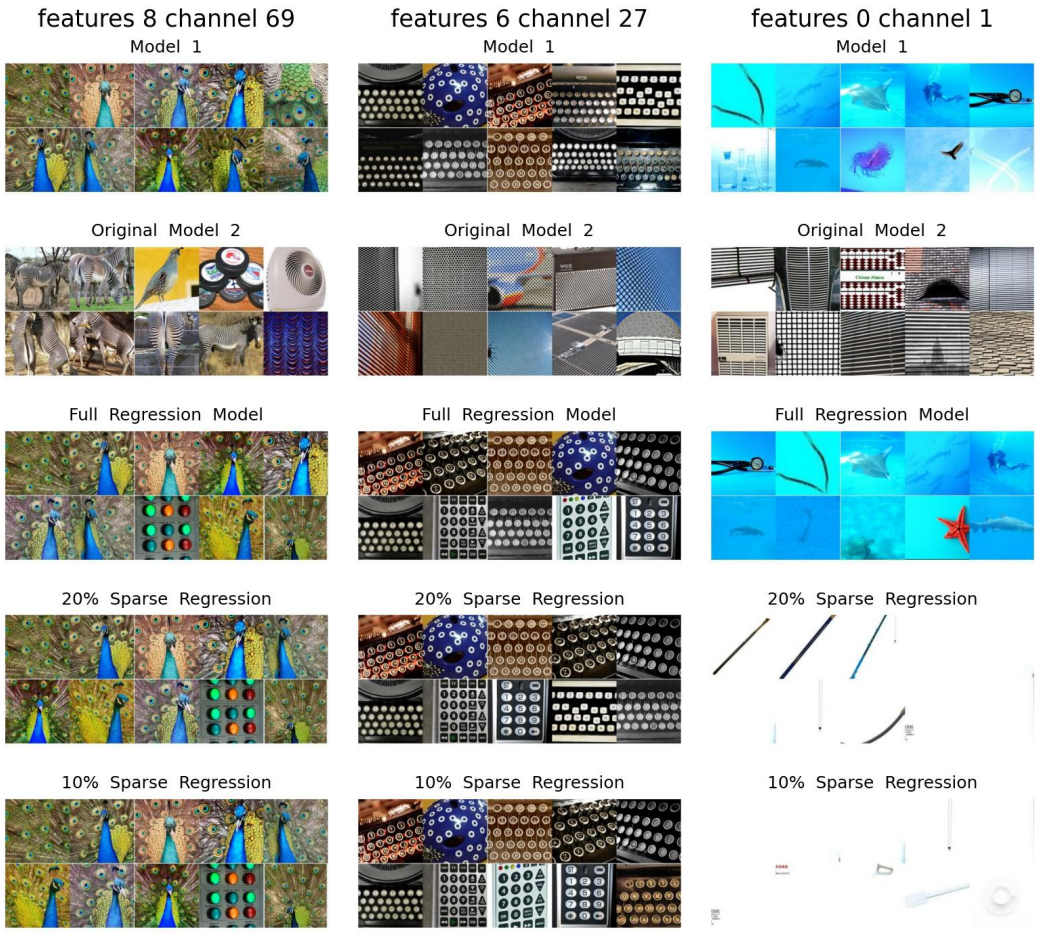

Figure 3.5: **Top images aligned after applying regression model. Left** and **middle** columns show the examples where even with only 10% sparse regression weights, the images are still similar to model 1. However, there are also some cases where after applying sparse weight, the images changed. Examples are shown in **right** row.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In this work, we explored the internal mechanisms of model merging in deep neural networks through the lens of feature visualization. While previous studies demonstrated that permutation-based alignment enables linear mode connectivity, they provided little insight into how features are preserved or distorted in the merged models. By visualizing the top-activated images across various models, both before and after alignment, we gained a deeper understanding of how individual features behave through permutation and interpolation.

Our results showed that many features are consistently aligned across independently trained models, but a significant portion remains hard to match. These unmatched features often correspond to rare or distributed representations that are not well captured by one-to-one alignment. We also found that feature similarity can be improved through regression-based alignment, and that the resulting weights are often sparse—indicating that only a few channels contribute meaningfully to cross-model feature approximation.

Altogether, our findings support the hypothesis that feature-level redundancy and sparsity underlie the success of merging strategies. However, they also highlight fundamental limitations of existing permutation-based methods. These insights provide a more mechanistic explanation of merging performance and point toward new directions for building more flexible and interpretable model merging frameworks.

4.2 Future Work

Similarity Metrics Our experiments revealed limitations in current quantitative similarity metrics, which do not always align with visual or qualitative observations. Metrics such as correlation and Kendall-τ may not always capture the semantic similarities between activation patterns. For instance, two channels may respond to conceptually similar inputs (e.g., animals or textures) but produce low metric scores. Future work should explore alternative metrics that are more consistent with visual intuition and potentially better suited for tasks involving model comparison and merging. Such metrics could lead to more reliable evaluations of feature similarity and support the design of more effective model merging strategies.

Model Merging with Sparse Autoencoders The sparsity observed in linear combinations of features between models suggests that representations are often distributed rather than aligned one-to-one. This motivates the use of sparse autoencoders to learn a disentangled latent space where overlapping features from multiple models can be more easily aligned or combined. By enforcing sparsity, we can encourage the emergence of monosemantic units, which are easier to interpret and merge. Future work can also investigate training autoencoders jointly across multiple models to discover shared and distinct feature subspaces, potentially enabling more robust merging across diverse architectures or tasks.

Feature Visualization Across Architectures While our study focused on model merging methods acting on identical architectures, an important direction for future work is to apply feature visualization techniques to compare and analyze models across different architectures. Understanding how features are represented in structurally distinct networks, such as AlexNet, ResNet, or Vision Transformers, can provide deeper insights into the generality and transferability of learned representations. Visualization can help reveal which features are shared across architectures and which are architecture-specific, potentially guiding cross-model merging strategies or transfer learning approaches. Additionally, it may expose limitations in current visualization techniques when applied to deeper or more complex models, motivating new tools for interpretability in diverse models.

Synthetic Feature Visualization Our analysis primarily relied on dataset-based top-activated images to interpret learned features. However, synthetic feature visualization, using optimized or generative inputs, may reveal additional properties of merged models, such as robustness, feature superpositions, or latent biases. Future work can incorporate synthetic feature optimization (e.g., via GANs) to analyze how merging affects abstract feature hierarchies, and to identify whether merged models exhibit unexpected behaviors not easily detectable through dataset examples alone.

Appendix

A Additional Results for Standard Trained AlexNet Models

We trained 5 AlexNet models on full ImageNet Dataset for 90 epochs with SGD optimization. Learning rate starts at 0.01 and decays by a factor of 10 every 30 epochs. All experiments were conducted using a single V100 GPU with 16GB of memory.

A.1 Top Images for Permutation Based Model Merging

Model 2 is permuted to align with model 1. Both direct merge and permuted merged models are studied, as well as repaired merge model. Top activated images from original model 2 before permutation is also included. Visually for most channels, top images are getting more similar after permutation, and the merging model shares the same similarity with slightly change in repaired model. (Figure A) In the case where top images from permuted model are not similar to those from model 1, we get different cases for merged model images. See Figure B, C, D.

A.2 Top Images for Aligning Multiple Models to One

We permuted 4 models against model 1. Visually, we can see that the similarity of the top images from 5 models varies from channel to channel. Some channels have similar images from all 5 models, and some have images totally dissimilar from all 5 models. There are also channels with similar images from 3 model and another type of similar images from 2 other models. Furthermore, We compute the CLIP-δ, Kendall-τ and correlation metrics for each channel compared to model 1. Means and standard deviations are also shown. These metrics match our visual intuition. For individual model, a higher Kendall-τ and correlation value or a lower CLIP-δ usually means top images are more similar to those in model 1. Cross all 5 models, a higher mean with lower standard deviation implies similar images, and a lower mean with higher standard deviation show dissimilarity. Feature visualization results from selected channels are shown in Figure E, F, G.

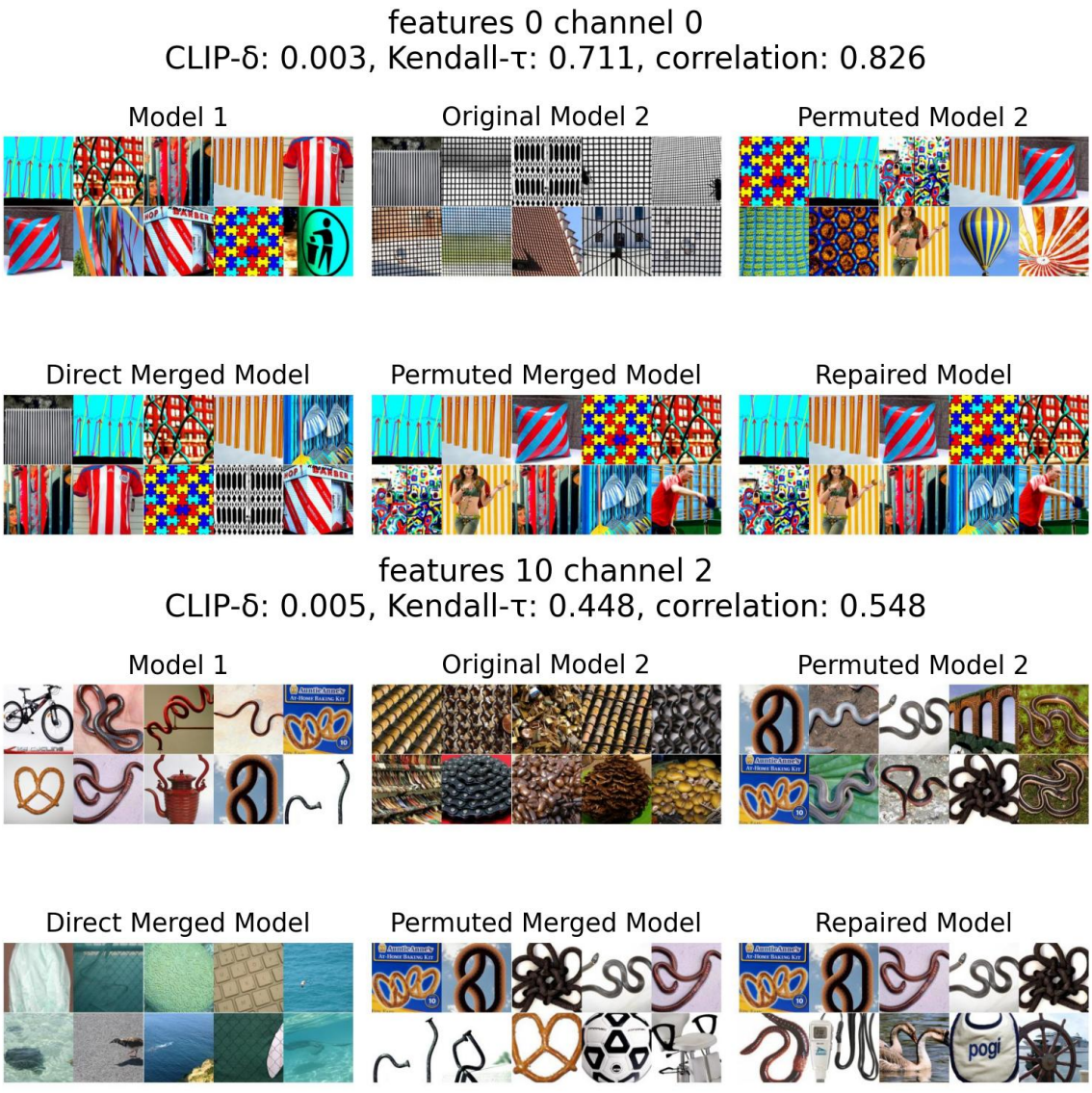

Figure A: Similar top images from model 1 and permuted model 2 with similar results from merged models.

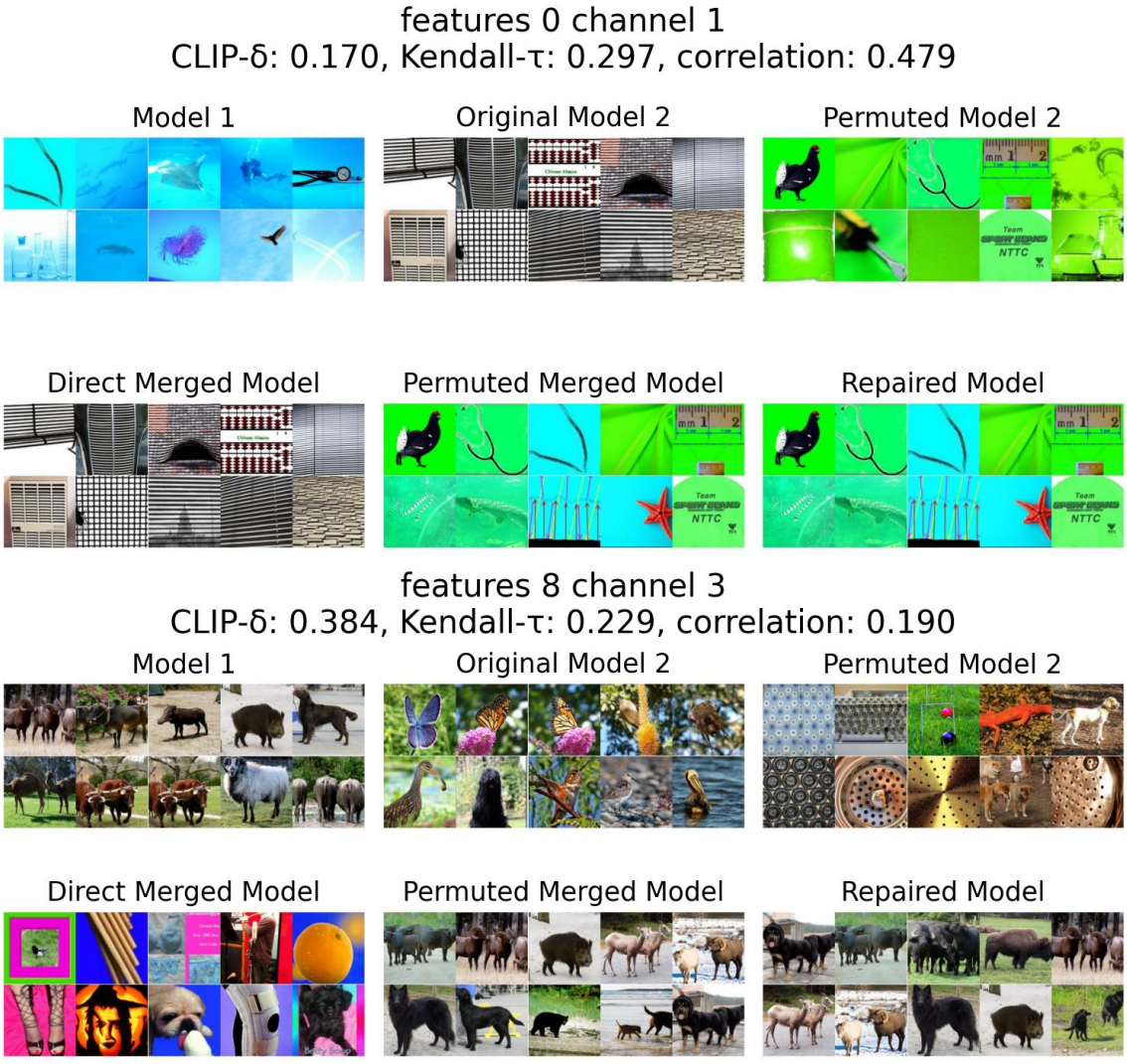

Figure B: Dissimilar top images from model 1 and permuted model 2 with images from one model dominate the top images of merged mode.

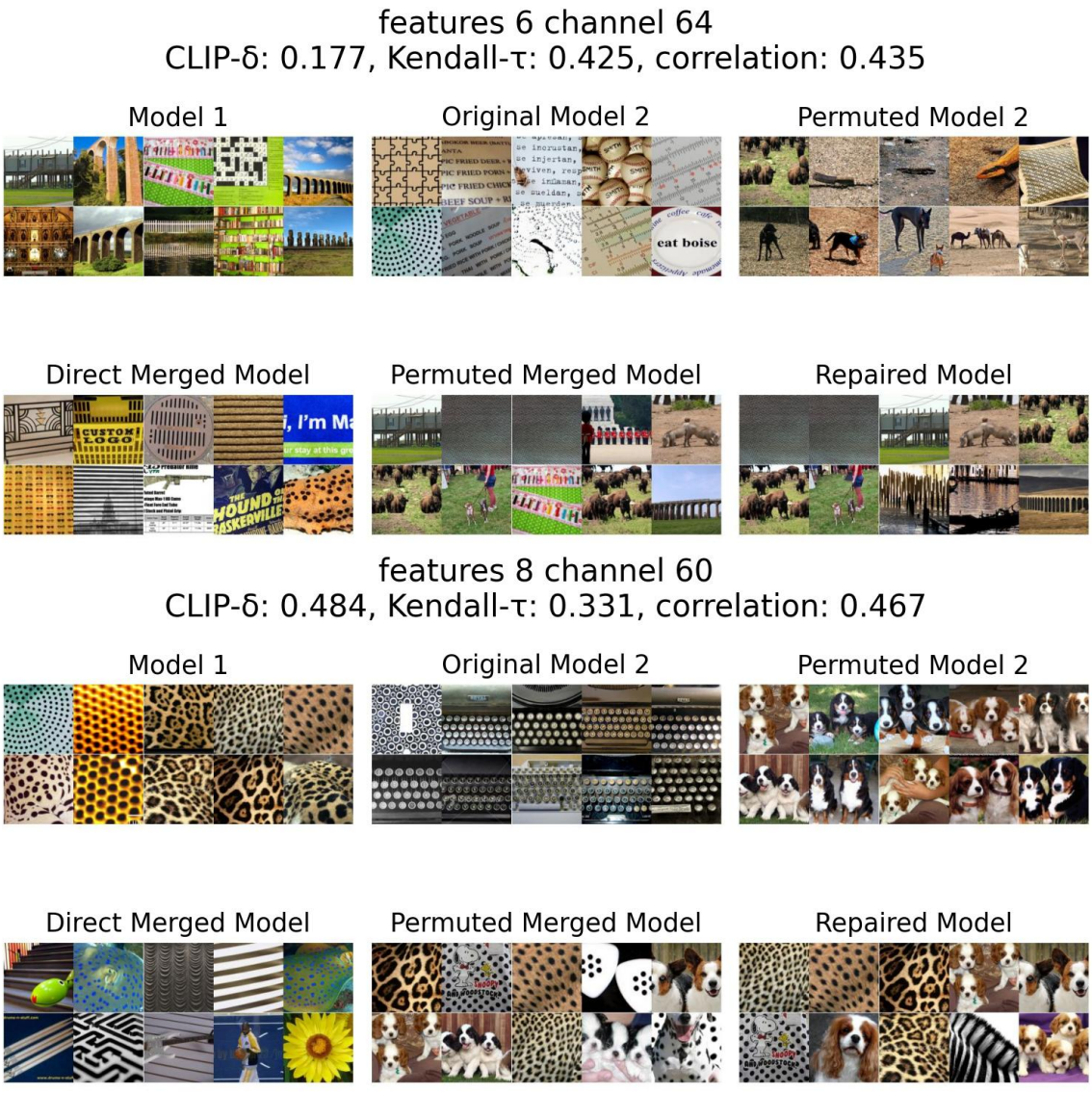

Figure C: Dissimilar top images from model 1 and permuted model 2 with images from both models appear in the top images of merged model.

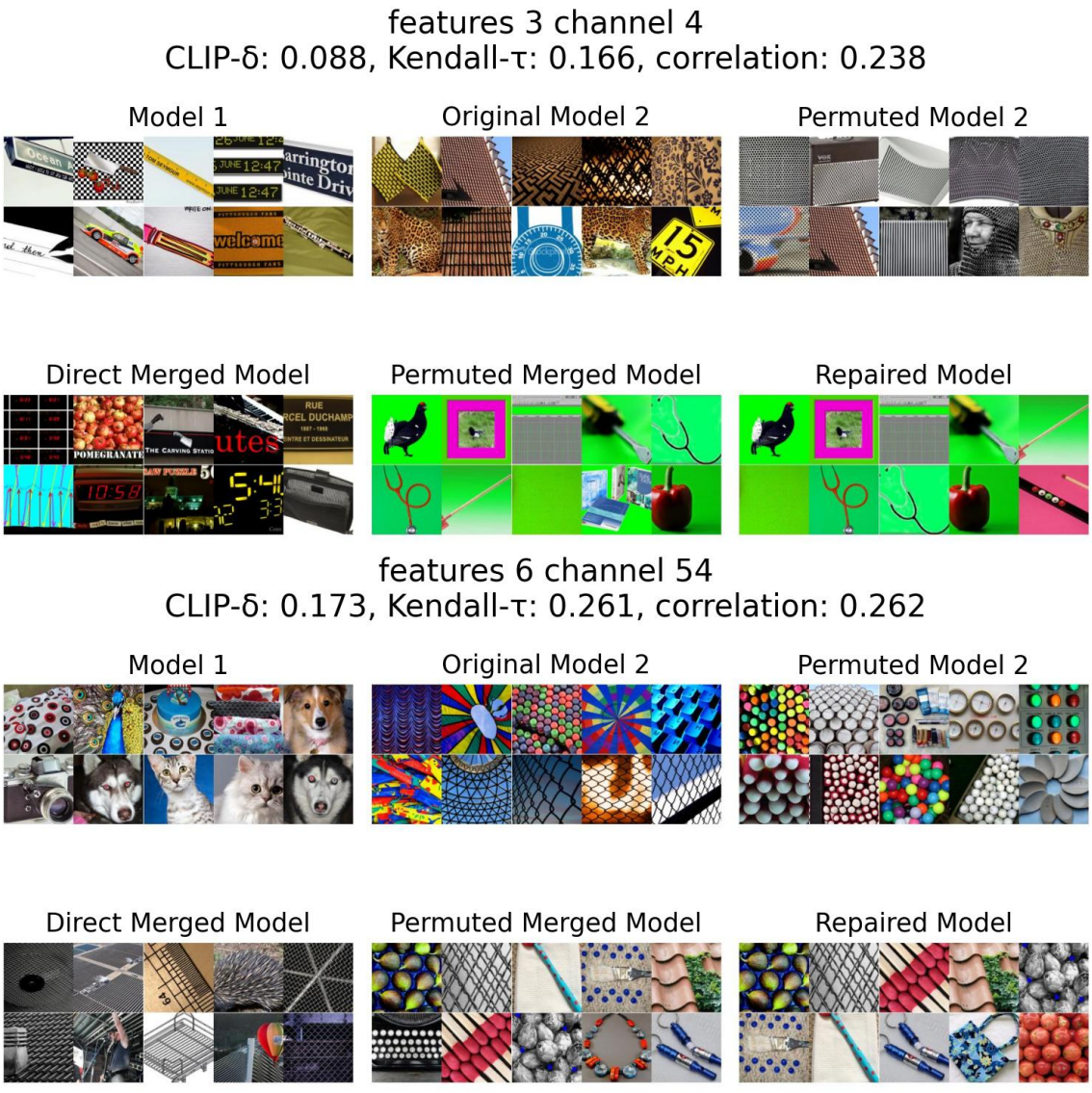

Figure D: Dissimilar top images from model 1 and permuted model 2 with most top images of merged model do not belong to any of the 2 models

A.3 Top Images for Linear Regression

We create a linear regression model, which take all the features in a given layer of model 2 and regress them onto the features of model 1. We compute the activation of each convolutional channels from both models with 5000 images randomly selected from ImageNet dataset, i.e., 5 images per classes. Then for each layer, we implement a linear regression model, such that for activation values of each channel in model 1, we have a linear combination of all channels in model 2 to approximate them. Afterwards, we feed the entire dataset to model 2, apply the regression model on the activations, and extract the top 10 activated images of each channels. Results for linear regression experiments are in Figure H, I. For majority of the channels, top activated images are aligned with even top 5 weights used.

B Results for AlexNet Models Trained with Stratified Dataset

We trained 2 AlexNet model with stratified dataset. One model is trained with 80% of 500 first image classes and 20% of the last 500 image classes, and another model is trained on the rest data. Both models are trained with SGD optimizer for 90 epochs. Learning rate starts at 0.01 and decays by a factor of 10 every 30 epochs.

B.1 Permutation and Merge Results

Figure J shows merge results for AlexNet model trained with unbalanced data. We carry the same experiment as described in **A.1**. Validation accuracies improve greatly compared to direct merging. Repaired merge models also perform better than the permuted only models.

Figure E: Examples where all top activated images from different model are very similar.

Figure F: These channels have similar top activated images from most models but not all 5.

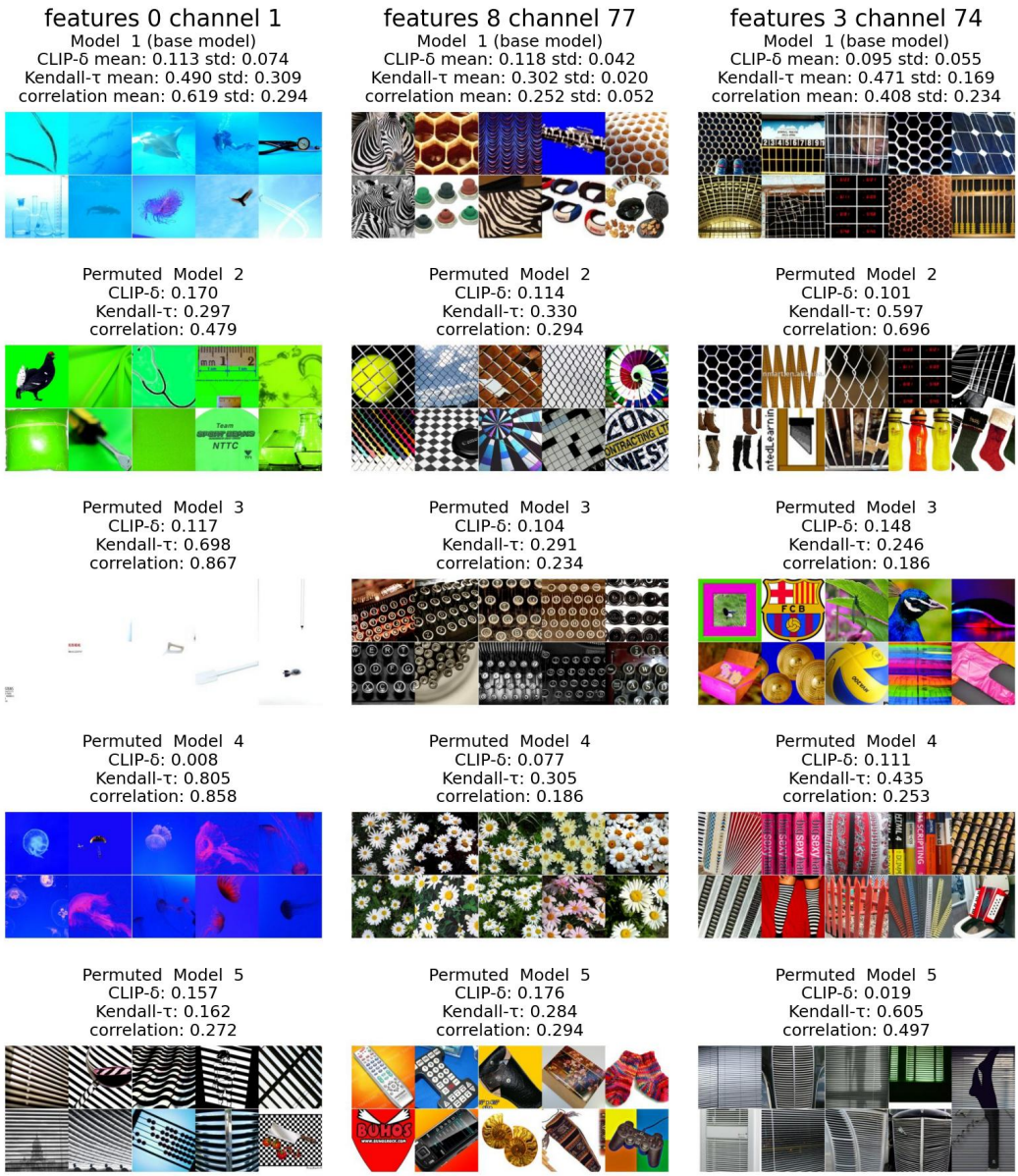

Figure G: Top activated images are mostly distinct across models even after alignment. This may indicate hard to learn or rarely learned features.

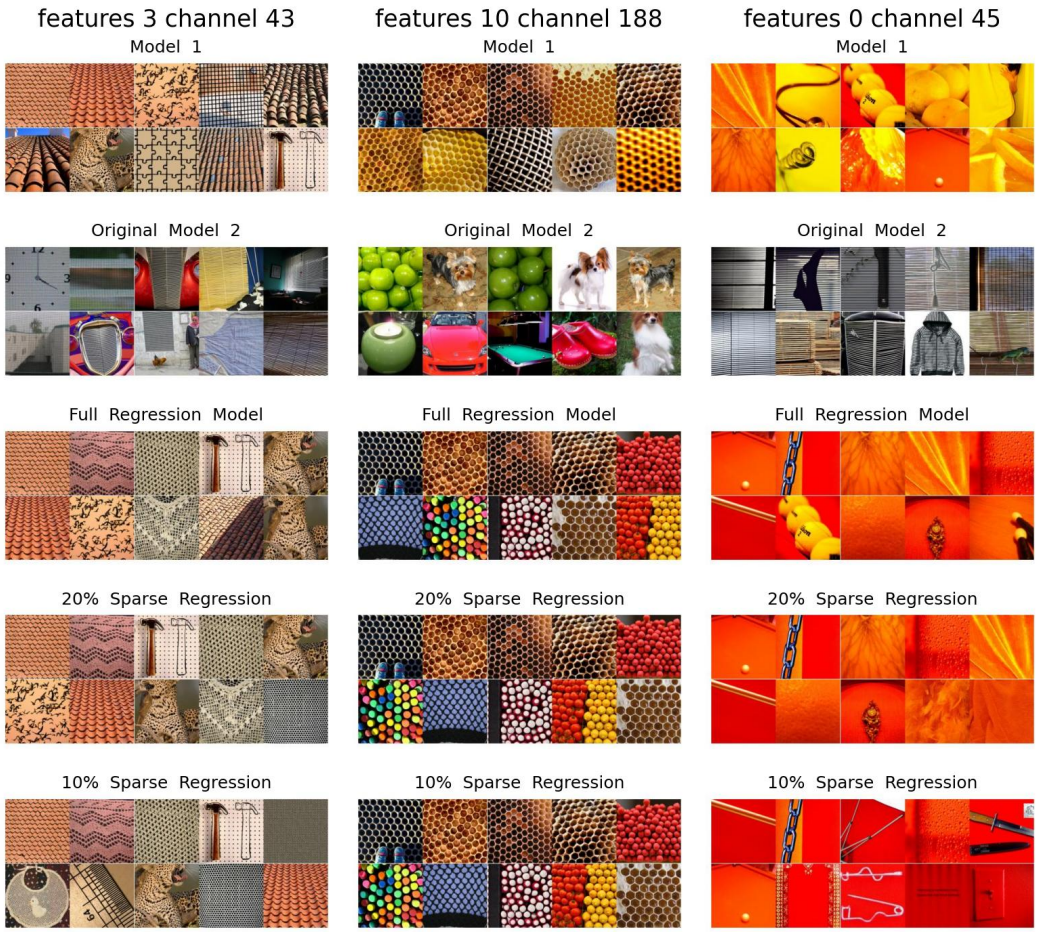

Figure H: Similar Top Activated Images After Regression. In most channels features are similar with only 10% sparse regression weights.

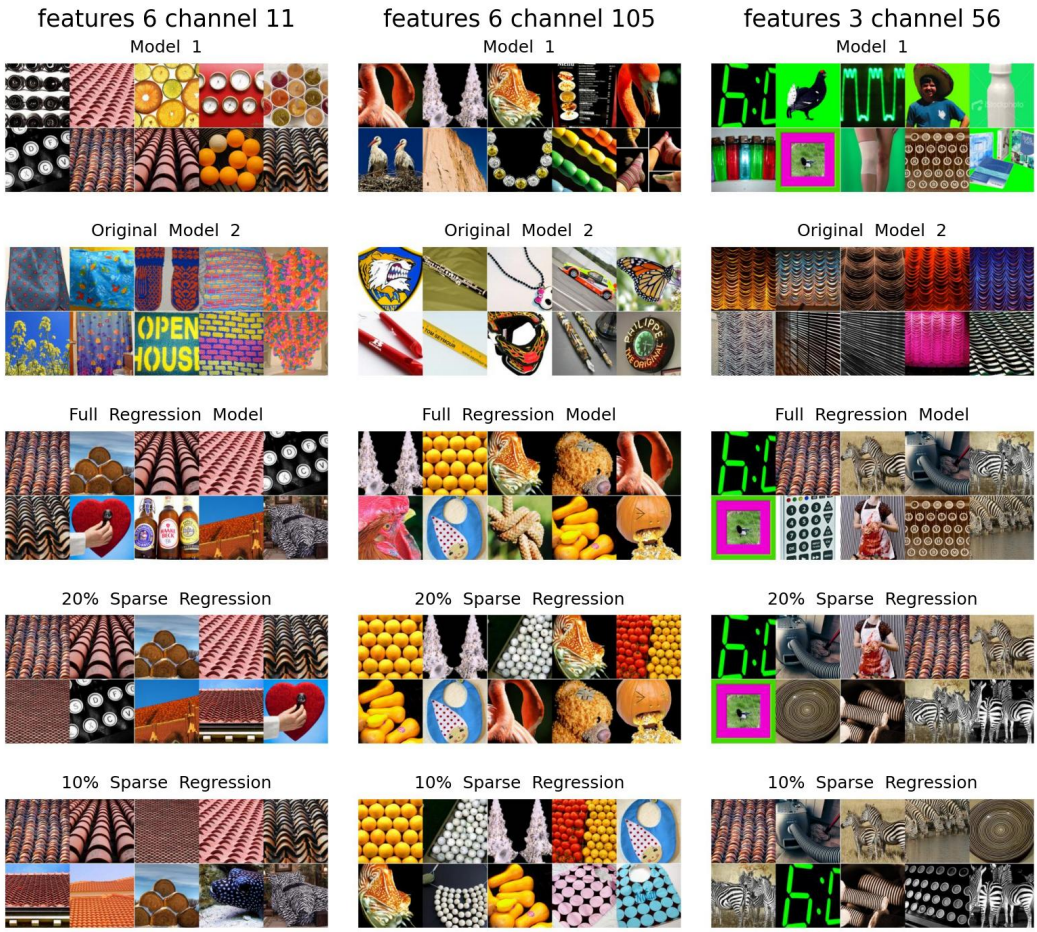

Figure I: Dissimilar Top Activated Images With Sparse Regression. In some channels, full weights regression model have similar top activated images, while similarity decrease with sparse weights.

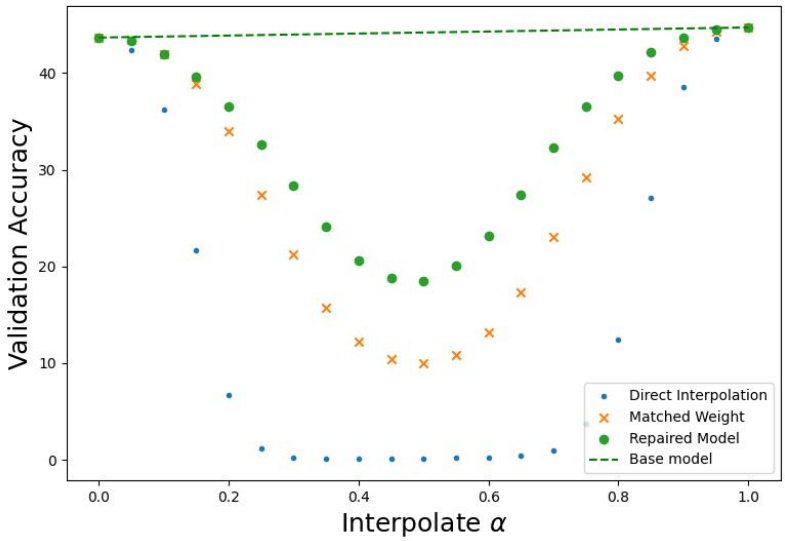

Figure J: Merge Results for AlexNet Trained with Stratified Dataset.

B.2 Top Images for Permutation Based Model Merging

Model 2 is permuted to align with model 1. Both direct merge and permuted merged models are studied, as well as repaired merge model. Top activated images from original model 2 before permutation is also included. Figure K, L show some selected results. We observe similar pattern as standard trained models. It is also observed that features are harder to align compared to standard trained models. This is probably because of the unbalanced training dataset.

C Results for ResNet18 Models

We trained 5 ResNet18 models. Each models are trained on full ImageNet Dataset for 100 epochs with SGD optimization. Learning rate starts at 0.1 and decays by a factor of 10 every 30 epochs.

C.1 Permutation and Merge Results

Figure M shows merge results for ResNet model. Same experiments as in **A.1** are performed. We observe some accuracy improvement with permuted only models, especially when α is less than

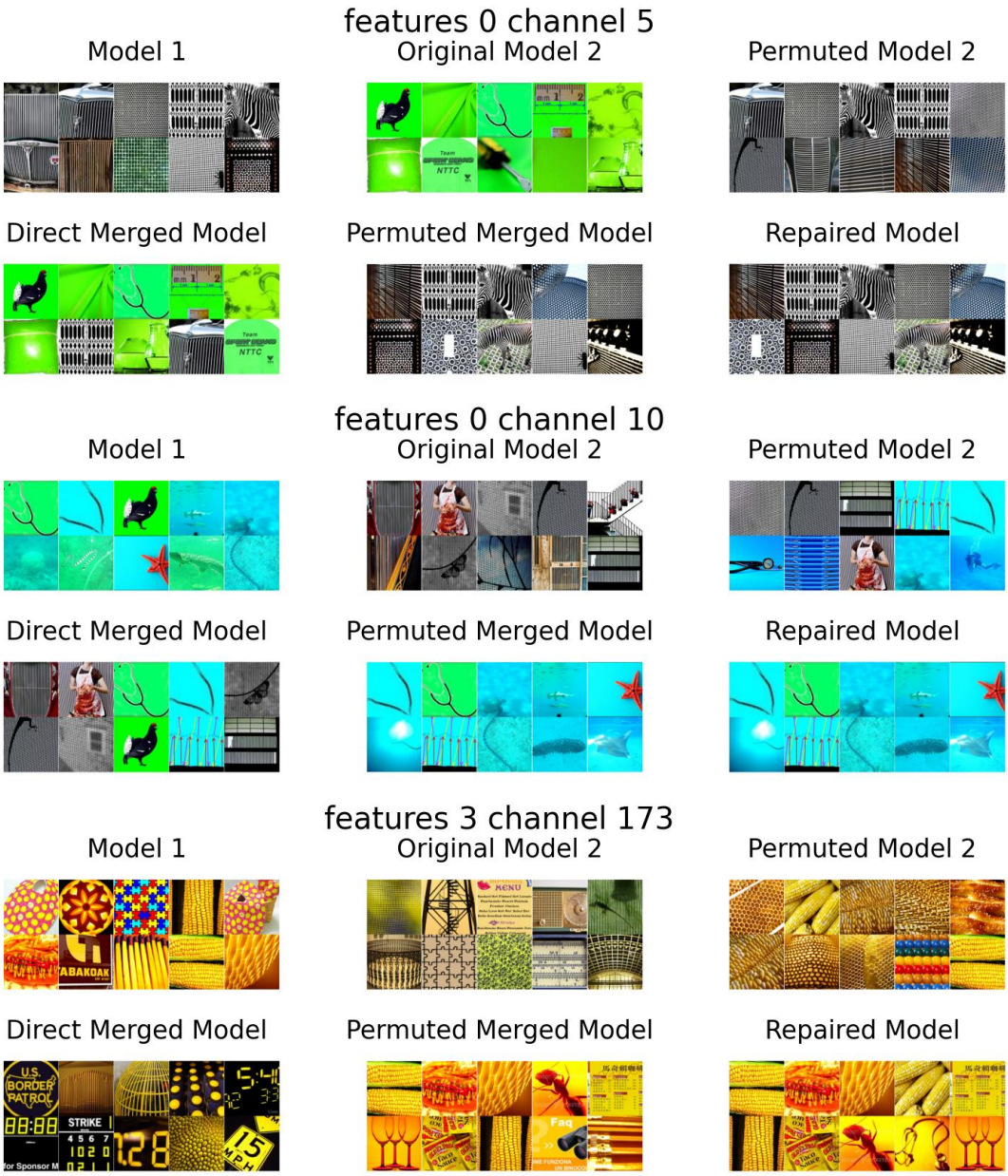

Figure K: Top activated images from selected channels for AlexNet models trained with stratified dataset. Here are some results for aligned model with similar top images.

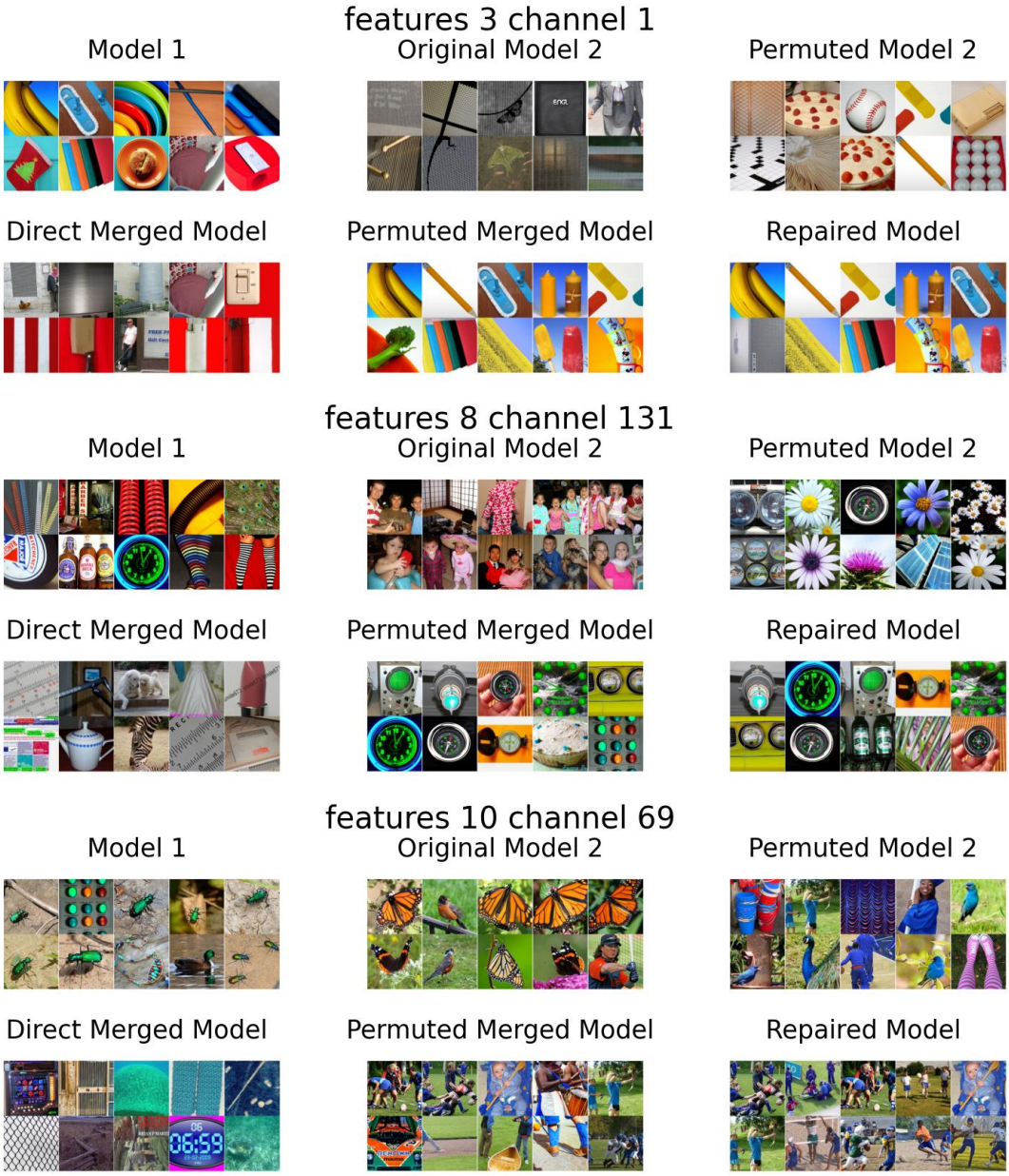

Figure L: Top activated images from selected channels for AlexNet models trained with stratified dataset. We also observe channels which do not have similar top images after permutation.

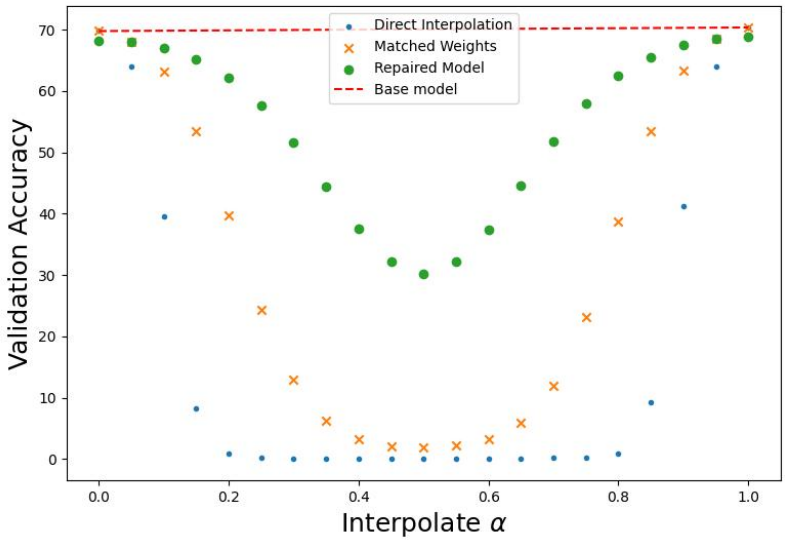

Figure M: Merge Results for Standard Trained ResNet18.

0.4 and greater then 0.6. After rescaling, the validation accuracies are improved significantly.

C.2 Top Images for Permutation Based Model Merging

Due to the ResNet18 model architecture and permutation algorithm, the feature visualization experiments are not performed with each convolutional layer, but with each residual block instead. There total 8 residual blocks in a ResNet18 model. Similar to previous experiments, we visualize top-10 images that activate each block most in following models: model 1, model 2, permuted model 2, directly merged model and permuted and merged model. Results are shown in Figure N, O. We also observe similar pattern as AlexNet models are observed.

C.3 Top Images for Aligning Multiple Models to One

We then permute three more model (model 3, 4, 5) and visualize top-10 images that activate each block most in each model. Results are shown in Figure P, Q. We can see similar pattern as AlexNet models are observed.

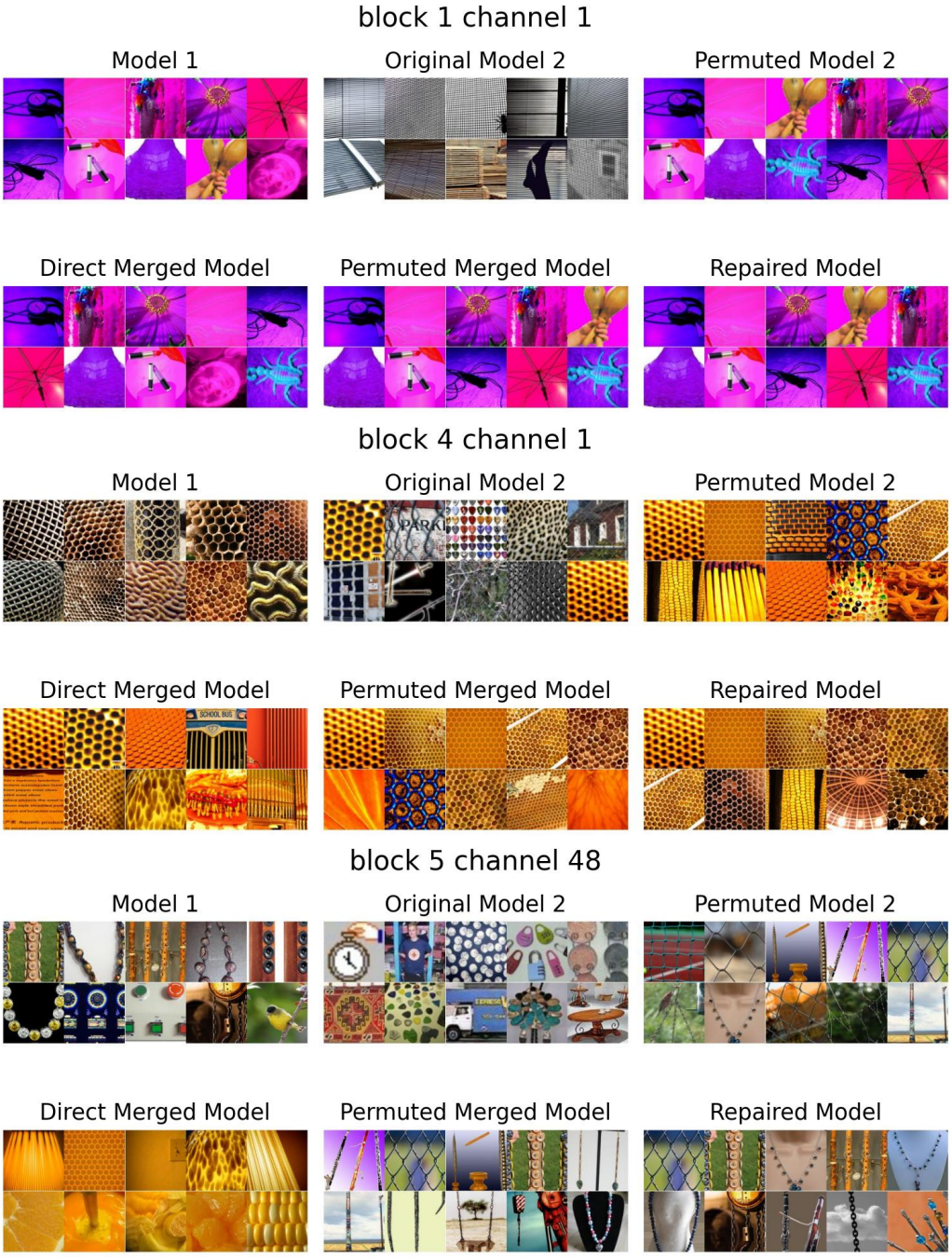

Figure N: Top images for 2 ResNet18 models. Model 2 is permuted to align with model 1. Top images from original model 2 and direct merged model are also shown. Top images shows similarities between model 1 and permuted model 2. In the case of dissimilarity, we observe that top images from merged model after permutation can have images from both model 1 and permuted model 2 or from none of them, which shares the same pattern as AlexNet models.

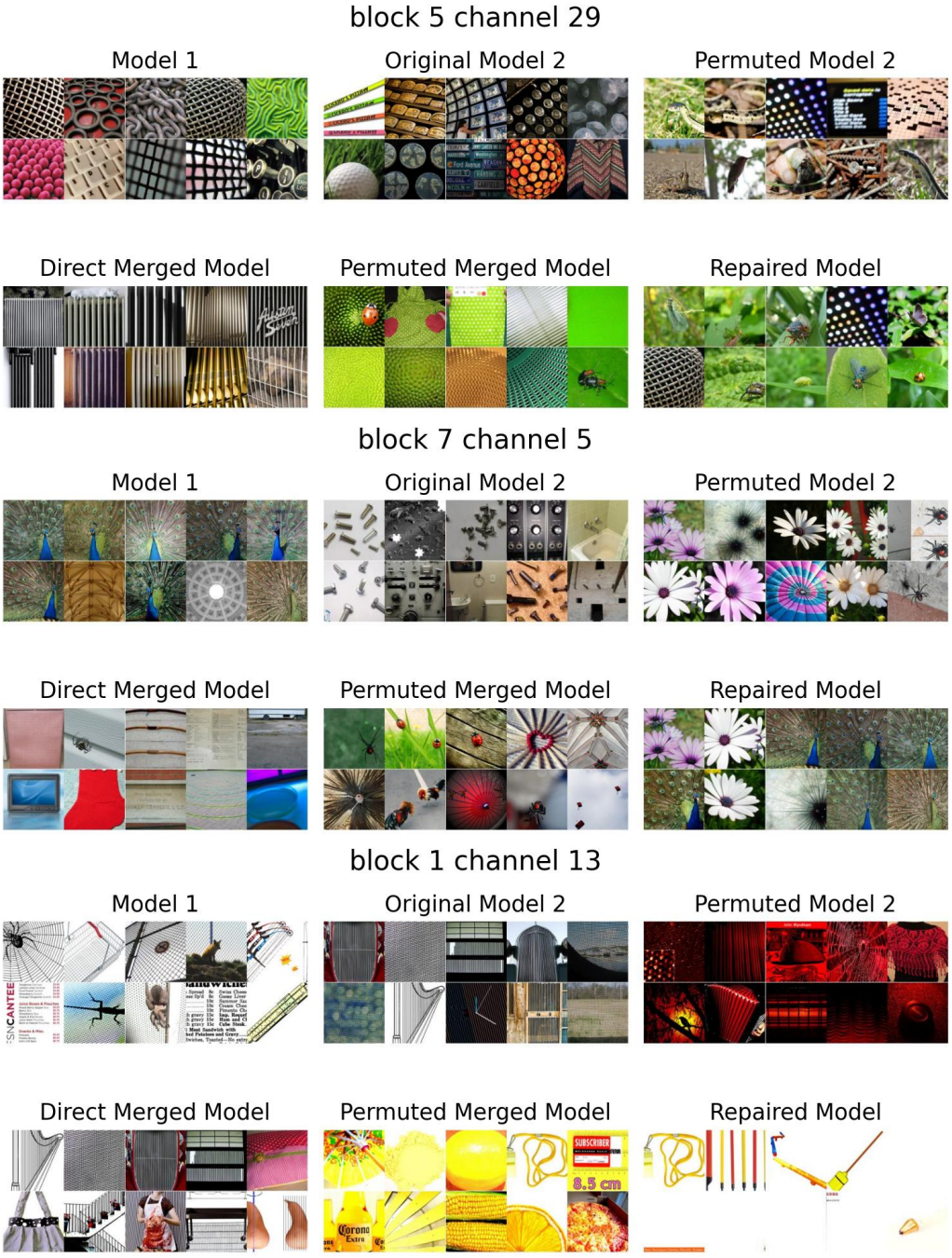

Figure O: Top images for 2 ResNet18 models. Model 2 is permuted to align with model 1. Top images from original model 2 and direct merged model are also shown. Top images shows similarities between model 1 and permuted model 2. In the case of dissimilarity, we observe that top images from merged model after permutation can have images from both model 1 and permuted model 2 or from none of them, which shares the same pattern as AlexNet models.

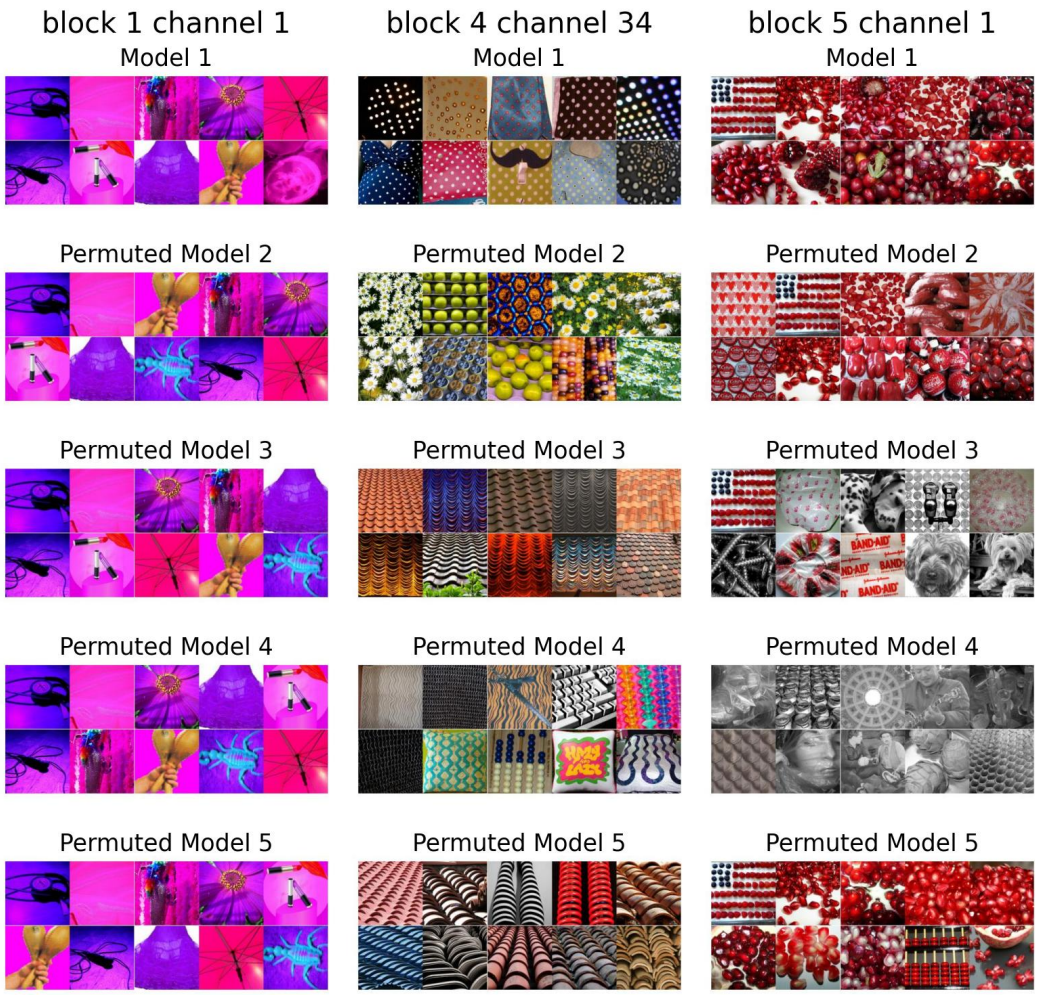

Figure P: Top activated images from 3 ResNet18 models. Model 2 to 5 are are more aligned to model 1 after permutation. In some cases, we have only some of the models visually aligned to model 1. There are also cases that neither of the 2 models shows FV similarity after permutation.

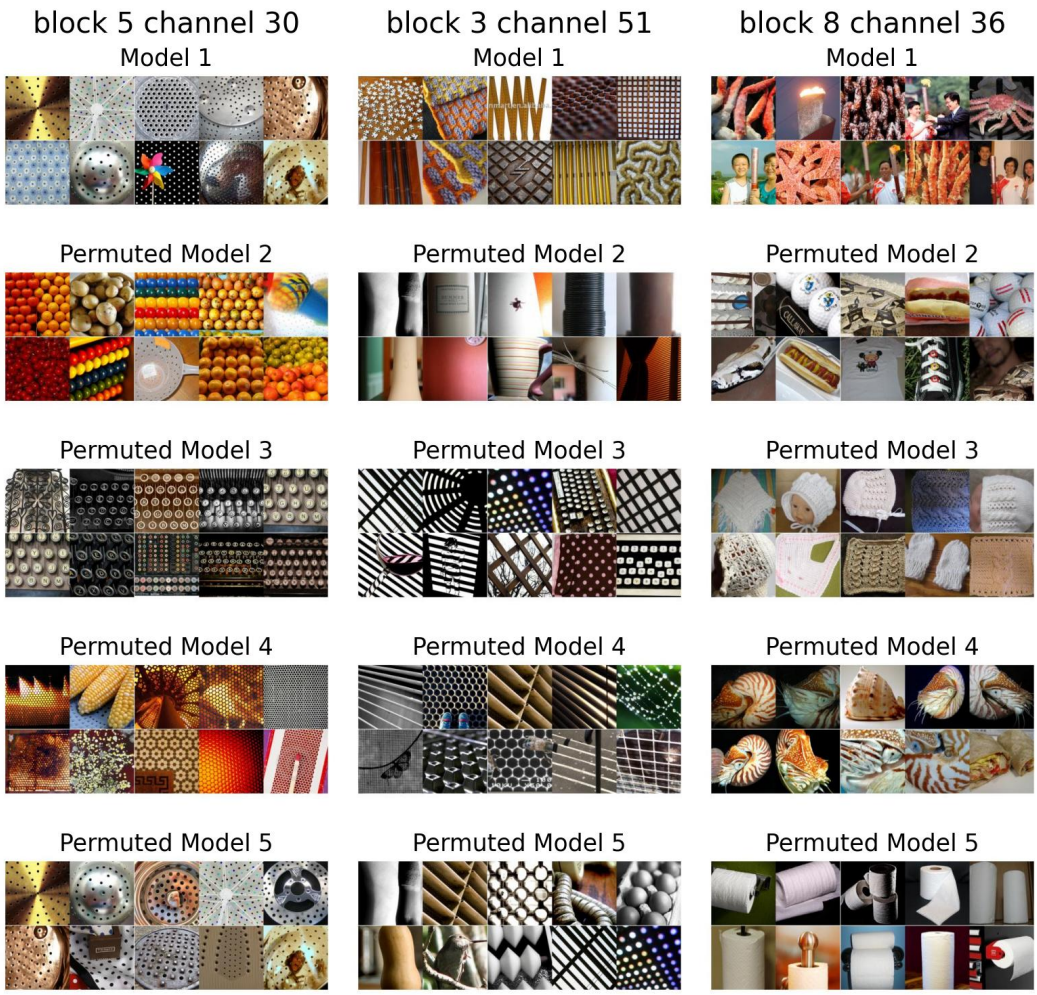

Figure Q: Top activated images from 3 ResNet18 models. Model 2 to 5 are are more aligned to model 1 after permutation. In some cases, we have only some of the models visually aligned to model 1. There are also cases that neither of the 2 models shows FV similarity after permutation.

References

Ainsworth, S. K., Hayase, J., & Srinivasa, S. (2023). *Git re-basin: Merging models modulo permutation symmetries.*

Bereska, L., & Gavves, E. (2024). *Mechanistic interpretability for ai safety – a review.* Retrieved from https://arxiv.org/abs/2404.14082

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., ... Olah, C. (2023). Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. (https://transformer-circuits.pub/2023/monosemantic-features/index.html)

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2021). A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, *6*(1), 25-45. Retrieved from https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12028 doi: https://doi.org/10.1049/cit2.12028

Cunningham, H., Ewart, A., Riggs, L., Huben, R., & Sharkey, L. (2023). *Sparse autoencoders find highly interpretable features in language models.* Retrieved from https://arxiv.org/abs/2309.08600

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

Desai, K., & Johnson, J. (2021). Virtex: Learning visual representations from textual annotations. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 11162–11173).

Draxler, F., Veschgini, K., Salmhofer, M., & Hamprecht, F. A. (2019). *Essentially no barriers in neural network energy landscape.*

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., ... Olah, C. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*. (https://transformer-circuits.pub/2021/framework/index.html)

Entezari, R., Sedghi, H., Saukh, O., & Neyshabur, B. (2022). *The role of permutation invariance in linear mode connectivity of neural networks.*

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009, 01). Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., ... Song, D. (2018). *Robust physical-world attacks on deep learning models.* Retrieved from https://arxiv.org/abs/1707.08945

Frankle, J., Dziugaite, G. K., Roy, D., & Carbin, M. (2020, 07). Linear mode connectivity and the lottery ticket hypothesis. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (Vol. 119, pp. 3259–3269). PMLR.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., ... Wu, J. (2024). Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., & Wilson, A. G. (2018). *Loss surfaces, mode connectivity, and fast ensembling of dnns.*

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). *Generative adversarial networks.* Retrieved from https://arxiv.org/abs/1406.2661

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). *Explaining and harnessing adversarial examples.* Retrieved from https://arxiv.org/abs/1412.6572

Gorton, L. (2024). *The missing curve detectors of inceptionv1: Applying sparse autoencoders to inceptionv1 early vision.* Retrieved from https://arxiv.org/abs/2406.03662

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition.*

Horoi, S., Camacho, A. M. O., Belilovsky, E., & Wolf, G. (2024). *Harmony in diversity: Merging neural networks with canonical correlation analysis.* Retrieved from https://arxiv

.org/abs/2407.05385

Jordan, K., Sedghi, H., Saukh, O., Entezari, R., & Neyshabur, B. (2023). *Repair: Renormalizing permuted activations for interpolation repair.*

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images.*

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc.

Li, Y., Yosinski, J., Clune, J., Lipson, H., & Hopcroft, J. (2016). *Convergent learning: Do different neural networks learn the same representations?*

Long, T., Gao, Q., Xu, L., & Zhou, Z. (2022, 07). A survey on adversarial attacks in computer vision: Taxonomy, visualization and future directions. *Computers & Security*, *121*, 102847. doi: 10.1016/j.cose.2022.102847

Mahendran, A., & Vedaldi, A. (2014). *Understanding deep image representations by inverting them.* Retrieved from https://arxiv.org/abs/1412.0035

Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). *Progress measures for grokking via mechanistic interpretability.* Retrieved from https://arxiv.org/abs/2301.05217

Nanfack, G., Fulleringer, A., Marty, J., Eickenberg, M., & Belilovsky, E. (2023). *Adversarial attacks on the interpretation of neuron activation maximization.*

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). *Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.* Retrieved from https://arxiv.org/abs/1605.09304

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

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, *37*, 3311-3325. Retrieved from https://

api.semanticscholar.org/CorpusID:14208692

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... Sutskever, I. (2021). *Learning transferable visual models from natural language supervision.* Retrieved from https://arxiv.org/abs/2103.00020

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). *Hierarchical text-conditional image generation with clip latents.* Retrieved from https://arxiv.org/abs/2204.06125

Sariyildiz, M. B., Perez, J., & Larlus, D. (2020). Learning visual representations with caption annotations. In *Computer vision–eccv 2020: 16th european conference, glasgow, uk, august 23–28, 2020, proceedings, part viii 16* (pp. 153–170).

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). *Deep inside convolutional networks: Visualising image classification models and saliency maps.*

Stoica, G., Bolya, D., Bjorner, J., Ramesh, P., Hearn, T., & Hoffman, J. (2024). *Zipit! merging models from different tasks without training.* Retrieved from https://arxiv.org/abs/2305.03053

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.* Retrieved from https://arxiv.org/abs/2211.00593

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., ... Schmidt, L. (2022). *Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time.* Retrieved from https://arxiv.org/abs/2203.05482

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). *Understanding neural networks through deep visualization.* Retrieved from https://arxiv.org/abs/1506.06579

Zou, C., Nanfack, G., Horoi, S., & Belilovsky, E. (2024). *Understanding permutation based model merging with feature visualizations.*