"Understanding Robustness Lottery": A Comparative Visual Analysis of Neural Network Pruning Approaches

Zhimin Li, Shusen Liu, Xin Yu, Kailkhura Bhavya, Jie Cao, Diffenderfer James Daniel, Peer-Timo Bremer, Valerio Pascucci

Abstract— Deep learning approaches have provided state-of-the-art performance in many applications by relying on extremely large and heavily overparameterized neural networks. However, such networks have been shown to be very brittle, not generalize well to new uses cases, and are often difficult if not impossible to deploy on resources limited platforms. Model pruning, i.e., reducing the size of the network, is a widely adopted strategy that can lead to more robust and generalizable network – usually orders of magnitude smaller with the same or even improved performance. While there exist many heuristics for model pruning, our understanding of the pruning process remains limited. Empirical studies show that some heuristics improve performance while others can make models more brittle or have other side effects. This work aims to shed light on how different pruning methods alter the network's internal feature representation, and the corresponding impact on model performance. To provide a meaningful comparison and characterization of model feature space, we use three geometric metrics that are decomposed from the common adopted classification loss. With these metrics, we design a visualization system to highlight the impact of pruning on model prediction as well as the latent feature embedding. The proposed tool provides an environment for exploring and studying differences among pruning methods and between pruned and original model. By leveraging our visualization, the ML researchers can not only identify samples that are fragile to model pruning and data corruption but also obtain insights and explanations on how some pruned models achieve superior robustness performance.

Index Terms—neural network pruning, robustness, XAI, information visualization

1 Introduction

Recent developments in deep learning have produced significant advances on a variety of application areas [13, 29, 30]. However, this performance is often achieved through extremely large neural networks consuming substantial resources. Further, these models are difficult to deploy and prone to over-fitting leading to poor generalization and fragile behavior [12]. Network pruning, which removes neurons and/or connections from a model, is a common approach to mitigate some of these challenges as compressing models can reduce both their computational footprint and their inherent redundancy [7,23] without significant performance loss. While model pruning can achieve comparable accuracy as the original dense models, some recent works [11, 19] have demonstrated that the resulting sparse models are brittle to out-of-distribution shifts [2]. For example, common, real-world corruptions can reduce the accuracy of such models by up to 40% for images from ImageNet-C [9]. This degradation of robustness has raised serious concerns on the practical viability of pruned models, especially in safety-critical applications such as autonomous driving.

Recent results [3] have demonstrated both theoretically and empirically that these problems are a byproduct of the pruning methodologies rather than a fundamental limitation of sparse networks. Instead, the *"CARD hypothesis"* [3] theoretically indicates that sparse networks with accuracy and robustness comparable to dense models exist. Furthermore, in some instances, it was empirically demonstrated that pruning can in fact improve both the accuracy and robustness of models compared to their dense baselines. This is especially surprising as making any model, let alone a pruned version, more robust to out-of-distribution shift has proven difficult. Nevertheless, it remains unclear why certain pruning techniques positively or negatively affect robustness. Providing an in-depth understanding will not only support a real-world deployment of such models but might also lead to even more advanced pruning approaches. To date, it is unclear what properties of these models can be attributed to their improved performance and the model pruning community does not have comprehensive introspection tools to answer these important questions. Such an effort is hampered by the opaque nature of neural networks and the lack of a dedicated system for model comparison and evaluation in the context of neural network pruning.

In this paper, we aim to fill this crucial gap by introducing a visual analytical system for understanding differences among representative pruning methods, and measuring and interpreting model behavior under various pruning strategies. As a general goal, we hope to understand the effect of model pruning on multiple levels, e.g., how pruning affects individual samples, how pruning alters a model's internal representation, and how the performance of various models differs for unseen or corrupted data, etc. By utilizing the proposed tool, ML researchers will be able to obtain answers to analytical questions and develop hypothesis for a variety of open questions in the network pruning research, e.g., why some samples [11] are more affected by pruning; why and how certain pruning techniques are better than others; and why certain pruned models [4] can have better robustness performance than a state-of-the-art dense weight trained model.

To achieve the goal of building an comprehensive introspection system for analyzing model pruning, we will make advances on both computation and visualization fronts. On the computation side, one essential challenge arises from the need to compare the latent representations of models to understand how making changes to them affects the final prediction. However, a neural network's feature representation usually lies in a high dimensional space that contains hundreds if not thousands of dimensions without explicit semantic or labels. Comparing such spaces is a non-trivial task, especially considering the behavior of classifier can be sensitive to small changes in the feature representation. Traditional dimensionality reduction methods [34, 38] are not suitable solutions as, for complex latent spaces, they will invariably induce information loss that could significantly impact the trustworthiness of the downstream analysis. A potential solution to this challenge is to preserve all high-dimensional relationships in the data for our comparison task. Since our goal is to understand how latent space changes affect the final prediction, e.g., image classification result, we simply need to understand what aspects of the feature representation directly contribute to the prediction. As long as we can faithfully encode this information, we can achieve a meaningful comparison of the

- *Zhimin Li and Valerio Pascucci are with Scientific Computing and Imaging Institute. E-mail: zhimin,pascucci@sci.utah.edu*
- *Xin Yu and Jie Cao are with University of Utah. E-mail: xin.yu,jcao @utah.edu.*
- *Shusen Liu, Kailkhura Bhavya, Diffenderfer James Daniel, and Peer-Timo Bremer are with Lawrence Livermore National Laboratory. E-mail: liu42, kailkhura1, diffenderfer2, bremer5@llnl.gov.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

high-dimensional feature representations.

In this work, we propose a set of geometrically inspired metrics (namely *Angle*, *Length*, *Margin*) derived from a direct decomposition of classification loss function (i.e., cross entropy). The proposed design allows us to achieve an intuitive comparison of network representations by isolating the most crucial information while removing other variations and noises that do not directly impact the model prediction. Despite the potential of these metrics in providing crucial analytical insights, the success and the usability of the tool hinges on effective visual encoding and interactive design to communicate complex comparison results to users for domain understanding. To tackle the visualization challenge, we introduce two novel visual encodings that help convey the key differences between model's latent representation and their impact on the prediction: *Local Geometry Plot* and *Global Geometry Plot*. The *Local Geometry Plot* provides a direct way to compare and contrast the latent representation geometry for a particular object class by encoding the *Angle* and *Length* metrics in a scatterplot like layout. On the other hand, the *Global Geometry Plot* provides a direct encoding of all object classes via a parallel coordinates (PCPs) like plot where the axis encodes the *Angle* metric (see Section 5). As our target users are ML researchers, we believe a visual encoding design that leverages well-established design (e.g., scatterplot and PCPs) would reduce the learning curve and provide better accessibility. Moreover, since many crucial insights can only be obtained through comparison between different methods, the overall design of the linked interface are centered around the ability to provide constrictive visualization, i.e., visualize the difference, or provide the same view of the two data side-by-side. By utilizing the proposed visual analytic system, researchers can explore where and how pruning methods differ, identify and explain why a subset of samples are vulnerable to model pruning and data perturbation, and provide insights and explanations behind why one model is more robust than another.

Our key contributions are summarized as follows:

- Three geometric metrics for understanding the structure of a neural network's feature representation, evaluating and comparing different model pruning techniques (section 5).
- Novel visual encoding that leverages these geometric metrics for communicating the key differences among feature representations of various models (section 6.3).
- A dedicated introspective visualization system for analyzing and understanding major pruning methods over different architectures and datasets (section 4, 6).
- Extensive case study that involves state-of-the-art models to demonstrate the usability of the proposed visualization system (section 7).

2 Related Work

In this section, we discuss various directions that are related to neural network pruning and visualization approaches, and discuss their relationship with respect to the proposed approach.

Network Pruning In this work, we focus on evaluating and comparing neural network pruning approaches [4, 5, 11, 16, 27, 27], most of which are originated from the ML community. Yann et al. [16] proposed a pruning method based on the assumption that a well optimized neural network model reach to a function's minimum and its second derivative can indicate the important of weights. Frankle and Carbin [5] proposed a lottery ticket hypothesis that a sparse subnetwork with the same initialization can reach similar accuracy of a dense network after training. Ramanujan, et al. [27] and Diffenderfer and Kailkhura [4] showed that an untrained subnetwork can be found within a dense network that has the same performance as a weight trained model. Hooker et al. [11] introduced pruning identified exemplars (PIE) which highlights a subset of samples that are more vulnerable to pruning than the other samples. To provide a more comprehensive understanding of these techniques, we discuss the pruning problem in-depth and explain the difference among popular pruning methods in Section 3.1.

A notable visualization work in this context is CNNPruner [18]. Guan et al. designed a visual analytics system that enables users to interactively perform pruning and explore the trade-off between model accuracy and pruning ratio. However, their motivation and goal arise from the question of how to design a human-in-the-loop interactive pruning system. Instead, we aim to evaluate and understand different pruning methods and their robustness in relationship to model's internal representation. Particularly, our framework provides explanation of why certain samples are more vulnerable to the network pruning and why random untrained subnetworks are surprisingly robust to common corruptions.

Compression/Pruning for Model Explanation In the visualization community, apart from the aforementioned interactive network pruning work [18], the majority of related works on network pruning focus on the model interpretation problem. Junpeng, et al. [36] used model distillation techniques to compress the size of the model and combined it with a deep generative model to understand model's reaction to a sample's neighbor. Summit [10] proposed two aggregation techniques (activation and neuron-influence aggregation) to select critical neurons (e.g., 7.5% weights) to build the attribution graph. With their feature visualization, the system tries to show an overview of the network model. However, the selected attribute graph is a subnetwork and the accuracy of the model is not verified. Liu, et al. [20] designed a visualization system to understand how adversarial examples affect a model's prediction by visualizing the critical subnetwork that preserves the same prediction accuracy of the complete network with these selected samples. The critical subnetwork is auto-selected by an optimization process. Kahng, et al. [14] designed a visualization system ACTIVIS, which used for a model's hidden layer activation pattern exploration. The system designs an activation matrix which shows the top n-th activation neurons at a time to compare the activation of different samples or different categories input. The authors used these most active neurons to select the subnetwork. Rather than explaining the prediction or focus rationale behind individual prediction, our study focus on comparing state-of-the-art model pruning methods in the context of model robustness under common corruptions.

Network Loss Function One key contribution of the proposed work is the introduction of a set of geometric metrics for describing and comparing the latent space of the model. The rationale behind the choice of these metric is that they are directly derived from a decomposition of the loss function, which captures the key structure that is relevant for model prediction. The choice of loss functions has a significant impact on the geometric feature of the latent space in the neural network model, i.e., the common loss function to optimize the neural network for classification task is the cross-entropy loss, which induces streak like structures corresponds to different classes (see section 2). Several works [21, 22] have explored the relationship between the loss function and the corresponding feature induced in latent space. However, these insights are used to highlight the potential issues of the loss function and motivate the design of new loss function. For example, Guo et al. have pointed out that neural network is over-confident under the training with certain loss functions [6]. Wen [37] adds an additional central loss, which forces the same class samples' last-layer feature embedding cluster together, with the cross-entropy loss to jointly optimize the neural network. Liu [21, 22] decouples the dot product behavior of the optimization function and design new optimization functions that optimize the angle of the last layer's feature embedding. In contrast, our work focuses on utilizing geometrically inspired metrics for an in-depth understanding of the relationship between the decomposition of loss function and the structure of latent space to form a basis for comparing the performance differences among different pruning methods.

Model Evaluation and Model Robustness Neural networks have shown superhuman performance on clean test dataset but they fall short on robustness by performing poorer on out-of-distribution data [2]. This brittleness issue is more prominent for pruned models [11], which makes it crucial to evaluate them on common corruptions arising in real world applications. To evaluate the performance on neural networks in the real world, several corruption benchmark datasets have been proposed. Hendrycks and Dietterich [9] developed corruption robustness benchmarking datasets CIFAR-10/100-C, ImageNet-C, and ImageNet-R to facilitate robustness evaluations of CIFAR and ImageNet classification models. Sun, et al. [32] and Mintun, et al. [25] further designed new corruption types to complement [9]. In addition to image classification, benchmarking datasets for object detection and point cloud classification were developed in [24] and [33], respectively. Motivated by the work on corruption robustness benchmarking, in this work we evaluate a range of pruned classifiers on not only on clean test data but also on corrupted datasets.

3 Domain Background

In this section, we discuss the basic terminologies, pruning techniques, and evaluation methods used in this study.

3.1 Network Pruning

Here, we introduce the pruning approaches used in this study for evaluation and analysis. In this paper, we will focus on unstructured pruning which removes redundant network weights. Nevertheless, the analysis pipeline also works for structured pruning which prunes entire neurons or filters. As summarized in [1], most works in network pruning start with scoring the model parameters based on their potential impact on the network performance, selecting weights of least importance to remove from the network, and optionally performing retraining to gain back performance degradation due to pruning.

We will explore the following pruning methods.

- **Random pruning**: Randomly select a set of weights and remove them from a neural network model.

- **Magnitude pruning**: Score the weights with their absolute values and prune the ones with the smallest scores.

- **First order Taylor expansion**: Prune the weights that would have the least impact on the loss function of the model if removed. Given the training data $\mathcal{D}$ and the neural network with its weights ϕ, the impact on the loss function $\mathcal{L}$ after removing a single weight $\boldsymbol{\phi}_m$ can be formulated as

  $$\mathcal{I}_m = |\mathcal{L}(\mathcal{D}, \hat{\boldsymbol{\phi}}) - \mathcal{L}(\mathcal{D}, \boldsymbol{\phi})|, \ \textit{s.t.}\ \hat{\boldsymbol{\phi}}_m = 0.$$

  where $\hat{\boldsymbol{\phi}}$ denotes the weights after pruning. We can estimate $\mathcal{I}_m$ by the functional Taylor expansion as

  $$\begin{aligned}\mathcal{I}_m = |&\nabla\mathcal{L}(\mathcal{D}, \boldsymbol{\phi})(\hat{\boldsymbol{\phi}} - \boldsymbol{\phi}) \\ &+ \frac{1}{2}(\hat{\boldsymbol{\phi}} - \boldsymbol{\phi})^T\nabla^2\mathcal{L}(\mathcal{D}, \boldsymbol{\phi})(\hat{\boldsymbol{\phi}} - \boldsymbol{\phi}) \\ &+ O(\|\hat{\boldsymbol{\phi}} - \boldsymbol{\phi}\|^3)|\end{aligned}$$

  where $\nabla\mathcal{L}$ and $\nabla^2\mathcal{L}$ indicate the first-order gradient of the weights and second-order gradient (Hessian matrix) respectively. Recent works [17, 26] revisit this gradient-based method and assume that (i) the higher than the first order terms are ignored for efficient approximation (ii) removing $\boldsymbol{\phi}_m$ will not affect the remaining weights, so $\hat{\boldsymbol{\phi}}_m = 0, \hat{\boldsymbol{\phi}}_i = \boldsymbol{\phi}_i\ \forall i \neq m$. Hence,

  $$\mathcal{I}_m \approx |\nabla\mathcal{L}(\mathcal{D}, \boldsymbol{\phi}_m)\boldsymbol{\phi}_m|$$

  Although we focus on the above method in our paper, our proposed pipeline also works for the other methods derived from OBS [8] with the higher order Taylor expansion.

- **Multi-prize lottery tickets (MPTs)**: This strategy searches for a performant sparse subnetwork within a randomly initialized network and can further compress the network by applying weight binarization. Counter to the traditional training paradigm of learning the network weights, this approach learns which randomly initialized weights should be retained to improve performance by optimizing over surrogate scores that indicate the importance of each weight to network performance. In our experiments, we make use of biprop (Algorithm 1 in [4]). This methodology is built on the *multi-prize lottery ticket hypothesis* [4] which proposes that sufficiently overparameterized randomly initialized networks contain sparse subnetworks that, without any training, can perform comparably to dense networks and are amenable to weight binarization. Theoretical proofs supporting this hypothesis have been established [4,31]. Further experimental efforts have established that these MPTs learned using biprop are comparably or more robust than dense networks learned using traditional weight optimization in work demonstrating that certain model compression algorithms are capable of producing compact, accurate, and robust deep neural networks, or CARDs [3].

3.2 Model Robustness Evaluation

Besides accuracy on clean test data, we use the cifar10-C dataset [9], which is the cifar10 test dataset corrupted with 19 common corruption types from four categories: noise, blur, weather, and digital corruptions. These corruptions preserve the semantic content of images and humans can recognize these image without trouble. The dataset contains 10000 unique images and each corruption technique has severity-level from one to five where a larger number denotes a more severe corruption. An image has 95 corrupted images and the total size of the dataset is 950000 images.

4 Tasks Analysis

As previously mentioned, existing model pruning schemes exhibit diverse behavior in terms of corruption robustness. Unfortunately, it is not clear why certain pruning techniques affect robustness in one way or another. In this work, we aim to provide an in-depth understanding of this phenomenon which will not only support a real-world deployment of such models but could also lead to more advanced pruning approaches. Specifically, we are interested in answering the following questions.

- Given two models pruned with different pruning techniques, what aspects of the models make their accuracy and robustness differ from each other?

- Which properties cause a model to be significantly more robust than the other?

- Which model has disentangled and better/robust representations for clean vs corrupted data?

- Why certain samples are more vulnerable to the model pruning?

These questions are articulated by domain experts, which are critical for comparing and designing better pruning strategies. To help domain experts improve their understanding of model behavior and answer the above questions, we distill these questions into the following four requirements to drive the design of the visualization system.

- **R1 - Model Performance Comparison Overview**: model pruning experts often perform experiments on multiple architectures, different pruning methods, and various evaluation benchmarks. A succinct presentation of these results can guide domain experts to figure out the pros and cons of different pruning solutions and narrow down their analysis to the most interesting subset for a detailed examination.

- **R2 - Individual Model Behavior Summary**: summary of certain representative model properties over a large number of samples is useful for domain experts to diagnose a model's behavior. It helps domain experts understand what mistakes the model will make and what decisions the model will be confident about.

- **R3 - Latent Space Comparison**: comparing different models' latent space provides domain experts a visual understanding of how models behave after pruning. Such a comparison can provide insights into why a model is more robust than the other and why certain samples are vulnerable to pruning or corruption.

- **R4 - Subset Samples Examination**: model pruning leads to varying impacts on a model's decision on different samples or labels. Sample examination reveals the set of samples that are most vulnerable to the network pruning and guides domain experts to figure out what features cause such vulnerability.

5 A Geometric View of Latent Space

Fulfilling above mentioned requirements is a non-trivial task. In particular, how to display a model's behavior summary over a large amount of data beside prediction accuracy and how to perform latent space comparison are difficult questions to answer. To address these challenges, in this section, we define the class direction, and corresponding three geometric metrics (*angle*, *length*, and *margin*) on a neural network's logit layer's feature encoding, which is the accumulated result of previous layers' transformations and directly used for a model's final prediction.

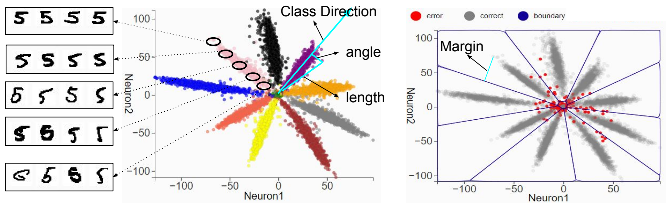


Fig. 1. The feature representation of data samples in the logit layer with 2 neurons can be visualized directly. The star shape is samples' latent space representation of a model trained with the cross entropy loss. The decision boundary in the logit layer can also be approximated and visualized by sampling the 2d space. Four geometric features that are connected with the loss function can be identified in the visualization.

Before getting into the detail description of the geometric features, we present an example to explain the intuition of what the geometric features looks like in the logit layer. Fig. 1 visualizes samples' feature representation of a trained neural network, where the logit layer has only 2 neurons. These 2d feature vectors display a star shape, which samples that are far away from the origin have more distinguishable features and samples that close to the origin are ambiguous with other labels. Class direction is the direction that across the middle of samples that belong to a certain category. *Angle* and *length* metrics are marked in the plot, which is the angle with the class direction and a sample's distance to the origin, respectively. The plot on the right shows the same feature vectors but with the decision boundaries of the classification. These geometric properties not only exist in 2d space but also can be generalize to high dimensional space to help summarize crucial aspects of the latent space structure and form the basis for cross model comparison. At the end of this section, we will discuss the impact of the curse of dimensionality and also how these geometric metrics' correlation with model robustness.

5.1 Class Direction

The loss function used for training the neural network is critical for shaping the geometry of the latent feature embedding. Cross-entropy loss and negative log-likelihood loss are default loss functions used for classification tasks. For a given example x with a ground truth label $y = i$, loss function can be formulated as $L_{loss} = -log(P(y = i|x))$ where $P(y = i|x)$ is the predicted probability for a model for the label $y = i$ with a value range in $[0, 1]$. A large $P(y = i|x)$ will achieve a small loss L_{loss}.

Without loss of generality, we assume that the neural network is composed of two parts: an encoder h that transforms input x into a feature representation vector $\vec{X} = h(x)$ and a classifier c that is used for producing the predicted probability $P(y = i|x) = c(\vec{X})$. We consider that the classifier part c consists of a fully-connected layer, and a softmax activation function, which is a general configuration in convolution neural network models. $\mathcal{C}$ is the number of classes for a classification task and $\vec{X}$ is the m-dimensional feature embedding of $\vec{X} \in \mathbb{R}^m$. The last fully-connected layer in classifier part c parameterized with weight $\boldsymbol{W} \in \mathbb{R}^{m \times \mathcal{C}}$ will take this m-dimensional feature vector as input, and project it into $\mathcal{C}$ scores, then output the predicted probability via soft-max activation function.

In equation (1), we can rewrite the weight matrix $\boldsymbol{W} = \{\vec{W}_i | 0 < i \leq \mathcal{C}, \vec{W}_i \in \mathbb{R}^m\}$ as the set of weights corresponding to each of the $\mathcal{C}$ classes. Hence, the predicted probability $P(y = i|x)$ is determined by the dot products of feature representation $\vec{X}$ and each target label j's neuron weight $\vec{W}_j$.

$$P(y=i|x) = \frac{e^{\vec{W}_i \cdot \vec{X}}}{\sum_{j=0}^{\mathcal{C}} e^{\vec{W}_j \cdot \vec{X}}} = \frac{e^{\|\vec{W}_i\| \|\vec{X}\| \cos\theta_i}}{\sum_{j=0}^{\mathcal{C}} e^{\|\vec{W}_j\| \|\vec{X}\| \cos\theta_j}} \tag{1}$$

$$= \frac{1}{\sum_{j \neq i}^{\mathcal{C}} \frac{e^{\|\vec{W}_j\| \|\vec{X}\| \cos\theta_j}}{e^{\|\vec{W}_i\| \|\vec{X}\| \cos\theta_i}} + 1} \tag{2}$$

$$= \frac{1}{\sum_{j \neq i}^{\mathcal{C}} e^{\|\vec{X}\| (\|\vec{W}_j\| \cos\theta_j - \|\vec{W}_i\| \cos\theta_i)} + 1} \tag{3}$$

$$= \frac{1}{\sum_{j \neq i}^{\mathcal{C}} e^{\|\vec{X}\| (C_j \cos\theta_j - C_i \cos\theta_i)} + 1} \tag{4}$$

The dot product operation (denoted as $\cdot$) can be interpreted as that the feature embedding $\vec{X}$ of every input example being projected onto each label j's $\vec{W}_j$ direction and multiply with $\|\vec{W}_j\|$. Here, $\|\vec{W}_j\|$ is a constant, and the predicted probability is determined by the L2 norm $\|\vec{X}\|$ and the angles θ_j between the directions of $\vec{X}$ and each $\vec{W}_j$. Based on the above intuition, we define the fixed weight vector $\vec{W}_j$ parameterizing the classifier part c as the **class direction** of category j in the network's last layer latent space.

5.2 Angle, Length, and Margin

Previous discussion has defined the class direction which can be extracted from the build-in weights in the neural network model. Based on the above definition, we introduce three geometric features of the last layer latent space - angle, length and margin, and the impact of data corruption on them.

Angle metric is the geometric angle θ between class directions and feature vectors. A small angle between the class direction $\vec{W}_i$ means the sample will be predicted as label i with high probability. To understand why angle metric matters, we can examine the softmax function in formula (1) in which the angle is θ_i. When pruning a well-trained neural network with the classifier part c fixed, $\|\vec{W}_i\|$ or $\|\vec{W}_j\|$ is a constant value which does not get affected by the feature embedding $\vec{X}$ and can be replaced with a constant value C_i or C_j. Since a given prediction shares the same $\|\vec{X}\|$, the angle value θ between class directions and feature vectors is the only variable which determines the prediction result. Assume the angle between feature vectors and the other class directions maintain the same, decreasing the angle θ_i will decrease the value of equation (5) which will increase the probability of equation (4). If a feature vector has similar angles with respect to two class directions then the model will give high probability to both categories. If the model makes an incorrect prediction on a sample, then the feature vector of this sample often has a large angle with the target class direction.

Length metric is the L2 norm of feature vectors which mainly affects the confidence or probability of the prediction. First of all, if the $\|X\|$ is close to zero, then we have the chance classifier $P = \frac{1}{\mathcal{C}}$ where $\mathcal{C}$ is the total number of labels. If the model makes a correct prediction for sample with ground truth label i, then,

$$C_j * \cos(\theta_j) - C_i * \cos(\theta_i) < 0, j \neq i, j \in 1, .., \mathcal{C}.$$

To understand how length affects the prediction, let us assume that the angle between the feature vectors and semantic directions are fixed, and we only adjust the value of length. Referring to equation (4), increasing the $\|X\|$ will increase the probability of prediction i and decrease the loss. If the model makes an error prediction of class k, then,

$$C_k * \cos(\theta_k) - C_i * \cos(\theta_i) > 0, k \neq i, k \in 1, .., \mathcal{C}$$

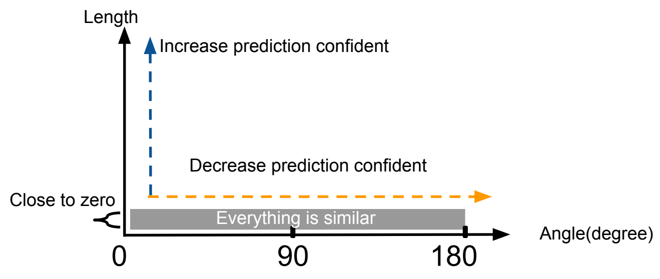


Fig. 2. A summary of how length of feature vector and its angle with the target class direction affect the prediction.

Under this condition, increasing $\|\vec{X}\|$ will decrease the probability of sample belonging to label i and increase the loss. In the later section, we will show that most samples, which are predicted wrong, often have small $\|\vec{X}\|$ value. These feature vectors will have overall small loss if model can not predict them correctly. Overall, how the prediction is affected by length and angle are depicted in Fig. 2. Increasing the angle between the semantic directions will decrease the prediction probability and increasing length will increase the probability. However, if the length is too small the model will not be able to make a proper prediction.

Margin metric is the minimum distance to the decision boundary. A large margin can tolerate severe data corruption and large perturbation in the input sample. In equation (1), a sample with the feature embedding $\vec{X}$ belonging to label i needs to have largest output probability of the model that needs to satisfy

$$\vec{W_i} \cdot \vec{X} - \vec{W_j} \cdot \vec{X} > 0, j \neq i, j \in 1, .., \mathcal{C}.$$

The decision boundary of label i is constructed with $n-1$ hyperplanes $(\vec{W_i} - \vec{W_j}) \cdot \vec{X} = 0, j \neq i, j \in 1, .., \mathcal{C}$. The minimum distance of a feature vector to decision boundary is the minimum distance of the feature vector X to all $n-1$ hyper-planes:

$$min\{\frac{\|(\vec{W_i} - \vec{W_j}) \cdot \vec{X}\|}{\|(\vec{W_i} - \vec{W_j})\|}, \quad j \neq i, \ j \in 1, .., \mathcal{C}\}. \tag{5}$$

Note that if a model makes a wrong prediction on a sample, then the margin value will be multiplied with negative one to indicate the error.

5.3 Correlation Between Geometric Metrics and Corruption Robustness

To attribute the robustness to three geometric features, we perform an experiment to measure the correlation between robustness and geometric properties. Note that the robustness is defined as the average accuracy on corrupted data cifar10c (i.e., applying $c = 19$ corruption types and $s = 5$ severity levels to each clean test image x_i and transforming it to a set of corrupted images $x_i^{c,s}$):

$$robustness = \sum_{i=1}^{N} robustness_i, \tag{6}$$

$$robustness_i = \sum_{c=1}^{19} \sum_{s=1}^{5} \mathcal{I}_{\{y_i = f(x_i^{c,s})\}}, \tag{7}$$

where $\mathcal{I}$ is an indicator function taking value 1 if the prediction is correct on corrupted image and 0 otherwise. Furthermore, $robustness_i$ denotes per sample robustness quantifying the level of invariance to corruptions.

We use three convolution neural network architectures – Alexnet, VGG16, and Resnet18, to evaluate the relationship between these geometric metrics and corruption robustness. The result of our experiment is summarized in Table 1, in which rc-angle, rc-l2, and rc-margin denote the Pearson correlation coefficient between the robustness and the angle, length, and margin, respectively. Overall, angle and margin to the decision boundary show strong correlation with the robustness. However, the length is moderately correlated with robustness on VGG16 and Alexnet, and it is weakly correlated with robustness on Resnet18. The correlation between length and robustness is not consistent across different architectures. On VGG16, the correlation factor is 0.5783 which is significantly different from 0.3401 on Resnet18.

CNN Architecture	rc-angle	rc-l2	rc-margin.
Resnet18	-0.8148	0.3401	0.7794
VGG16	-0.8107	0.5783	0.8234
Alexnet	-0.8572	0.5557	0.7681

Table 1. This table shows the Pearson correlation coefficient between different geometric features and models' robustness. Angle and margin show significant correlation with robustness. The correlation between l2 and robustness is moderate or weak.

5.4 The Effect of the Curse of Dimensionality

High dimensional geometry is affected by the curse of dimensionality [35] such that the intuitions humans have on two or three dimensional spaces may not apply to high dimensional space. For example, the angle between two random vectors in high dimensional space will display counter-intuitive behavior. In Fig. 3, the top plot is the result of an experiment which is performed on vectors generated by uniform random sampling. With the increasing dimension, the mean angle between 10000 uniformly random sample vectors will converge to 90 (orthogonal) and their standard deviation will get small. The orthogonality between two vectors indicates that they do not contain any information about each other.

The phenomenon also affects the high dimensional latent space of well trained neural network models. For an untrained LeNet_5 models, the samples' mean angle and standard deviation with the class direction is 88.8 ± 6.8, which indicates uninformative feature vector. The bottom plot of Fig. 3 displays the average angle and standard deviation of MNIST dataset with their class directions on well trained LeNet_5 models. The x-axis represents dimension which is the number of neurons placed in the last feature layer of LeNet_5 models. As the number of neurons increase, the average angle of samples with the class direction also increase and the standard deviation will become smaller. In the following discussion of this study, it is a rare event that a feature vector can have a small angle (e.g., 0 degree) with its class direction in the popular well trained convolution architectures. Most of the angle between samples and the class direction is above 30 degrees and the angle of samples with unrelated class direction is near 90.

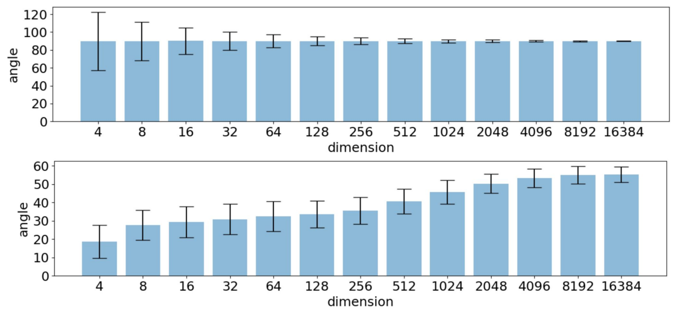


Fig. 3. Top plot shows that with increasing dimensions, the angle between two uniformly sampled random vectors will converge to 90 degrees with smaller standard deviations. The bottom plot shows that increasing the number of neurons in the last feature layer will lead to a large angle between samples and their class direction.

6 System Design

Leveraging the proposed geometric metrics, we introduce several novel visual encoding that incorporate these metrics for pruning evaluation and model comparison. In this section, we discuss the design rational for these views and explain how each of the views work together to address the corresponding domain requirements (see section 4).

6.1 Evaluation Table View

Having an overview of pruning approaches' prediction performance is an essential first step for the pruning analysis (**R1**). This process involves multiple model architectures, pruning approaches, evaluation metrics, and large number of samples' prediction outcome. We aggregate these information into a table format and design the evaluation table view.

In Fig. 4 ①, the visualization shows an evaluation overview of a convolution architecture for two pruning techniques (magnitude and Taylor, see section 3.1). The evaluation is performed to understand the whole test dataset of clean cifar10 and corrupted cifar10c [9]. In the visualization, each histogram represents the evaluation outcomes (i.e., accuracy) on a given corruption type, for models with varying pruned rates, which is defined as the fraction of weights removed by the pruning algorithm. The vertical position of histograms plot encodes the ranking of different pruning methods' performance. For example, the pruning method which generates a models with the best performance over the fog corruption dataset, its evaluation histogram will be placed on the top. As illustrated in 4 ①, when a model is selected, its performance on various corruption scenarios are highlighted. Under the fog background corruption evaluation, the maximum (or best) performance of magnitude pruning method is worse than the Taylor pruning approach. However, on rest of the corruption types, the magnitude pruning has better performance.

During the pruning analysis, domain experts may also be interested in different architectures and pruning techniques' performance on a subset of samples or samples from a specific class. To achieve this requirement, during the downstream analysis, our system enables users to select a subset of samples, and the evaluation table can automatically update the visualization to current samples' result (**R4**). Current visualization also enables the comparative analysis which shows the performance comparison (Fig. 4 ⑤) between current selected samples and the entire test dataset. The length of green bar indicates the amount of performance increase on the selected subset of samples compared to the accuracy of the overall test dataset. For example, the test accuracy of a model is 90% but the model's performance on the selected subset samples is 95%, then the green bar displays the 5% performance improvement. The red bar shows the opposite concept that reveals the the amount of decrease in the performance. Such an operation enables users to examine more details of model behavior and help understand model performance with different levels of granularity.

With the evaluation table, users can also select a model of interest for detailed comparison through other views. The selected combination is displayed in Fig. 4 ②) with detailed description.

6.2 Local Geometry Plot View

Model evaluation table provides a summary of model's overall performance, but we are unable to explore any class specific information. Local Geometry Plot performs class-specific evaluations, which helps in conveying how well samples from a specific class are classified and uncover the cause of potentially poor performance. Visualization in Fig. 4 ③ displays geometric metrics for five classes. The accuracy on the top shows that samples from the cat class have 81.7% accuracy, which is much less than rest of the samples, and the plane class has the best performance with a 95.0% accuracy. The *angle* and *length* metrics are presented as a scatter plot and the density distribution of each metric is placed along side each axis. The density distribution of the margin metric is placed at the top right. With in a single model, these geometric metric values often have similar ranges for each class. Among these points, gray color indicates the correctly classified samples and the red color indicates the incorrectly classified samples. In all five categories, these incorrect samples have larger *angle* and smaller *length* metric than the correct samples, and their *margin* metric is also relatively small.

In additional to exploring a model's latent space geometry and and its per-class performance, local analysis also enables comparison among different models (**R3**). Users can select two models from Fig.4 ② for comparison. Here, we select two models with different pruning ratios over the clean test dataset. Gray color highlights the reference model and yellow color highlights the compared model. The visualization in Fig. 4 ④ shows a class's geometric metrics' distribution shifts and the trajectory of samples' movement. From left to right, the visualization shows (b) the complete trajectory of all samples; (c) the trajectory of samples that are incorrectly classified by both reference model and compared model (purple color); (d) the trajectory of samples that are correctly classified by the reference model but incorrectly classified by the compared model (red); (e) the trajectory of samples that are incorrectly classified by the reference model but correctly classified by the compared model; (f) the samples that are correctly classified by both models (gray). The trajectory provides an intuitive visualization to convey the changes in sample predictions under the pruning operation. The potential clutter caused by large sample size is mitigated through a flexible selection interface. Users can use the "metric difference selection" to select the samples based on the amount the geometric metrics (i.e., *length*, *angle* or *margin*) are affect by the current pruning operation.

6.3 Global Geometry Plot View

The local geometry plot provides a detailed view regarding the behavior of a single class, however, for the comparison task the ability to have a more comprehensive summary is crucial. Global geometry plot gives a geometric overview of how model performs on all classes on the currently selected dataset (**R2**), which shows not only how well samples are classified, but also what other classes the model is confused with.

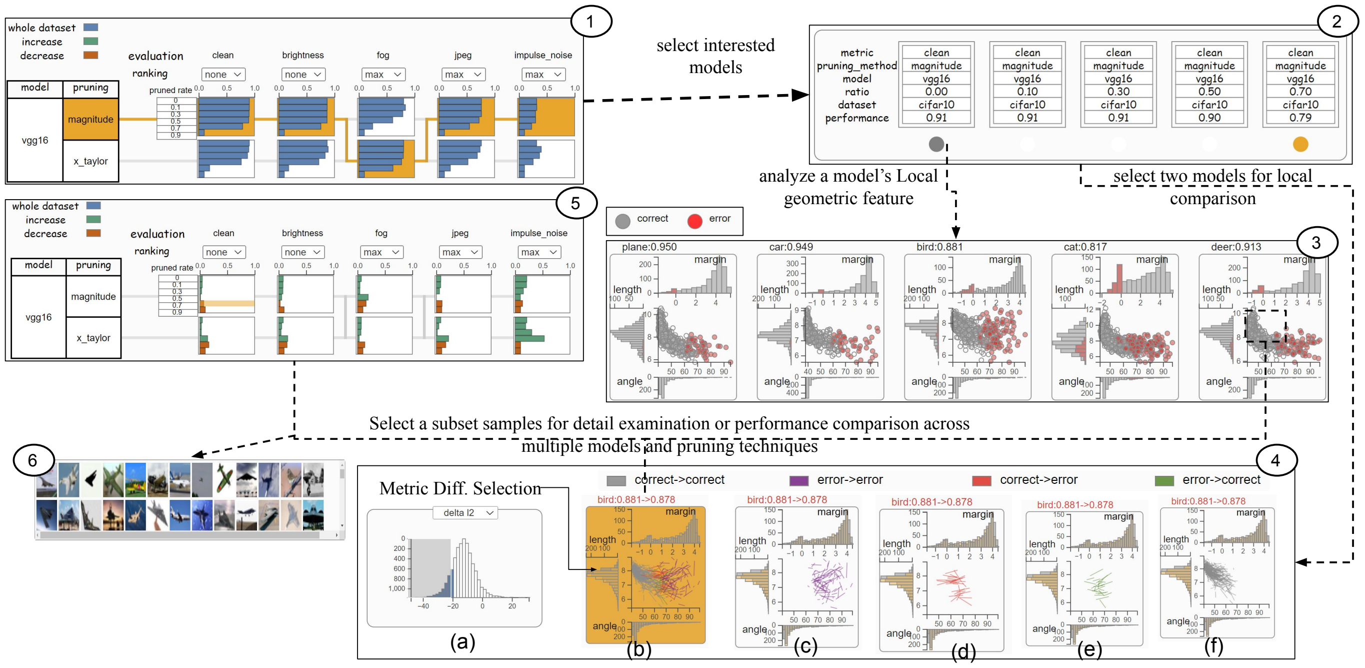
The global geometry plot uses parallel coordinate display in $n + 2$ dimension configuration, in which n is the number of classes (for dataset with large number of classes, a pre-selection can be applied to focus on subset classes to make visual encoding and exploration manageable). The first n dimensions represent the *angle* metric of a sample with respect to each class direction. For example, Fig. 5 ② displays samples belonging to the plane class. Most of the samples have smaller angles with respect to the direction of the plane class in comparison to other classes. However, these plane samples also have relatively small angles with the bird class (Fig. 5 ④) and the ship class (Fig. 5 ⑤), which may lead to incorrect classification. By further examining these samples, we can see similar shared background might be one contributing factor for this mistake.

Similarly with the local geometry plot, global geometry plot also enables the comparison among models and datasets (**R3**). In Fig. 5 ③, the visualization displays the same plane class but with samples that are corrupted with fog noise. Before corruption, plane class is only confused with the ship and the bird classes. However, in the presence of fog corruption, the uncertainty of model's prediction changes and the ambiguity between plane samples and cat samples increases.

The density plot (Fig. 5 ⑥) on the top of each cat class axis shows the distribution shift. Gray color highlights the reference dataset (original) and the yellow color highlights the compared dataset (corrupted). In the visualization, cat class samples' *angle* metrics shift to the smaller values in the presence of the corruption. Moreover, the corrupted dataset as a whole has smaller *margin* and *length* compared to the clean datasets. Our comparative interface allows one to extensively reason about similarities and differences among models pruned with various techniques. A detailed case study is presented in the next section.

7 Use Cases

In this section, we evaluate the usability of our system by demonstrating how the proposed system can provide valuable insights and impactful answers on behaviors of various pruning methods, particularly regarding the robustness of the pruned models to common corruptions.



metric					
pruning_method	clean	clean	clean	clean	clean
model	magnitude	magnitude	magnitude	magnitude	magnitude
ratio	vgg16	vgg16	vgg16	vgg16	vgg16
dataset	0.00	0.10	0.30	0.50	0.70
performance	cifar10	cifar10	cifar10	cifar10	cifar10
	0.91	0.91	0.91	0.90	0.79

Fig. 4. An analytical workflow with a model's local geometric examination. Users can start with selecting a list of interesting combinations, which includes a model, a pruning technique, and a corrupted dataset, based on the evaluation table ①. The detail of the selected combination are displayed in ②. Users can choose a single combination for detailed local geometric analysis over each label in ③ or select two combinations and compare their geometric features differences in ④. During the analysis, a subset selection of the samples during the analysis pipeline can highlight what kind of samples will confuse the model ⑥. Select a label category and examine its perform over all the combination ⑤ compared with the same result of whole cifar-10 testing dataset.

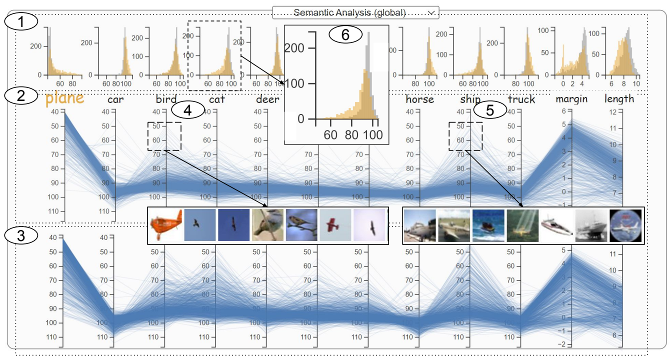
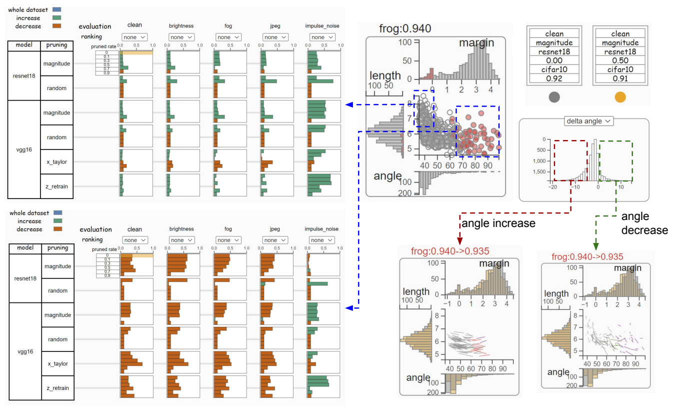


Fig. 5. VGG16 model's behavior over cifar10 plane samples, and the same data but corrupted with fog corruption. In the reference dataset ②, these samples are confused with bird and ship samples. In the fog corrupted version ③, plane samples can also be confused with cat samples. In the corrupted dataset, the l2 norm and margin shift to a small range of values.

7.1 How does pruning impact different samples in different models?

Model evaluation process often aggregates a model's prediction over all test samples, which misses critical information about model's behavior on a specific category or a subset of samples. However, in order to have an in-depth understanding of pruning method's behavior, it is important to understand how various methods affect a subset of samples, and whether they influence these samples differently. In Fig. 6, we demonstrate some of these observations on a magnitude pruned model. We see that the samples that have relative large *length* and small *angle* metric values often show resiliency to the different pruning operations across multiple network models and evaluations. On the other hand, samples with small *length* and large *angle* values display fragile behavior, and these samples can be affected by pruning and data corruptions dramatically.



clean	clean
magnitude	magnitude
resnet18	resnet18
0.00	0.50
cifar10	cifar10
0.92	0.91

Fig. 6. An overview of evaluation comparison across multiple models and pruning techniques of frog samples. The sample with large *length* and small *angle* are less affected by the pruning and different model architecture. They tend to show better performance on different evaluations. On the other hand, the samples with smaller *length* and large *angle* are fragile over different pruning approaches and models. Meanwhile, pruning operation can show both positive and negative impact on samples' geometry.

Model pruning affects different samples in different ways. By comparing two VGG16 models with the same performance accuracy but one of them is pruned 50% with magnitude pruning, we see that some samples decrease their angle with their class direction but the other set of samples increase the angle. In Fig. 6), two local geometric views show the positive and negative impact. Through the use of "metric difference selection" control (see Fig.4④(a)), we can isolate samples with reduced *angle* metric during the pruning process, which indicates increased performance.

7.2 How do different types of pruning schemes affect learned representations?

Model pruning is often evaluated with prediction accuracy on clean test data which does not reveal their impact on model's internal representation, i.e, latent spaces, which may help shed light on their true capability. Here, we demonstrate a use case in which multiple models' latent spaces that are generated by different pruning approaches from the same model architecture are compared in terms of their representation on clean test data.

In Fig. 8, the original model is a VGG16 that trained on cifar10 dataset. After training, the accuracy of the final model is 91%. Here we show samples from plane class and their *angle* with respect to each class direction is displayed in Fig. 8 ①. We can see that the initial dense (or unpruned) model may confuse the plane samples with bird and ship samples. With the first order Taylor pruning, the accuracy of the model drops to 77% and the pruned model (Fig. 8 ②) increases the chance of confusing the plane samples with the bird samples. The probability that plane samples are confused with ship samples slightly decreases, while the chance of confusing with truck samples and deer samples increases. For random pruning, only 10% of weights are removed from the model can damages the performance of the model significantly. The operation decreases the model's (Fig. 8 ③) performance to 73%, and changes the plane samples' *angles* dramatically such that they become closer to the cat's class direction. However, the ambiguity between plane class and cat class is not the main concern in the dense well trained model and the model generated by other pruning approaches. The magnitude pruning generates a model(Fig. 8 ④) with a comparable 79% accuracy through removing 70% of the weights. However, the resulting model amplifies the ambiguity of plane samples with bird and ship samples. The uncertain between plane and frog samples also increase. The next model (Fig. 8 ⑤) is the retrained model after pruning 99% weights of a model, and retraining the model with the original data to recover the loss in accuracy. The *angle* of plane class samples does not completely recover along the plane class direction, and the ambiguity among plane, bird and ship samples still exist in the the new representation. The last model (Fig. 8 ⑥) uses the untrained VGG16 model and pruned with MPTs. The *angle* of the plane samples is relatively small compared to other class direction. Meanwhile, the *angle* with other class directions is close to 90 degrees and they have much small standard deviation compared to other models. From the above comparison, we can tell that *different model pruning methods can impact model's geometry differently and in certain cases amplify the existing mistakes or cause model to make new mistakes*.

To better understand the details of geometric changes after magnitude and MPTs pruning, we take a closer look through the local geometric plots in Fig. 7. The top five plots show the geometric changes of plane samples with different fraction of weight pruned using magnitude pruning. From the visualization, we can see that 10% weight pruning seems not to affect most of the samples' geometry metrics. With 30% weights pruned, we can see a notable shift in samples' geometry metrics, as indicated by the obvious trajectories in the plots. With 50% and 70% weights pruned,the geometric structure of samples have significant change and the predicted label of some samples are flipped. Without retraining, straight forward magnitude pruning only shifts the geometric metric distribution in the negative performance direction (increase *angle*, decrease *length* and *margin*). The bottom five local geometric visualization in Fig. 7 show the geometry metrics changes under different pruning ratio using MPT approach. The pruning operation will have positive impact until the percentage of the pruning reach a certain threshold. After this threshold, further pruning only cause negative impact on model's performance.

Overall, pruning approaches can have a drastically different impacts on a model's latent spaces' geometric structure even if the final pruned models have a similar prediction accuracy. In this case, the first order Taylor pruning method does not significantly transforms the geometric structure but amplifies the ambiguity that exists in the original dense model. However, random pruning and the magnitude pruning not only amplify the ambiguity in the original model but also changes the geometric structure which causes new mistakes that does not exist in

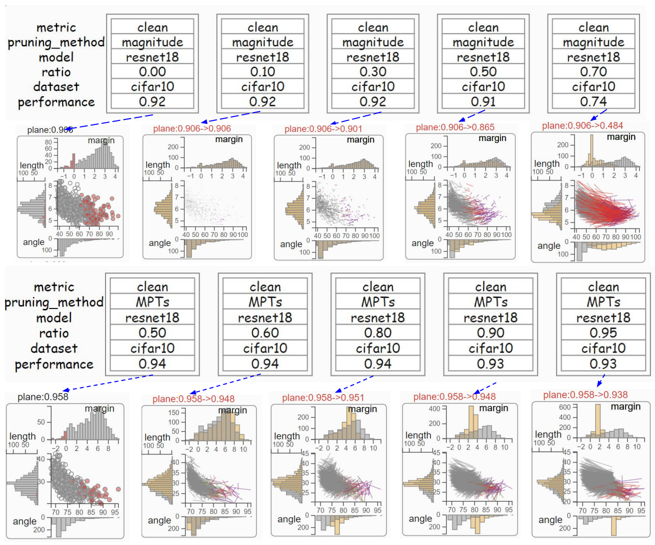


Fig. 7. Magnitude pruning will decrease the geometric features margin, *length* and increase the *angle*. MPTs continue to increase the *margin*, *length* but decrease the *angle*. This trend stops when more than 90% of weights are pruned.

the original dense model. The models generated by MPT approach demonstrate more stable latent space structure than rest of the models and the method may be able to refine the geometric feature of the well trained model. This result is rather surprising for ML researchers, as it shows that *MPT pruning can be used to refine the model geometry as opposed to the popular belief that pruning only maintains or hurts the original model's behavior.* Moreover, the observation in Fig. 8 also raises the question whether the new metrics should be designed to incorporate class prediction uncertainty of individual samples for pruning method evaluation.

7.3 Why are MPT pruned models more robust than other (dense and pruned) models?

MPTs method has been proven [4] to produce compact models, which not only have high prediction accuracy and small model size but also models that are significantly more robust to various data corruptions than regular weight trained models[1]. However, it is still unclear why and how MPT approach achieves such a performance gain. Here, we show that our visualization tool can help develop hypothesis towards answering this question by comparing the latent spaces' geometric structure between the traditional weight trained VGG16 and the VGG16 model generated by MPT. We then carry out a simple quantitative verification of the hypothesis – the insights learned with our analysis can be crucial for future development of robust models.

In Fig. 9, the top panel (Fig. 9 ①,②) shows the geometric difference of samples from truck class with and without the JPGE corruption in the original weight unpruned trained VGG16 model. The model is trained on cifar10 dataset with 200 epochs and the final accuracy on clean test data is 91%. With the same test dataset but corrupted with the JPEG compression, the accuracy of the model drops to 76%. The models are able to distinguish truck images from samples of other classes even though some samples may be slightly confused with car samples (①). Once the dataset is corrupted, the model displays confusion with multiple categories such as plane/ship (②) and models' global geometric features on corrupted data is not as coherent as the clean dataset. The local geometric visualization on the right shows a summary of the geometric feature distribution shift of the truck samples. The overall accuracy drops from 93.6% to 84.0%. The *margin* of samples to the decision boundary, *angle*, and *length* of the samples have a noticeable shift. This pattern is expected as the model has not seen the corrupted data during training.

[1]https://robustbench.github.io/div_cifar10_corruptions_heading

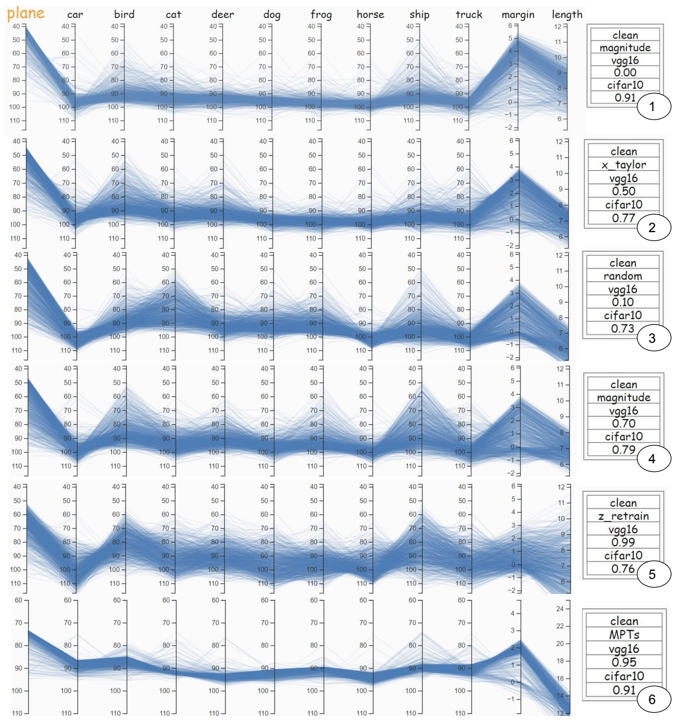


Fig. 8. Different pruning approaches on the same weight trained model end up with similar model accuracy. However, these models can have distinct geometric representation which causes different models making different mistakes.

The bottom panel (Fig. 9 ③,④) illustrates the same comparison, but using the models generated by MPT method, i.e., start with an untrained VGG16 model then 95% of the weights are pruned, resulting a model with a 91% accuracy. The performance of the model declines to 85% when tested on the corruption dataset which is significantly better than the performance of dense weight trained model (76%). For samples belonging to the truck class, the performance of the model only drops slightly from 93.7% to 92.2%. The *angle* distribution of samples with each class direction displays a distinct pattern, i.e., the dense VGG16 has much larger variance but smaller mean *angle*, whereas the MPT model has much a smaller variance but larger mean. For MPTs, the global geometric visualization comparison ③,④ shows that the difference between corrupted and clean data is minor and the relative local geometric visualization reveals that these three geometric distribution have much less distribution shift compared with ①,②.

Such an observation motivates a hypothesis: *geometric of MPTs is less sensitive to different corruption types than that of the regular weight trained models.* To further verify the hypothesis, we compare relative *margin* distribution shift over the cifar10c dataset between two models. The relative *margin* change is defined as $\frac{m_{original}-m_{corrupted}}{m_{original}}$, where $m_{original}$ is the *margin* value of a sample in a model and $m_{corrupted}$ is the *margin* value of the same sample corrupted with different perturbation of corruption techniques in the same model. As mentioned in the previous section, each sample of cifar10 in cifar10c dataset has 95 corrupted samples which lead to 95 relative *margin* changes. In order to reduce the impact of outliers, We remove minimum 0.5% and maximum 0.5% values from each model to perform the comparison. The result is displayed in Fig. 10, where the orange curve is the density distribution of the relative *margin* change from MPTs model and the blue curve is the same distribution from the unpruned weight trained model. The orange curve has relatively smaller *margin* change than the blue curve which further corroborates our hypothesis that geometric features in the MPTs are less sensitive to common data corruptions than the unpruned weight trained VGG16. *This finding has potentially significant implication for robust machine learning as it suggests that to design a robust neural network, one should not only optimize for the*

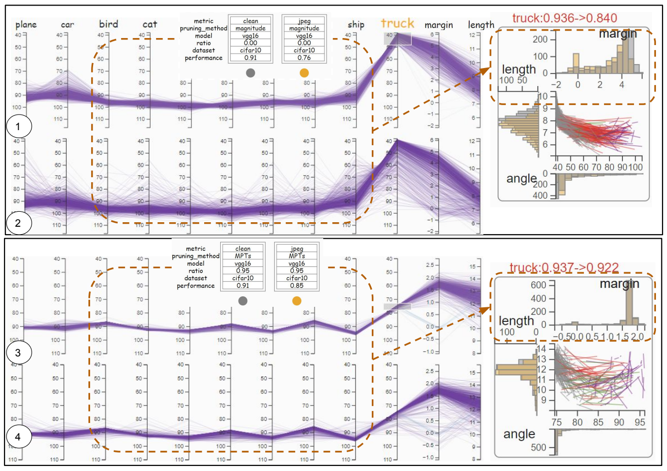


Fig. 9. Top visualization displays a comparison between truck samples of cifar10 on a weight trained VGG16 and the same samples but corrupted with the JPEG compression. The same comparison is performed on the VGG16 which is generated by MPT method. The JPEG corruption causes more angle variations with multiple class directions and minimum distant shift in the weight train VGG16 than the VGG16 generated by the MPTs, which gives insights of why the models generated by MPTs method can be more robust than the regular weight trained model.

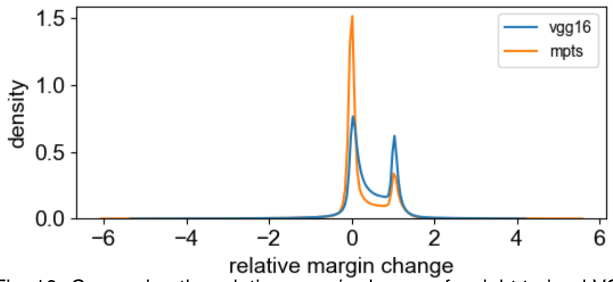


Fig. 10. Comparing the relative *margin* change of weight trained VGG16 and the untrained VGG16 but 95% pruned MPT. Two models have the same prediction accuracy on cifar10 dataset. The MPT model has smaller relative *margin* shift than the regular weight trained VGG16 over all types of corruptions in cifar10c dataset.

train accuracy but also incorporate additional geometric constraints during training.

8 Discussion and Conclusion

In this work, we introduced three geometrically inspired metrics in the neural network latent space for a comparative study of how widely adopted (and state-of-the-art) model pruning approaches impact neural network models' internal representation and performance. The proposed visualization system is able to highlight the key differences between pruning techniques that are unknown to ML researchers. The visualization also provided valuable insights for explaining model robustness from a geometric perspective and answered why certain pruning methods produce surprisingly robust models while others reduce model robustness.

One potential limitation of the current system is the scalability of the parallel coordinate view. As for more complex datasets, such as Imagenet (1000 classes), visualizing all *class direction* at the same is not practical. However, practically speaking, it is important to note that even with a more scalable visual encoding, there is a limit to how many classes a user can meaningfully examine at the same time. Accordingly, a pre-selection, or ranking, can be adopted to greatly mitigate this challenge. In the future, we plan to develop an easy-to-use interface to help users select a subset of the interesting classes with different criteria, e.g., classes mostly influenced by network pruning or data perturbation. Another potential improvement of the current system for future work comes from additional introspection ability, i.e., can we combine other model explanation tools such as saliency map [28] or concept-based explanation [15] with the proposed geometry metric, to better articulate the exact semantics changes induced by pruning.

Acknowledgments

Thanks for the discussion assistant from multiple persons.

References

[1] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[2] S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, and D. Song. Anomalous example detection in deep learning: A survey. *IEEE Access*, 8:132330–132347, 2020.

[3] J. Diffenderfer, B. R. Bartoldson, S. Chaganti, J. Zhang, and B. Kailkhura. A winning hand: Compressing deep networks can improve out-of-distribution robustness. *CoRR*, abs/2106.09129, 2021.

[4] J. Diffenderfer and B. Kailkhura. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network, 2021.

[5] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[8] B. Hassibi and D. Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[9] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[10] F. Hohman, H. Park, C. Robinson, and D. H. P. Chau. S ummit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics*, 26(1):1096–1106, 2019.

[11] S. Hooker, A. Courville, G. Clark, Y. Dauphin, and A. Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019.

[12] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[13] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[14] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. A cti v is: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97, 2017.

[15] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

[16] Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[17] N. Lee, T. Ajanthan, and P. H. Torr. Snip: Single-shot network pruning based on connection sensitivity. *ICLR*, 2019.

[18] G. Li, J. Wang, H.-W. Shen, K. Chen, G. Shan, and Z. Lu. Cnnpruner: Pruning convolutional neural networks with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1364–1373, 2020.

[19] L. Liebenwein, C. Baykal, B. Carter, D. Gifford, and D. Rus. Lost in pruning: The effects of pruning neural networks beyond test accuracy. *Proceedings of Machine Learning and Systems*, 3:93–138, 2021.

[20] M. Liu, S. Liu, H. Su, K. Cao, and J. Zhu. Analyzing the noise robustness of deep neural networks. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 60–71, 2018. doi: 10.1109/VAST.2018.8802509

[21] W. Liu, Z. Liu, Z. Yu, B. Dai, R. Lin, Y. Wang, J. M. Rehg, and L. Song. Decoupled networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2771–2779, 2018.

[22] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.

[23] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

[24] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[25] E. Mintun, A. Kirillov, and S. Xie. On interaction between augmentations and corruptions in natural corruption robustness. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In *CVPR*, pp. 11264–11272, 2019.

[27] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11893–11902, 2020.

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

[29] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[30] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[31] K. Sreenivasan, S. Rajput, J.-Y. Sohn, and D. Papailiopoulos. Finding everything within random binary networks. *arXiv preprint arXiv:2110.08996*, 2021.

[32] J. Sun, A. Mehra, B. Kailkhura, P.-Y. Chen, D. Hendrycks, J. Hamm, and Z. M. Mao. Certified adversarial defenses meet out-of-distribution corruptions: Benchmarking robustness and simple baselines. *arXiv preprint arXiv:2112.00659*, 2021.

[33] J. Sun, Q. Zhang, B. Kailkhura, Z. Yu, C. Xiao, and Z. M. Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022.

[34] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[35] M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pp. 758–770. Springer, 2005.

[36] J. Wang, L. Gou, W. Zhang, H. Yang, and H.-W. Shen. Deepvid: Deep visual interpretation and diagnosis for image classifiers via knowledge distillation. *IEEE transactions on visualization and computer graphics*, 25(6):2168–2180, 2019.

[37] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pp. 499–515. Springer, 2016.

[38] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.