

Interpretable Diversity Analysis: Visualizing Feature Representations in Low-Cost Ensembles

Tim Whitaker

*Department of Computer Science
Colorado State University
Fort Collins, CO, USA
timothy.whitaker@colostate.edu*

Darrell Whitley

*Department of Computer Science
Colorado State University
Fort Collins, CO, USA
whitley@cs.colostate.edu*

Abstract—Diversity is an important consideration in the construction of robust neural network ensembles. A collection of well trained models will generalize better if they are diverse in the patterns they respond to and the predictions they make. Diversity is especially important for low-cost ensemble methods because members often share network structure in order to avoid training several independent models from scratch. Diversity is traditionally analyzed by measuring differences between the outputs of models. However, this gives little insight into how knowledge representations differ between ensemble members. This paper introduces several interpretability methods that can be used to qualitatively analyze diversity. We demonstrate these techniques by comparing the diversity of feature representations between child networks using two low-cost ensemble algorithms, Snapshot Ensembles and Prune and Tune Ensembles. We use the same pre-trained parent network as a starting point for both methods which allows us to explore how feature representations evolve over time. This approach to diversity analysis can lead to valuable insights and new perspectives for how we measure and promote diversity in ensemble methods.

Index Terms—Neural Networks, Machine Learning, Ensemble Methods, Diversity, Interpretability, Feature Visualization

I. INTRODUCTION

Ensemble learning has long been known to be an effective technique for improving generalization on machine learning tasks. Instead of making predictions with a single model, ensembles train multiple independent models and combine their predictions together. The combination of several models can help to reduce bias or variance that a single model might have. Diversity is an important topic in ensemble learning as it is a critical component in building robust ensembles that generalize well [10], [19].

As neural networks and datasets have become larger, efficient algorithms have been introduced to reduce the computational cost of training several models from scratch. In general, this is done by sharing information between ensemble members in some way. For example, Snapshot Ensembles train a single model and save checkpoints of the model throughout time [8]. Prune and Tune Ensembles train a single parent model and then spawn independent child networks by pruning random parameters from the parent followed by fast retraining [23]. Diversity becomes especially important for these low-cost ensemble methods as information sharing invariably leads to higher correlation between predictions.

Many metrics have been introduced to measure diversity in classifier ensembles [10]. Most of these metrics involve analyzing the differences between output predictions of different members. While this approach enables easy comparisons between classifiers, output differences can vary greatly depending on the accuracy of individual members. For example, a group of random models can be highly diverse but have terrible performance and a group of perfectly accurate models can have low diversity but perform well. Because of this, output diversity metrics can be misleading when comparing the results of different ensemble methods.

Meanwhile, neural network interpretability methods continue to grow in popularity and importance in machine learning research [25]. This field is shifting the preconceived notions that deep neural networks act as black box models. By utilizing feature visualization and saliency attribution techniques, it becomes possible to explore *how*, *what*, and *why* neural networks make decisions by visualizing the types of patterns that specific neurons within the network respond to.

These methods are especially critical for evaluating low-cost ensemble methods that share network structure, because we can meaningfully compare feature representations of the same neuron in two different networks and visualize how they evolve over time. We explore this idea with Snapshot Ensembles and Prune and Tune Ensembles by using the same pre-trained Inception-V3 network as a starting point for both methods [8], [23]. We generate child networks for each ensemble by training on ImageNet and we then create feature visualizations for each neuron in each of the child networks. We quantify the representational diversity by measuring the similarity between the visualizations using well established perceptual hashing algorithms [2].

Our results indicate that Prune and Tune Ensembles display significantly more representational diversity than Snapshot Ensembles for each of the five benchmark perceptual hashing algorithms we used to measure similarity. The unique network topology of Prune and Tune children encourages each ensemble member to learn unique patterns. This insight could be highly effective for building more robust ensemble techniques in the future as our experiments offer a new perspective for analyzing diversity. The application of interpretability techniques can be invaluable for better understanding how to measure and promote diversity in low-cost ensembles.

II. BACKGROUND

A. Low-Cost Ensemble Learning

Several methods have been introduced to reduce the costs of training ensembles, which generally share either network structure, gradient information, or learned parameters among ensemble members. This can be effective at reducing computational cost, as each network can leverage the knowledge learned from other members without needing to be trained from scratch. However this invariably reduces diversity between members and can have significant impact on generalization performance [10]. In this paper, we explore diversity using two recent low-cost ensemble learning algorithms, Prune and Tune Ensembles and Snapshot Ensembles.

B. Prune and Tune Ensembles

Prune and Tune Ensembling (PAT) is an efficient low-cost ensemble learning method that leverages ideas from evolutionary and temporal ensembles in order to efficiently create accurate and diverse networks through pruning.

Prune and Tune Ensembles work by first training a single parent network and then spawning child networks by cloning and dramatically pruning the parent using random or anti-random sampling strategies. Each of the child networks are then fine tuned with a cyclic learning rate schedule for a small number of epochs. Pruning and tuning a previously trained network has an extremely low computational cost. Because the parent is already optimized, the child networks converge with only a few epochs of additional training. One can easily create a parent and dynamically generate many diverse child networks at a cost that is only slightly more than training a single network.

As the child networks are all derived from an identical parent network and tuned for a small number of epochs, one would assume that the children would be highly correlated. However, results have shown that Prune and Tune Ensembles produce extremely accurate ensembles with good output diversity. We aim to better explore the feature representations of these child networks in order to understand how sparse network structure affects diversity.

Anti-random pruning was introduced in Prune and Tune Ensembles as a technique for encouraging more diversity among pairs of child networks [13], [23]. Anti-random pruning creates mirrored pairs of child networks, such that whenever we randomly prune the parent to create a child, a sibling is created that inherits the opposite set of parameters.

Consider a binary bit string $M = \{x_0, \dots, x_n : x \in \{0, 1\}\}$, that is randomly generated with 50% sparsity where 1 represents parameters that are kept and 0 represents parameters that are pruned. The anti-random network then is created by reversing the polarity of all the bits in the mask M , such that:

$$\hat{\theta}_1 = \theta \circ M \quad \text{and} \quad \hat{\theta}_2 = \theta \circ (1 - M) \quad (1)$$

where $\hat{\theta}_i$ are the parameters of the child network, θ are the parameters of the parent network and \circ denotes the Hadamard product.

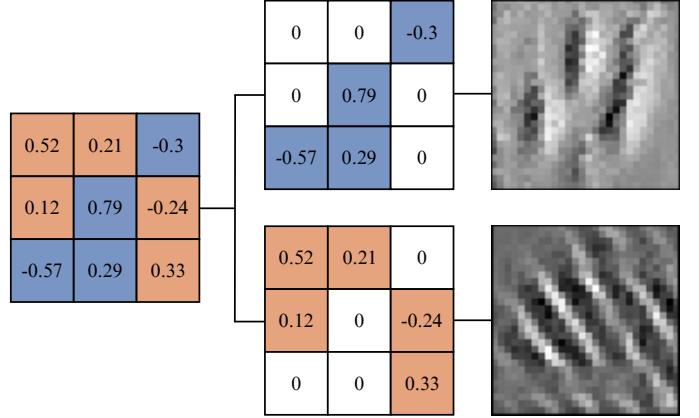


Fig. 1: A parent filter split into two anti-random child filters. The resulting networks learn diverse feature representations as a result of their unique topologies.

The resulting two child networks maximize the Cartesian Distance, CD , between the two binary bit masks a and b , where $a = M$ and $b = 1 - M$.

$$CD(a, b) = \sqrt{|a_1 - b_1| + \dots + |a_n - b_n|} \quad (2)$$

This process is repeated N times to create an ensemble of size $2N$, where each child network has exactly 50% sparsity.

In order to recover accuracy after pruning, each child undergoes a small tuning phase using a cyclic one-cycle learning rate schedule. One-cycle tuning is a two phase schedule that inversely cycles between growth and decay for both learning rate and momentum [18]. The first phase grows the learning rate from η_{min} to η_{max} , while momentum decays from μ_{max} to μ_{min} . The cycle then flips to decay the learning rate from η_{max} to η_{min} and the momentum from μ_{min} to μ_{max} . Both learning rate and momentum are updated using cosine annealing.

$$F(t) = \alpha_0 + \frac{1}{2}(\alpha_1 - \alpha_0)(1 + \cos(\frac{t}{t_{max}}\pi)) \quad (3)$$

where $F(t)$ is the value at iteration t , t_{max} is the total number of iterations, α_0 is the initial value we anneal from and α_1 is the final value we anneal to.

C. Snapshot Ensembles

Snapshot Ensembling is a popular low-cost ensemble learning approach that works by training only a single model and saving checkpoints of that model's state over time.

Training is broken up into several cycles, where for each cycle the learning rate is decayed from a large value to a small value. The large rates encourage the model to move further in the parameter space before converging to a local optima which is then saved [8]. This process then repeats, with the learning rate resetting to the large value at the beginning of each new cycle.

The total number of ensemble members that are created for the final ensemble is governed by the cycle length and total training budget. The cycle lengths need to be long enough

Table I: Comparison between Low Cost Ensembles using WideResNet-28-10 Architecture. M refers to the total number of models in the final ensemble. Besides Prune and Tune (PAT) and Snapshot Ensembles, other methods include: Dropout [7], TreeNet [11], Batch Ensembles [22], Fast Geometric Ensembles [5], and Multiple-Input Multiple-Output (MIMO) [6]. * indicates published results taken from [6], [12], [23].

Method	CIFAR-10			CIFAR-100		
	ACC	NLL	ECE	ACC	NLL	ECE
Dropout*	95.9	0.15	0.024	79.6	0.83	0.05
Treenet (M=3)*	95.9	0.25	0.018	80.8	0.77	0.05
Batch (M=4)*	96.2	0.14	0.02	81.5	0.74	0.05
SnapShot (M=5)	96.27	0.13	0.02	82.1	0.66	0.04
FGE (M=12)	96.35	0.13	0.02	82.3	0.65	0.04
MIMO (M=3)*	96.4	0.13	0.01	82.0	0.69	0.02
PAT (M=6)	96.48	0.11	0.005	82.7	0.63	0.01

to allow for single models to optimize well, however this can limit the potential number of ensemble members created. Within each cycle, the learning rate is decayed with cosine annealing, and the learning rate at any given iteration can be described with:

$$a(t) = F(\text{mod}(t - 1, \lceil T/M \rceil)) \quad (4)$$

where $a(t)$ is the learning rate at iteration t , $F()$ is the cosine annealing function, T is the total number of iterations, and M is the cycle length used for each model.

Snapshot Ensembles are able to create ensembles with good performance without requiring any more computation than a single model would have. However, snapshots taken early in the training process tend to perform worse and snapshots taken later tend to be highly correlated and less diverse, thus resulting in less effective ensembles.

To provide context on the relative performance of these approaches, we include Table 1 which presents results comparing Accuracy (ACC), Negative Log Likelihood (NLL) and Expected Calibration Error (ECE) for both Prune and Tune Ensembles and Snapshot Ensembles as well as several other recent and low cost ensemble methods on CIFAR-10 and CIFAR-100. All methods are trained with roughly the same computational budget and use a standardized WideResNet-28-10 convolutional model. Prune and Tune Ensembles outperform all other methods with an accuracy of 96.48 on CIFAR-10 and 82.7 on CIFAR-100 when accounting for a fixed training budget while Snapshot Ensembles result in accuracies of 96.27 and 82.1 respectively.

D. Output Diversity

Diversity has long been known to be an important consideration in ensemble learning [1], [10], [21]. This is often explained in ensemble literature with an example of the bias-variance decomposition of the mean squared error (MSE) [1], [21]. Given a model's prediction f and a true target value from an unknown test distribution y , the mean squared error is defined to be the expectation of the squared distance between

the models predictions and the true target distribution. Given an ensemble of M equally weighted estimators, the decomposition is extended to produce the bias-variance-covariance decomposition [1].

$$\overline{\text{bias}} = \frac{1}{M} \sum_i (E[f_i] - y) \quad (5)$$

$$\overline{\text{var}} = \frac{1}{M} \sum_i E[(f_i - E[f_i])^2] \quad (6)$$

$$\overline{\text{covar}} = \frac{1}{M(M-1)} \sum_i \sum_{j \neq i} E[(f_i - E[f_i])(f_j - E[f_j])] \quad (7)$$

$$\text{MSE} = \overline{\text{bias}^2} + \frac{1}{M} \overline{\text{var}} + (1 - \frac{1}{M}) \overline{\text{covar}} \quad (8)$$

Generalization error for single models relies upon the optimization of both bias and variance, where the tuning of a model towards high bias can cause it to miss important features and the tuning of a model toward high variance can cause it to be highly sensitive to noise. When the decomposition is extended to an ensemble, the generalization performance additionally depends on the covariance between models. Ideally, ensemble methods that prioritize diversity will be able to reduce covariance without increasing the bias or variance of individual models [1], [21].

In a classification context, there is no standardized analog to the bias-variance decomposition [1]. Instead, several metrics have been introduced as a means to quantify diversity in classification ensembles [10]. The most popular of which are the Kullback-Leibler Divergence and Prediction Disagreement Ratio [4], [6], [12].

Kullback-Leibler Divergence, also known as relative entropy, approximately measures how different one probability distribution is from one another. This operates on the output probabilities of each ensemble member and the average is measured over all pairwise combinations.

$$d_{KL}(f_1, f_2) = \frac{1}{N} \sum_{i=1}^N f_1(x_i) \log \left(\frac{f_1(x_i)}{f_2(x_i)} \right) \quad (9)$$

where N is the number of test samples and $f_i(x_i)$ is the output probabilities for a given model f_i and test sample x_i .

Prediction Disagreement Ratio (PDR) instead measures the differences between only the predicted class instead of differences between the full output distributions.

$$d_{PDR}(f_1, f_2) = \frac{1}{N} \sum_{i=1}^N \text{argmax}(f_1(x_i)) \neq \text{argmax}(f_2(x_i)) \quad (10)$$

where N is the number of test samples and $\text{argmax}(f_i(x_i))$ is the predicted class label for model f_i and test sample x_i .

Table II presents results from several low-cost ensemble papers on a benchmark CIFAR-10 experiment with a total training budget of 200 epochs [6], [12], [23]. The results indicate that output diversity metrics do not directly correlate

Table II: Prediction Disagreement Ratio (PDR) and KL divergence between ensemble members on CIFAR-10 with WideResNet-28x10. Results reported from [6], [12], [23]

Methods	$d_{PDR} \uparrow$	$d_{KL} \uparrow$	Acc \uparrow
Treenet	0.010	0.010	95.9
BatchEnsemble	0.014	0.020	96.2
EDST Ensemble	0.026	0.057	96.4
MIMO	0.032	0.081	96.4
Dense Ensemble	0.032	0.086	96.6
Snapshot Ensemble	0.034	0.068	96.3
DST Ensemble	0.035	0.095	96.4
Prune and Tune Ensemble	0.036	0.090	96.5

with the most accurate ensembles. For example, Dense Ensembles have a prediction disagreement ratio or 0.032 and a KL-divergence of 0.086 with an accuracy of 96.6. These diversity metrics are lower than Prune and Tune (PAT) Ensembles and Dynamic Sparsity Training (DST) Ensembles despite having a higher accuracy. It's clear that diversity metrics alone are not sufficient to fully analyze the generalization capabilities of different methods.

III. INTERPRETABLE DIVERSITY ANALYSIS

The efficacy of low-cost methods often rely on a trade-off between training cost, model accuracy, and member diversity. However, the current state of diversity analysis focuses heavily on output diversity, which gives little insight into how networks actually develop and represent diversity.

This chapter explores diversity by visualizing the feature spaces of child networks generated in Prune and Tune Ensembles and compares them to checkpoint models in Snapshot Ensembles. These networks offer an excellent foundation for exploring diversity for a number of reasons.

1) Prune and Tune Ensembles demonstrate excellent accuracy and output diversity, despite the fact that each child network inherits parameters from a shared parent and is tuned for only a small number of epochs. Visualizing the feature representations of these models can provide an explanation as to how and why these methods produce robust ensembles.

2) Child networks in Prune and Tune Ensembles and Snapshot Ensembles are derived from an identical parent. We can look at the index of the same neuron in two different networks or checkpoints and visualize how feature representations diverge and change over time and space.

3) Sparsity has long been an important topic in deep neural network research. Trained networks can have a significant number of parameters removed and continue to maintain, or even exceed the original network's accuracy with little additional training. Visualizing how sparsity affects the underlying knowledge representations can be useful for better understanding the behavior of deep neural networks.

A. Feature Visualization

Feature visualization uses optimization to create images that maximize or minimize activations of specific parts of the network. This is done by first initializing an image with

random noise. This image is then fed into a fixed network and gradient descent is used to update the pixel values of the input image. The resulting images display the types of patterns a particular neuron or channel responds strongly to.

Various forms of regularization have been shown to improve image quality, including frequency penalization, transformation robustness, and learned priors (such as color distribution) [14]. Additionally, images are preconditioned by decorrelating and whitening the input such that gradient descent happens in Fourier Space, with frequencies scaled to have the same energy [14]. The 2D Fourier Transform for an x -by- y image X is defined as:

$$\mathcal{F}(u, v) = \sum_{i=0}^x \sum_{j=0}^y e^{-2\pi i/x} e^{-2\pi j/y} X_{i,j} \quad (11)$$

We then aim to maximize the output activations of a specific convolutional filter by optimizing the following objective:

$$F(X) = - \sum_{x,y} h_{n,x,y,z}(X) \quad (12)$$

where h is the output of a neuron, X is the input image, n is the layer, z is the channel, and x and y are the spatial positions of the neuron within a channel. In order to generate an image that minimizes neuron activation, the sign is flipped for the objective. The image is then optimized using a standard gradient descent based optimizer:

$$X_{n+1} = X_n - \eta \nabla F(X_n) \quad (13)$$

where η is the learning rate and ∇ is the gradient of the objective function with respect to the input image.

B. Saliency Maps

Saliency Maps visualize the gradient of a prediction with respect to the input image, highlighting the parts of the input that a network responds strongly to. This is done by first performing a forward pass through the network with a given input image. The saliency is computed by setting all outputs of the non-predicted class to 0 and back propagating the predicted class score back through to the inputs [16]. The resulting gradient is then normalized and plotted using a heatmap, where larger values correspond to more salient parts of the input space.

$$G(X) = \frac{\partial Y_c}{\partial X} \quad (14)$$

where the saliency map $G(X)$ is the gradient of the output of the predicted class label Y_c with respect to the input image X .

SmoothGrad was later introduced as an extension to the method described above that reduces noise in the visualizations by averaging the gradient over several Gaussian perturbed inputs [17].

$$SG(X) = \frac{1}{N} \sum_{i=1}^N \frac{\partial Y_c}{\partial X + \gamma_i} \quad (15)$$

where $SG(X)$ is the SmoothGrad output for an input image X and Y_c is the output of the predicted class label c . The input

Snapshot Ensemble										Prune and Tune Ensemble											
Parent																					
Child 1																					
Child 2																					
Metric																					
A-Hash	28	36	27	31	28	40	37	24	39	30	P-Hash	32	34	34	36	38	36	30	34		
D-Hash	26	29	33	29	34	36	31	38	34	39	W-Hash	26	30	28	34	30	28	26	38	34	
C-Hash	23	38	20	16	28	26	57	44	33	49	Mean	27.0	33.4	28.4	27.6	29.6	35.2	38.2	33.6	34.8	37.2
Mean	27.0	33.4	28.4	27.6	29.6	35.2	38.2	33.6	34.8	37.2											

Table III: Feature visualizations of a random selection of neurons in child networks from a Snapshot Ensemble and a Prune and Tune Ensemble. Values reported are the Hamming distances between the two image hashes for each child network visualization. Bold values represent greater distance between representations. The sparsity from Prune and Tune Ensembles results in much more diverse feature representations than two subsequent checkpoints in Snapshot Ensembles.

X is perturbed with Gaussian noise sampled from a normal distribution $\gamma_i \sim \mathcal{N}(0, \sigma^2)$.

C. Quantifying Visual Diversity

The most common image similarity metrics are generally used to measure image quality as a response to noise, corruption, or compression against a ground truth image. These include the mean squared error, peak signal-to-noise ratio, structural similarity index, spatial correlation coefficient, spectral angle mapper, and universal image quality index. These metrics tend to be highly sensitive to minute differences and noisy when applied to images that contain substantially different content.

We instead introduce a robust image hash similarity based approach to quantifying diversity. Image hashing algorithms compress images into binary bit strings such that images that are visually similar will result in hashes that are similar. The bits of the image hash are spatially significant, so the similarity between two hashes can be described by the Hamming Distance, which is the number of positions in which the bits of two binary strings differ. Images that are more similar will therefore have a lower Hamming Distance than images that are diverse [24].

We use several popular image hashing algorithms for measuring the distances between feature visualizations, including: average hash, perceptual hash, difference hash, wavelet hash, and color hash [2]. Average hash, difference hash, wavelet hash and perceptual hash all start by resizing a given image to

an 8x8 pixel square and converting it to grayscale. The hash is then constructed by assigning a 1 or 0 for each pixel in the block according to some heuristic. Average Hash assigns a 1 if the pixel value is greater than the mean pixel value. Perceptual Hash does the same except all values are converted to the frequency domain using a discrete cosine transformation first. Wavelet hash uses a discrete wavelet transform instead of a discrete cosine transform. Difference hash can be described as a gradient hash where the difference between pixel values is compared to the mean difference between pixel values. Finally, color hash maintains color information by skipping the grayscale step and instead using the hue, saturation, and value space of an image separately.

IV. EXPERIMENTS

We conduct a large scale comparison between child networks in a Snapshot Ensemble and a Prune and Tune Ensemble. We start with an open source Inception network pre-trained on ImageNet [15]. Following the hyperparameters given in the Snapshot Ensemble paper, we continue training of the pre-trained network for two cycles of 40 epochs in length with a cyclic cosine-annealing decay schedule with an initial learning rate of 0.1 and a final learning rate of 1e-5 [8].

We use the same pre-trained network as the parent for the Prune and Tune Ensemble as well. Two child networks are created using anti-random pruning with an unstructured sparsity target of 50%. Each child is fine tuned for 40 epochs using a one-cycle learning rate schedule with a max learning

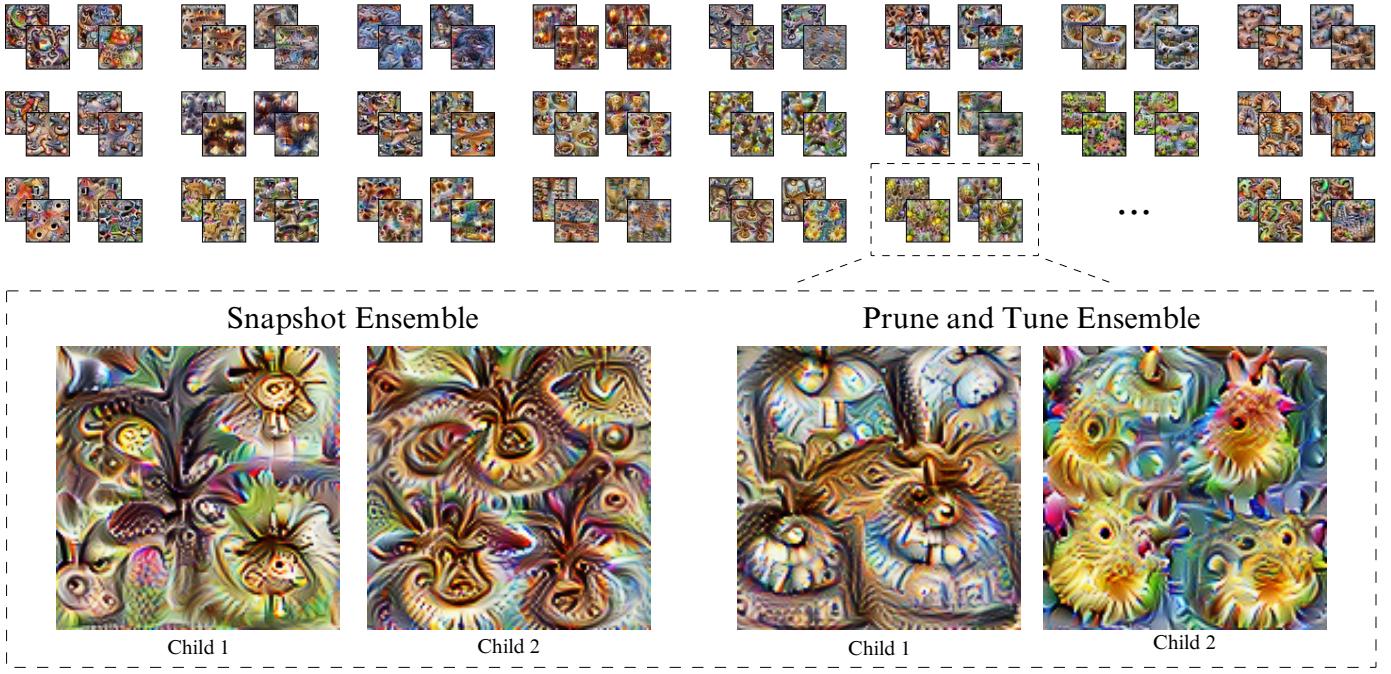


Fig. 2: A sample of 1024 feature visualizations from the final convolutional layer in both Snapshot and Prune and Tune Ensembles. Each cluster of four visualizations are from the same neuron location in each of the different Inception child networks derived from the same shared parent network.

Method	Average Hash	Perceptual Hash	Difference Hash	Wavelet Hash	Color Hash
Snapshot Ensemble	30.85 ± 0.27	30.77 ± 0.25	31.55 ± 0.28	30.94 ± 0.28	27.56 ± 0.39
Prune and Tune Ensemble	31.16 ± 0.28	31.19 ± 0.25	31.60 ± 0.27	31.17 ± 0.27	32.74 ± 0.51

Table IV: Mean distances using a variety of image hash algorithms over all 1024 feature visualizations from the final convolutional layer between Snapshot and Prune and Tune sibling networks. **Bolded** values correspond to greater distance and greater diversity between child feature representations.

rate of 0.1 and a final learning rate of 1e-5. We choose these hyperparameters to match the computational budget defined in Snapshot Ensembles [8].

Using the child networks constructed above, we create feature visualizations of identical neurons in each of the networks [9], [20]. Feature visualizations are optimized using ADAM with a learning rate of 0.05 for 1024 steps. Several random data augmentations are applied at each step, including: jittering by up to 8 pixels, scaling by a factor between 0.95 and 1.05, rotating by an angle between -5 and 5 degrees, and jittering a second time by up to 4 pixels.

We then compute 64 bit image hashes for each visualization (average hash, perceptual hash, difference hash, wavelet hash, and color hash) and report the Hamming Distance between each pair of child network visualization hashes.

We first present feature visualizations from several neurons selected at random from different layers within each network. Table III displays these sample visualizations where each column corresponds to the same neuron in each of the different networks. Prune and Tune children often focus on differ-

ent elements present in the parent network’s visualization, while Snapshot checkpoints appear to be much more highly correlated with the parent. The Hamming distances between each pair of image hashes supports this, with Prune and Tune Ensembles being more distant than those created with Snapshot Ensembles.

We extend these feature visualizations by analyzing every channel in the final convolutional layer. Thus, for each pair of child networks, 1024 feature visualizations are created for each. All of these visualizations are hashed using the algorithms described above and then the distance between sibling network hashes are measured. These final neurons are especially relevant as they represent the final stage of feature extraction before being fed into the fully-connected classifier layers. They most closely illustrate how each child network represents concepts and the differences over all the neurons in this layer will provide a comprehensive overview on the diversity between network representations.

Figure 2 displays a subset of these visualizations along with Table IV that reports the mean Hamming distances between

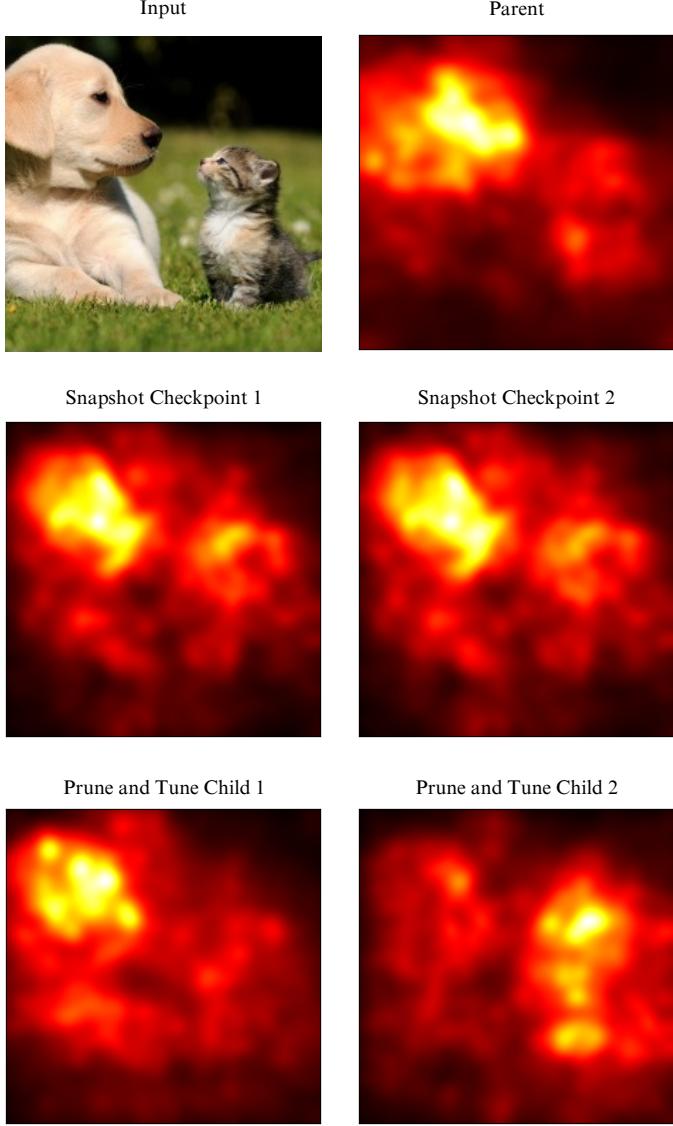


Fig. 3: An example of how saliency maps can be useful with ambiguous input. We include SmoothGrad saliency maps for a shared parent network, two Snapshot Ensemble checkpoints, and two Prune and Tune child networks. Both Snapshot checkpoints focus on the dog while the Prune and Tune children split their attention between the dog and the cat. The diversity encouraged by Prune and Tune Ensembles helps to reduce bias present in the parent network.

	RMSE	A-Hash	P-Hash	D-Hash	W-Hash
SSE	0.06 ± 0.001	10.01 ± 0.04	18.12 ± 0.04	10.00 ± 0.04	15.13 ± 0.04
PAT	0.07 ± 0.001	11.13 ± 0.04	19.95 ± 0.04	11.03 ± 0.04	15.97 ± 0.04

Table V: Mean distances between Snapshot Ensemble (SSE) and Prune and Tune Ensemble (PAT) saliency maps for all 50,000 samples in the ILSVRC2012 ImageNet validation set.

each pair of child networks over all feature visualization hashes. For every hashing algorithm we see more diversity in the Prune and Tune representations than in the Snapshot Ensemble.

We then explore how each ensemble method interprets input by creating SmoothGrad saliency maps for each of the 50,000 samples in the ImageNet ILSVRC2012 validation set [3]. Table V reports the average Root Mean Square Error, Average Hash, Perceptual Hash, Difference Hash, and Wavelet Hash distance between each pair of child networks over all 50,000 images. Prune and Tune Ensembles display greater diversity when considering the importance of input data compared to Snapshot Ensembles. Additionally, Figure 3 provides an example visualization of how saliency maps can be especially useful when dealing with ambiguous inputs. We feed an image containing a puppy and a kitten to each child network in the Snapshot and Prune and Tune Ensemble and compute their saliency maps using SmoothGrad. The parent network and the Snapshot checkpoints focus primarily on the puppy in the input image. The Prune and Tune children split their attention, where one child focuses on the puppy and the other on the kitten. This diversity can be valuable in building robust ensembles as techniques that encourage diverse representations can help to reduce bias (focus on the puppy) that is inherent in the parent network.

V. CONCLUSIONS

Diversity is an important property of robust neural network ensembles. However, traditional measures of diversity focus only on model outputs and give little insight into how ensemble members represent knowledge differently. We introduce a combination of interpretability methods and perceptual hashing as an effective approach for qualitatively analyzing diversity and measuring the similarity between knowledge representations.

This is especially important for low-cost ensemble methods as they tend to share information between members in order to reduce computational cost. In particular, temporal ensembles and evolutionary ensembles create members that inherit parameters and network structure from previous iterations. This allows for meaningful comparisons between inherited neurons at the same location in different child networks.

Our experiments demonstrate the efficacy of feature visualization and saliency maps with two powerful low-cost ensemble methods, Snapshot Ensembles and Prune and Tune Ensembles. We find that the sparsity and unique network topology of Prune and Tune children is highly effective for encouraging diverse feature representations when compared to Snapshot Ensembles.

Interpretability methods are making deep neural networks more accessible and understandable, and we believe the introduction of these methods to ensemble learning can provide better insights into diversity and aid in the construction of more robust low-cost ensembles.

VI. ACKNOWLEDGMENTS

This work is supported by National Science Foundation award IIS-1908866.

REFERENCES

- [1] G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. *Information Fusion*, 6:5–20, 03 2005.
- [2] J. Buchner. Imagehash. <https://github.com/JohannesBuchner/imagehash>, 2021.
- [3] J. Deng, W. Dong, R. Socher, L. Li, L. Kai, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [4] S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2020.
- [5] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *arXiv preprint:1802.10026*, 2018.
- [6] M. Havasi, R. Jenatton, S. Fort, J. Liu, J. Snoek, B. Lakshminarayanan, A. Dai, and D. Tran. Training independent subnetworks for robust prediction, 2021.
- [7] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.
- [8] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. Hopcroft, and K. Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.
- [9] L. Kiat. Lucent. <https://github.com/greentfrapp/lucent>, 2021.
- [10] L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181–207, 05 2003.
- [11] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint:1511.06314*, 2015.
- [12] S. Liu, T. Chen, Z. Atashgahi, X. Chen, G. Sokar, E. Mocanu, M. Pechenizkiy, Z. Wang, and D. Mocanu. Freetickets: Accurate, robust and efficient deep ensemble by training with dynamic sparsity, 2021.
- [13] Y. Malaiya. Antirandom testing: getting the most out of black-box testing. In *Proceedings of Sixth International Symposium on Software Reliability Engineering. ISSRE'95*, pages 86 – 95, 1995.
- [14] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. <https://distill.pub/2017/feature-visualization>.
- [15] PyTorch. Torchvision pre-trained models. <https://pytorch.org/vision/stable/models.html>, 2022.
- [16] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3145–3153. JMLR.org, 2017.
- [17] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise, 2017.
- [18] L. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2018.
- [19] P. Solla and A. Krogh. Learning with ensembles: How overfitting can be useful. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996.
- [20] Tensorflow. Lucid. <https://github.com/tensorflow/lucid>, 2021.
- [21] N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 90–95 vol.1, 1996.
- [22] Y. Wen, D. Tran, and J. Ba. Batchensemble: An alternative approach to efficient ensemble and lifelong learning, 2020.
- [23] T. Whitaker and D. Whitley. Prune and tune ensembles: Low-cost ensemble learning with sparse independent subnetworks. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 2022.
- [24] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions. 2010.
- [25] Y. Zhang, P. Tiño, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021.