There Is No Magic Subnetwork: An Interpretable Analysis of Diversity in Prune and Tune Ensembles

Anonymous
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Diversity is an important consideration in the construction of robust neural network ensembles. A collection of well trained models will generalize better to unseen data if they are diverse in the patterns they respond to and the predictions they make. Encouraging diversity becomes especially important for low-cost ensemble methods, as members often share network structure or training epochs in order to avoid training several independent networks from scratch. The most popular methods for measuring diversity focus on analyzing the differences between the outputs of models. However, this black box approach gives little insight into how diversity develops among ensemble members. In this paper, we introduce network interpretability methods as an effective approach for analyzing diversity in evolutionary and temporal ensembles. We apply these methods to Prune and Tune Ensembles, where we demonstrate that independent subnetworks derived from an identical parent can learn extremely diverse feature representations with little computational cost. This qualitative approach to diversity analysis can lead to valuable insights and new perspectives for how we view and encourage diversity in ensemble methods.**

***Index Terms*—Neural Networks, Machine Learning, Ensemble Methods, Network Intrepretability, Ensemble Diversity, Prune and Tune Ensembles**

I. Introduction

Diversity has long been known to be an important consideration in ensemble learning [17], [28]. Assuming equivalent model performance, a group of diverse models should have better generalization capabilities than a group of less diverse models. Diversity becomes especially important for low-cost ensemble methods as they tend to reduce computational cost by sharing information between members in some way.

Many metrics have been introduced to measure diversity in classifier ensembles [17]. Most of these metrics focus on measuring the differences between predictions in the output space of models, which allows for easy comparisons between classifiers of any type. However, diversity metrics that act on outputs and predictions tend to be closely tied to accuracy. An ensemble can display great diversity but perform much worse than another ensemble if the accuracy of the members is not great. Because of this, traditional diversity metrics can be hard to fairly compare between methods.

Meanwhile, neural network interpretability continues to be a growing trend in machine learning research [42]. This field is shifting the preconceived notions that deep neural networks act as black box models. By utilizing visualization and attribution techniques, it becomes possible to explore *how* and *what* deep neural networks learn.

The application of interpretability methods to ensemble learning could lend interesting insights into how diversity develops and is represented among ensemble members. We explore this idea with a recent low-cost ensemble learning approach, Prune and Tune Ensembles [38]. Prune and Tune Ensembles create members by cloning and pruning a trained parent network several times. Because each child is derived from an identical parent, interpretability methods allow us to visualize exactly how the child networks diverge as a result of the unique network structures each child inherits via pruning. This new approach to analyzing diversity extends to a variety of low-cost ensemble methods, including evolutionary and temporal ensembles alike.

We present several experiments to explore diversity in Prune and Tune Ensembles. We begin with a benchmark comparison of output diversity with other modern low-cost ensemble learning algorithms. We then apply several interpretability techniques, including feature visualization, activation grids, and saliency maps, to analyze the differences in feature representations between child networks. Additionally, we explore perceptual hashing as a method for quantifying these differences. We report the hamming distance between child networks using several popular hashing algorithms, including: average hash, perceptual hash, difference hash, wavelet hash, and color hash [5]. We demonstrate that the feature representations between child networks in a Prune and Tune Ensemble are more diverse than those that are developed in a temporal ensemble.

Our experiments can offer a new perspective on the role that sparsity plays in deep convolutional networks. The Lottery Ticket Conjecture introduced the notion that the success of deep neural networks is due to gradient descent seeking out and training a specific subset of well-initialized weights [8]. Prune and Tune Ensembles create child networks that explicitly cut these lottery ticket subnetworks in half. We find that these independent subnetworks learn extremely diverse feature representations with very little training, suggesting that *there is no magic subnetwork* that governs generalization performance. Combining several independent subnetworks with diverse representations can be a powerful way to improve generalization and build robust low-cost ensembles.

II. Background

A. Low-Cost Ensemble Learning

Several methods have been introduced to address the large costs of training ensembles of deep neural networks, which can be broadly categorized as pseudo-ensembles, temporal ensembles, and evolutionary ensembles [1], [2], [31]. All of these approaches attempt to strike a balance between computational cost, model accuracy, and ensemble diversity. As each of these approaches share network structure, parameters, or training epochs in some way, generalization is often heavily impacted by the amount of diversity that each of the members are able to capture without loss of test accuracy.

Pseudo-Ensembles: Members are trained as a part of a single monolithic network architecture. Methods like Dropout [29], DropConnect [35] and Stochastic Depth Networks [14] mask different parts of a network for each batch during training. This can be described as a way to train an exponential number of subnetworks that are implicitly ensembled at test time. Multi-Input Multi-Output (MIMO) trains a single network with duplicated input and output heads [12]. TreeNets use locally connected branches with separate output heads [18]. BatchEnsemble decomposes weights into a Hadamaard product of a shared set of weights and a rank-one matrix for each member [37].

Pseudo-Ensembles are memory-efficient, but tend to be less diverse than other ensemble methods as network structure is implicitly shared [1].

Temporal Ensembles: Members are created by saving the intermediate model states of a single model throughout training [31]. Horizontal Voting Ensembles take the most recent states from a contiguous block of epochs [40]. Snapshot Ensembles make use of cyclic learning rates that encourage long jumps to new locations in the parameter space before converging [13]. Fast Geometric Ensembles extend Snapshot Ensembles by looking for minima along high-accuracy pathways [10].

Temporal Ensembles are computationally efficient as ensemble members are created within a single network's training process. However, model states taken from the early phases of training tend to perform worse and model states taken later in training tend to be highly correlated.

Evolutionary Ensembles: Members are generated via perturbative processes on parent networks. Evolution Strategies (ES, CMA-ES, NES) generate populations of networks by adding random noise to the weights of a parent [11], [24], [39]. Neuroevolution of Augmenting Topologies (NEAT) creates populations by complexifying the topology along with modifying the weights of a parent network [30]. MotherNets hatch children with function preserving morphisms that add additional neurons and layers around a core parent network [36].

Evolutionary Ensembles produce diverse networks at the cost of much larger memory requirements and convergence times.

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.

Prune and Tune Ensembles work by 1) training a single parent network, 2) spawning child networks by cloning and dramatically pruning the parent using random or anti-random sampling strategies, and 3) fine tuning each of the child networks with a cyclic learning rate schedule for a small number of epochs. A key idea is that pruning and tuning a previously trained network has an extremely low computational cost. Because the parent is already optimized, the child networks all converge with only a few epochs of additional training. One can easily create a parent and dynamically generate many child networks at a cost that is only slightly more than training a single network.

As the children are all derived from an identical parent network, anti-random pruning and one-cycle tuning were introduced as effective methods for encouraging diversity among the child networks [21], [27].

Parent Network Flexibility: Prune and Tune Ensembles can use any network architecture or training methodology for the parent. Pre-trained networks can be used to save additional computation. This paper explores the diversity of Prune and Tune Ensembles on image classification problems with deep convolutional networks. The principles introduced here could theoretically apply to a wide array of machine learning problems and model architectures.

Creating Child Networks With Anti-Random Pruning: Anti-random pruning was introduced as a technique for encouraging diversity among pairs of child networks derived from an identical parent. 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 neural network F with hidden layers parameterized by θ, optimized by any standard deep learning training algorithm, G. Child networks are created by cloning the parent network F and pruning a significant number of hidden parameters. We implement pruning by taking the Hadamard product between a binary mask and a set of parameters.

Consider a binary bit string $M = \{x_0, ..., x_n : x \in \{0, 1\}\}$, that is randomly generated with 50% sparsity where 1 represents parameters that we keep 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:

$$f_1 = \theta \circ M \qquad \text{and} \qquad f_2 = \theta \circ (1 - M)$$

where f_i is a child network, θ denotes the parameters of the parent network and $\circ$ denotes the Hadamard product.

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

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

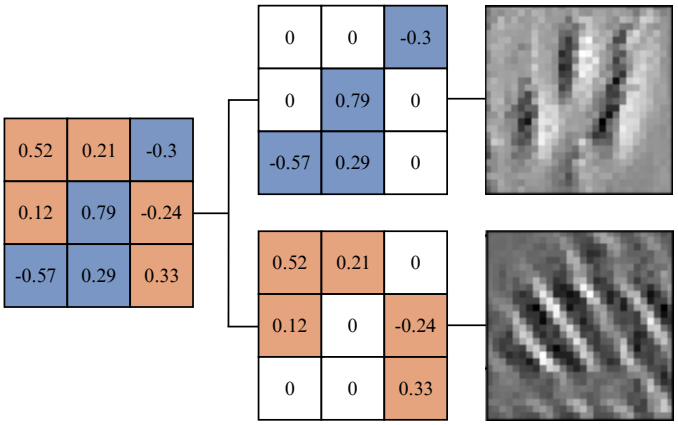

Fig. 1. A parent filter split into two anti-random child filters. The resulting networks learn diverse feature representations as a result of their unique topologies.

In Prune and Tune Ensembles, this process is repeated N times to create an ensemble of size $2N$, where each child network has exactly 50% sparsity.

Pruning Selection and Structure: We have described networks with an all encompassing parameter vector θ. However, neural networks are hierarchical structures made up of many different layers, f_i, with varying numbers of parameters, θ_i.

$$F(x) = f_n(...f_2(f_1(x;\theta_1);\theta_2)...;\theta_n)$$

Random pruning methods account for the imbalanced nature of neural networks by considering the distribution of parameter selection and structure.

Pruning structure refers to whether individual parameters (unstructured) or entire neurons (structured) are removed. Structured pruning results in more significant cost savings as the resulting networks are made smaller. Unstructured pruning often results in models with better predictive power at higher levels of sparsity [3].

Pruning selection is distinguished between global and local pruning. Global methods pool all parameters together before pruning, irrespective of their locations. Local pruning instead removes a fixed amount of parameters per layer or filter. Global pruning ends up removing more parameters from larger layers than smaller layers. When anti-random networks are generated, this imbalance can lead to layer collapse which halts information flow through the network. Local pruning ensures that all child networks have a consistent distribution of sparsity throughout the network.

One-Cycle Tuning: In order to recover accuracy after pruning, each child undergoes a small tuning phase using a cyclic learning rate schedule. One-cycle tuning is a two phase schedule that inversely cycles between growth and decay for both learning rate and momentum [27]. 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.

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

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.

Table I. Comparison between Low Cost Ensembles using WideResNet-28-10 Architecture. Prune and Tune (PAT) produced better generalization and used at least 15% fewer FLOPS than any other low cost ensemble method. * indicates published results taken from [12], [19], [38].

	CIFAR-10			CIFAR-100		
Method	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

Making Predictions: Prune and Tune Ensembles use model averaging to combine predictions from ensemble members.

$$y_e = argmax(\frac{1}{S}\sum_{i=1}^{S}\sigma(f_i(x)))$$

where y_e is the ensemble prediction, S is the number of members in the ensemble (the ensemble size), σ is the softmax function and $f_i(x)$ is the output of the individual ensemble member i.

Table 1 presents results comparing Accuracy (ACC), Negative Log Likelihood (NLL) and Expected Calibration Error (ECE) for several low cost ensemble methods for CIFAR-10 and CIFAR-100. Prune and Tune Ensembles yield excellent results.

III. Exploring Diversity

Diversity is a crucial factor in the success of ensemble generalization. We use this chapter to explore diversity in Prune and Tune Ensembles between Anti-Random child networks. These networks offer an excellent foundation for exploring diversity for a number of reasons.

1) Prune and Tune Ensembles demonstrate excellent empirical results, but there has not been enough exploration into *why* they perform so well. Visualizing the feature representations of these models can lend an explanation to the performance we witness despite these ensembles requiring relatively little computation.

2) Child networks are derived from an identical parent. We can look at the same neuron in two different networks and visualize how feature representations diverge and change as a result of pruning different parts of the space from a shared parent.

3) There is a notion introduced in a popular line of pruning research, the lottery ticket conjecture, that suggests that the

success of deep neural networks is due to gradient descent seeking out and training a specific subset of weights called the lottery ticket subnetwork [8]. Anti-random networks explicitly destroy the lottery ticket, yet still perform remarkably well. Exploring these independent subnetworks could lead to a better understanding of pruning and the role that sparsity plays in deep convolutional networks.

Experimental Details: All experiments are conducted on a single home computer. We train and evaluate all models using an Nvidia GTX-1080ti GPU. Output diversity experiments use a WideResNet28x10 model trained on CIFAR-10 [16], [41]. The parent model is trained for 140 epochs and six anti-random child networks are tuned for an additional 10 epochs. Interpretability experiments are conducted with an Inception-V1 model pre-trained on ImageNet [6], [32]. Child networks are fine tuned for 10 epochs on the 2012 ILSVRC validation dataset [23]. We use Stochastic Gradient Descent with Nesterov momentum for all model optimization.

A. Output Diversity

The generalization performance of neural network ensembles tend to increase with the number of well trained and diverse models it contains. This is often explained in ensemble literature by considering the bias-variance decomposition of the mean squared error (MSE) [4], [34].

$$\begin{aligned} MSE(x) &= E[(f(x) - y)^2] \\ &= (E[f(x)] - y)^2 + E[f(x) - E[f(x)]]^2 \\ &= bias[f(x)]^2 + var[f(x)] \end{aligned}$$

When used with a ensemble of estimators, the variance component breaks down further to produce the bias-variance-covariance decomposition [4].

$$\overline{bias^2} = \frac{1}{M}\sum_i (E[f_i] - y)$$

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

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

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

Ideally, ensemble methods that prioritize diversity will be able to reduce covariance without increasing the bias or variance.

In a classification context, there is no standardized analog to the bias-variance decomposition [4]. Methods that only output labels (such as decision trees or k-nearest neighbors) differ from methods that output probability distributions. Methods that use majority vote differ from those that average outputs. Because of this, several metrics have been used to quantify diversity in classification ensembles [17].

Recent low-cost ensemble approaches have focused on Kullback-Leibler Divergence and Prediction Disagreement Ratio [7], [12], [19].

Kullback-Leibler Divergence: Also known as relative entropy, KL Divergence 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.

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

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): Rather than comparing the differences of the output distributions, PDR measures the differences in only the predicted classes.

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

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

Table II. Prediction Disagreement Ratio (PDR) and KL divergence between ensemble members on CIFAR-10 with WideResNet-28x10. Methods marked with * are results reported from [12], [19]

Methods	$d_{PDR} \uparrow$	$d_{KL} \uparrow$
Treenet*	0.010	0.010
BatchEnsemble*	0.014	0.020
LTR Ensemble*	0.026	0.057
EDST Ensemble*	0.026	0.057
Prune and Tune Ensemble	0.036	0.090
MIMO*	0.032	0.081
Dense Ensemble*	0.032	0.086

Table 1 presents results from a benchmark comparison on CIFAR-10 with a total training budget of 200 epochs. Prune and Tune Ensembles produce larger measures of diversity for KL and PDR metrics compared to several other low cost ensembling methods.

B. Feature Visualization

One limitation of metrics such as LK and PDR is that they give little insight into how the representations of features differ between ensemble members. Feature visualization uses optimization to create images that maximize or minimize activations of specific parts of the network. The resulting images display the types of patterns a neuron or channel responds strongly to.

A naive approach, one that performs gradient descent on the raw pixels of an image, tends to result in images with a lot of high frequency noise and nonsensical patterns [22]. Various forms of regularization can be used, including frequency penalization, transformation robustness, and learned priors [22]. However, these techniques reduce high frequencies in the gradient rather than the visualization itself. To alleviate this, 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 [22]. The 2D Fourier Transform for an x-by-y image X is defined as:

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

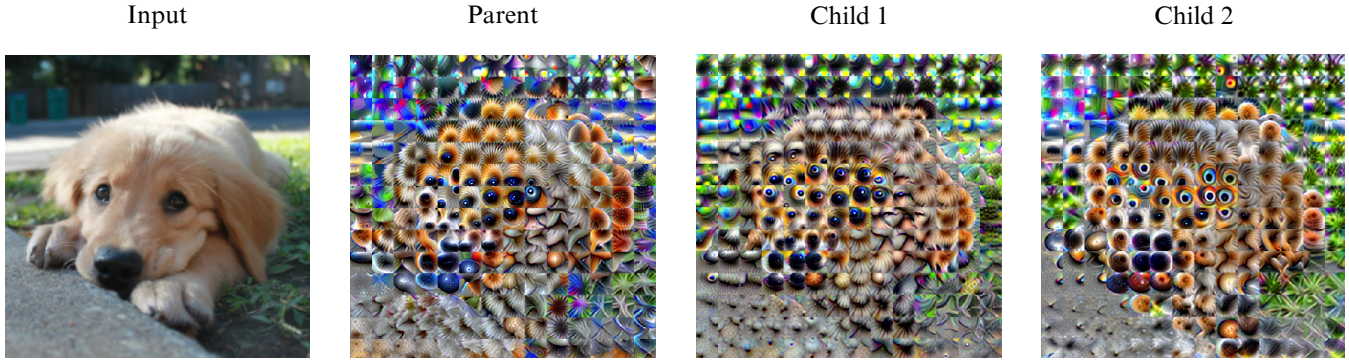

Fig. 2. Activation grids for a parent and two anti-random child networks. Grid cells are feature representations for the image patch at layer 4c in an Inception-V1 network.

We maximize the activation of a specific convolutional filter by optimizing the following objective:

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

where h is the activation 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 activations, the sign is flipped for the objective. The image is then optimized using any standard gradient descent based optimizer:

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

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

We explore feature visualizations by using the open source PyTorch Lucent library [15], inspired by its TensorFlow predecessor, Lucid [33]. We select a random neuron from each inception block in the network. We then generate feature visualizations by optimizing an image in a decorrelated fourier transformed space. We use the ADAM optimizer with a learning rate of 0.05 and train for 1024 steps. Several augmentations are applied at each step to improve image quality, including: jittering by up to 16 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 8 pixels.

Quantifying Visual Diversity: There are a large number of similarity metrics that can be used to measure differences between images, including the mean square error, peak signal-to-noise ratio, structural similarity index, spatial correlation coefficient, spectral angle mapper, and universal image quality index. These approaches are generally used to measure image quality as a response to noise, corruption, or compression against a ground truth image.

However, these image quality metrics tend to be highly sensitive to minute differences, and we found these measures to be too noisy for an apt comparison. As feature visualizations can vary wildly, we suggest a hash 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. We compare the hashes of the feature visualizations for each child network and report the Hamming distance between the two, where the Hamming distance is equal to the number of positions in which the bits of the two hashes differ.

Figure 3 presents the results of several feature visualizations between anti-random networks in a Prune and Tune Ensemble, along with two sequential snapshots in a Snapshot Ensemble. Each Prune and Tune Child is tuned for 10 epochs. The Snapshots are taken by training the same parent network for an additional 10 epochs each, using the schedule described in the snapshot paper [13] where the max learning rate is $\eta = 0.1$.

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 [5]. We see a stark difference in the diversity of child networks in Prune and Tune Ensembles compared to two late checkpoints in a Snapshot Ensemble.

C. Activation Grids

Rather than generating images that maximize or minimize the activations for an individual neuron, activation grids instead allow us to visualize how combinations of neurons respond to a specific input. Looking at the combinations of all neurons in a layer allows us to visualize how a network may interpret different parts of an image as a whole.

Optimization is again done in the decorrelated Fourier Space as described in the previous section. The input image is divided into a grid of image patches. The optimization objective then is to maximize the dot product between the original image patch and the combination of all neuronal activations in a given layer.

$$F(X) = -\sum_{i} \sum_{x,y,z} h_{n,x,y,z}(X_i) \cdot X_i$$

where h is the activation of a neuron, X is the input image, X_i is an image patch, n is the layer, z is the channel, and x and y are the spatial positions of the neuron within a channel.

Figure 2 displays how a parent network and two optimized child networks respond to an identical input. The resulting grids do display quite a lot of diversity in representations, with notable differences around the eyes and nose of the dog.

Prune and Tune Ensembles

Model	3a:51	3b:131	4a:307	4b:423	4c:34	4d:164	4e:178	5a:228	5b:70
Parent									
Child 1									
Child 2									

Metric	3a:51	3b:131	4a:307	4b:423	4c:34	4d:164	4e:178	5a:228	5b:70
A-Hash	**31**	**31**	30	**34**	**40**	**37**	24	**39**	**30**
P-Hash	**30**	**32**	**40**	**32**	**36**	**38**	**36**	**30**	**34**
D-Hash	28	**31**	**34**	**31**	**36**	**31**	**38**	**34**	**39**
W-Hash	**36**	**34**	**30**	**30**	**38**	28	26	**38**	**34**
C-Hash	**36**	**34**	18	42	**26**	**57**	**44**	**33**	**49**
Mean	**32.2**	**32.4**	30.4	33.8	**35.2**	**38.2**	**33.6**	**34.8**	**37.2**

Snapshot Ensembles

Model	3a:51	3b:131	4a:307	4b:423	4c:34	4d:164	4e:178	5a:228	5b:70
Parent									
Snapshot 1									
Snapshot 2									

Metric	3a:51	3b:131	4a:307	4b:423	4c:34	4d:164	4e:178	5a:228	5b:70
A-Hash	**31**	28	**31**	32	28	36	**27**	31	28
P-Hash	**30**	26	34	30	32	34	34	28	28
D-Hash	**29**	30	29	30	26	29	33	29	34
W-Hash	30	26	28	28	26	**30**	**28**	34	30
C-Hash	29	26	**35**	**54**	23	38	20	16	28
Mean	29.8	27.2	**33.4**	**34.8**	27.0	33.4	28.4	27.6	29.6

Fig. 3. A collection of neurons randomly chosen from each convolutional block in an Inception-V1 model. Neurons are labeled "layer-index:neuron-number". Metrics include Average Hash (A-Hash), Perceptual Hash (P-Hash), Difference Hash (D-Hash), Wavelet Hash (W-Hash), and Color Hash (C-Hash). We report the Hamming Distance between the hashes of the two child feature visualizations. **Bolded** values correspond to greater distances between child filters.

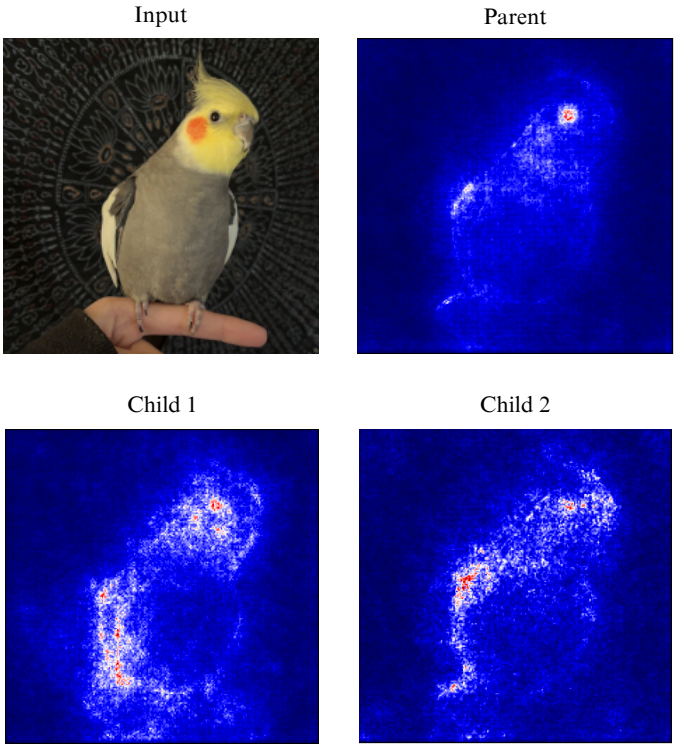

Fig. 4. SmoothGrad saliency maps for a parent network and two anti-random children. Gradients are averaged over 10 runs where the input is perturbed with gaussian noise $\gamma \sim \mathcal{N}(0, 1)$.

D. Saliency Maps

Saliency Maps visualize parts of the input space that a network responds strongly too. This is done by first performing a forward pass through the network with a given input image. The gradient is then computed by setting all non-predicted outputs to 0 and backpropagating the predicted class score back through to the inputs [25].

SmoothGrad was later introduced as an extension to the vanilla method that reduces noise in the visualizations by averaging the gradient over several gaussian perturbed inputs [26]. The resulting saliency maps are then described as:

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

where, $G(X)$ is the SmoothGrad output for an input image X and Y_c is the output of the predicted class label c with all other outputs set to 0. The input X is perturbed with gaussian noise sampled from a normal distribution $\gamma_i \sim \mathcal{N}(0, \sigma^2)$.

Figure 4 displays the results of SmoothGrad applied to a parent and two anti-random child networks. We see a large difference in regions of interest between the parent and child networks. The parent network tends to focus strongly on the eye, while the child networks spread their focus around the body, wings and feet as well.

Table III. Comparison between two pairs of anti-random child networks and the lottery ticket subnetwork from the same parent network. All models are tuned for 10 epochs after pruning. We report the mean accuracy (ACC), negative log likelihood (NLL), and expected calibration error (ECE).

Network	ACC ↑	NLL ↓	ECE ↓
Parent	96.20	0.1369	0.0183
Child 1	95.38	0.1506	0.0184
Child 2	95.27	0.1542	0.0204
Child 3	95.72	0.1445	0.0158
Child 4	95.48	0.1471	0.0172
Lottery Ticket	95.48	0.1582	0.0219

IV. There Is No Magic Subnetwork

The Lottery Ticket Hypothesis is a recent and popular approach for finding sparse networks that train well from scratch. By training a dense network, pruning the lowest magnitude weights, and rewinding the network to the original weights, the resulting sparse network should train as well as the original dense network. These high performing subnetworks are called lottery tickets as they are thought to be winners of the weight initialization lottery [8].

Several subsequent papers have found that the lottery ticket procedure fails for larger datasets and deeper networks with modern learning rate schedules [9], [20]. Despite this, an untested conjecture in the original lottery ticket hypothesis, continues to be a popular interpretation for why deep neural networks are successful.

The Lottery Ticket Conjecture: *Stochastic Gradient Descent seeks out and trains a subset of well-initialized weights. Dense, randomly-initialized networks are easier to train than the sparse networks that result from pruning because there are more possible subnetworks from which training might recover a winning ticket.*

Assume there is a lottery ticket subnetwork which has an oversized impact on generalization. Creating a pair of anti-random child networks destroys the lottery ticket subnetwork with overwhelming probability. Furthermore, if one child did inherit more of the lottery ticket subnetwork, the other child must inherit less. Thus if there is a lottery ticket subnetwork which impacts generalization, we should see a decline in performance in either of the anti-random child networks, but this is not what happens. In Table 3 we take a parent network, compute the lottery ticket subnetwork, then generate two pairs of anti-random children. In terms of accuracy, negative log likelihood and expected calibration error, the anti-random children are indistinguishable from the lottery ticket.

Our experiments indicate that large networks can be completely partitioned using random pruning methods and still maintain excellent performance after a small amount of tuning. We suggest that *there is no magic subnetwork* that governs generalization performance. These independent subnetworks learn diverse representations of features and the combination of diverse representations is important in building robust and effective systems.

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 differ in feature space. We apply several neural network interpretability methods to aid in visualizing how networks in evolutionary and temporal ensembles develop diverse representations. We use feature visualization to generate images that maximize specific neurons; we use activation grids to visualize how combinations of neurons from a layer react to input; and we use saliency maps to visualize important parts of the input space.

We explore these methods with Prune and Tune Ensembles, a recent low-cost ensemble learning approach that dynamically generates child networks with anti-random pruning methods. We visualize how feature representations in children diverge as a result of pruning different parts of a shared space. Our experiments indicate that these child networks learn extremely diverse representations with very little training. We use several perceptual hash algorithms to quantify the distance between feature representations and we see much greater diversity between Prune and Tune children than sequential Snapshot states.

Anti-random children learn diverse feature representations as a result of their unique network topology. 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.

References

[1] P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. *ArXiv preprint:1412.4864*, 2014.

[2] H.G. Beyer and H.P. Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1:3–52, 03 2002.

[3] D. Blalock, J. Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning?, 2020.

[4] G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. *Information Fusion*, 6:5–20, 03 2005.

[5] J. Buchner. Imagehash. https://github.com/JohannesBuchner/imagehash, 2021.

[6] 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.

[7] S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2020.

[8] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[9] J. Frankle, G. Dziugaite, D. Roy, and M. Carbin. Stabilizing the lottery ticket hypothesis, 2020.

[10] 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.

[11] N. Hansen, S. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation. *Evolutionary computation*, 11(1):1–18, 2003.

[12] M. Havasi, R. Jenatton, S. Fort, J. Liu, J. Snoek, B. Lakshminarayanan, A. Dai, and D. Tran. Training independent subnetworks for robust prediction, 2021.

[13] 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.

[14] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, pages 646–661, 2016.

[15] L. Kiat. Lucent. https://github.com/greentfrapp/lucent, 2021.

[16] A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, May 2012.

[17] 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.

[18] 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.

[19] 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.

[20] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning, 2019.

[21] 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.

[22] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[24] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

[25] 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.

[26] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise, 2017.

[27] 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.

[28] P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Machine Learning Research*, 15(56):1929–1958, 2014.

[30] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127, June 2002.

[31] A. Swann and N. Allinson. Fast committee learning: preliminary results. *Electronics Letters*, 34:1408–1410, 1998.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions, 2014.

[33] Tensorflow. Lucid. https://github.com/tensorflow/lucid, 2021.

[34] 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.

[35] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proc Intern. Conf. Machine Learning (ICML)*, pages 1058–1066, 2013.

[36] A. Wasay, B. Hentschel, Y. Liao, S. Chen, and S. Idreos. Mothernets: Rapid deep ensemble learning. *arXiv preprint:1809.04270*, 2018.

[37] Y. Wen, D. Tran, and J. Ba. Batchensemble: An alternative approach to efficient ensemble and lifelong learning, 2020.

[38] 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.

[39] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, and J. Schmidhuber. Natural evolution strategies, 2011.

[40] J. Xie, B. Xu, and Z. Chuang. Horizontal and vertical ensemble with deep representation for classification. *arXiv preprint: 1306.2759*, 2013.

[41] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint: 1605.07146*, 2016.

[42] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021.