Low-Cost Ensemble Learning: Surveying Efficient Methods for Training Multiple Neural Networks

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

***Abstract*—Ensemble learning has long been known to be a reliable and consistent way to improve generalization performance across a wide range of machine learning tasks. Instead of training and making predictions with a single model, ensembles use several independent models and combine their predictions together. However, training several independent models from scratch can become prohibitively expensive as deep neural networks continue to grow in both scale and complexity. Many approaches have been introduced to alleviate these large computational costs, often balancing tradeoffs between training cost, inference cost, storage cost, member diversity, and population size. These low-cost methods make ensemble learning much more accessible to researchers and practitioners, while significantly improving generalization performance in resource constrained environments. This work aims to provide a comprehensive overview of the state of this field.**

I. Introduction

Ensemble learning is effective at improving generalization performance across a wide range of machine learning tasks. Instead of training and making predictions with a single model, ensembles instead use several independent models and combine their predictions together. The combination of multiple predictions can help to reduce bias or variance and can significantly improve performance when compared to a single model [2], [5].

Ensembles of neural networks in particular have demonstrated excellent results, being used to win many high profile machine learning competitions [24], [29], [42]. However, as modern neural networks continue to grow in both scale and complexity, the cost associated with ensemble learning quickly becomes untenable. A single state of the art network requires several decades worth of GPU hours and hundreds of thousands of dollars in cloud compute costs to train [3], [30]. Inference and storage can also be costly with these networks as they contain billions of parameters and require hundreds of gigabytes of memory.

Low-cost ensemble methods have become increasingly important as they significantly reduce the computational cost associated with traditional ensemble learning, while retaining many generalization benefits that ensemble learning affords.

Several methods have been introduced to alleviate ensemble cost that we broadly categorize as pseudo-ensembles, temporal ensembles, or evolutionary ensembles. Pseudo-ensembles train several models under the guise of a single monolithic architecture. Temporal ensembles encompass methods that save snapshots or checkpoints of a single model throughout training. Evolutionary ensembles leverage concepts like mutation, recombination, and selection to spawn new child networks. While these methods vary significantly in their approach to reducing cost, the primary idea behind each is to share either network structure, gradient information, or learned parameters among ensemble members.

The sharing of information between ensemble members is effective at reducing training cost, as each ensemble member avoids the need to be trained from scratch. However, this invariably reduces diversity between members, which can have significant impact on generalization performance [20]. If each model makes the same predictions or learns the same feature representations, then there is little benefit to be gained from combining predictions. Diversity plays a critical role in ensemble learning research and the advantages and disadvantages of low-cost methods often relies on a trade-off between training cost, model accuracy, and member diversity.

The majority of recent low-cost ensemble learning papers focus primarily on supervised learning and image classification. Experiments include several standardized benchmarks with deep convolutional residual networks on the CIFAR and ImageNet datasets and evaluations of robustness using corrupted and perturbed validation datasets. While these benchmarks are mostly standardized, there are small inconsistencies with the training budget and population sizes between methods, which can make it hard to fairly compare some of the published results.

In this paper, we aim to provide a comprehensive overview of the state of low-cost ensemble learning. We discuss a number of the most popular low-cost ensemble methods and organize them into a new taxonomy based on their primary means for reducing cost. We compare and contrast these methods and discuss published results on standard benchmarks and diversity evaluations. Finally, we discuss limitations of the current state of the field and introduce promising future research directions. This field has significant potential impact for improving state of the art while making ensemble learning more accessible to researchers and practitioners.

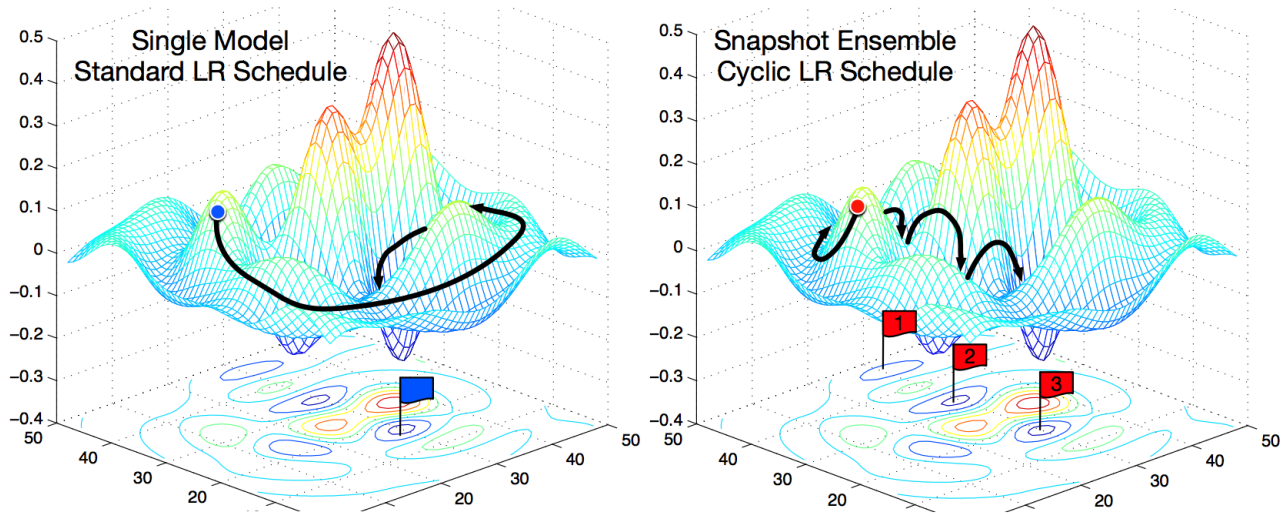

Fig. 1. An illustration from Huang et al. demonstrating the optimization trajectory of a standard network in the leftmost figure and a snapshot ensemble in the rightmost figure. A cyclic learning rate schedule encourages convergence to unique local optima that are subsequently saved for the ensemble [15].

II. Overview of Low-Cost Methods

There is no established taxonomy for categorizing low-cost ensemble learning approaches in the literature. However, we do note commonalities among methods and we find that many approaches tend to cluster around three primary ideas. We categorize these ideas under the names: pseudo-ensembles, temporal ensembles, and evolutionary ensembles.

A. Pseudo-Ensembles

Bachman et al. formalized the term pseudo-ensemble by describing these methods as: a (possibly infinite) collection of child models spawned from a parent model by perturbing it according to some noise process [1]. The authors use the term "spawned" to describe the creation of ensemble members, implying that independent networks are created or generated. However, the paper further elaborates and ultimately describes pseudo-ensembles as techniques that implicitly train ensembles under the guise of a single monolithic architecture [1]. This distinguishes pseudo-ensembles from other methods where independent members are created and trained separately.

Dropout is the canonical example of a pseudo-ensemble where neurons are randomly masked during training for each mini-batch of data. Dropout is essentially training a unique subnetwork (ensemble member) for each forward and backward pass. During inference, dropout is turned off so that no neurons are masked, which results in each of the subnetworks being implicitly ensembled together [31].

Several variants of Dropout have since been introduced. DropConnect masks individual connections rather than neurons [37]. Stochastic Depth Networks mask entire layers of deep residual networks [16]. Since these techniques mask large parts of the network during training, significantly more forward passes are needed to fully train a network. However, the resulting networks tend to converge to better optima than their standard counterparts.

Some recent works build on the ideas of parameter sharing in pseudo-ensembles by explicitly encoding relationships in the network architectures themselves. TreeNets are architectures that consist of zero or more shared layers that then split into independent branches, each with its own output head. During training, the shared layers accumulate the gradients from each independent branch and during testing, each path from input to one of the outputs can be treated as an independent ensemble member [21].

Multiple-Input Multiple-Output (MIMO) is a recent variation on TreeNets that uses multiple input heads and multiple output heads with no independent branches. Instead, the entire network is shared. Each input head is fed different samples and each of the output heads is trained to predict the corresponding input. [11]

BatchEnsembles build on parameter sharing by decomposing weight matrices into a Hadamard product of a single shared set of weights (slow weights) and a rank-1 matrix (fast weights) for each member [39]. The rank-1 matrix is represented by two single column vectors which significantly reduces the storage cost for each member, and the Hadamaard product is much cheaper computationally than matrix multiplication which results in little computational overhead compared to inference with a single network.

Knowledge Distillation was introduced to alleviate the inference and storage costs of ensembles and large networks. This approach is based the idea that you can transfer the knowledge learned from a large network to a small network by training the small network using the predictions of a trained large network, instead of the raw training labels. This enables the small network to learn features that would be hard to

Table I. A generalized overview of some of the benefits and drawbacks of various low-cost ensemble learning strategies. All categories are relatively and subjectively graded according to their primary cost reduction idea. Individual methods within each category may differ in their strengths or weaknesses from the proposed scale.

Method	Training	Inference	Storage	Diversity	Population Size
Pseudo	✓	✓✓	✓✓	-	-
Temporal	✓✓	-	-	✓	✓
Evolutionary	✓	-	-	✓✓	✓✓

disentangle if trained from scratch. This can be done in the context of ensemble learning by training a single network on the predictions made by a fully trained ensemble. The single network then approximates the behavior of the ensemble without needing to store and test all of the models separately [14].

Pseudo-ensembles are also closely related to recent robustness and optimization trends in neural network training. Stochastic Weight Averaging (SWA) is an optimization technique where the weights of a model are averaged over several states taken from the final training epochs. These final epochs often use a very small learning rate and the weights tend to bounce around a single optima. Instead of using these checkpoints as independent models and averaging their predictions, instead the weights themselves at these locations are averaged to create a more robust single model [17].

Model Soups operate on a similar principle to Stochastic Weight Averaging and have recently shown state-of-the-art performance on large image classification benchmarks. Instead of taking the final epoch checkpoints of a single model, Model Soups average the weights of several independent models that are derived from the same pre-trained network but are each fine tuned with different hyperparameters [43].

Pseudo-ensembles are much more memory efficient than other low-cost ensemble methods as members are embedded within a single network structure. These techniques are generally very fast at performing inference, as forward passes do not need to be independently computed for each ensemble member. This makes these techniques highly valuable for environments where memory and inference cost is prioritized, like in embedded systems or edge computing. However, pseudo-ensembles do tend to need more training epochs to converge and the resulting models are less diverse than other ensemble methods, as large parts of the network structure are shared.

B. Temporal Ensembles

Fast Committee Learning introduced the idea that ensembles can be created by selecting and saving checkpoints over a single network's training trajectory [34]. We call related approaches temporal ensembles since these methods are based on saving model states over time.

Horizontal Voting Ensembles experimented with saving a contiguous number of states from every single epoch taken late in the training phase. Checkpoints taken early in training will be much less accurate than those taken later in training, suggesting that its worth using less diverse models if their overall accuracy is better [44].

Snapshot Ensembles make use of a repeating cyclic schedule that slowly decays the learning rate 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 saved at the end of each cycle [15].

Fast Geometric Ensembles build on Snapshot Ensembles by analyzing the loss surfaces found at and between local optima. The local optima found at the end of each snapshot cycle tend to be connected by simple curves that maintain training and test accuracy throughout. By interpolating the weights between optima found using the snapshot ensemble algorithm, Fast Geometric Ensembles can generate much larger ensembles by saving model states across these high accuracy pathways [9].

FreeTickets is a recent method that combines ideas from sparse neural network training and temporal ensemble learning. FreeTickets use a dynamic sparse training algorithm that isolates a unique sparse subnetwork at each cycle which is optimized with a cyclical learning rate schedule. At the end of every optimization cycle, the sparse model state is saved before updating the connectivity for the next cycle. The differences between the network structure of each member encourages greater diversity between members [22].

Temporal Ensembles are very efficient for training, since they typically don't need any more training time than a single network. The performance of these temporal ensembles often compete with traditional dense ensembles at a significant fraction of the compute cost. Inference and storage is more expensive than pseudo-ensembles since each ensemble member needs to be evaluated independently. Additionally, temporal ensembles tend to have smaller potential population sizes than evolutionary ensembles as member generation is dependent on cycle length and the total training epoch budget. The members of certain temporal ensembles can struggle with diversity as model checkpoints taken later in the training process tend to be highly correlated.

C. Evolutionary Ensembles

Evolutionary computation has been a popular alternative to gradient based optimization for decades. Evolutionary methods utilize concepts inspired by biological evolution like natural selection, mutation, and recombination to optimize populations over time. The usage of populations in evolutionary algorithms leads to a natural connection to ensemble learning [7]. We describe low-cost ensemble methods as evolutionary if the primary idea is to spawn new child networks that inherit parameters or network structure from some parent(s) as a result of evolutionary operators.

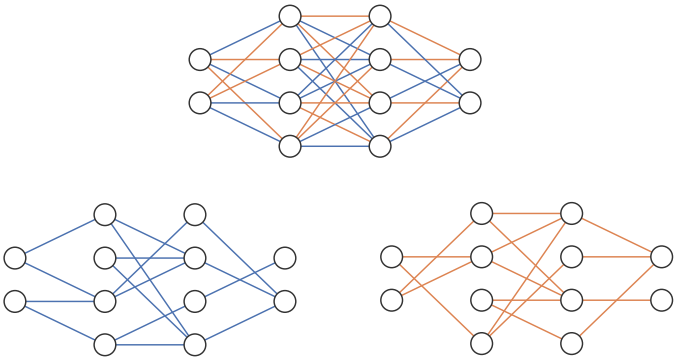

Fig. 2. Prune and Tune Ensembles generate diverse children with unique network connectivity by pruning random parameters from a shared and fully trained parent.

Traditional evolutionary methods have mostly been concerned with single hypothesis testing, which means that optimization is geared towards selecting a single network with the best generalization performance from a population [7], [23]. Evolution Strategies are the most popular of these, which describes a family of algorithms that generate child networks by selecting the best models according to fitness rankings and mutating them with noise perturbations [28]. Several variations of evolution strategies have been introduced such as Natural Evolution Strategies (NES) and Covariance Matrix Adaptation Evolution Strategies (CMA-ES). NES generates new populations according to a weighted average over the previous population distribution, where the models are weighted according to their fitness [41]. NES approximates a natural gradient as mutations encourage models to move in the direction of previously successful models. CMA-ES is similar to NES, but uses an incrementally updating covariance matrix to better explore the state space [10]. The covariance matrix uses pairwise fitness rankings to skew distributions rather than a simple weighted average, however this is extremely costly with high dimensional spaces.

Evolutionary methods have also explored mutations to the network architecture itself. Neuroevolution of Augmenting Topologies (NEAT) is perhaps the most popular of these methods, which starts with a small and simple population that is slowly complexified over time with additional neurons and connections, incorporating concepts like speciation to encourage diversity [32]. Weight Agnostic Neural Networks have shown that evolution of network architectures with a single shared value for all weights can encode solutions to reinforcement learning problems [8].

While evolution is highly scalable due to its distributed nature, it tends to be extremely sample inefficient. Modern neural networks can contain billions of parameters which makes gradient based optimization much more efficient. Recent evolutionary works have aimed to combine gradient optimization with evolutionary ideas.

Evolutionary Ensembles with Negative Correlation Learning (EENCL) incorporates a standard gradient based training cycle for each evolutionary generation. Multiple independent networks are trained simultaneously with the inclusion of a correlation penalty error term to encourage diversity. At each generation, the fittest individuals are selected as parents for the next generation, and child networks are spawned with normally distributed weight perturbations [23].

It's only recently that low-cost methods have been introduced that build on some of these evolutionary principles. MotherNets and Prune and Tune Ensembles are two such prominent examples, which both work by first training a single parent network and then spawning children with mutated network architectures that are then further optimized.

MotherNets "hatch" children from a small, trained network where additional neurons and layers are added around this trained core network. The weights of the child networks are adjusted using function preserving network morphisms, in order to ensure that the added layers and neurons do not catastrophically affect the core networks functionality. Diversity is encouraged by adding different amounts of neurons and layers to each child [38].

Prune and Tune Ensembles (PAT) instead spawn children by cloning and randomly pruning a large, trained network. Each child ends up with a unique network connectivity as a result of pruning. The children all converge in very few epochs as they are derived from a fully trained parent network. Optimization of the children is done with a cyclic learning rate schedule to encourage more diversity between siblings [40].

Pure evolutionary methods have struggled with the large dimensionality of modern deep neural networks. These recent low-cost methods (MotherNets and PAT) offer a glimpse of how a hybrid of gradient based optimization and evolutionary ideas can result in fast training and excellent diversity. These low-cost evolutionary methods offer a unique advantage over other approaches in that it is extremely cheap to generate new ensemble members. Once a parent network is trained, child networks tend to converge quickly while maintaining diversity thanks to unique network connectivity and learning rate schedules that encourage divergent movement in the parameter space. Additionally, the decoupled nature of child generation and parent training enables more flexibility as pre-trained models could be used to save significant amounts of compute. Temporal Ensembles and Pseudo-Ensembles have necessary architectural or training requirements that limits their flexibility.

III. Diversity in Low-Cost Ensembles

Diversity has long been known to be an important consideration in ensemble learning [2], [20], [36]. This is often explained in ensemble literature with an example of the bias-variance decomposition of the mean squared error (MSE) [2], [36]. 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. Bias is the difference between the expectation of the model and the true targets and variance is the squared difference between the models predictions and its mean.

Fig. 3. A suggested approach for interpretable diversity analysis. Displayed is a collection of feature visualizations for specific neurons selected from child networks in a Snapshot Ensemble and a Prune and Tune Ensemble. Identical parent networks were used for both ensemble methods and each child is optimized for the same number of epochs. The same neurons were chosen in each model for visualization. Snapshot Ensemble networks appear to be much more correlated than Prune and Tune children, suggesting that altered network structure in evolutionary ensembles can significantly impact diversity.

$$bias = E[f] - y$$
$$var = E[(f - E[f])^2]$$
$$MSE = E[(f-y)^2] = (E[f]-y)^2 + E[(f-E[f])^2]$$
$$MSE = bias^2 + var$$

Given an ensemble of M equally weighted estimators, the decomposition can be further extended to produce the bias-variance-covariance decomposition [2].

$$\overline{bias} = \frac{1}{M}\sum_i (E[f_i] - y)$$
$$\overline{var} = \frac{1}{M}\sum_i E[(f_i - E[f_i])^2]$$
$$\overline{covar} = \frac{1}{M(M-1)}\sum_i \sum_{j\neq i} E[(f_i - E[f_i])(f_j - E[f_j])]$$
$$MSE = \overline{bias^2} + \frac{1}{M}\overline{var} + (1 - \frac{1}{M})\overline{covar}$$

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 [2], [36].

Several metrics have been used to quantify diversity in classification ensembles [20]. The most popular of which are the Pairwise Output Correlation, Kullback-Leibler Divergence, and Prediction Disagreement Ratio.

Pairwise Output Correlation: The average correlation between the raw outputs for each pair of models in the ensemble.

$$d_{corr}(f_1, f_2) = \frac{1}{N}\sum_{i=1}^{N} \frac{cov(f_1(x_i), f_2(x_i))}{\sigma_{f_1(x_i)}\sigma_{f_2(x_i)}}$$

where N is the number of test samples, $cov(f_1(x_i), f_2(x_i))$ is the covariance between two network output vectors for input x_i, and $\sigma_{f(x_i)}$ is the standard deviation of the output vector.

Kullback-Leibler Divergence: Also known as relative entropy, KL Divergence approximately measure 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 for each sample, PDR measures the pairwise ratio of disagreements between the predicted classes of models.

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

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.

These diversity metrics all focus on differences between the outputs of ensemble members. However, one limitation of output diversity is that it is intrinsically tied to model accuracy [20]. An ensemble can display great diversity but perform much worse than another ensemble if the accuracy of individual members is slightly worse. So, while these metrics are fairly standardized across recent low-cost ensemble methods, it can be hard to fairly compare different methods based solely on diversity without also taking model performance into account.

A. Interpretable Diversity Analysis

A growing trend in machine learning research is the study of neural network interpretability [25]. This field encompasses techniques that allow for human visualization and interpretation of the internal feature representations of neural networks.

Feature visualization is the most popular interpretability method, where the pixel values of an input image are optimized to produce an output that maximizes activations of specific neurons, filters, or layers of a network [27].

This was originally done with gradient descent on the raw pixels of a randomized input image. However, this resulted in images with a lot of high frequency noise and nonsensical patterns [27]. To alleviate this, better visualizations were created with regularization techniques like transformation augmentations, frequency penalizations, and Fourier decorrelation [27].

Figure 3 illustrates an example of how feature visualizations could be useful in investigating diversity between child networks from two different low-cost ensemble algorithms. A pretrained Inception-V3 model [35] is used as a parent network for both a snapshot ensemble and prune and tune ensemble. The snapshot ensemble continues training the parent for two snapshot cycles of 20 epochs each with the child models being saved at the end of each cycle. The prune and tune children are created by cloning and pruning 50% of the parameters of the parent and continued training for 20 epochs each using a one cycle learning rate.

Since the same parent network is used for both methods and all child networks inherit parameters from the same model, we can visualize the same neurons in each model to explore how the diversity develops at the feature level in these methods. A random selection of neurons were chosen, and feature visualizations were created using the open source pytorch library lucent [18]. From a qualitative standpoint, the feature visualizations in snapshot ensembles appear to be much more correlated than those from prune and tune ensembles, suggesting that network architecture can significantly impact the diversity of networks derived from identical parents.

There is potentially great value in applying these kinds of techniques to low-cost ensemble methods to visualize how diversity develops and is represented among different ensemble members that share network structure or parameters.

IV. Experiments

Most low-cost ensemble papers include a standardized benchmark classification task on the CIFAR-10 and CIFAR-100 datasets with a modern wide residual network architecture. There are notably fewer results on the much larger scale ImageNet dataset, likely due to the much higher computational costs and resources required.

A. Datasets

CIFAR: consists of 60,000 small natural colored images of 32x32 pixels in size. Those 60,000 images are split up into 50,000 training images and 10,000 testing images. CIFAR-10 samples from 10 classes of images, while CIFAR-100 samples from 100 classes of images. CIFAR-100 is more difficult than CIFAR-10 as each class will have only 500 training samples compared to 5,000 in CIFAR-10 [19].

ImageNet: is a much larger scale dataset consisting of over 14 million images corresponding to over 100,000 class labels, called synonym sets, from the WordNet project. The most commonly used subset of ImageNet, ILSVRC 2012, was introduced in a seminal computer vision competition and is now the standard for large scale image classification projects. This subset consists of 1,281,167 full color images that correspond to one of 1,000 different class labels. A validation set is publicly available that consists of 50,000 images. This subset is used for all reported results and all images are resized to 224x224 pixels [4].

Corrupted Datasets: In addition, these methods are also evaluated on corrupted versions of CIFAR-10, CIFAR-100 and ImageNet in order to evaluate the robustness to various perturbations [13]. These additional test sets are generated by adding 20 different kinds of image corruptions (gaussian noise, snow, blur, pixelation, etc.) at five different levels of severity to the original CIFAR and ImageNet test sets. The total number of images in each of the corrupted CIFAR sets is 1,000,000 and the total number of images in corrupted ImageNet is 5,000,000.

B. Models

ResNet [12]: ResNets are one of the most popular convolutional vision architectures in deep learning. They popularized skip connections between groups of layers in order to improve gradient flow and enable networks to grow much deeper than was previously possible. The ImageNet experiments use ResNet-50, a 50 layer version that contains $\sim$ 25 million parameters [12].

Wide ResNet [45]: Wide ResNets were introduced as a variant to the original ResNet, addressing the problem of diminishing feature reuse in deep residual networks. Wide ResNets use much wider convolutional layers in the residual blocks, which demonstrate better performance with significantly fewer layers than their ResNet counterparts. The CIFAR experiments use WRN-28-10, a 28 layer variant that contains $\sim$ 36 million parameters [45].

C. Method Details

All methods are trained with Stochastic Gradient Descent with Nesterov momentum $\mu = 0.9$ and weight decay $\gamma = 0.0005$ [33]. Unless stated otherwise, all methods use a step-wise decay schedule for the learning rate. An initial learning rate of $\eta_1 = 0.1$ is used for 50% of the training budget which decays linearly to $\eta_2 = 0.001$ at 90% of the training budget. The learning rate is kept constant at $\eta_2 = 0.001$ for the final 10% of training. For CIFAR, a batch size of 128 is used for training and random crop, random horizontal flip, and mean standard scaling data augmentations are applied for all approaches. For ImageNet, a batch size of 4096 is used. Data is resized to 256x256, center cropped to 224x224 and scaled with mean standard normalization.

Table II. Results for ensembles of WideResNet-28-10 models on both CIFAR-10 and CIFAR-100 as well as ensembles of ResNet-50 on ImageNet. Results obtained from [11], [22], [40]. cAcc, cNLL, and cECE correspond to corrupted test sets.

Methods (CIFAR-10/WRN-28-10)	Acc ↑	NLL ↓	ECE ↓	cAcc ↑	cNLL ↓	cECE ↓	FLOPs ↓	Epochs ↓
Independent Model	96.0	0.159	0.023	76.1	1.050	0.153	3.6e17	200
Monte Carlo Dropout	95.9	0.160	0.024	68.8	1.270	0.166	1.00x	200
TreeNet (M=3)	95.9	0.258	0.018	75.5	0.969	0.137	1.52x	250
SSE (M=5)	96.3	0.131	0.015	76.0	1.060	0.121	1.00x	200
FGE (M=12)	96.3	0.126	0.015	75.4	1.157	0.122	1.00x	200
PAT (M=6) (AR + 1C)	96.5	0.113	0.005	76.2	0.972	0.081	0.85x	200
BatchEnsemble (M=4)	96.2	0.143	0.021	77.5	1.020	0.129	4.40x	250
MIMO (M=3)	96.4	0.123	0.010	76.6	0.927	0.112	4.00x	250
EDST (M=7)	96.4	0.127	0.012	76.7	0.880	0.100	0.57x	850
DST (M=3)	96.4	0.124	0.011	77.6	0.840	0.090	1.01x	750
Dense Ensemble (M=4)	96.6	0.114	0.010	77.9	0.810	0.087	1.00x	800

Methods (CIFAR-100/WRN-28-10)	Acc ↑	NLL ↓	ECE ↓	cAcc ↑	cNLL ↓	cECE ↓	FLOPs ↓	Epochs ↓
Independent Model	79.8	0.875	0.086	51.4	2.700	0.239	3.6e17	200
Monte Carlo Dropout	79.6	0.830	0.050	42.6	2.900	0.202	1.00x	200
TreeNet (M=3)	80.8	0.777	0.047	53.5	2.295	0.176	1.52x	250
SSE (M=5)	82.1	0.661	0.040	52.2	2.595	0.145	1.00x	200
FGE (M=12)	82.3	0.653	0.038	51.7	2.638	0.137	1.00x	200
PAT (M=6) (AR + 1C)	82.7	0.634	0.013	52.7	2.487	0.131	0.85x	200
BatchEnsemble (M=4)	81.5	0.740	0.056	54.1	2.490	0.191	4.40x	250
MIMO (M=3)	82.0	0.690	0.022	53.7	2.284	0.129	4.00x	250
EDST (M=7)	82.6	0.653	0.036	52.7	2.410	0.170	0.57x	850
DST (M=3)	82.8	0.633	0.026	54.3	2.280	0.140	1.01x	750
Dense Ensemble (M=4)	82.7	0.666	0.021	54.1	2.270	0.138	1.00x	800

Methods (ImageNet/ResNet50)	Acc ↑	NLL ↓	ECE ↓	cAcc ↑	cNLL ↓	cECE ↓	FLOPs ↓	Epochs ↓
Independent Model	76.1	0.943	0.039	40.5	3.200	0.105	4.8e18	90
TreeNet (M=2)	78.1	0.852	0.017	42.4	3.052	0.073	1.33x	150
BatchEnsemble (M=4)	76.7	0.944	0.050	41.8	3.180	0.110	4.40x	135
MIMO (M=2)	77.5	0.887	0.037	43.3	3.030	0.106	2.00x	150
DST (M=2)	78.3	0.914	0.060	43.7	2.910	0.057	1.12x	400
EDST (M=4)	77.7	0.935	0.064	42.6	2.987	0.058	0.87x	310
Dense Ensemble (M=4)	77.5	0.877	0.031	42.1	2.990	0.051	1.00x	360

[9], [11], [15], [22]. The **Independent Model** is a baseline single model result. The **Dropout Model** includes dropout layers between convolutional layers in the residual blocks at a rate of 30% for the CIFAR experiments and 10% for the ImageNet experiments. **Snapshot Ensembles** (SSE) use a cosine annealing learning rate with an initial learning rate $\eta = 0.1$ for a cycle length of 40 epochs [15]. **Fast Geometric Ensembles** (FGE) use a pre-training routine for 156 epochs. A curve finding algorithm then runs for 22 epochs with a cycle length of 4, each starting from checkpoints at epoch 120 and 156. **TreeNets**, [21], **BatchEnsemble** is trained for 250 epochs on CIFAR and 135 epochs on ImageNet [39]. **MIMO** and **TreeNet** [11] are trained for 250 epochs on CIFAR and 150 epochs on ImageNet. **FreeTickets** introduces several configurations for building ensembles. We include their two best configurations for Dynamic Sparse Training (DST, M=3, S=0.8) and Efficient Dynamic Sparse Training (EDST, M=7, S=0.9) on CIFAR and (DST, M=2, S=0.8) and (EDST, M=4, S=0.8) on ImageNet. **Prune and Tune Ensembles** (PAT) train a single parent network for 140 epochs. Six children are created with anti-random pruning (50% sparsity) and tuned with a one-cycle learning rate for 10 epochs. The tuning schedule starts at $\eta_1 = 0.001$, increases to $\eta_2 = 0.1$ at 1 epoch and then decays to $\eta_3 = 1e-7$ using cosine annealing for the final 9 epochs.

D. Metrics

All methods include results for Accuracy (Acc), Negative Log Likelihood (NLL), and Expected Calibration Error (ECE). Results on the corrupted datasets are prepended with c (cAcc, cNLL, cECE). The total number of floating point operations (FLOPs) and training epochs are also reported.

$$Acc = \frac{1}{N}\sum_{i}^{N} \mathbb{I}(\hat{y}_i = y_i)$$

where N is the number of samples, $\hat{y}$ is the predicted class label, and y is the actual class label.

$$NLL = \frac{1}{N}\sum_{i}^{N}(-\sum_{c}^{C} log\frac{exp(\hat{y}_{n,y_n})}{\sum_c exp(\hat{y}_{n,c})}y_{n,c})$$

Table III. Prediction Disagreement Ratio (PDR) and KL divergence between ensemble members on CIFAR-10 with WideResNet-28x10. Results reported from [11], [22], [40]

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
DST Ensemble	0.035	0.095	96.4
Prune and Tune Ensemble	0.036	0.090	96.5

where N is the number of samples, C is the number of classes, $y_{n,\hat{y_n}}$ is the predicted value for the correct class y_n, $y_{n,c}^{\hat{}}$ is the predicted output for class c, and $y_{n,c}$ is the actual output for class c.

$$ECE = \sum_{b} \frac{n_b}{N} |acc(b) - conf(b)|$$

where n_b is the number of predictions in bin b, N is the total number of samples, and acc(b) and conf(b) are the accuracy and confidence of bin b respectively [26].

A subset of methods also report prediction disagreement ratio and KL divergence between ensemble members on the CIFAR-10 experiment.

V. Limitations and Future Work

Low-cost ensemble algorithms vary significantly in how training time is organized. We find that there is a lot of variation in population size, total FLOP(s), and total training epochs amongst methods, with several papers reporting and comparing results that do not use identical training budgets. In all cases the training procedures are transparent, however it can be hard to fairly assign significance due to the differences in training configuration. Additionally, the time and resource intensive nature of ImageNet has led to a lack of results on this benchmark compared to the small scale and easily accessible CIFAR datasets.

Low-cost ensemble research primarily focuses on image classification with ResNets and Wide ResNets. New model architectures like Vision Transformers are reporting state of the art results and growing in popularity in computer vision [6]. There is a lot of value in exploring the efficacy of low-cost ensembles with novel architectures and in completely different domains like natural language processing, reinforcement learning or continual learning.

The current state of the art computer vision models are often pretrained with unsupervised learning on massive datasets like the JFT-300M and JFT-3B [46]. These large scale datasets could have different training and generalization dynamics that would be interesting to explore with low-cost ensemble methods.

There is also a lot of potential for further exploring diversity in low-cost ensembles in a standardized and systematic way. There are serious limitations with output diversity metrics as they give little insight into how and why different predictions are made.2 There is a lot of potential for a new qualitative approach to diversity analysis that could lead to valuable insights about how diversity develops or could be encouraged in low-cost ensembles.

VI. Conclusion

Ensemble learning has long been known to be an effective and powerful way to improve performance on a number of machine learning tasks. As deep neural networks have grown in popularity and scale, the computational costs associated with training several independent networks from scratch has become expensive. Low-cost ensemble methods have introduced ways to share information between members that can significantly reduce computational cost while retaining the benefits to generalization that ensemble learning affords.

This paper aims to provide a comprehensive survey of the state of low-cost ensemble learning. This paper discusses, summarizes, and categorizes a number of popular methods under a cost reduction taxonomy consisting of pseudo-ensembles, temporal ensembles, and evolutionary ensembles. Pseudo-ensembles train multiple members as if they were a part of a single monolithic architecture. Temporal ensembles train a single model and save snapshots of it throughout time. Evolutionary ensembles spawn new children with inherited parameters from parents using evolutionary operators like mutation, selection, or recombination. Each of these categories has benefits and drawbacks according to a balance between population size, training efficiency, storage cost, inference cost, and implementation flexibility.

Ultimately, cost is able to be reduced by sharing information between members. This invariably reduces diversity, which is a significant factor in the generalization performance of ensembles. We discuss the importance of diversity in these papers and we introduce interpretable diversity analysis as a promising avenue for future research. We include several benchmark experiments on CIFAR and ImageNet and discuss some limitations with fairly comparing the various methods.

Low-Cost Ensemble Learning is a really interesting field with significant potential impact. These low-cost methods improve generalization performance on a wide variety of tasks while making ensemble learning more accessible for researchers and practitioners. The ideas introduced in these papers leave a solid foundation for future research and they become more important as machine learning projects continue to grow in scale.

References

[1] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles, 2014.

[2] Gavin Brown, Jeremy L. Wyatt, and Peter Ti328;o. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, 6(55):1621–1650, 2005.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[7] Christian Gagné, Michèle Sebag, Marc Schoenauer, and Marco Tomassini. Ensemble learning for free with evolutionary algorithms ? 2007.

[8] Adam Gaier and David Ha. Weight agnostic neural networks, 2019.

[9] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *arXiv preprint:1802.10026*, 2018.

[10] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[11] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M. Dai, and Dustin Tran. Training independent subnetworks for robust prediction, 2021.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint: 1503.02531*, 2015.

[15] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free, 2017.

[16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, pages 646–661, 2016.

[17] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2018.

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

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

[20] Ludmila Kuncheva and Chris Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181–207, 05 2003.

[21] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks, 2015.

[22] Shiwei Liu, Tianlong Chen, Zahra Atashgahi, Xiaohan Chen, Ghada Sokar, Elena Mocanu, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Freetickets: Accurate, robust and efficient deep ensemble by training with dynamic sparsity, 2021.

[23] Yong Liu, Xin Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[24] Gábor Melis. Dissecting the winning solution of the higgsml challenge. In Glen Cowan, Cécile Germain, Isabelle Guyon, Balázs Kégl, and David Rousseau, editors, *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, volume 42 of *Proceedings of Machine Learning Research*, pages 57–67, Montreal, Canada, 13 Dec 2015. PMLR.

[25] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[26] Jeremy Nixon, Mike Dusenberry, Ghassen Jerfel, Timothy Nguyen, Jeremiah Liu, Linchuan Zhang, and Dustin Tran. Measuring calibration in deep learning, 2019.

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

[28] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

[29] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[30] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model, 2022.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

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

[33] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

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

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

[36] 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.

[37] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[38] Abdul Wasay, Brian Hentschel, Yuze Liao, Sanyuan Chen, and Stratos Idreos. Mothernets: Rapid deep ensemble learning. *arXiv preprint:1809.04270*, 2018.

[39] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: An alternative approach to efficient ensemble and lifelong learning, 2020.

[40] Tim Whitaker and Darrell Whitley. Prune and tune ensembles: Low-cost ensemble learning with sparse independent subnetworks, 2022.

[41] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, and Jürgen Schmidhuber. Natural evolution strategies, 2011.

[42] Russell D. Wolfinger and Pei-Yi Tan. Stacked ensemble models for improved prediction accuracy. In *SAS Global Forum 2017 Proceedings*, 2017.

[43] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022.

[44] Jingjing Xie, Bing Xu, and Zhang Chuang. Horizontal and vertical ensemble with deep representation for classification. *arXiv preprint: 1306.2759*, 2013.

[45] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint: 1605.07146*, 2016.

[46] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12104–12113, June 2022.