Learning Neural Network Subspaces

Mitchell Wortsman [1] **Maxwell Horton** [2] **Carlos Guestrin** [2] **Ali Farhadi** [2] **Mohammad Rastegari** [2]

Abstract

Recent observations have advanced our understanding of the neural network optimization landscape, revealing the existence of (1) paths of high accuracy containing diverse solutions and (2) wider minima offering improved performance. Previous methods observing diverse paths require multiple training runs. In contrast we aim to leverage both property (1) and (2) with a single method and in a single training run. With a similar computational cost as training one model, we learn lines, curves, and simplexes of high-accuracy neural networks. These neural network subspaces contain diverse solutions that can be ensembled, approaching the ensemble performance of independently trained networks without the training cost. Moreover, using the subspace midpoint boosts accuracy, calibration, and robustness to label noise, outperforming Stochastic Weight Averaging.

1. Introduction

Optimizing a neural network is often conceptualized as finding a minimum in an objective landscape. Therefore, understanding the geometric properties of this landscape has emerged as an important goal. Recent work has illuminated many intriguing phenomena. Garipov et al. (2018); Draxler et al. (2018) determine that independently trained models are connected by a curve in weight space along which loss remains low. Additionally, Frankle et al. (2020) demonstrate that networks which share only a few epochs of their optimization trajectory are connected by a linear path of high accuracy. However, the connected regions in weight space found by Garipov et al. (2018); Draxler et al. (2018); Frankle et al. (2020) require approximately twice the training time compared with standard training, as two separate minima are first identified then connected.

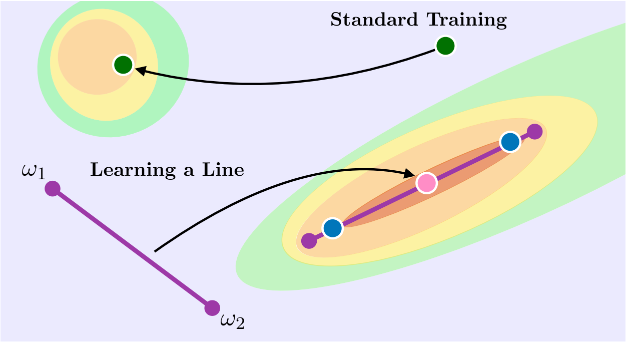

Figure 1. Schematic for learning a line of neural networks compared with standard training. The midpoint outperforms standard training in terms of accuracy, calibration, and robustness. Models near the endpoints enable high-accuracy ensembles in a single training run.

This work is motivated by the existence of connected, functionally diverse regions in solution space. In contrast to prior work, our aim is to directly parameterize and learn these neural network subspaces from scratch in a single training run. For instance, when training a line (Figure 1) we begin with two randomly initialized endpoints and consider the neural networks on the linear path which connects them. At each iteration we use a randomly sampled network from the line, backpropagating the training loss to update the endpoints. Central to our method is a regularization term which encourages orthogonality between the endpoints, just as two independently trained networks are orthogonal (Fort et al., 2019). When the line settles into a low loss region we find that models from opposing ends are functionally diverse.

In addition to lines, we learn curves and simplexes of high-accuracy neural networks (Figure 2). We also uncover benefits beyond functional diversity. Lines and simplexes identify and traverse large flat minima, with endpoints near the periphery. The midpoint corresponds to a less sharp solution, which is associated with better generalization (Dziugaite & Roy, 2018). Using this midpoint corresponds to ensembling in weight space, producing a single model which requires no additional compute during inference. We find that taking the midpoint of a simplex can boost accuracy, calibration, and robustness to label noise.

The rest of the paper is organized via the following contributions:

[1] University of Washington (work completed during internship at Apple). [2] Apple. Correspondence to: Mitchell Wortsman <mitchnw@cs.washington.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

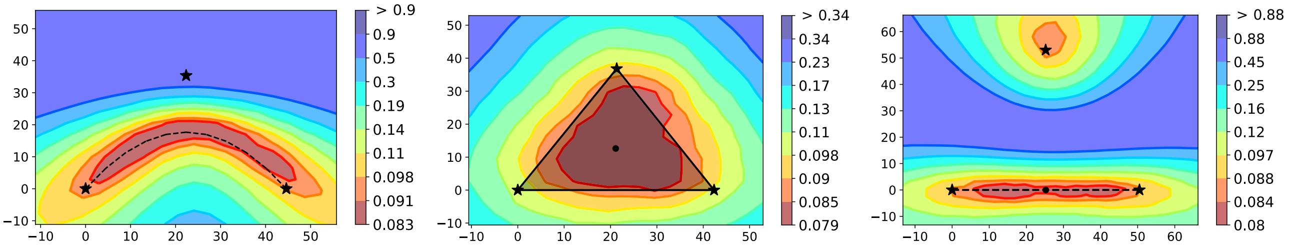

Figure 2. Test error on a two dimensional plane for three learned subspaces for cResNet20 (CIFAR10)—a quadratic Bezier curve (left), a simplex with three endpoints (middle), and a line (right). The subspace parameters ω_1, ω_2 and ω_3 are plotted and used to construct the plane, except for the line for which ω_3 was taken to be a solution obtained via standard training. Note that although ω_3 is used to define the Bezier curve (left), it never passes through it. Visualization as in Garipov et al. (2018) with ω_1 at the origin.

1. We contextualize our work via 5 observations regarding the objective landscape (section 2).
2. We introduce a method for learning diverse and high-accuracy lines, curves, and simplexes of neural networks (section 3).
3. We show that lines and curves found in a single training run contain models that approach or match the ensemble accuracy of independently trained networks (subsection 4.2).
4. We find that taking the midpoint of a simplex provides a boost in accuracy, calibration, and robustness (subsection 4.3; subsection 4.4).

2. Preliminaries and Related Methods

We highlight a few recent observations which have advanced understanding of the neural network optimization landscape (Dauphin et al., 2014; Li et al., 2018a;b; Fort & Jastrzebski, 2019; Evci et al., 2019; Frankle, 2020; Oswald et al., 2021). We remain in the setting of image classification with setup and notation drawn from Frankle et al. (2020).

Consider a neural network $f(\mathbf{x}, \theta)$ with input $\mathbf{x}$ and parameters $\theta \in \mathbb{R}^n$. For initial random weights θ_0 and SGD randomness ξ, the weights at epoch t are given by $\theta_t = \mathsf{Train}^{0 \to t}(\theta_0, \xi)$. Additionally let $\mathsf{Acc}(\theta)$ denote the test accuracy of network f with parameters θ. The first three observations pertain to the setting where two networks are trained with different SGD noise—consider $\theta_T^1 = \mathsf{Train}^{0 \to T}(\theta_0, \xi_1)$ and $\theta_T^2 = \mathsf{Train}^{0 \to T}(\theta_0, \xi_2)$. The observations are unchanged when θ_T^1 and θ_T^2 have differing initializations.

Observation 1. (Lakshminarayanan et al., 2017) Ensembling θ_T^1 and θ_T^2 in output space—making predictions $\hat{\mathbf{y}} = \frac{1}{2}\left(f\left(\mathbf{x}, \theta_T^1\right) + f\left(\mathbf{x}, \theta_T^2\right)\right)$—boosts accuracy, calibration, and robustness. This is attributed to functional diversity meaning $f\left(\cdot, \theta_T^1\right)$ and $f\left(\cdot, \theta_T^2\right)$ make different errors.

Observation 2. (Frankle et al., 2020; Fort et al., 2020) Ensembling θ_T^1 and θ_T^2 in weight space—making predictions with the network $f\left(\mathbf{x}, \frac{1}{2}\left(\theta_T^1 + \theta_T^2\right)\right)$—fails, achieving no better accuracy than an untrained network.

Definition 1. A *connector* between neural network weights $\psi_1, \psi_2 \in \mathbb{R}^n$ is a continuous function $\mathsf{P} : [0,1] \to \mathbb{R}^n$ such that $\mathsf{P}(0) = \psi_1$, $\mathsf{P}(1) = \psi_2$, and the average accuracy along the connector is at least the average accuracy given by the weights at the endpoints. Equivalently, if $\mathcal{U}$ denotes the uniform distribution then $\mathbb{E}_{\alpha \sim \mathcal{U}([0,1])}[\mathsf{Acc}(\mathsf{P}(\alpha))] \gtrapprox \frac{1}{2}(\mathsf{Acc}(\psi_1) + \mathsf{Acc}(\psi_2))$. In the language of *connectors*, Observation 2 states that there does not exist a *linear* connector between θ_T^1 and θ_T^2.

Observation 3. (Garipov et al., 2018; Draxler et al., 2018) There exists a *nonlinear* connector P between θ_1^T and θ_2^T, for instance a quadratic Bezier curve.

Observation 4. (Frankle et al., 2020) There exists a *linear* connector when part of the optimization trajectory is shared. Instead of branching off at θ_0, let $\theta_k = \mathsf{Train}^{0 \to k}(\theta_0, \xi)$ and consider $\theta_{k \to T}^i = \mathsf{Train}^{k \to T}(\theta_k, \xi_i)$ for $i \in \{1, 2\}$. For $k \ll T$, $\mathsf{P}(\alpha) = (1 - \alpha)\theta_{k \to T}^1 + \alpha\theta_{k \to T}^2$ is a linear connector.

Observation 4 generalizes to the higher dimensional case (Appendix H) for which a convex hull of neural networks attains high accuracy. To consider higher dimensional connectors we discuss one additional definition. Let $\mathcal{U}\left(\Delta^{m-1}\right)$ refer to the uniform distribution on $\Delta^{m-1} = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \sum_i \boldsymbol{\alpha}_i = 1, \boldsymbol{\alpha}_i \geq 0\}$ and let $\mathbf{e}_i$ refer to the standard basis vector (all zeros except for position i which is 1). Note that Δ^{m-1} is often referred to as the $m-1$ dimensional probability simplex.

Definition 2. An *m-connector* on $\psi_1, ..., \psi_m \in \mathbb{R}^n$ is a continuous function $\mathsf{P} : \Delta^{m-1} \to \mathbb{R}^n$ such that $\mathsf{P}(\mathbf{e}_i) = \psi_i$ and $\mathbb{E}_{\boldsymbol{\alpha} \sim \mathcal{U}(\Delta^{m-1})}[\mathsf{Acc}(\mathsf{P}(\boldsymbol{\alpha}))] \gtrapprox \frac{1}{m}\sum_{i=1}^m \mathsf{Acc}(\psi_i)$. This definition formalizes that in Fort & Jastrzebski (2019). In this work we will primarily focus on linear m-connectors which have the form $\mathsf{P}(\boldsymbol{\alpha}) = \sum_i \boldsymbol{\alpha}_i \psi_i$.

Linear m-connectors are implicitly used by Izmailov et al. (2018) in *Stochastic Weight Averaging (SWA)*. SWA uses a high constant (or cyclic) learning rate towards the end of training to bounce around a minimum while occasionally saving checkpoints. SWA returns the weight space ensemble (average) of these models, motivated by the observation that SGD solutions often lie at the edge of a minimum and averaging moves towards the center. The averaged solution is less sharp, which may lead to better generalization (Chaudhari et al., 2019; Dziugaite & Roy, 2018; Foret et al., 2020).

Observation 5. (Izmailov et al., 2018) If weights $\psi_1,...\psi_m$ lie at the periphery of wide and flat low loss region, then $\mathsf{Acc}\left(\frac{1}{m}\sum_{i=1}^m \psi_i\right) > \frac{1}{m}\sum_{i=1}^m \mathsf{Acc}(\psi_i)$.

SWA is extended by SWA-Gaussian (Maddox et al., 2019) (which fits a Gaussian to the saved checkpoints) and Izmailov et al. (2020) (who considers the subspace which they span). These techniques advance Bayesian deep learning—methods which aim to learn a distribution over the parameters. Other Bayesian apporaches include variational methods (Blundell et al., 2015), MC-dropout (Gal & Ghahramani, 2016), and MCMC methods (Welling & Teh, 2011; Zhang et al., 2020). However, variational methods tend not to scale to larger networks such as residual networks (Maddox et al., 2019). Moreover, a detailed empirical study by Fort et al. (2019) recently observed that many Bayesian models tend to capture the local uncertainty of a single mode but are much less functionally diverse than independently trained networks which identify multiple modes. Ensembling models sampled from the learned distribution is therefore inferior in terms of accuracy and robustness.

Other related techniques include *Snapshot Ensembles (SSE)* (Huang et al., 2017) which use a cyclical learning rate with multiple restarts, saving checkpoints prior to each restart. Fast Geometric Ensembles (Garipov et al., 2018) employs a similar strategy but does not begin saving checkpoints until later in training. Other methods to efficiently train and evaluate ensembles include BatchE (Wen et al., 2020). Although their method is compelling, BatchE requires longer training for ensemble members to match standard training accuracy.

To summarize, connectors—high-accuracy subspaces of neural networks—have two useful properties:

- Property 1: They contain models which are functionally diverse and may be ensembled in output space (Observations 1 & 3).
- Property 2: Taking the midpoint of the subspace (ensembling in weight space) can improve accuracy and generalization (Observation 5).

Prior work satisfying Property 1 requires multiple training runs. Subspaces satisfying Property 2 yield solutions that are less functionally diverse (Fort et al., 2019). Our aim is to leverage both Property 1 and 2 in a single training run.

Algorithm 1 TrainSubspace

```
Input: $\mathsf{P}$ with domain $\Lambda$ and parameters $\{\omega_i\}_{i=1}^m$, network $f$, train set $\mathcal{S}$, loss $\ell$, and scalar $\beta$ (e.g. a line has $\Lambda = [0,1]$ and $\mathsf{P}(\alpha;\omega_1,\omega_2) = (1-\alpha)\omega_1 + \alpha\omega_2$).
Initialize each $\omega_i$ independently.
for batch $(\mathbf{x},\mathbf{y}) \subseteq \mathcal{S}$ do
    Sample $\boldsymbol{\alpha}$ uniformly from $\Lambda$.
    $\theta \leftarrow \mathsf{P}(\boldsymbol{\alpha};\{\omega_i\}_{i=1}^m)$
    $\hat{\mathbf{y}} \leftarrow f(\mathbf{x},\theta)$
    Sample $j,k$ from $\{1,...,m\}$ without replacement.
    $\mathcal{L} \leftarrow \ell(\hat{\mathbf{y}},\mathbf{y}) + \beta\cos^2(\omega_j,\omega_k)$
    Backprop $\mathcal{L}$ to each $\omega_i$ and update with SGD & momentum using estimate $\frac{\partial\mathcal{L}}{\partial\omega_i} = \frac{\partial\ell}{\partial\theta}\frac{\partial\mathsf{P}}{\partial\omega_i} + \beta\frac{\partial\cos^2(\omega_j,\omega_k)}{\partial\omega_i}$.
end for
```

3. Method

In a single training run, we find a connected region in solution space comprised of high-accuracy and diverse neural networks. To do so we directly parameterize and learn the parameters of a subspace.

First consider learning a line. Recall that the line between $\omega_1 \in \mathbb{R}^n$ and $\omega_2 \in \mathbb{R}^n$ in weight space is $\mathsf{P}(\alpha;\omega_1,\omega_2) = (1-\alpha)\omega_1 + \alpha\omega_2$ for α in the domain $\Lambda = [0,1]$. Our goal is to learn parameters ω_1, ω_2 such that $\mathsf{Acc}(\mathsf{P}(\alpha;\omega_1,\omega_2))$ is high for all values of $\alpha \in \Lambda$ ($\mathsf{Acc}(\theta)$ denotes the test accuracy of the neural network f with weights θ). Equivalently, our aim is to learn a high-accuracy connector between ω_1 and ω_2 (Definition 1).

More generally we consider subspaces defined by $\mathsf{P}(\cdot,\{\omega_i\}_{i=1}^m) : \Lambda \to \mathbb{R}^n$. We experiment with two shapes in addition to lines:

1. One-dimensional Bezier curves with a single bend $\mathsf{P}(\alpha;\omega_1,\omega_2,\omega_3) = (1-\alpha)^2\omega_1 + 2\alpha(1-\alpha)\omega_3 + \alpha^2\omega_2$ for $\alpha \in \Lambda = [0,1]$.
2. Simplexes with m endpoints $\{\omega_i\}_{i=1}^m$. A simplex is the convex hull defined by $\mathsf{P}(\boldsymbol{\alpha};\{\omega_i\}_{i=1}^m) = \sum_{i=1}^m \boldsymbol{\alpha}_i\omega_i$. The domain Λ for $\boldsymbol{\alpha}$ is the probability simplex $\{\boldsymbol{\alpha} \in \mathbb{R}^m : \sum_i \boldsymbol{\alpha}_i = 1, \boldsymbol{\alpha}_i \geq 0\}$.

Our training objective is to minimize the loss ℓ for all network weights θ such that $\theta = \mathsf{P}(\boldsymbol{\alpha},\{\omega_i\}_{i=1}^m)$ for some $\boldsymbol{\alpha} \in \Lambda$. Recall that for input $\mathbf{x}$ and weights θ a neural network produces output $\hat{\mathbf{y}} = f(\mathbf{x},\theta)$. Given the predicted label $\hat{\mathbf{y}}$ and true label $\mathbf{y}$ the training loss is a scalar $\ell(\hat{\mathbf{y}},\mathbf{y})$.

If we let $\mathcal{D}$ denote the data distribution and $\mathcal{U}(\Lambda)$ denote the uniform distribution over Λ, our training objective without regularization is to minimize

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\left[\mathbb{E}_{\boldsymbol{\alpha}\sim\mathcal{U}(\Lambda)}\left[\ell(f(\mathbf{x},\mathsf{P}(\boldsymbol{\alpha},\{\omega_i\}_{i=1}^m)),\mathbf{y})\right]\right]. \tag{1}$$

In practice we find that achieving significant functional diversity along the subspace requires adding a regularization term with strength β which we describe shortly. For now we proceed in the scenario where $\beta = 0$. Algorithm 1 is a stochastic approximation for the objective in Equation 1—we approximate the outer expectation with a batch of data and the inner expectation with a single sample from $\mathcal{U}(\Lambda)$.

Specifically, for each batch $(\mathbf{x}, \mathbf{y})$ we randomly sample $\boldsymbol{\alpha} \sim \mathcal{U}(\Lambda)$ and consider the loss

$$\ell(f(\mathbf{x}, \mathsf{P}(\boldsymbol{\alpha}, \{\omega_i\}_{i=1}^m)), \mathbf{y}). \tag{2}$$

If we let $\theta = \mathsf{P}(\boldsymbol{\alpha}, \{\omega_i\}_{i=1}^m)$ denote the single set of weights sampled from the subspace, we can calculate the gradient of each parameter ω_i as

$$\frac{\partial \ell}{\partial \omega_i} = \frac{\partial \ell}{\partial \theta} \frac{\partial \mathsf{P}(\boldsymbol{\alpha}, \{\omega_i\}_{i=1}^m)}{\partial \omega_i}. \tag{3}$$

The right hand side consists of two terms, the first of which appears in standard neural network training. The second term is computed using P. For instance, in the case of a line, the gradient for an endpoint ω_1 is

$$\frac{\partial \ell}{\partial \omega_1} = (1 - \boldsymbol{\alpha}) \frac{\partial \ell}{\partial \theta}. \tag{4}$$

Note that the gradient estimate for each ω_i is aligned but scaled differently. As is standard for training neural networks we use SGD with momentum. In Appendix A we examine Equation 1 in the simplified setting where the landscape is convex. In Appendix B we approximate the inner expectation of Equation 1 with multiple samples.

The method as described so far resembles Garipov et al. (2018), though we highlight some important differences. Garipov et al. (2018) begin by independently training two neural networks and subsequently learning a connector between them, considering curves and piecewise linear functions with fixed endpoints. Our method begins by initializing the subspace parameters randomly, using the same initialization as standard training (Kaiming normal (He et al., 2015)). The subspace is then fit in a single training run.

This contrasts significantly with standard training. For instance, when learning a simplex with m endpoints we begin with m random weight initializations and consider the subspace which they span. During training we move this entire subspace through the objective landscape.

Regularization. We have outlined a method to train high-accuracy subspaces of neural networks. However, as illustrated in subsection 4.2 (Figure 6), subspaces found without regularization do not contain models which achieve high accuracy when ensembled, suggesting limited functional diversity. To promote functional diversity, we want to encourage distance between the parameters $\{\omega_i\}_{i=1}^m$. Fort et al. (2019) show that independently trained models have weight vectors with a cosine similarity of approximately 0, unlike models with a shared trajectory. Therefore, we encourage all pairs ω_j, ω_k to have a cosine similarity of 0 by adding the following regularization term to the the training objective (Equation 1):

$$\beta \cdot \mathbb{E}_{j \neq k}\left[\cos^2(\omega_j, \omega_k)\right] = \beta \cdot \mathbb{E}_{j \neq k}\left[\frac{\langle \omega_j, \omega_k \rangle^2}{\|\omega_j\|_2^2 \|\omega_k\|_2^2}\right]. \tag{5}$$

In Algorithm 1 we approximate this expectation by sampling a random pair ω_j, ω_k for each training batch. Unless otherwise mentioned, β is set to a default value of 1. We do not consider L_2 distance since networks with batch normalization can often have weights arbitrarily scaled without changing their outputs.

Layerwise. Until now our investigation has been layer agnostic—we have treated neural networks as weight vectors in $\mathbb{R}^n$. However, networks have structure and connectivity which are integral to their success. Accordingly, we experiment with an additional stochastic approximation to Equation 1. Instead of approximating the inner expectation with a single sample $\boldsymbol{\alpha} \sim \mathcal{U}(\Lambda)$ we independently sample different values of $\boldsymbol{\alpha}$ for weights corresponding to different layers. In Appendix H we extend the analysis of Frankle et al. (2020) to this *layerwise* setting.

4. Results

In this section we present experimental results across benchmark datasets for image classification (CIFAR-10 (Krizhevsky et al., 2009), Tiny-ImageNet (Le & Yang, 2015), and ImageNet (Deng et al., 2009)) for various residual networks (He et al., 2016; Zagoruyko & Komodakis, 2016). Unless otherwise mentioned, β (Equation 5) is set to a default value of 1. The CIFAR-10 (Krizhevsky et al., 2009) and Tiny-ImageNet (Le & Yang, 2015) experiments follow Frankle et al. (2020) in training for 160 epochs using SGD with learning rate 0.1, momentum 0.9, weight decay 1e-4, and batch size 128. For ImageNet we follow Xie et al. (2019) in changing batch size to 256 and weight decay to 5e-5. All experiments are conducted with a cosine annealing learning rate scheduler (Loshchilov & Hutter, 2016) with 5 epochs of warmup and without further regularization (unless explicitly mentioned). When error bars are present the experiment is run with 3 random seeds and mean±std is shown. Additional details found in Appendix D, including SWA hyperparameters and the treatment of batch norm layers (which mirror SWA (Izmailov et al., 2018)). As discussed in subsection D.2, memory/FLOPs overhead is not significant as feature maps (inputs/outputs) are much larger than the number of parameters for convolutional networks. Code available at `https://github.com/apple/learning-subspaces`.

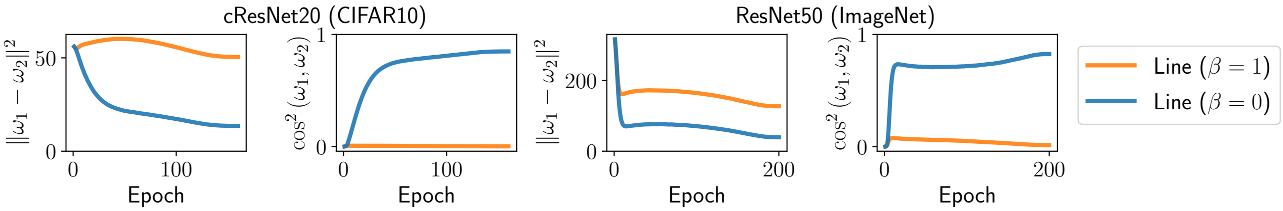

Figure 3. L_2 distance and squared cosine similarity between endpoints ω_1, ω_2 when training a line. β denotes the strength (scale factor) of the regularization term $\beta \cos^2(\omega_j, \omega_k) = \beta \langle \omega_1, \omega_2 \rangle^2 / \left(\|\omega_j\|_2^2 \|\omega_k\|_2^2\right)$ which is added to the loss to encourage large, diverse subspaces.

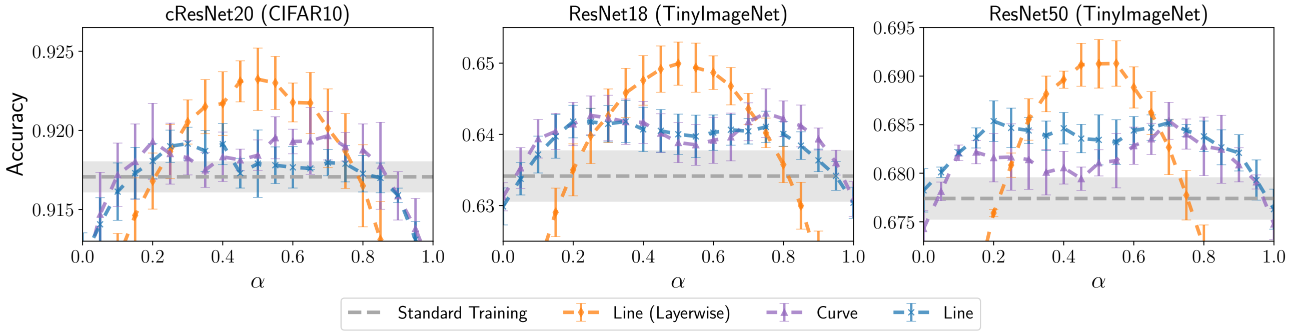

Figure 4. Visualizing model accuracy along one-dimensional subspaces. The accuracy of the model at point $\alpha \in [0, 1]$ along the subspace matches or exceeds standard training for a large section of the subspace (especially towards the subspace center).

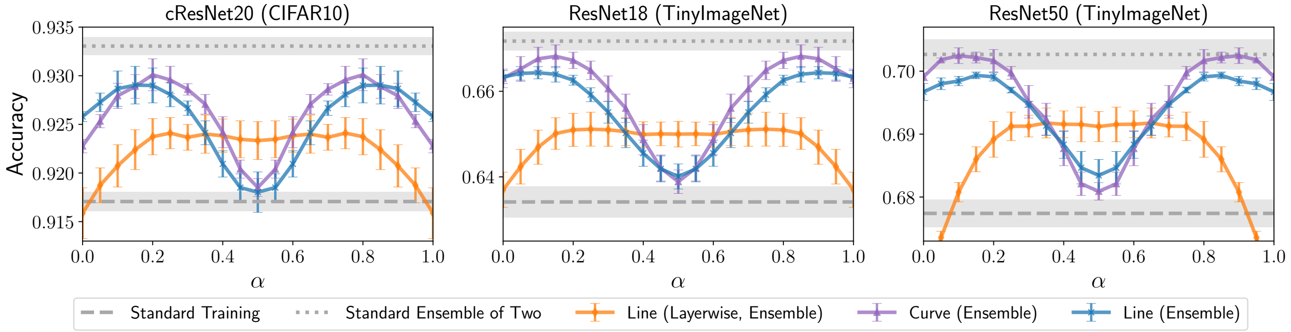

Figure 5. Accuracy when two models from the subspace are ensembled—at point α we plot the accuracy when models $\mathsf{P}(\alpha)$ and $\mathsf{P}(1 - \alpha)$ are ensembled. Performance approaches the ensemble of two independently trained networks, denoted "Standard Ensemble of Two".

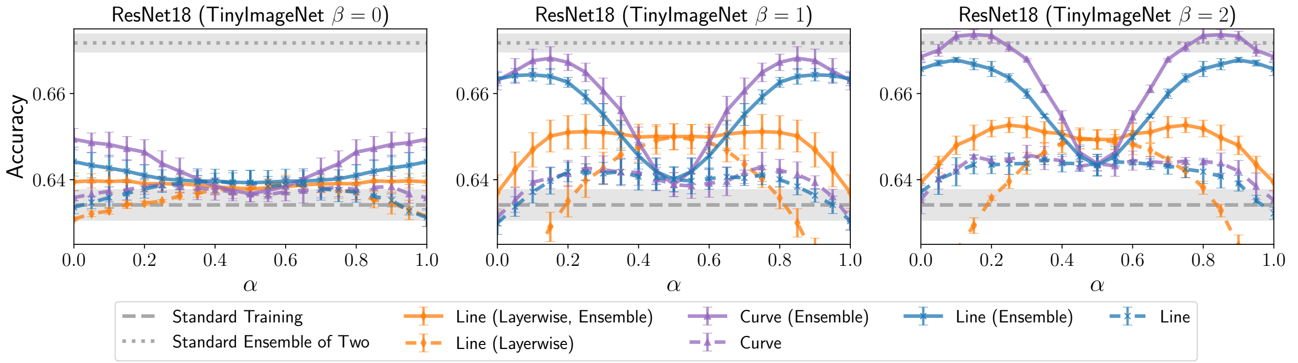

Figure 6. Visualizing both model and ensemble accuracy along one-dimensional subspaces for different regularization strengths β. Regularization (Equation 5) tends to produce a subspace with more accurate and diverse models. Note that the visualization format of Figure 4 and Figure 5 are combined, a technique we will use throughout the remainder of this work. For each subspace type, **(1)** accuracy of a model with weights $\mathsf{P}(\alpha)$ is shown with a dashed line and **(2)** accuracy when the output of models $\mathsf{P}(\alpha)$ and $\mathsf{P}(1 - \alpha)$ are ensembled is shown with a solid line and denoted **(Ensemble)**.

4.1. Subspace Dynamics

We begin with the following question: when training a line, how does the shape vary throughout training and how is this affected by β, the regularization coefficient? Figure 3 illustrates L_2 distance $\|\omega_1 - \omega_2\|_2$ and cosine similarity squared $\cos^2(\omega_1, \omega_2)$ throughout training. Recall that ω_1 and ω_2 denote the endpoints of the line which are initialized independently. Since a line is constructed using only two endpoints, the regularization term (Equation 5) simplifies to $\beta \cos^2(\omega_1, \omega_2)$.

When $\beta = 1$ the endpoints of a line become nearly orthogonal towards the end of training (in CIFAR10 they remain orthogonal throughout). Although L_2 distance isn't explicitly encouraged, it remains significant. Notably, for CIFAR10 the endpoints remain approximately as far apart throughout training as randomly initialized weights. For ResNet50 on ImageNet the L_2 distance between endpoints remains substantial (≈ 127), compared to ≈ 173 for independently trained solutions. Note that in both cases weight decay pushes trained weights towards the origin. When $\beta = 0$ there is no term encouraging separation between ω_1 and ω_2. However, they still remain a distance apart (13 for CIFAR10 and 40 for ImageNet). Further analysis is conducted in Appendix E, revealing that initializing ω_1 and ω_2 with the same shared weights has surprisingly little effect on the final cosine and L_2 distance.

4.2. Accuracy Along Lines and Curves

Next we investigate how accuracy varies along a one-dimensional subspace. For brevity let $\mathsf{P}(\alpha)$ denote the weights at position α along the subspace, for $\alpha \in [0, 1]$. We are interested in two quantities: (1) the accuracy of the neural network $f(\cdot, \mathsf{P}(\alpha))$ and (2) the accuracy when the outputs $f(\cdot, \mathsf{P}(\alpha))$ and $f(\cdot, \mathsf{P}(1-\alpha))$ are ensembled. Quantity (1) will determine if the subspace contains accurate solutions. Quantity (2) will demonstrate if the subspace contains diverse solutions which produce high-accuracy ensembles.

Quantities (1) and (2) are illustrated respectively by Figure 4 and Figure 5 In both Figure 4 and Figure 5 the regularization strength β remains at the default value of 1, while Figure 6 provides analogous results for $\beta \in \{0, 1, 2\}$. Note that *Layerwise* indicates that the layerwise training variant is employed (as described in section 3).

The baselines included are standard training and a standard ensemble of two independently trained networks (requiring twice as many training iterations). In Appendix F we experiment with additional baselines. There are many interesting takeaways from Figure 4, Figure 5, and Figure 6:

1. Not only does our method find a subspace of accurate solutions, but for $\beta > 0$ accuracy can improve over standard training. We believe this is because standard training solutions lie towards the periphery of a minimum (Izmailov et al., 2018) whereas our method traverses the the minimum. Solutions at the center tend to be less sharp than at the periphery, which is associated with better generalization (Dziugaite & Roy, 2018). These effects may be compounded by the regularization term, which leads the subspaces towards wider minima.
2. The ensemble of two models towards the endpoints of the subspace approaches, matches, or exceeds the ensemble accuracy of two independently trained models. This is notable as the subspaces are found in only one training run.
3. Subspaces found through the layerwise training variant have more accurate midpoints ($\alpha = 0.5$) but less accurate ensembles.

4.3. Performance of a Simplex Midpoint

The previous section provided empirical evidence that the midpoint of a line (simplex with two endpoints) can outperform standard training in the same number of epochs, and hypothesized two explanations for this observation. In this section we demonstrate that this trend is amplified when considering a simplex with m endpoints for $m > 2$.

Accuracy. The accuracy of a single model at center of a simplex is presented by Figure 7. The boost over standard training is significant, especially for TinyImageNet and higher dimensional simplexes. Recall that when training a simplex with m endpoints we initialize m separate networks and, for each batch, randomly sample a network in their convex hull. We then use the gradient to move this $m-1$ dimensional subspace through the objective landscape. It is not obvious that this method should converge to a high-accuracy subspace or contain high-accuracy solutions.

We compare a simplex with m endpoints with SWA (Izmailov et al., 2018) when m checkpoints are saved and averaged, to maintain parity in the number of stored model parameters. For layerwise training our method outperforms or matches SWA in every case. We speculate that this may be true either because our midpoint lies closer to the minimum center than the stochastic average, or because our method finds a wider minimum then SWA. We are training a whole subspace, whereas SWA constructs a subspace after training. SWA can only travel to the widest point of the current minimum, while our method searches for a large flat minimum.

Robustness to Label Noise; Calibration. Figure 8 demonstrates that taking the midpoint of a simplex boosts robustness to label noise and improves expected calibration error

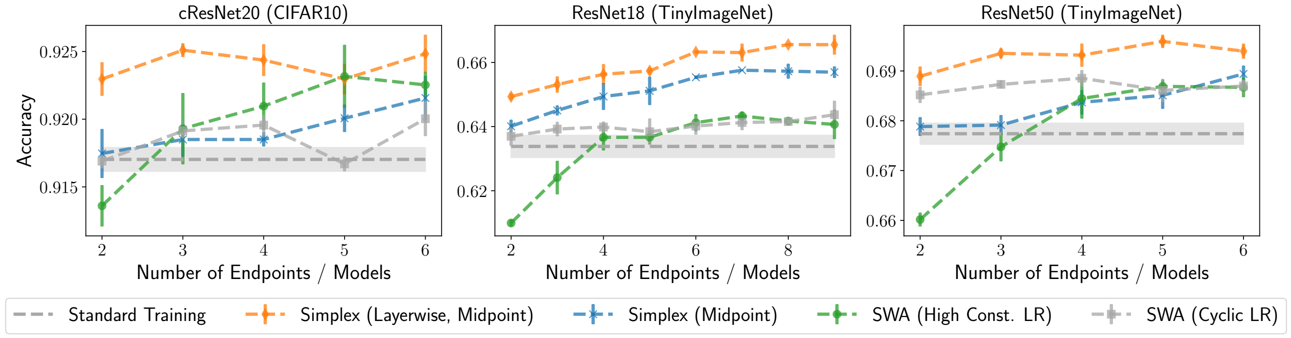

Figure 7. The model at the center of a learned simplex with m endpoints improves accuracy over standard training and SWA (Izmailov et al., 2018). A solution towards the center of a minimum tends to be less sharp than at the periphery, which is associated with better generalization (Dziugaite & Roy, 2018).

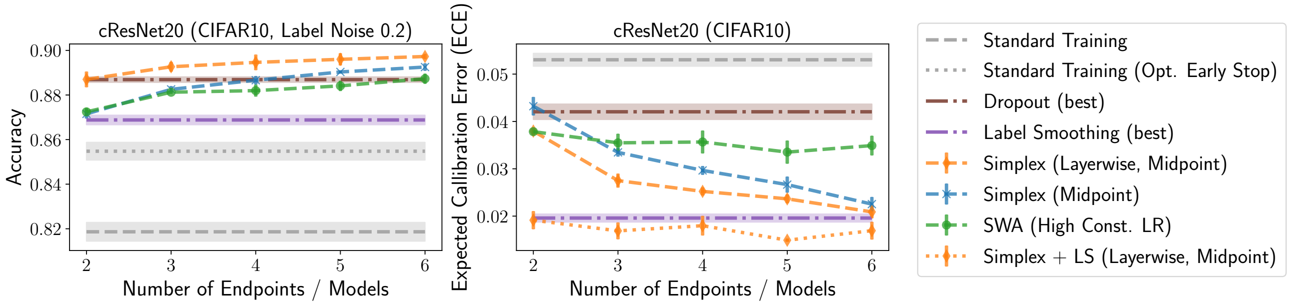

Figure 8. Using the model at the simplex center provides robustness to label noise and improved calibration. For *Dropout* and *Label Smoothing* we run hyperparameters $\{0.05, 0.1, 0.2, 0.4, 0.8\}$ and report the best. For Simplex + LS we add label smoothing.

(ECE) for cResNet20 on CIFAR10. Note that CIFAR10 with label noise c indicates that before training, a fraction c of training data are assigned random labels (which are fixed for all methods). In addition to a SWA baseline we include optimal early stopping (the best training accuracy for standard training, before over-fitting), label smoothing (Müller et al., 2019), and dropout (Srivastava et al., 2014). Label smoothing and dropout have a hyperparameter for which we try values $\{0.05, 0.1, 0.2, 0.4, 0.8\}$ and report the best result for each plot. Expected calibration error (ECE) (Guo et al., 2017) measures if prediction confidence and accuracy are aligned. A low ECE is preferred, since models with a high ECE are overconfident when incorrect or underconfident when correct.

4.4. ImageNet Experiments

In this section we experiment with a larger dataset—ImageNet (Deng et al., 2009)—for which networks are less overparameterized. In Figure 9 we visualize accuracy over a line, showing both (1) the accuracy of the neural network $f(\cdot, \mathsf{P}(\alpha))$ and (2) the accuracy when the outputs of the networks $f(\cdot, \mathsf{P}(\alpha))$ and $f(\cdot, \mathsf{P}(1-\alpha))$ are ensembled. In addition to testing the network on the clean dataset (left column), we show accuracy under the *snow* and *contrast* dataset corruptions found in ImageNet-C (Hendrycks & Dietterich, 2019). Finally, in the right column we show the relative difference in accuracy between two models on the line. There are two interesting findings from this experiment: **(1)** it is possible to find a subspace of models, even on ImageNet, that matches or exceeds the accuracy of standard training. **(2)** Models along the line can exhibit varied robustness when faced with corrupted data.

Finding **(2)** can be examined through the lens of *underspecification* in deep learning. D'Amour et al. (2020) observe that independently trained models which perform identically on the clean test set behave very differently on downstream tasks. Here we observe this behavior for models in the same linearly connected region found in a single training run. This is a promising observation in the case that a validation set exists for downstream domains. In Appendix G we experiment with all corruptions types in ImageNet-C and demonstrate that the models we find tend to exhibit more robustness than standard training.

The WideResNet50 and ResNet50 in Figure 9 are respectively trained for 100 and 200 epochs (for both our method and the baseline). The smaller ResNet50 is trained for longer as, when trained for 100 epochs, the accuracy of the ResNet50 subspace falls slightly below that of standard training. However, when trained for even longer, the accuracy exceeds that of standard training. This trend is illustrated by Figure 10 which shows how accuracy and

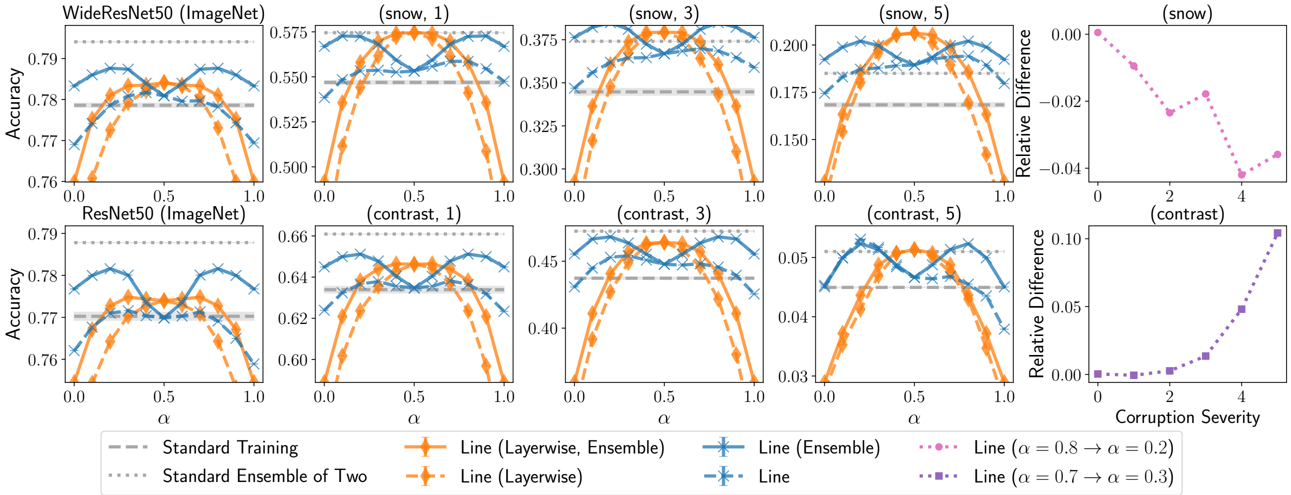

Figure 9. Accuracy along one-dimensional subspaces (with the same visualization format as Figure 6) tested on (left column) ImageNet (Deng et al., 2009) and (middle columns) ImageNet-C (Hendrycks & Dietterich, 2019) for corruption types *snow* and *contrast* with severity levels 1, 3, and 5. Relative difference in accuracy for two models on a line is shown in the rightmost column—models on the line with the similar performance on the clean test set exhibit varied performance on corrupted images (D'Amour et al., 2020).

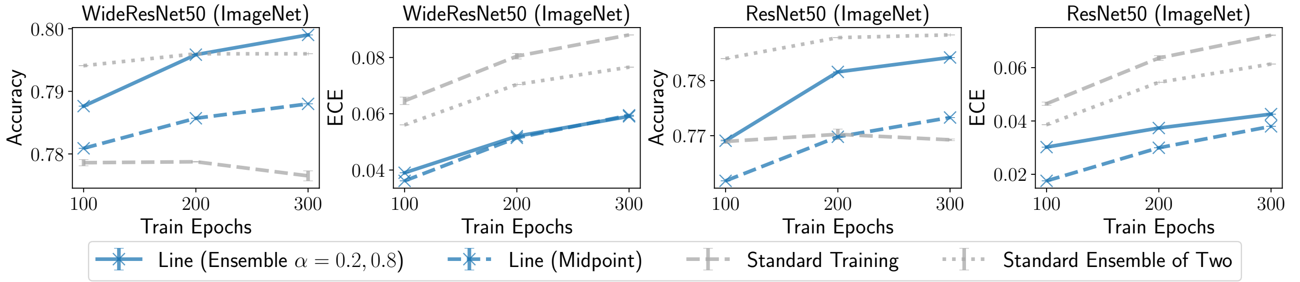

Figure 10. Accuracy and Expected Calibration Error (ECE) for the midpoint of a line trained for $\{100, 200, 300\}$ epochs on ImageNet. The models at the midpoint of a line are more callibrated and, when all models are trained for longer, more accurate.

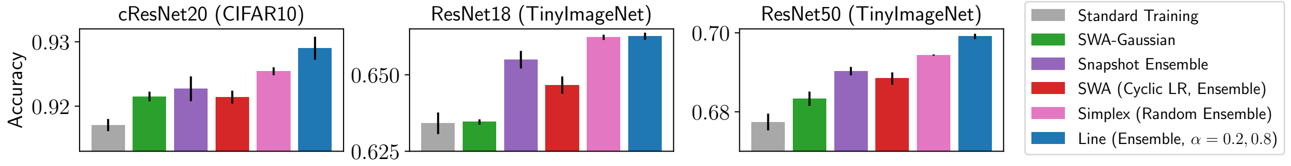

Figure 11. Ensembling 6 models drawn randomly from a 6 endpoint simplex compared with a 6 model Snapshot Ensemble (Huang et al., 2017), an ensemble of 6 SWA checkpoints (Izmailov et al., 2018), and 6 samples from a gaussian fit to the SWA checkpoints.

expected calibration error (ECE) (Guo et al., 2017) change as a function of training epochs. The subspace midpoint is consistently more calibrated than models found through standard training.

Finally, Figure 12 (left) demonstrates that the midpoint of a line outperforms standard training and optimal early stopping for various levels of label noise.

4.5. Randomly Ensembling from the Subspace

In Figure 11 we experiment with drawing multiple models from the simplex and ensembling their predictions. We consider a simplex with 6 endpoints and draw 6 models randomly (with the same sampling strategy employed during training) and refer to the resulting ensemble as *Simplex (Random Ensemble)*. We also experiment with a 6 model Snapshot Ensemble (Huang et al., 2017), ensembling 6 SWA checkpoints using a cyclic learning rate (this differs slightly, but resembles FGE (Garipov et al., 2018)), and SWA-Gaussian (Maddox et al., 2019). Additional details for the baselines are provided in subsection D.4. Surprisingly, ensembling 2 models from opposing ends of a linear subspace is still more accurate. Finally, in Appendix C we investigate the possibility of efficiently ensembling from a subspace without the cost.

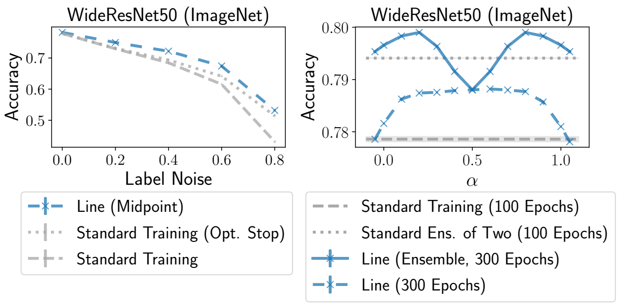

Figure 12. (left) Taking the midpoint of a line provides robustness to label noise on ImageNet compared with standard training and optimal early stopping. (right) It is possible for linearly connected models to individually attain an accuracy that is at or below standard training, while their ensemble performance is above that of standard ensembles.

4.6. Is Nonlinearity Required?

Garipov et al. (2018); Draxler et al. (2018) demonstrate that there exists a nonlinear path of high accuracy between two independently trained models. Independently trained models are functionally diverse, resulting in high-performing ensembles. However, the linear path between independently trained models encounters a high loss barrier (Frankle et al., 2020; Fort et al., 2020). In this section we aim to provide empirical evidence which answers the following question: is this energy barrier inevitable? Is it possible for linearly connected models to individually attain an accuracy that is at or below that of standard training, while their ensemble performance is at or above that of standard ensembles? In Figure 12 (right) we demonstrate that, for WideResNet50 on ImageNet trained for 100 epochs, this high loss barrier is not necessary. In this one case we are concerned with existence and not training efficiency, so we find the requisite linearly connected models by training a line for 300 epochs and interpolating slightly off the line (considering $\alpha = -0.05, 1.05$).

5. Conclusion

We have identified and traversed large, diverse regions of the objective landscape. Instead of constructing a subspace post training, we have trained lines, curves, and simplexes of high-accuracy neural networks from scratch. However, our understanding of neural network optimization has evolved significantly in recent years and we expect this trend to continue. We anticipate that future work will continue to leverage the geometry of the objective landscape for more accurate and reliable neural networks.

Acknowledgements

For insightful discussions, helpful suggestions, and support we thank Rosanne Liu, Jonathan Frankle, Joshua Susskind, Gabriel Ilharco Magalhães, Sarah Pratt, Ludwig Schmidt, ML Collective, Vivek Ramanujan, Jason Yosinski, Russ Webb, Ivan Evtimov, and Hessam Bagherinezhad. MW acknowledges Apple for providing internship support.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015. URL http://arxiv.org/abs/1512.01274.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 27, pp. 2933–2941. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/17e23e50bedc63b4095e3d8204ce063b-Paper.pdf.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2018.

Dziugaite, G. K. and Roy, D. Entropy-SGD optimizes the prior of a PAC-Bayes bound: Generalization properties of entropy-SGD and data-dependent priors. In Dy, J.

and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1377–1386, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/dziugaite18a.html.

Evci, U., Pedregosa, F., Gomez, A., and Elsen, E. The difficulty of training sparse neural networks. *arXiv preprint arXiv:1906.10732*, 2019.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Fort, S. and Jastrzebski, S. Large scale structure of neural network loss landscapes. In *Advances in Neural Information Processing Systems*, pp. 6709–6717, 2019.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *arXiv preprint arXiv:2010.15110*, 2020.

Frankle, J. Revisiting" qualitatively characterizing neural network optimization problems". *arXiv preprint arXiv:2012.06898*, 2020.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pp. 8789–8798, 2018.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pp. 1169–1179. PMLR, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7:7, 2015.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018a.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Advances in neural information processing systems*, pp. 6389–6399, 2018b.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32:13153–13164, 2019.

Müller, R., Kornblith, S., and Hinton, G. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

Oswald, J. V., Kobayashi, S., Sacramento, J., Meulemans, A., Henning, C., and Grewe, B. F. Neural networks with late-phase weights. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=C0qJUx5dxFb.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., and Schmidt, L. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33, 2020.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wen, Y., Tran, D., and Ba, J. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Xie, S., Kirillov, A., Girshick, R., and He, K. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1284–1293, 2019.

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

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient mcmc for bayesian deep learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkeS1RVtPS.