# ON NONDETERMINISM AND INSTABILITY IN NEURAL NETWORK OPTIMIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Optimization nondeterminism causes uncertainty when improving neural networks, with small changes in performance difficult to discern from run-to-run variability. While uncertainty can be reduced by training multiple copies of a model with different random seeds, doing so is time-consuming, costly, and makes reproducibility challenging. Despite this, little attention has been paid towards establishing an understanding of this problem. In this work, we establish an experimental protocol for understanding the effect of optimization nondeterminism on model diversity, which allows us to study the independent effects of a variety of sources of nondeterminism. Surprisingly, we find that each source of nondeterminism all have similar effects on multiple measures of model diversity. To explain this intriguing fact, we examine and identify the instability of model training, when taken as an end-to-end procedure, as the key determinant. We show that even one-bit changes in initial model parameters result in models that converge to vastly different values. Last, we demonstrate that recent methods in accelerated model ensembling hold promise for reducing the effects of instability on run-to-run variability.

## 1 INTRODUCTION

Consider this common scenario: you have a baseline "current best" model, and are trying to improve it. Now, one of your experiments has produced a model whose metrics are slightly better than the baseline. Yet you have your reservations — how do you know the improvement is "real", and not due to random fluctuations that cause run-to-run variability?

Similarly, consider performing hyperparameter optimization, in which there are many possible values for a set of hyperparameters, and you find minor differences in performance between them. How do you pick the best hyperparameters, and how can you be sure that you've actually picked wisely?

In both scenarios, the standard practice is to perform multiple independent training runs of your model to understand its variability. While this does indeed help address the problem, it can be extremely wasteful, increasing the time required for effective research, using more computing power, and making reproducibility more difficult, while still leaving some uncertainty.

Ultimately, the source of this problem is the nondeterminism in optimizing models, in which each training run produces a separate model with its own performance characteristics. Nondeterminism itself is caused by many factors — while the most salient cause is the random initialization of parameters, other sources exist, including random shuffling of training data, per-example stochasticity of data augmentation, any explicit random operations (e.g. dropout (Srivastava et al., 2014)), asynchronous model training (Recht et al., 2011), and even nondeterminism in low-level libraries such as cuDNN (Chetlur et al., 2014), which are present to improve throughput on hardware accelerators.

Despite the clear impact nondeterminism has on the efficacy of modeling, relatively little attention has been paid towards understanding its mechanisms, even in the classical supervised setting. In this work, we establish an experimental protocol for analyzing the impact of nondeterminism in model training, allowing us to quantify the independent effect of each source of nondeterminism. In doing so, we make a surprising discovery: each source has nearly the same effect on the variability of final model performance. Further, we find each source produces models of similar diversity, as measured by correlations between model predictions, functional changes in model performance while ensembling, and state-of-the-art methods of model similarity (Kornblith et al., 2019). To emphasize one particularly interesting result: nondeterminism in low-level libraries like cuDNN matters just as much with respect to model diversity and variability as varying the entire network initialization.

We explain this mystery by demonstrating that it can be attributed to an inherent numerical *instability* in optimizing neural networks — when training with SGD-like approaches, we show that small changes to initial parameters result in large changes to final parameter values. In fact, the instabilities in the optimization process are extreme: *changing a single weight by the smallest possible amount within machine precision ($\sim 6 * 10^{-11}$) causes nearly as much variability as all other sources combined*. Therefore, any source of nondeterminism that has any effect at all on model weights is doomed to inherit at least this level of variability.

Last, we present promising results in reducing the effects of instability. While we find that many approaches result in no apparent change, we demonstrate that methods for accelerated model ensembling (Huang et al., 2017) actually do reduce the variability of trained models without an increase in model training time, providing the first encouraging signs for tractability of the problem.

## 2 Related Work

NONDETERMINISM. Relatively little prior work has studied the effects of nondeterminism on model optimization. Within reinforcement learning, nondeterminism is recognized as a significant barrier to reproducibility and evaluating progress in the field (Nagarajan et al., 2018; Henderson et al., 2018; Islam et al., 2017; Machado et al., 2018). In the setting of supervised learning, though, the focus of this work, the problem is much less studied. Madhyastha & Jain (2019) aggregate all sources of nondeterminism together into a single random seed and analyze the variability of model attention and accuracy as a function of it across various NLP datasets. They also propose a method for reducing this variability (see Sec. A for details of our reproduction attempt). More common in the field, results across multiple random seeds are reported (see Erhan et al. (2010) for a particularly extensive example), but the precise nature of nondeterminism's cause of variability goes unstudied.

INSTABILITY. We use the term "stability" to refer to numerical stability, in which a stable algorithm is one for which the final output (converged model) does not vary much as the input (initial parameters) are changed. Historically, the term "stability" has been used both in learning theory (Bousquet & Elisseeff, 2002), in reference to vanishing and exploding gradients (Haber & Ruthotto, 2017), and in the adversarial robustness community for a particular form of training (Zheng et al., 2016).

## 3 Nondeterminism

Many sources of nondeterminism exist when optimizing neural networks, each of which can affect the variability and performance of trained models. We begin with a very brief overview:

PARAMETER INITIALIZATION. When training a model, parameters without preset values are initialized randomly according to a given distribution, *e.g.* a Gaussian with mean 0 and variance determined by the number of input connections to the layer (Glorot & Bengio, 2010; He et al., 2015).

DATA SHUFFLING. In stochastic gradient descent (SGD), the overall gradient is approximated by the gradient on a random subset of examples. Most commonly, this is implemented by shuffling the training data, after which the data is iterated through in order. Shuffling may happen either once, before training, or in between each epoch of training, the variant we use in this work.

DATA AUGMENTATION. A very common practice, data augmentation refers to randomly altering each training example to artificially expand the training dataset. For example, in the case of images, it is common to randomly flip an image, which encourages invariance to left/right orientation.

STOCHASTIC REGULARIZATION. Some forms of regularization, such as Dropout (Srivastava et al., 2014), take the form of stochastic operations in a model during training. Dropout is the most common instance of this type of regularization, with a variety of others also in relatively common use, such as DropConnect (Wan et al., 2013), variational dropout (Gal & Ghahramani, 2016), and variable length backpropagation through time (Merity et al., 2017), among many others.

LOW-LEVEL OPERATIONS. An underlooked source of nondeterminism, the very libraries that many deep learning frameworks are built on, such as cuDNN (Chetlur et al., 2014) often are run nondeterministically for performance reasons. This nondeterminism is small in magnitude — in one test we performed this caused a difference of roughly $0.003\%$. In the case of cuDNN, the library we

test with, it is possible to disable nondeterministic behavior, incurring a speed penalty typically on the order of $\sim 15\%$. However, unlike the other sources of nondeterminism, it is not possible to "seed" this nondeterminism; it is only possible to turn it on or off, but not control its nondeterministic behavior.

### 3.1 Protocol for Testing Effects of Nondeterminism

Diversity in Performance. Our protocol for testing the effects of sources of nondeterminism is based on properly controlling for each source. In general, suppose there are $N$ sources of nondeterminism, with source $i$ controlled by a seed $S_i$. To test the effect of source $i$, we keep all values $S_j, j \neq i$ set to a constant, and vary $S_i$ with $R$ different values, where $R$ is the number of independent training runs performed. For sources of nondeterminism which cannot be effectively seeded, such as cuDNN, we indicate one of these values as the deterministic value, which it must be set to when varying the other sources of nondeterminism.

For example, suppose that we wish to study three sources of nondeterminism, denoting $S_1$ the seed for random parameter initialization, $S_2$ for training data shuffling, and $S_3$ for cuDNN, where $S_3 = 1$ is the deterministic value for cuDNN. To test the effect of random parameter initialization, with a budget of $R = 30$ training runs, then we set $S_3$ to the deterministic value of 1, $S_2$ to an arbitrary constant (also 1 for simplicity), and test 30 different values of $S_1$. All together, this corresponds to training models for $(S_1, S_2, S_3) \in \{(i, 1, 1)\}_{i=1}^{30}$, producing a set of 30 models. To look at variability according to a particular evaluation metric (*e.g.* cross-entropy or accuracy for classification), we calculate the standard deviation (across all $R = 30$ models) of the metric. Note that it is also possible to test the effects of several sources of nondeterminism in tandem this way. For example, to test all sources of nondeterminism together, the set of models can be changed to $(S_1, S_2, S_3) \in \{(i, i, 0)\}_{i=1}^{R}$.

Diversity in Representation. Beyond looking at diversity of test set generalization, though, it is worth examining how different the representations of trained models actually are — even though the diversity in performance might be similar between models trained with different types of nondeterminism, it might be the case that one type of nondeterminism produces models that have learned largely similar concepts, with the variance in generalization due to other factors. We examine this question through three analyses that focus on the functional behavior of models:

First, and more straightforward, we consider the average correlation between the predictions of two models, *i.e.* the expectation (across pairs of models from the same nondeterminism source), of the correlation of predictions, calculated across examples and classes. For example, for a classification problem, the predicted logits from each of $R$ models are flattened into vectors of length $N * C$ (with $N$ the number of test examples and $C$ the number of classes), and we calculate the mean correlation coefficient of the predictions across all $\binom{R}{2}$ pairs of models. We use Spearman's $\rho$ for the correlation coefficient, but note that others such as Pearson's $r$ are possible and yield similar conclusions.

The second analysis we perform examines the change in performance from ensembling two models from the same source of nondeterminism. The intuition is as follows: If a pair of models are completely redundant, then ensembling them would result in no change in performance. However, if models actually learn different representations, then we expect an improvement from ensembling, with a greater improvement the greater the diversity in a set of models. Denoting by $f(S_i)$ some particular evaluation metric $f$ calculated on the predictions of model $S_i$, this change is equivalent to:

$$\frac{1}{\binom{R}{2}} \sum_{i=1}^{R} \sum_{j=i+1}^{R} \left( f\left(\frac{S_i + S_j}{2}\right) - \frac{f(S_i) + f(S_j)}{2} \right) \tag{1}$$

Last, for a more detailed view of learned representations within a network, we consider a state-of-the-art method for measuring the similarity of learned neural network representations, centered kernel alignment (CKA) (Kornblith et al., 2019), which has been used to analyze models trained with different random initializations, widths, and even entirely different architectures. We use the linear version of CKA, which Kornblith et al. (2019) found performed similarly to an RBF kernel.

### 3.2 Experiments in image classification

We begin our study of nondeterminism with the fundamental task of image classification. We test our protocol using CIFAR-10 (Krizhevsky et al., 2009), a 10-way classification dataset with $50{,}000$

| Nondeterminism Source | Accuracy SD (%) | Cross-Entropy SD | Pairwise Corr. | Ensemble $\Delta$ (%) |
|---|---|---|---|---|
| Parameter Initialization | $0.22 \pm 0.02$ | $0.0080 \pm 0.0005$ | 0.872 | 1.74 |
| Data Shuffling | $0.25 \pm 0.02$ | $0.0087 \pm 0.0005$ | 0.871 | 1.74 |
| Data Augmentation | $0.22 \pm 0.02$ | $0.0075 \pm 0.0005$ | 0.872 | 1.83 |
| cuDNN | $0.21 \pm 0.01$ | $0.0087 \pm 0.0007$ | 0.873 | 1.64 |
| Data Shuffling + cuDNN | $0.21 \pm 0.01$ | $0.0083 \pm 0.0006$ | 0.871 | 1.77 |
| Data Shuffling + Aug. + cuDNN | $0.21 \pm 0.01$ | $0.0077 \pm 0.0005$ | 0.871 | 1.72 |
| All nondeterminism sources | $0.26 \pm 0.02$ | $0.0075 \pm 0.0005$ | 0.871 | 1.72 |

Table 1: The effect of each source of nondeterminism and several combinations of nondeterminism sources for ResNet-14 on CIFAR-10. The second and third columns give the standard deviation of accuracy and cross-entropy across 100 runs, varying only the nondeterminism source (700 trained models total). Also given are error bars, corresponding to the standard deviation of the standard deviation. The fourth and fifth columns give the average pairwise Spearman's correlation coefficient between predictions and the average change in accuracy from ensembling two models (Sec. 3.1).

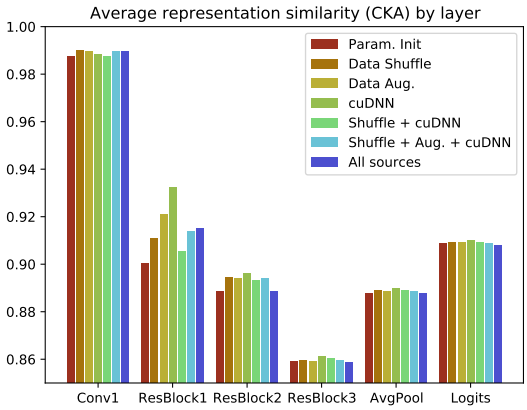


Figure 1: Average representation similarity as determined by CKA (Kornblith et al., 2019) for ResNet-14 on CIFAR-10 across nondeterminism sources. Layers are selected uniformly across the network, ranging from the first convolutional layer (Conv1), the output of each residual block (ResBlock1,2,3), the penultimate average pooling layer (AvgPool), and the logits themselves (Logits). Shown is the average CKA value between pairs of models.

training images of resolution $32 \times 32$ pixels and $10{,}000$ for testing. In these initial experiments, we use a 14-layer ResNet model (He et al., 2016), trained with a cosine learning rate decay (Loshchilov & Hutter, 2016) for 500 epochs with a maximum learning rate of $.40$, three epochs of learning rate warmup, a batch size of 512, momentum of $0.9$, and weight decay of $5 \cdot 10^{-4}$, obtaining a baseline accuracy of $90.0\%$. Data augmentation consists of random crops and horizontal flips. All experiments were done on two NVIDIA Tesla V100 GPUs with `pytorch` (Paszke et al., 2019).

We show the results of our protocol in this setting in Table 1. What we find is surprising — while there are slight differences, each source of nondeterminism has very similar effects on the variability of final trained models, measured across both accuracy and cross-entropy. In fact, random parameter initialization, arguably the most salient form of nondeterminism, does not stand out based on any of these metrics. Even when combining multiple sources of nondeterminism, there's remarkably little difference — nothing more than one or two standard deviations compared to any individual source of nondeterminism. In short, each source of nondeterminism has roughly the same effect on the variability in final model performance, with no combination of nondeterminism sources a factor of even two more important than any other.

Turning toward representational diversity on a per-layer level, average CKA values across 6 representative layers are shown in Fig. 1, done for pairwise combinations of 25 models (for computational expediency). Consistent with other analyses, CKA reveals that while there are some differences in representational similarity between models trained with different sources of nondeterminism, especially the output of the first residual block, by and large these differences are small, easily dwarfed in size by differences across layers.

| Nondeterminism Source | PPL SD | Ensemble PPL $\Delta$ |
|---|---|---|
| Parameter Initialization | $0.20 \pm 0.01$ | -2.07 |
| Stochastic Operations | $0.19 \pm 0.01$ | -2.08 |
| All nondeterminism sources | $0.18 \pm 0.01$ | -2.07 |

Table 2: The effect of each source of nondeterminism for a QRNN on Penn Treebank; 100 runs per row. Note that lower PPL is better, so changes in Ensemble PPL are negative.

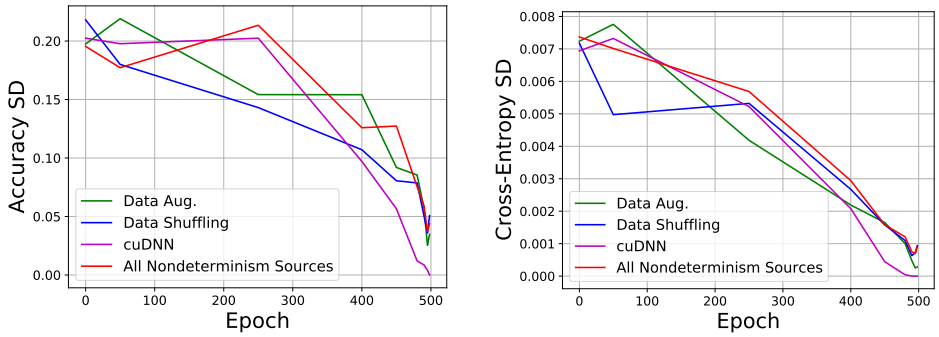


Figure 2: The effect of the onset of nondeterminism on the variability of accuracy and cross-entropy in converged models. Each point corresponds to training a set of 20 models deterministically for a certain number of epochs, then enabling a source of nondeterminism by varying its seed starting from that epoch and continuing through the end of training.

### 3.3 Experiments in language modeling

Here we show that this phenomenon is not unique to image classification, applying the same experimental protocol to language modeling. For these experiments, we employ a small quasi-recurrent neural network (QRNN) (Bradbury et al., 2016) on Penn Treebank (Marcus et al., 1993), using the publicly available code of Merity et al. (2017). This baseline model uses a 256-dimensional word embedding size, 512 hidden units per layer, and 3 layers of recurrent units, obtaining a perplexity (PPL) of 75.49 on the Penn Treebank test set.

For this task, two sources of nondeterminism are relevant: the random parameter initialization, and multiple stochastic operations including variations of dropout and variable length backpropagation through time, which share a common seed. To measure diversity in performance, PPL is the most widely-accepted metric, and for diversity in representation we focus on benefits from ensembling because CKA was not designed for variable-length input and common computing libraries (Virtanen et al., 2020) are not efficient enough to calculate $O(R^2)$ correlation coefficients with such large inputs. We show results in Table 2, where we find even less difference in variability between the two sources of nondeterminism considered across both diversity metrics, showing that the phenomenon isn't unique to image classification or ResNets.

### 3.4 Nondeterminism throughout training

One hypothesis for the cause of this phenomenon is that models are particularly sensitive to changes in the initial phase of learning, which recent work has shown to be the case in other contexts (Achille et al., 2019; Frankle et al., 2020). With our experimental protocol, this is straightforward to test: If this were the case, then training networks identically for the first $N$ epochs and only then introducing these sources of nondeterminism would result in a significantly lower diversity in trained models, measured across any metric of choice, *e.g.* variance in cross-entropy or accuracy. Furthermore, by varying $N$, we can actually determine *when* in training each source of nondeterminism has its effect (for sources that vary over the course of training, *i.e.* not random parameter initialization).

We do this experiment for the ResNet-14 model on CIFAR-10 in Fig. 2, where we find that it is *not* only the beginning of training that is sensitive to changes in nondeterminism, refuting this hypothesis about the onset of the effects of nondeterminism. Instead, the effect of nondeterminism is nearly as high when enabling it even after 50 epochs (10% of the way through training), and we see only a gradual reduction in final model variability as the onset of nondeterminism is moved later and later.

| Nondeterminism Source | Accuracy SD (%) | Cross-Entropy SD | Pairwise Corr. | Ensemble $\Delta$ (%) |
|---|---|---|---|---|
| Random Bit Change | $0.21 \pm 0.01$ | $0.0070 \pm 0.0004$ | 0.874 | 1.82 |

Table 3: The effect of instability — randomly changing a single weight by one bit for ResNet-14 on CIFAR-10. Also see Table 1 for comparison.

| Nondeterminism Source | PPL SD | Ensemble PPL $\Delta$ |
|---|---|---|
| Random Bit Change | $0.19 \pm 0.01$ | -2.07 |

Table 4: The effect of instability for a QRNN on Penn Treebank. Also see Table 2 for comparison.

# 4 INSTABILITY

Why does each source of nondeterminism have a similar effect on model diversity? We answer this question by finding the smallest possible change that produces as diverse a range of models, revealing the *instability* in optimizing neural networks.

## 4.1 INSTABILITY AND NONDETERMINISM

To demonstrate, consider again a ResNet-14 trained on CIFAR-10, with a cross-entropy loss on the test set of $0.3519$ and accuracy of $90.0\%$. Now consider another model trained in an identical fashion, with exactly equal settings for all sources of nondeterminism, but one extremely small change: we randomly pick a single weight in the first layer and change its value by the smallest amount possible in a 32-bit floating point representation, *i.e.* an addition or subtraction of a single bit in the least-significant digit. As an example, in one model this changed a value from approximately $-0.0066514308$ to $-0.0066514313$, a difference on the order of $5 \cdot 10^{-10}$.

What happens when we optimize this model, which differs from the original only by a miniscule amount, in a single weight, itself within a single layer? By the end of the first epoch of training, still in a phase where the learning rate is warming up, the new model already differs in accuracy by $0.18\%$ compared to the original ($25.74\%$ vs $25.56\%$). With one more epoch of training the difference is a larger $2.33\%$ ($33.45\%$ vs $31.12\%$), and by the end of the third epoch, as the learning rate warmup has just ended, the difference is a staggering $10.42\%$ ($41.27\%$ vs $30.85\%$). Finally, at the end of training, the differences shrink again, with the new model converged to an accuracy of $90.12\%$ and a cross-entropy of $0.34335$. When viewing the optimization process end-to-end, with the initial parameters as the input and a given performance metric as the output, this implies a condition number $\frac{\|\delta f\|}{\|\delta x\|}$ of at least $1.8 \cdot 10^7$ for cross-entropy and $2.6 \cdot 10^8$ for accuracy.

To identify whether instability is a root cause for why each source of nondeterminism appears equally important, we can follow our testing protocol from Sec. 3 — this time, our source of nondeterminism is randomly picking a different weight to change in each model training run, then either incrementing or decrementing it to the next available floating-point value. We show the results in Table 3 for image classification on CIFAR-10 (*c.f.* Table 1 for comparison) and Table 4 for language modeling on Penn Treebank (*c.f.* Table 2), where we find that even this miniscule change produces roughly as much variability in model performance as every other source of nondeterminism.

From this, it is easy to see why every other source of nondeterminism has similar effects to each other — so long as a nondeterminism source causes any change to model weights at all, whether by changing the input slightly, altering the gradient in some way, or some other effect, it will have *at least* as much model variability as caused by the instability of model optimization. As a consequence, model instability is a barrier that must be overcome if one wishes to mitigate the effects of even a single source of nondeterminism.

## 4.2 WHY OPTIMIZATION IS INSTABLE

Instability is due to network depth, with any network greater than a single layer exhibiting instability:

| Ensemble | Accuracy SD (%) | Cross-Entropy SD | Pairwise Corr. |
|---|---|---|---|
| Snapshot | $0.19 \pm 0.02$ | $0.0044 \pm 0.0003$ | 0.957 |
| Vanilla ($N = 1$) | $0.26 \pm 0.02$ | $0.0075 \pm 0.0005$ | 0.871 |
| Vanilla ($N = 2$) | $0.19 \pm 0.02$ | $0.0044 \pm 0.0004$ | 0.929 |
| Vanilla ($N = 3$) | $0.15 \pm 0.02$ | $0.0033 \pm 0.0005$ | 0.951 |
| Vanilla ($N = 4$) | $0.16 \pm 0.02$ | $0.0030 \pm 0.0004$ | 0.963 |
| Vanilla ($N = 5$) | $0.12 \pm 0.02$ | $0.0028 \pm 0.0004$ | 0.970 |

Table 5: Snapshot ensembles compared with regular ensembles for ResNet-14 on CIFAR-10, with all nondeterminism sources enabled. $N$ denotes the number of component models in a regular ("Vanilla") ensemble. The snapshot ensemble is based on 100 runs of model training, and all vanilla ensembles are drawn from a pool of 100 independent model runs.

Theoretically, a linear model, optimized with a cross-entropy and with an appropriate learning rate schedule, should always converge to a global minimum due to convexity. However, we find an even stronger property: when we modify the initial weights by a single bit, beyond simply converging to the same weights, the entire optimization trajectory stays close to that of an unperturbed model, never differing by more than a vanishingly small amount. At convergence, a set of linear models trained in this was with only single random bit changes had a final accuracy SD of 0 (*i.e.* no changes in any test set predictions) and cross-entropy SD of $\sim 3 \cdot 10^{-8}$, far below that of any deeper model.

In contrast, instability occurred as soon as a single hidden layer was added, with an accuracy SD of 0.28 and cross-entropy SD of 0.0055 for a model with a fully-connected hidden layer, and an accuracy SD of 0.14 and cross-entropy SD of 0.0024 when the hidden layer is convolutional, both a factor of $10{,}000$ greater than the linear model (see Table 7 for full details).

## 5 A Path Forward

Here we identify and demonstrate how recent techniques in accelerated model ensembling hold promise for reducing the variability caused by nondeterminism and instability. For learnings on approaches which *did not* reduce variability, we direct the reader to Sec. A.

The approach we focus on is Snapshot Ensembles (Huang et al., 2017), which creates an ensemble of models during the training of a single model. It does this via usage of a cyclic learning rate schedule, where the members of the ensemble are drawn from instances during training when the learning rate goes to 0 (before rising again due to the cyclic learning rate). We compare a Snapshot ensemble approach with 5 cycles in its learning rate to ordinary ensembling, in which multiple models are trained entirely from scratch.

Results are shown in Table 5, where for reasons of space we focus on CIFAR-10 with all sources of nondeterminism enabled, though we have separately observed these results generalize to other models and sources of nondeterminism. The snapshot ensemble we identified, despite training only a single model, had variability in accuracy and cross-entropy comparable to an ensemble of 2, and pairwise correlation similar to an ensemble of 3 or 4. By reducing the variability in model performance without increasing training time, this identifies accelerated model ensembling as a promising approach for further reducing the trained model variability and mitigating the effects of instability.

## 6 Generalization Experiments

In this section we detail additional experiments showing our nondeterminism and instability claims generalize to other settings. For computational reasons (each result requires training the same model end-to-end $R = 100$ times), we restrict our attention to modestly-sized datasets and models.

On CIFAR-10, in addition to the ResNet-14 experimented with in Sec. 3 and Sec. 4 and the linear and single hidden layer models from Sec. 4.2, we experiment with 6-, 10-, and 18-layer ResNet variants, VGG-11 (Simonyan & Zisserman, 2014), and a 50%-capacity ShuffleNetv2 (Ma et al., 2018), allowing further exploration of the effect of model size and architecture on nondeterminism and instability. As shown in Table 6, the observations around instability and its relationship to

| Nondeterminism Source | Accuracy SD (%) | Cross-Entropy SD | Pairwise Corr. | Ensemble $\Delta$ (%) |
|---|---|---|---|---|
| CIFAR-10: ResNet-6 | | | | |
| Parameter Initialization | $0.50 \pm 0.04$ | $0.0123 \pm 0.0011$ | 0.925 | 2.17 |
| All nondeterminism sources | $0.43 \pm 0.03$ | $0.0111 \pm 0.0007$ | 0.924 | 2.17 |
| Random bit change | $0.41 \pm 0.02$ | $0.0100 \pm 0.0006$ | 0.925 | 2.12 |
| CIFAR-10: ResNet-10 | | | | |
| Parameter Initialization | $0.23 \pm 0.01$ | $0.0064 \pm 0.0004$ | 0.912 | 2.13 |
| All nondeterminism sources | $0.23 \pm 0.01$ | $0.0068 \pm 0.0005$ | 0.911 | 2.13 |
| Random bit change | $0.25 \pm 0.02$ | $0.0068 \pm 0.0005$ | 0.913 | 2.08 |
| CIFAR-10: ShuffleNetv2-50% | | | | |
| All nondeterminism sources | $0.22 \pm 0.02$ | $0.0115 \pm 0.0008$ | 0.692 | 1.39 |
| Random bit change | $0.21 \pm 0.01$ | $0.0071 \pm 0.0007$ | 0.695 | 1.36 |
| CIFAR-10: VGG-11 | | | | |
| All nondeterminism sources | $0.18 \pm 0.01$ | $0.0070 \pm 0.0005$ | 0.806 | 0.94 |
| Random bit change | $0.16 \pm 0.01$ | $0.0060 \pm 0.0004$ | 0.811 | 0.89 |
| MNIST | | | | |
| Parameter Initialization | $0.047 \pm 0.0036$ | $0.0025 \pm 0.0002$ | 0.941 | 0.064 |
| All nondeterminism sources | $0.046 \pm 0.0032$ | $0.0023 \pm 0.0001$ | 0.939 | 0.068 |
| Random bit change | $0.035 \pm 0.0026$ | $0.0011 \pm 0.0001$ | 0.989 | 0.011 |

Table 6: Generalization experiments of nondeterminism and instability with other architectures on CIFAR-10 and additional experiments on a high-accuracy dataset (MNIST). Each row is computed from the statistics of 100 trained models (*i.e.* 1,300 models total for this table).

nondeterminism generally hold for these architectures, with a close correspondence between the magnitude of effects for a random bit change and each of the four metrics considered. Interestingly, for ResNet models the pairwise correlation was generally decreasing with model size, indicating that greater variability in prediction may be a general trend among larger models.

We also perform experiments on MNIST (LeCun et al., 1998), allowing us to test whether our observations hold for tasks with very high accuracy — $99.14\%$ for our baseline model. As before, we find similar effects of nondeterminism for parameter initialization and all nondeterminism sources, including a comparable effect (albeit slightly smaller) from a single random bit change, highlighting that the instability of training extends even to datasets where the goal is simpler and model performance is higher. Of note, though, is the relative smaller effect of a single bit change on pairwise ensemble metrics of diversity, suggesting that the magnitude of instability might be at least partially related to the interplay of model architecture, capacity, and degree of overfitting.

## 7 Conclusion

In this work, we have shown two surprising facts: First, though conventional wisdom is that run-to-run variability in model performance is primarily determined by random parameter initialization, many sources of nondeterminism actually cause similar levels of variability. Second, the source of this phenomenon is the instability of model optimization, in which changes on the order of $10^{-10}$ in a single weight at initialization can have as much effect as reinitializing to completely random values. Model instability itself is a property of the depth of a model, with any model with more than single layer suffering from it when optimized with SGD. Beyond identifying and explaining this property of neural network optimization, we have also identified a promising direction for reducing the variability in model performance and representation without incurring the cost of training multiple copies of a model: accelerated model ensembling.

We hope that our work sheds light on a complex phenomenon that affects both researchers and users of deep learning. Ultimately, we also hope that, aided by this understanding, the field is able to make progress in improving the stability of model training, accelerating progress for all.

## REFERENCES


Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. In *International Conference on Learning Representations*, 2019.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2 (Mar):499–526, 2002.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb): 625–660, 2010.

Jonathan Frankle, David J Schwab, and Ari S Morcos. The early phase of neural network training. *arXiv preprint arXiv:2002.10365*, 2020.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027, 2016.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Pranava Madhyastha and Rishabh Jain. On model stability as a function of random seed. In *Conference on Computational Natural Language Learning*, pp. 929–939, 2019.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Prabhat Nagarajan, Garrett Warnell, and Peter Stone. The impact of nondeterminism on reproducibility in deep reinforcement learning. 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pp. 693–701, 2011.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066, 2013.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 4480–4488, 2016.

## A APPROACHES THAT DON'T REDUCE INSTABILITY

In the process of finding an approach that reduces run-to-run variability of models (Sec. 5), we experimented with many approaches which all failed to make a dent in improving variability and stability. For the benefit of the field, here we provide our experiences with these approaches which did not succeed in improving stability, despite the intuitive arguments for why they might help.

USING A DIFFERENT OPTIMIZER. Since instability and nondeterminism are both a property of optimization, it is conceivable that use of a different optimizer might be able to lessen the degree of instability in model training. We experimented SGD with various values of momentum, ranging from 0 for pure SGD to $0.999$ for a momentum-driven optimizer, but none succeeded in reduce instability. In addition, we experimented with the Adam (Kingma & Ba, 2014), as a representative of the class of adaptive learning rate algorithms, but this, too, had no effect on stability.

DURATION OF TRAINING. Noticing that the effects of nondeterminism seemed to accumulate during the course of training (Fig. 2), it seemed reasonable that varying the duration of training might have an effect. However, varying the duration of training from anywhere between 50 and $2{,}000$ epochs on CIFAR-10 all produced models with a similar variance in performance as the results in the rest of this work (which used 500 epochs), even though the absolute performance differed by up to $\sim 2\%$.

AGGRESSIVE STOCHASTIC WEIGHT AVERAGING. Inspired by the success found by Madhyastha & Jain (2019), we tried using Aggressive Stochastic Weight Averaging (ASWA), a variant of SWA (Izmailov et al., 2018). However, we could not get the model to converge to a reasonable degree of performance with the original formulation due to update sizes that decreased too rapidly, and though we were able to modify it to converge successfully, the output variance remained as high as the other models.

GRADIENT CLIPPING. With the intuition that instability might be caused by spurious gradients of large magnitude, we experimented with clipping the norm of gradients (using `pytorch`'s implementation of `torch.nn.utils.clip_grad.clip_grad_norm_`. Like other approaches, though, this had no effect on model variability.

WEIGHT AUGMENTATION. A very experimental approach, to reduce instability we experimented with taking an averaged gradient around the current set of parameters at each step, approximated by sampling a random weight offset before doing a forward or backward pass through the model. Intuitively, this would encourage optimization to not be too sensitive to the current value of weights; however, in practice this simply didn't affect the variance or stability of the model.

## B ADDITIONAL LINEAR AND 2-LAYER RESULTS

In Table 7 we include results for linear networks and 2-layer networks (1 hidden layer) on CIFAR-10, omitted from the main text due to space constraints. Of note in these results is the relative smaller effect of a single bit change for a 2-layer network where the hidden layer is convolutional — still much larger than for the linear model, but significantly smaller than for any other non-linear model, suggesting a similar effect to what was observed on MNIST in that degree of instability might be related to the interplay of model and dataset.

## C SUBTLETIES IN EVALUATION

While we have done our best to present our experimental protocol as plainly as possible, one subtlety arises when interpreting results. First, recall that in our example from Sec. 3.1, testing the effects of random initialization corresponded to training models for $(S_1, S_2, S_3) \in \{(i, 1, 1)\}_{i=1}^{R}$, where $S_1$ is the seed for random initialization, $S_2$ is the seed for training data shuffling, and $S_3$ is set to 1 to indicate the deterministic mode for cuDNN. The subtlety arises in that the resulting distribution of $(S_1, S_2, S_3) \in \{(i, 1, 1)\}_{i=1}^{R}$ is not necessarily the same as the distribution where $S_2$ is set to

| Nondeterminism Source | Accuracy SD (%) | Cross-Entropy SD | Pairwise Corr. | Ensemble $\Delta$ (%) |
|---|---|---|---|---|
| CIFAR-10: Linear model | | | | |
| Parameter Initialization | $0.03 \pm$ 2e-3 | $0.0002 \pm$ 1e-5 | 0.997 | -4e-6 |
| All nondeterminism sources | $0.10 \pm 0.01$ | $0.0007 \pm$ 4e-5 | 0.996 | 0.06 |
| Random bit change | $0.00 \pm 0.00$ | 2e-8 $\pm$ 1e-9 | 1.000 | 0.00 |
| CIFAR-10: One hidden layer (fully-connected) | | | | |
| Parameter Initialization | $0.31 \pm 0.02$ | $0.0054 \pm 0.0004$ | 0.941 | 1.38 |
| All nondeterminism sources | $0.30 \pm 0.02$ | $0.0057 \pm 0.0005$ | 0.937 | 1.49 |
| Random bit change | $0.28 \pm 0.02$ | $0.0055 \pm 0.0005$ | 0.945 | 1.31 |
| CIFAR-10: One hidden layer (convolutional) | | | | |
| Parameter Initialization | $0.26 \pm 0.02$ | $0.0043 \pm 0.0003$ | 0.974 | 0.64 |
| All nondeterminism sources | $0.22 \pm 0.01$ | $0.0044 \pm 0.0003$ | 0.973 | 0.68 |
| Random bit change | $0.14 \pm 0.01$ | $0.0024 \pm 0.0002$ | 0.993 | 0.18 |

Table 7: Linear and 2-layer experiments on CIFAR-10

a different arbitrary constant value, *e.g.* $S_2 = 2$. Due to this, there may be minor discrepancies when comparing the diversity in performance between two different sources of nondeterminism (albeit nothing likely to change general conclusions). Combined with the natural sampling variability implicit in only training a finite number of models, this can lead to paradoxical results such as the standard deviation for a particular metric being slightly higher for a random bit change as compared to an entirely different random parameter initialization. While we have separately validated that the general conclusions of our results hold when varying a few of these constant factors (*i.e.* running experiments where $S_2$ is set to 2 and 3, in this example), it is difficult to resolve the discrepancy entirely without models according to the full cross-product of random seeds, which is prohibitive due to the exponential number of such models.