Nondeterminism and Instability in Neural Network Optimization

Cecilia Summers [1] **Michael J. Dinneen** [1]

Abstract

Nondeterminism in neural network optimization produces uncertainty in performance, making small improvements difficult to discern from run-to-run variability. While uncertainty can be reduced by training multiple model copies, doing so is time-consuming, costly, and harms reproducibility. In this work, we establish an experimental protocol for understanding the effect of optimization nondeterminism on model diversity, allowing us to isolate the effects of a variety of sources of nondeterminism. Surprisingly, we find that all sources of nondeterminism have similar effects on measures of model diversity. To explain this intriguing fact, we identify the instability of model training, taken as an end-to-end procedure, as the key determinant. We show that even one-bit changes in initial parameters result in models converging to vastly different values. Last, we propose two approaches 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. 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 run-to-run variability?

Similarly, consider hyperparameter optimization, in which many possible values exist for a set of hyperparameters, with 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 train multiple independent copies of your model to understand its variability. While this helps address the problem, it is extremely wasteful, using more computing power, increasing the time required for effective research, and making reproducibility difficult, all while still leaving some uncertainty.

Ultimately, the source of this problem is nondeterminism in model optimization — randomized components of model training that cause each run to produce different models with their own performance characteristics. Nondeterminism itself occurs due to many factors: while the most salient source is the random initialization of parameters, other sources exist, including random shuffling of training data, stochasticity in data augmentation, explicit random operations (e.g. dropout (Srivastava et al., 2014)), asynchronous training (Recht et al., 2011), and even nondeterminism in low-level libraries such as cuDNN (Chetlur et al., 2014).

Despite the clear impact nondeterminism has on the efficacy of modeling, relatively little attention has been paid towards understanding its mechanisms. 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 can matter 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 *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 the initialization of a single weight by the smallest possible amount within machine precision ($\sim 6 \cdot 10^{-11}$) produces nearly as much variability as all other sources combined*. Therefore, any source of nondeterminism with any effect at all on model weights inherits at least this level of variability.

Last, we present promising results in reducing the effects of

[1]Department of Computer Science, University of Auckland, Auckland, New Zealand. Correspondence to: Cecilia Summers <cecilia.summers.07@gmail.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

instability on run-to-run variability. While we find that many approaches result in no apparent change, we propose and demonstrate two approaches that reduce model variability without any increase in model training time: accelerated model ensembling and test-time augmentation. Together, these provide the first encouraging signs for the tractability of this problem. Code has been made publicly available.[1]

2. Related Work

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

Instability. We use the term "stability" in a manner analogous to numerical stability (Higham, 2002), where a stable algorithm is one for which the final output (converged model) does not vary much as the input (initial parameters) are changed. In other contexts, the term "stability" has been used both in learning theory (Bousquet & Elisseeff, 2002) and in reference to vanishing and exploding gradients (Haber & Ruthotto, 2017).

3. Nondeterminism

Many sources of nondeterminism exist in neural network optimization, each of which affects the variability 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 zero-mean Gaussian with variance determined by the number of input connections to the layer (Glorot & Bengio, 2010; He et al., 2015).

Data Shuffling. In stochastic gradient descent, the gradient is approximated on a random subset of examples, commonly implemented by using small batches of data iteratively in a shuffled training dataset (Bottou, 2012). Shuffling may happen either once, before training, or in between each epoch of training, the variant we use in this work.

Data Augmentation. A common practice, data augmentation refers to randomly altering each training example to artificially expand the training dataset (Shorten & Khoshgoftaar, 2019). For example, randomly flipping images encourages invariance to left/right orientation.

Stochastic Regularization. Some types of regularization, such as Dropout (Srivastava et al., 2014), take the form of stochastic operations internal to a model during training. Other instances of this include DropConnect (Wan et al., 2013) and variable length backpropagation through time (Merity et al., 2017), among many others.

Low-level Operations. Often underlooked, many libraries that deep learning frameworks are built on, such as cuDNN (Chetlur et al., 2014), typically run nondeterministically in order to increase the speed of their operations. This nondeterminism is small when evaluated in the context of a single operation — in one test we performed it caused an output difference of 0.003%. In the case of cuDNN, the library we test, it is possible to disable nondeterministic behavior at a speed penalty on the order of $\sim 15\%$. However, unlike other nondeterminism sources, it is not possible to "seed" this; it is only possible to turn it on or off.

3.1. Protocol for Testing Effects of Nondeterminism

Performance Variability. Our protocol for testing the effects of sources of nondeterminism is based on properly controlling for each source. Formally, suppose there are N sources of nondeterminism, with source i controlled by 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, denote 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 $R = 100$ training runs, we set S_3 to the deterministic value of 1, S_2 to an arbitrary constant (typically 1 for simplicity), and test 100 different values of S_1. All together, this corresponds to training models for each of $(S_1, S_2, S_3) \in \{(i, 1, 1)\}_{i=1}^{100}$. To measure variability of a particular evaluation metric (*e.g.* cross-entropy or accuracy for classification), we calculate the standard devia-

[1] https://github.com/ceciliaresearch/nondeterminism_instability

tion (across all $R = 100$ models) of the metric. Note that it is also possible to test the effect of several sources of nondeterminism in tandem this way, *e.g.* by considering $(S_1, S_2, S_3) \in \{(i, i, 0)\}_{i=1}^{R}$ to measure the joint effect of all three sources in this example.

Representation Diversity. We also examine differences in the *representation* of trained models, complementary to variability in test set performance — this allows us to differentiate cases where two sources of nondeterminism have similar performance variability but actually produce models with disparate amounts of representational similarity. In order to rigorously examine this, we consider four distinct analyses of the functional behavior of models:

The first and simplest metric we consider is the average disagreement between pairs of models, with higher disagreement corresponding to higher diversity and variability. In contrast to our other metrics, this considers only the argmax of a model's predictions, which makes it the most limited but also the most interpretable of the group. This metric has also been used recently to compare similarity in the context of network ensembles (Fort et al., 2019).

Second, 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. Concretely, for a classification task, the predicted logits from each of R models are flattened into vectors of length $N * C$ (with N test examples and C classes), and we calculate the mean correlation coefficient of the predictions across all $\binom{R}{2}$ pairs of models. We use Spearman's ρ for the correlation coefficient, but note that other metrics are possible and yield similar conclusions. For this metric, a lower score indicates a more diverse set of models.

The third analysis we perform examines the change in performance in ensembling two models from the same source of nondeterminism. Intuitively, 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 ensembling should create an improvement, 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, and $\frac{S_i+S_j}{2}$ the ensemble of models S_i and S_j, this metric is formally determined by:

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

Last, for a more detailed view of learned representations internal to a network, we consider a state-of-the-art method for measuring the similarity of neural network representations, centered kernel alignment (CKA) (Kornblith et al., 2019), which has previously 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.* found to perform similarly to more complicated RBF kernels.

3.2. Experiments in Image Classification

We begin our study of nondeterminism with the fundamental task of image classification. We execute our protocol with CIFAR-10 (Krizhevsky et al., 2009) as a testbed, a 10-way classification dataset with 50,000 training images of resolution 32×32 pixels and 10,000 images 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 linear 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. Across all measures of performance variability and representation diversity, what we find is surprising and clear — while there are slight differences, each source of nondeterminism has very similar effects on the variability of final trained models. In fact, random parameter initialization, arguably the form of nondeterminism that variability in performance is most commonly attributed to, does not stand out based on any metric, and even combinations of multiple sources of nondeterminism produce remarkably little difference — all are within a maximum of 20% (relative) of each other.

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

3.3. Experiments in Language Modeling

Here we show that this phenomenon is not unique to image classification by 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 model uses a 256-dimensional word embedding, 512 hidden units per layer, and 3 layers of recurrent units, obtaining a perplexity (PPL) of 75.49 on the Penn Treebank test set.

Nondeterminism Source	Accuracy SD (%)	Cross-Entropy SD	Pairwise Disagree (%)	Pairwise Corr.	Ensemble Δ (%)
Parameter Initialization	0.23 ± 0.02	0.0074 ± 0.0005	10.7	0.872	1.82
Data Shuffling	0.25 ± 0.02	0.0082 ± 0.0005	10.6	0.871	1.81
Data Augmentation	0.23 ± 0.02	0.0072 ± 0.0005	10.7	0.872	1.83
cuDNN	0.22 ± 0.01	0.0083 ± 0.0007	10.5	0.873	1.76
Data Shuffling + cuDNN	0.21 ± 0.01	0.0077 ± 0.0005	10.6	0.871	1.80
Data Shuffling + Aug. + cuDNN	0.22 ± 0.01	0.0074 ± 0.0005	10.7	0.871	1.84
All Nondeterminism Sources	0.26 ± 0.02	0.0072 ± 0.0005	10.7	0.871	1.82

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 each standard deviation. The fourth, fifth, and sixth columns give the average percentage of examples models disagree on, the average pairwise Spearman's correlation coefficient between predictions, and the average change in accuracy from ensembling two models, respectively (Sec. 3.1).

Nondeterminism Source	PPL SD	Pairwise Disagree (%)	Ensemble PPL Δ
Parameter Initialization	0.20 ± 0.01	17.3	-2.07
Stochastic Operations	0.19 ± 0.01	17.3	-2.08
All Nondeterminism Sources	0.18 ± 0.01	17.4	-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 for language modeling tasks, so changes in PPL from ensembling are negative.

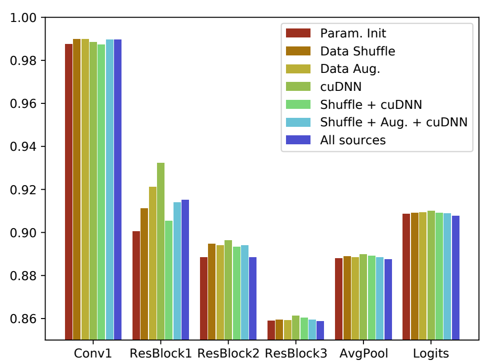

Figure 1. Average CKA representation similarity (Kornblith et al., 2019) for pairs of ResNet-14 models on CIFAR-10 across nondeterminism sources and a variety of network layers.

For this task, two sources of nondeterminism are relevant: random parameter initialization, and stochastic operations, including a variation of dropout and variable length backpropagation through time, which share a common seed. To measure performance variability, PPL is the most widely-accepted metric, and for diversity in representation we focus on only two metrics (pairwise disagreement and benefits from ensembling) because CKA was not designed for variable-length input and standard 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 almost no difference across all diversity metrics, showing the phenomenon generalizes beyond image classification and ResNets.

3.4. Nondeterminism Throughout Training

One hypothesis for the this phenomenon's cause is the sensitivity of optimization in the initial phase of learning, which recent work has demonstrated 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 models identically for the first N epochs and only then introducing nondeterminism would result in significantly less variability in final trained models, measured across all metrics. 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 perform this experiment for the ResNet-14 model on CIFAR-10 in Fig. 2, where we find that the beginning of training is not particularly sensitive to nondeterminism. Instead, model variability is nearly as high when enabling nondeterminism even after 50 epochs, and we see only a gradual reduction in final model variability as the onset of nondeterminism is moved later and later.

4. Instability

Why does each source of nondeterminism have similar effects on model variability? We approach this question by finding the smallest possible change that produces the same amount of variability. In doing so, we find that only an extremely tiny change is necessary, thereby demonstrating the *instability* in optimizing neural networks.

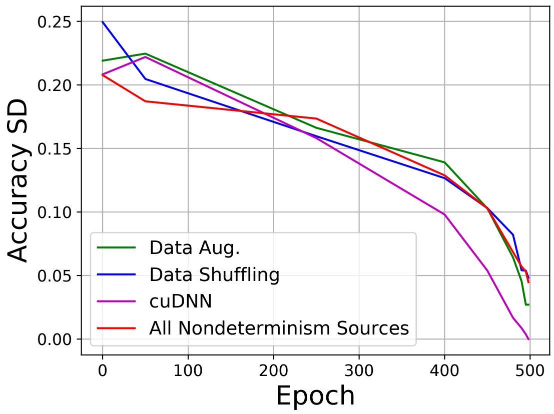

Figure 2. The effect of the onset of nondeterminism on the variability of accuracy in converged models. Each point corresponds to training 100 models deterministically for a certain number of epochs (x-axis), then enabling a given source of nondeterminism by varying its seed starting from that epoch and continuing through to the end of training, then measuring the accuracy SD (y-axis).

4.1. Instability and Nondeterminism

To demonstrate, we perform a simple experiment: First we deterministically train a simple ResNet-14 model on CIFAR-10, achieving a test cross-entropy of 0.3519 and accuracy of 90.0%. Then, we train another model 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 possible amount 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, this could change a value from -0.0066514308 to -0.0066514313, a difference on the order of $5 \cdot 10^{-10}$.

What happens when we optimize this model, different from the original by only a single bit? By the end of the first epoch of training, with the learning rate still warming up, the new model already differs in accuracy by 0.18% (25.74% vs 25.56%). In one more epoch the difference is a larger 2.33% (33.45% vs 31.12%), and after three epochs, the difference is a staggering 10.42% (41.27% vs 30.85%). Finally, at the end of training the model weights converge, with the new model obtaining an accuracy of 90.12% and a cross-entropy of 0.34335, substantially different from the original despite only a tiny change in initialization. Viewing the optimization process end-to-end, with the initial parameters as the input and a given performance metric as the output, this demonstrates a condition number $\frac{\|\delta f\|}{\|\delta x\|}$ of $1.8 \cdot 10^7$ for cross-entropy and $2.6 \cdot 10^8$ for accuracy.

We can more rigorously test this using our 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 small 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 — so long as nondeterminism produces any change in model weights, whether by changing the input slightly, altering the gradient in some way, or any other effect, it will produce *at least* as much model variability as caused by the instability of model optimization.

4.2. Instability and Depth

Instability occurs in networks of more than a single layer.

Due to convexity, linear models optimized with a cross-entropy loss and an appropriate learning rate schedule always converge to a global minimum. However, in practice we find an even stronger property: when initial weights are modified by a single bit, beyond simply converging to the same final value, 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 way 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 $\sim 1 \cdot 10^{-7}$, far below that of any deeper model.

In contrast, instability occurs as soon as a single hidden layer was added, with an accuracy SD of 0.28 and cross-entropy SD of 0.0051 for a model whose hidden layer is fully-connected, and an accuracy SD of 0.14 and cross-entropy SD of 0.0022 when the hidden layer is convolutional, both a factor of $10{,}000$ greater than the linear model. See Supplementary Material for full details and a visualization of the effects of instability during training.

5. Reducing Variability

Here we identify and demonstrate two approaches that partially mitigate the variability caused by nondeterminism and instability. See the Supplementary Material for learnings on approaches which were unsuccessful in reducing variability.

Accelerated Ensembling. As previously mentioned, the standard practice for mitigating run-to-run variability is to train multiple independent copies of a model, gaining a more robust performance estimate by measuring a metric of interest over multiple trials. Ensembling is a similar alternative approach, which shares the intuition of multiple independent training runs, but differs in that the predictions themselves are averaged and the performance of the ensembled model

Nondeterminism Source	Accuracy SD (%)	Cross-Entropy SD	Pairwise Disagree (%)	Pairwise Corr.	Ensemble Δ (%)
Random Bit Change	0.21 ± 0.01	0.0068 ± 0.0004	10.6	0.874	1.82

Table 3. The effect of instability — randomly changing a single weight by one bit during initialization for ResNet-14 on CIFAR-10.

Nondeterminism Source	PPL SD	Pairwise Disagree (%)	Ensemble PPL Δ
Random Bit Change	0.19 ± 0.01	17.7	-2.07

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

itself is measured. Indeed, as demonstrated in Table 5 (top), ensembles of larger models have less variability. However, since ensembling still requires training multiple copies of models, is does not reduce the computational burden caused by nondeterminism and instability.

To that end, we propose the use of recent *accelerated* ensembling techniques to reduce variability. Accelerated ensembling is a new research direction in which only one training run is needed (Huang et al., 2017; Garipov et al., 2018; Wen et al., 2020). While such techniques typically underperform ensembles composed out of truly independent models, the nature of their accelerated training can reduce variability without incurring additional cost during training. The approach we focus on is the Snapshot Ensemble (Huang et al., 2017), which uses a cyclic learning rate schedule, creating the members of an ensemble out of models where the learning rate is 0 in the cyclic learning rate schedule.

In Table 5 (bottom), we compare a snapshot ensemble ("Acc. Ens.") with 5 cycles in its learning rate (*i.e.* model snapshots are taken after every 100 epochs of training) to ordinary ensembling on CIFAR-10 with all sources of nondeterminism enabled. Despite training only a single model, the accelerated ensemble had variability in accuracy and cross-entropy comparable to an ensemble of two independently-trained models, with other metrics comparable to those of even larger ensembles. Across measures, accelerated ensembling reduces variability by an average of 48% relative.

Test-Time Data Augmentation. Test-time data augmentation (TTA) is the practice of augmenting test set examples using data augmentation, averaging model predictions made on each augmented example, and is typically used to improve generalization (Szegedy et al., 2015). Beyond improved generalization, though, TTA can be thought of as a form of ensembling in data-space (as opposed to the model-space averaging of standard ensembling), giving it potential for mitigating the variability due to nondeterminism.

In Table 5 (bottom), we show results on CIFAR-10 with horizontal flip TTA and image cropping TTA (details in Supplementary Material), and also experiment with combining accelerated ensembling with TTA. Simple flip TTA reduces variability across all metrics (21% relative reduction on average), standalone cropping reduces variability by 16% to 21% depending on the number of crops, and employing both as TTA pushes this up to 37%. Combined with accelerated model ensembling, variability is reduced by up 61% without any increase in training budget.

6. Generalization Experiments

In this section we detail additional experiments showing the generalization of our results on nondeterminism, instability, and methods for reducing variability to other datasets (MNIST, ImageNet) and model architectures. We compile our main generalization results in Table 6, with additional results in the Supplementary Material.

CIFAR-10. On CIFAR-10, in addition to the ResNet-14 employed throughout this work, we experiment with a smaller 6-layer variant, larger 18-layer variant, VGG-11 (Simonyan & Zisserman, 2014), and a 50%-capacity ShuffleNetv2 (Ma et al., 2018), with even more architectures in the Supplementary Material. As shown in Table 6, the observations around instability and its relationship to nondeterminism generally hold for these architectures, with a close correspondence between the magnitude of effects for a random bit change and each of the five metrics considered.

Turning towards our proposals (Sec. 5) for mitigating the effects of nondeterminism and instability on model variability, we find across all model architectures that both accelerated ensembling and test-time augmentation reduce variability across nearly all metrics, with perhaps larger relative reductions for larger models and the pairwise metrics. Only for the intersection of the smallest model (ResNet-6) and metrics of performance variability (Accuracy SD and Cross-Entropy SD) was there no benefit.

MNIST. Experiments on MNIST (LeCun et al., 1998), allow us to test whether our observations hold for tasks with very high accuracy — 99.14% for our relatively simple baseline model, which has two convolution and fully-connected layers. As before, we find similar effects of nondeterminism for parameter initialization and all nondeterminism sources,

Model	Training Cost	Accuracy SD (%)	Cross-Entropy SD	Pairwise Disagree (%)	Pairwise Corr.	Ensemble Δ (%)	Variability Reduction
Single Model	1×	0.26 ± 0.02	0.0072 ± 0.0005	10.7	0.871	1.82	*n/a*
Ensemble ($N = 2$)	2×	0.19 ± 0.02	0.0044 ± 0.0004	6.9	0.929	0.89	39%
Ensemble ($N = 3$)	3×	0.15 ± 0.02	0.0033 ± 0.0005	5.5	0.951	0.59	55%
Ensemble ($N = 4$)	4×	0.17 ± 0.02	0.0030 ± 0.0004	4.6	0.963	0.43	60%
Ensemble ($N = 5$)	5×	0.12 ± 0.02	0.0028 ± 0.0004	4.1	0.970	0.34	67%
Ensemble ($N = 10$)	10×	0.11 ± 0.02	0.0022 ± 0.0004	2.9	0.985	0.20	76%
Ensemble ($N = 20$)	20×	0.11 ± 0.04	0.0018 ± 0.0005	2.0	0.992	0.08	81%
Acc. Ens.	1×	0.19 ± 0.02	0.0044 ± 0.0003	6.1	0.957	0.63	48%
Single/Flip-TTA	1×	0.24 ± 0.02	0.0061 ± 0.0005	8.2	0.905	1.20	21%
Single/Crop25-TTA	1×	0.23 ± 0.02	0.0059 ± 0.0004	9.2	0.893	1.49	16%
Single/Crop81-TTA	1×	0.21 ± 0.01	0.0055 ± 0.0004	8.8	0.898	1.39	21%
Single/Flip-Crop25-TTA	1×	0.21 ± 0.02	0.0051 ± 0.0004	7.2	0.920	0.99	33%
Single/Flip-Crop81-TTA	1×	0.19 ± 0.01	0.0049 ± 0.0004	6.9	0.922	0.92	37%
Acc. Ens./Flip-TTA	1×	0.15 ± 0.01	0.0039 ± 0.0003	5.0	0.967	0.45	58%
Acc. Ens./Flip-Crop81-TTA	1×	0.16 ± 0.01	0.0033 ± 0.0002	4.6	0.972	0.38	61%

Table 5. Comparison of single and ensemble model variability on CIFAR-10 with proposed methods for reducing the effects of nondeterminism. For standard ensembles, N denotes the number of constituent models, "Acc. Ens." uses the Snapshot Ensemble method of accelerated ensembling, and [Single|Acc. Ens.]/[Flip|CropX|Flip-CropX]-TTA use either horizontal flips, crops (with X denoting the number of crops), or flips and crops for test-time augmentation on top of either regular single models or an accelerated ensemble. Also shown is the training time and average relative reduction in variability across metrics compared to the baseline 'Single Model". All results are based on 100 runs of model training.

including a comparable effect (albeit 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 metrics of diversity, further suggesting that the magnitude of instability might be at least partially related to the interplay of model architecture, capacity, and degree of overfitting.

In terms of the mitigations against variability, only test-time augmentation appeared to significantly help. For MNIST, the only augmentation employed was cropping, with a small 1-pixel padding (models were trained with no data augmentation). While the fact that accelerated ensembling did not result in improvements is not particularly important in practice (since MNIST models are fast to train), it is an interesting result, which we hypothesize is also related to the degree of overfitting (similar to ResNet-6 on CIFAR-10).

ImageNet. We perform larger-scale tests on ImageNet using 20 runs of a ResNet-18 (He et al., 2016), trained for 120 epochs, obtaining an average top-1 accuracy of 71.9% on the ImageNet validation set. Again, we find evidence supporting instability, with "Random Bit Change" having levels of variability comparable to models trained with all nondeterminism sources. For reducing variability, we focus our efforts on TTA, where we find modest improvements for both flipping-based and crop-based TTA on all metrics other than Accuracy SD, noting the large error bars of Accuracy and Cross-Entropy SD relative to their point estimates.

7. Conclusion

In this work, we have shown two surprising facts: First, though conventional wisdom holds that run-to-run variability in model performance is primarily determined by random parameter initialization, many sources of nondeterminism actually result in similar levels of variability. Second, a key driver 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 all weights to completely random values. We have also identified two approaches for reducing the variability in model performance and representation without incurring any additional training cost: ensembling in model-space via accelerated model ensembling, and ensembling in data-space via the application of test-time data augmentation.

Many promising directions for future work exist. One important line of inquiry is in developing stronger theoretic understanding of the instability in optimization, beyond the largely empirical evidence in our work. Another natural direction is improving upon the algorithms for reducing the effects of instability on model variability — although both accelerated ensembling and TTA help, they are far from solving the problem entirely and incur additional computation during test time. Last, it would be interesting to examine our findings on even larger models (*e.g.* transformers for NLP and image recognition) and problems outside the fully supervised setting. We hope that our work has shed light on a complex phenomenon that affects all deep learning researchers and inspires further research.

Nondeterminism Source	Accuracy SD (%)	Cross-Entropy SD	Pairwise Disagree (%)	Pairwise Corr.	Ensemble Δ (%)
CIFAR-10: ResNet-6					
Parameter Initialization	0.50 ± 0.04	0.0117 ± 0.0010	20.0	0.925	2.17
All Nondeterminism Sources	0.43 ± 0.03	0.0106 ± 0.0007	20.1	0.924	2.17
Random Bit Change	0.41 ± 0.02	0.0094 ± 0.0006	19.8	0.925	2.12
Single/Flip-Crop-TTA	0.44 ± 0.03	0.0096 ± 0.0006	15.6	0.949	1.41
Acc. Ens.	0.45 ± 0.03	0.0104 ± 0.0007	14.0	0.963	0.99
Acc. Ens./Flip-Crop-TTA	0.43 ± 0.03	0.0096 ± 0.0006	11.6	0.973	0.71
CIFAR-10: ResNet-18					
Parameter Initialization	0.15 ± 0.01	0.0067 ± 0.0005	4.7	0.814	0.71
All Nondeterminism Sources	0.18 ± 0.01	0.0073 ± 0.0005	4.8	0.808	0.75
Random Bit Change	0.13 ± 0.01	0.0060 ± 0.0005	4.7	0.830	0.73
Single/Flip-Crop-TTA	0.14 ± 0.01	0.0047 ± 0.0003	3.4	0.851	0.41
Acc. Ens.	0.13 ± 0.01	0.0038 ± 0.0003	2.9	0.884	0.31
Acc. Ens./Flip-Crop-TTA	0.11 ± 0.01	0.0029 ± 0.0002	2.2	0.909	0.19
CIFAR-10: ShuffleNetv2-50%					
Parameter Initialization	0.22 ± 0.01	0.0112 ± 0.0007	8.4	0.696	1.38
All Nondeterminism Sources	0.22 ± 0.02	0.0123 ± 0.0008	8.4	0.692	1.40
Random Bit Change	0.21 ± 0.01	0.0107 ± 0.0006	8.3	0.695	1.36
Single/Flip-Crop-TTA	0.18 ± 0.01	0.0093 ± 0.0007	6.5	0.762	0.90
Acc. Ens.	0.18 ± 0.01	0.0067 ± 0.0005	5.0	0.930	0.52
Acc. Ens./Flip-Crop-TTA	0.15 ± 0.01	0.0051 ± 0.0004	4.1	0.948	0.35
CIFAR-10: VGG-11					
Parameter Initialization	0.20 ± 0.01	0.0063 ± 0.0004	6.6	0.807	0.91
All Nondeterminism Sources	0.18 ± 0.01	0.0065 ± 0.0004	6.6	0.806	0.94
Random Bit Change	0.16 ± 0.01	0.0060 ± 0.0004	6.5	0.811	0.89
Single/Flip-Crop-TTA	0.15 ± 0.01	0.0042 ± 0.0003	4.2	0.892	0.36
Acc. Ens.	0.13 ± 0.01	0.0041 ± 0.0003	4.1	0.914	0.39
Acc. Ens./Flip-Crop-TTA	0.11 ± 0.01	0.0026 ± 0.0002	2.8	0.951	0.17
MNIST					
Parameter Initialization	0.047 ± 0.0036	0.0024 ± 0.0001	0.54	0.941	0.064
All Nondeterminism Sources	0.046 ± 0.0032	0.0022 ± 0.0001	0.56	0.939	0.068
Random Bit Change	0.035 ± 0.0026	0.0011 ± 0.0001	0.30	0.989	0.011
Single/Crop-TTA	0.039 ± 0.0025	0.0016 ± 0.0001	0.38	0.953	0.037
Acc. Ens.	0.050 ± 0.0031	0.0019 ± 0.0001	0.55	0.943	0.064
Acc. Ens./Crop-TTA	0.046 ± 0.0028	0.0013 ± 0.0001	0.40	0.956	0.039
ImageNet: ResNet-18					
All Nondeterminism Sources	0.10 ± 0.01	0.0027 ± 0.0004	20.7	0.814	1.94
Random Bit Change	0.09 ± 0.01	0.0026 ± 0.0004	20.6	0.815	1.91
Single/Flip-TTA	0.12 ± 0.02	0.0022 ± 0.0004	18.8	0.827	1.60
Single/Crop-TTA	0.10 ± 0.02	0.0023 ± 0.0003	19.8	0.815	1.72
Single/Flip-Crop-TTA	0.11 ± 0.01	0.0018 ± 0.0002	18.2	0.825	1.45

Table 6. Generalization experiments of nondeterminism and instability with other architectures on CIFAR-10, ImageNet, and MNIST. For CIFAR-10 and MNIST, each row is computed from the statistics of 100 trained models, and for ImageNet, each row is computed from 20 trained models. Within each section the most relevant comparisons to make are between "Random Bit Change" and "All Nondeterminism Sources" to evaluate instability, and between "All Nondeterminism Sources", "Acc. Ens.", and each TTA method to evaluate the efficacy of our proposals to mitigate the effects of nondeterminism and instability (all TTA models have all sources of nondeterminism enabled). Notation follows Tables 1 and 5, and all TTA cropping for CIFAR-10 uses the 81-crop variant.

References

Achille, A., Rovere, M., and Soatto, S. Critical learning periods in deep neural networks. In *International Conference on Learning Representations*, 2019.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

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

Bradbury, J., Merity, S., Xiong, C., and Socher, R. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

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

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

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.

Glorot, X. and Bengio, Y. 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.

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

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Higham, N. J. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

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.

Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

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

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

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

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

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

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. 1993.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Nagarajan, P., Warnell, G., and Stone, P. The impact of nondeterminism on reproducibility in deep reinforcement learning. 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

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

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

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

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.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066, 2013.

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