

Sufficient is better than optimal for training neural networks

Corresponding Author: Professor Greg van Anders

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

**** What are the noteworthy results?**

The work proposed and validated an idea of "sufficient training": It demonstrates better generalization performance by training neural networks to get "good enough" (near-optimal, higher training loss) solutions, instead of adopting the usual optimal ones on training set (i.e. lowest training loss).

The work achieves "sufficient training" using a physics motivated optimizer: it models neural network parameters as particles in a simulated physical molecular system and the loss function as the system potential. The negative gradient of the loss (obtained by automatic differentiation, same as backpropagation) pushes the "particles" (network parameters) towards a minimum of the loss function, similar to a force on the particles. A Nosé-Hoover chain (NHC) thermostat is used to control the temperature of this physical system by introducing virtual particles that interact with the network parameters. The NHC allows generating networks (and form ensembles) at any target temperature and thus control the level of fitting to the training set.

The idea of "sufficient training" is validated on two regimes: 1. retrofitting a pretrained neural network and reducing its overfitting on training set. This starts with a pretrained model at temperature 0 and gradually increases temperature. 2. "Ab initio" training: applying the above described sufficient training optimizer from the start of training. The system is trained at a fixed target temperature throughout training. The authors show that in both regimes, the proposed sufficient training optimizer brings benefits to the generalization performance compared to the baseline Adam optimizer.

The result is validated on two tasks on a few datasets: 1. classification: MNIST, HIGGS, IRIS datasets 2. Regression: fitting a noisy sine curve, Auto-MPG (S) single-variate regression, Auto-MPG (M) multivariate regression.

**** Will the work be of significance to the field and related fields? How does it compare to the established literature? If the work is not original, please provide relevant references.**

The general topic of sufficient training may be potentially interesting. However, I believe there are a few more things to be addressed in order to fully convince the general machine learning community.

1. Most people agree that after some level of training on a small dataset, neural networks can overfit and test errors can increase. The standard approach around overfitting in the current big data era is to use larger datasets and more data augmentation. Therefore, the significance of this approach (or any approach mitigating overfitting) depends on the problem domain. In general, overfitting seems not a significant problem in most big data domains. For demonstrating your approach, it may be useful to find some domains where it is always in a small data regime and training data cannot be obtained and/or augmented easily (like some specific astronomy, biology, medical domains).

2. The proposed sufficient training, although motivated by physical systems, is essentially an approach that tradeoff training performances with test performances. A baseline comparison to other simpler approaches may be needed to establish the necessity of your less simple physics formulations. For example, how does some simple noise injection schemes to parameters or combinations of early stopped models work compared to your formulation?

To make significant contribution in machine learning, it may benefit from closely investigating the problem and identifying the

key cause of overfitting instead of only proposing a large physics-based formulation and claim that it helps mitigating overfitting. This may require simplifying your system and identifying minimal components that make it work, which will provide more insight to machine learning researchers who do not share a physics background/interest.

**** Does the work support the conclusions and claims, or is additional evidence needed?**

As discussed above, I believe the comparison with similar techniques or simpler alternative is lacking. It is not yet demonstrated that this physics-motivated formulation is really necessary. Not enough simpler alternatives are discussed and compared.

Some claims in the main text and abstract are a bit exaggerated. The author seems to suggest that "sufficient training" is different from optimizing. A more reasonable view is that "sufficient training" is just another way of optimizing, since your approach is essentially a different type of optimizer, which rely on gradients computed by automatic differentiation via backpropagation. The way particles are pushed by forces of gradients in your system is essentially optimizing. The tone of "creating a new paradigm different from optimization" needs to be soften to avoid misleading if you still rely on backpropagation/automatic differentiation.

**** Are there any flaws in the data analysis, interpretation and conclusions? Do these prohibit publication or require revision?**

One thing I notice is that the results of your approach are all obtained with ensemble models. It is expected that ensemble models can benefit generalization, it may be fairer to compare with baselines in a non-ensemble fashion if possible. Or you can establish an ensemble baseline using popular ensemble techniques.

This ensemble vs. single model comparison issue need to be addressed in order to support the claim that your approach is better than baseline.

**** Is the methodology sound? Does the work meet the expected standards in your field**

Some general comments about the direction/approach are given in the above section about significance.

Regarding details and execution, I feel that the datasets used are on a simple/toy or uncommon side. MNIST, sine fitting, IRIS, etc. are all relatively toy datasets in machine learning nowadays. CIFAR10 and CIFAR100 have the same format as MNIST and are less "toy" --- it is pretty standard in the field to try on them. Optionally, if you can obtain some results on advanced tasks like modern NLP and ImageNet, it would make your work much more solid.

**** Is there enough detail provided in the methods for the work to be reproduced?**

I believe there is enough detail. Also the code is provided.

(Remarks on code availability)

The code has a README file and runs on default settings via one command line without any problem. The code consists of a demo of applying the proposed approach on learning a noisy sine curve.

Reviewer #2

(Remarks to the Author)

In this manuscript, the authors present a novel algorithm, termed "simmering," for training neural networks. By leveraging Nose-Hoover chain thermostats, a concept originating from molecular dynamics, the proposed approach aims to address the persistent challenge of overfitting in deep learning. While the idea is intriguing and has potential, the current work lacks sufficient experimental evidence to fully support the claims made.

The experiments conducted primarily utilize synthetic data or relatively simple datasets, including MNIST, HIGGS, and IRIS, in conjunction with baseline architectures such as LeNet-5. While these initial results provide some insight into the potential of the proposed method, they are insufficient to establish its robustness and generalizability. To strengthen the conclusions, I strongly recommend that the authors extend their evaluation to more complex and realistic datasets, such as ImageNet, and incorporate benchmark neural network architectures, such as Transformer or ResNet-101.

Furthermore, the manuscript would benefit from a more thorough comparison of the proposed method against existing regularization techniques commonly used in deep learning, particularly in mitigating overfitting. Examples include weight decay, dropout, and early stopping. Such comparisons are essential to position the proposed approach within the existing body of work and to demonstrate its advantages or limitations.

Finally, the clarity of the methodological description should be improved. As it stands, the explanation of the proposed approach is not sufficiently detailed to enable reproducibility. I recommend the authors provide a more comprehensive and precise account of the algorithm, including its implementation details and overall pipeline.

In summary, while the manuscript introduces a potentially promising approach, significant revisions are needed to

substantiate the claims and enhance the clarity and reproducibility of the work. With these improvements, the study could make a more meaningful contribution to the field.

(Remarks on code availability)

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

Most of my questions are reasonably well addressed by the authors' response. The newly added experiments on CIFAR-10 with ConvNets and NLP with Transformers make the work more solid. Yet some more details are needed for those experiments: 1. for all settings, both training and test performances should be shown, preferably in curves, as functions of training epochs. 2. did you use any data augmentation with CIFAR-10?

Regarding the new Figure 3 of model confidence. It would be more convincing if some statistics are shown with a larger sample size, instead of a single example. One example is good for illustration purpose but it is not sufficient to draw a conclusion on such a property.

(Remarks on code availability)

Reviewer #2

(Remarks to the Author)

The paper proposes a “sufficient training” method called Simmering to mitigate overfitting in deep learning. It uses a Nosé–Hoover Chain (NHC) thermostat to sample parameter configurations and aggregates predictions via ensemble. While applying NHC to deep learning is novel, the core gains against overfitting largely stem from ensembling, which is already well studied.

Simmering requires forward and backward passes over many parameter configurations, incurring substantial computational and time costs that scale poorly for large models. However, the paper provides no quantitative analysis or comparison of computational complexity, training time, or resource usage, and experiments are mostly limited to small-scale settings, making it difficult to assess real-world performance and scalability.

The authors claim that “ab initio” Simmering from random initialization outperforms baselines, but do not provide corresponding training/validation loss curves, leaving the evidence insufficient. The assertion in Supplementary Fig. 1 that Adam and Simmering follow distinct training trajectories is also questionable: the Adam trajectory reflects training from scratch and is expected to move normal to loss contours; the Simmering trajectory is a 100-epoch slice taken from the Adam run after near-convergence, where oscillations around a local minimum are typical. Thus, differences between Fig. 1a and Fig. 1b are likely due to different training phases rather than algorithmic mechanics. Moreover, the images in Fig. 3 are very low resolution, making human recognition difficult and weakening the persuasive power of the comparison.

The writing also needs improvement: the paper contains many mathematical symbols and formulas, but the exposition is not well organized and the notation is unclear, failing to present an intuitive algorithmic pipeline and hindering reproducibility. I recommend adding a complete training pipeline description (including hyperparameters, temperature/learning-rate schedules, sampling and ensembling strategy), providing complexity and resource-cost comparisons, releasing convergence curves for from-scratch training, and supplying higher-quality visualizations to substantiate the claims.

(Remarks on code availability)

Version 2:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The results seem overall interesting. A few things to improve:

1. I feel that Supplementary Figure 1 and 2 are much more informative than Table 1. Perhaps replace table 1 with those figures will make the main paper more convincing.

2. From Supplementary Figure 1, it seems that Simmering produces more "diverse" models that works better in an ensemble than traditional early stopping. Although this may be beyond the scope of current paper, it may be interesting to explore more along this direction.

3. In Supplementary Figure 2, Simmering converges much faster than the baseline method on Transformers, where it only uses 20 epochs where the baseline takes 60 epochs to reach the same level of performance. If this result is true, it could be more interesting the rest of the paper combined. Why is this convergence speedup not explored more in the paper? I find it hard to believe that it converges so much faster.

4. Main text Figure 3 about classification confidence, on the other hand, is much less interesting since it contains merely one illustrative example. It is hard to conclude anything from one example. If a more statistical measure is beyond the scope the paper, I suggest reducing the emphasis on this, by perhaps moving this one to the supplementary materials and add the Supplementary Figure 1 and 2 to the main text, together with related analyses.

(Remarks on code availability)

I looked at the original code in the initial review and it ran without problem. I have NOT yet checked the latest code about CIFAR10 and transformers (if any).

Reviewer #2

(Remarks to the Author)

My main concerns have been addressed, but I have one further suggestion. Although this is an interdisciplinary work, the proposed method is ultimately aimed at training neural network models. Therefore, the notations and pseudocode in the paper should follow the conventions of neural network training algorithms, rather than those of thermodynamics. This will enable researchers in the artificial intelligence field to better understand and utilize the proposed method.

(Remarks on code availability)

NA

Version 3:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

My suggestions and concerns have been addressed in the latest manuscript. The results presented in the paper are interesting and convincing to me.

(Remarks on code availability)

It would be nice if the author can provide the source code for the transformer based NLP experiment.

Reviewer #2

(Remarks to the Author)

My concerns have been addressed

(Remarks on code availability)

Open Access This Peer Review File is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

In cases where reviewers are anonymous, credit should be given to 'Anonymous Referee' and the source.

The images or other third party material in this Peer Review File are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

Manuscript ID: **NCOMMS-25-24798-T**
Title: **Simmering: sufficient is better than optimal for training neural networks**
Authors: Irina Babayan, Hazhir Aliahmadi, Greg van Anders

Response to Referee Reports

We thank both Referees for their constructive feedback on the manuscript. The Referees made a number of suggestions that prompted us to significantly extend our analysis and to clarify the presentation of our approach. We believe the new analyses (of CIFAR-10 and a transformer-based translation task) better situate our approach in the machine learning literature.

The reports are reproduced below in their entirety. The quotations are verbatim and indented in blue, sans-serif font. In some places we quote text directly from the revised manuscript using red, serif font; text is reproduced verbatim except in places where it involves reference numbers, which, since this report does not appear with the context of the full reference list, we have given author-date pairs instead.

In addition, we corrected some minor spelling and grammatical errors. We also made substantial clarifications to the Discussion and beginning of the Methods.

Following the response to the Referees, we append a highlighted version of the manuscript that indicates the changes we have made in response to the Referees' suggestions, and formatting issues.

We hope that you will find the revised manuscript suitable for publication in *Nature Communications*.

Report of Referee 1

From the report:

**** What are the noteworthy results?**

The work proposed and validated an idea of "sufficient training": It demonstrates better generalization performance by training neural networks to get "good enough" (near-optimal, higher training loss) solutions, instead of adopting the usual optimal ones on training set (i.e. lowest training loss).

The work achieves "sufficient training" using a physics motivated optimizer: it models neural network parameters as particles in a simulated physical molecular system and the loss function as the system potential. The negative gradient of the loss (obtained by automatic differentiation, same as backpropagation) pushes the "particles" (network parameters) towards a minimum of the loss function, similar to a force on the particles. A Nosé-Hoover chain (NHC) thermostat is used to control the temperature of this physical system by introducing virtual particles that interact with the network parameters. The NHC allows generating networks (and form ensembles) at any target temperature and thus control the level of fitting to the training set.

The idea of "sufficient training" is validated on two regimes: 1. retrofitting a pretrained neural network and reducing its overfitting on training set. This starts with a pretrained model at temperature 0 and gradually increases temperature. 2. "Ab initio" training: applying the above described sufficient training optimizer from the start of training. The system is trained at a fixed target temperature throughout training. The authors show that in both regimes, the proposed sufficient training optimizer brings benefits to the generalization performance compared to the baseline Adam optimizer.

The result is validated on two tasks on a few datasets: 1. classification: MNIST, HIGGS, IRIS datasets 2. Regression: fitting a noisy sine curve, Auto-MPG (S) single-variate regression, Auto-MPG (M) multivariate regression.

We thank the Referee for their careful examination of our results.

From the Report:

** Will the work be of significance to the field and related fields? How does it compare to the established literature? If the work is not original, please provide relevant references.

The general topic of sufficient training may be potentially interesting. However, I believe there are a few more things to be addressed in order to fully convince the general machine learning community.

1. Most people agree that after some level of training on a small dataset, neural networks can overfit and test errors can increase. The standard approach around overfitting in the current big data era is to use larger datasets and more data augmentation. Therefore, the significance of this approach (or any approach mitigating overfitting) depends on the problem domain. In general, overfitting seems not a significant problem in most big data domains. For demonstrating your approach, it may be useful to find some domains where it is always in a small data regime and training data cannot be obtained and/or augmented easily (like some specific astronomy, biology, medical domains).

We thank the Referee for this valuable opportunity to discuss the relevance and applicability of sufficient training in the era of big data.

We will start by addressing the relevance of alternative training paradigms, like sufficient training, in large data regimes. Although the proportionality between a model's size (in this case, the number of trainable neural network parameters) and the training dataset size plays a key role in overfitting risk, both computational and theoretical advancements have allowed model capacity growth to keep pace with dataset growth. State-of-the-art architectures such as ResNet, Transformer-based models like BERT and GPT, and large convolutional networks have been developed with degrees of freedom in the hundreds of millions to billions to address real-world problems in vision, language, and speech. Nonetheless, overfitting remains a persistent obstacle to generalizability even in these "big data" regimes. For example, the "double descent" training phenomenon shows that increasing model capacity beyond what is necessary for interpolation can initially worsen unseen data prediction error before performance improvements occur, (Belkin *et al.*, 2019) underscoring that the classical overfitting unseen data loss "peak" persists in modern settings. The generalization advantage observed by using dropout when learning datasets like ImageNet or CIFAR-10 indicates that even large models learning from large datasets can benefit significantly from regularization to combat overfitting. (Srivastava *et al.*, 2014) However, even under regularization, modern deep networks have been shown to have the capacity to perfectly memorize random labels and noise, (Zhang *et al.*, 2017) demonstrating their massive overfitting capacity. Overall, overfitting continues to manifest when employing optimization, even in "big data" regimes.

We acknowledge that, in addition to using larger datasets, data augmentation (DA) is another common overfitting mitigation technique aimed at combating the potential mismatch between model capacity and dataset size. While DA can reduce model variance and thereby improve generalizability, it is not generically suitable for learning complex data. By making the data covariance more isotropic, DA can erase informative data structures in high-dimensional feature spaces and thus increase model bias, particularly when applied too aggressively or to overparameterized models. Certain augmentations can also introduce train–test distribution shifts, harming performance, especially in regression tasks. (Lin *et al.*, 2024) Moreover, although DA tends to improve a model’s average classification accuracy, this can come at the expense of substantial drops (up to 20%) in individual class accuracy, as observed on ImageNet. (Kirichenko *et al.*, 2023) Finally, while DA has shown clear benefits for non-pretrained models and in small-data regimes, recent work (Longpre *et al.*, 2020) has found that these accuracy gains often do not translate to modern pretrained Transformer architectures, where task-agnostic DA fails to consistently improve performance.

Overall, overfitting is not a training phenomenon that is exclusive to small-data domains. Furthermore, employing data augmentation does not consistently improve generalizability, particularly when learning high-dimensional data or with modern architectures.

We thank the Referee for raising this important point; accordingly, we have added the following section to the introduction:

Even in large- and augmented-data regimes, with correspondingly advanced architectures such as ResNet (He *et al.*, 2016) and Transformers, (Vaswani *et al.*, 2017) modern networks can memorize noise or random labels with negligible training error while still exhibiting poor generalization, (Zhang *et al.*, 2017) indicating that overfitting is a fundamental learning challenge that transcends model and dataset size. (Zhang *et al.*, 2017; Belkin *et al.*, 2019; Srivastava *et al.*, 2014)

From the Report:

2. The proposed sufficient training, although motivated by physical systems, is essentially an approach that tradeoff training performances with test performances. A baseline comparison to other simpler approaches may be needed to establish the necessity of your less simple physics formulations. For example, how does some simple noise injection schemes to parameters or combinations of early stopped models work compared to your formulation?

the Referee presents a valuable opportunity for comparison with other overfitting avoidance methods.

We will start by addressing the comparison with noise injection. Generically, one cannot know a priori what noise process has affected the input and target data in the training set, unless additional information about the ground truth is available. Thus, the performance of noise injection is highly domain- and task-specific. We recognize that noise injection can nonetheless approximate the noise distribution of a dataset effectively. If the target data of a training dataset has additive Gaussian noise, and the neural network loss function is chosen to be mean-squared error, then the prediction uncertainty distribution of sufficient training is a generalization of that of Bayesian inference with noise injection, as Bayesian inference is a special case of maximum entropy distribution sampling (at $T = 1$). (Neal, 1993) We have expanded our description in Main Methods to highlight the relationship between noise injection and sufficient training but note that this correspondence is specific to a particular noise distribution and loss function choice and is not generically present. The Main Methods now reads:

... In the particular case where the training data targets have additive Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_t^2)$, and we choose mean-squared error (MSE) as the loss function, the prediction uncertainty distribution is structurally

similar to that of Bayesian inference with noise injection, but generalized to any $\beta = 1/T$. However, this equivalence between noise injection and sufficient training is a special case; in general sufficient training does not assume a form for the noise distribution. Given this training data target noise distribution and loss function choice, $p(t'|z', \theta)$ becomes

$$p(t'|z', \theta) = \left(\frac{1}{2\pi\sigma_t^2} \right)^{\frac{1}{2}} e^{-\frac{1}{2\sigma_t^2}(y(z';\theta)-t')^2}. \quad (21)$$

Then, we can expand $y(z'; \theta)$ around the free energy-minimizing $\theta = 0$ as

$$y(z'; \theta) \approx y(z'; 0) + \left. \frac{\partial y(z'; \theta)}{\partial \theta_\alpha} \right|_{\theta=0} \theta_\alpha. \quad (22)$$

Using Equation (21) and the Taylor-expansion of $y(z'; \theta)$ in Equation (22), we can evaluate Equation (20) to be

$$p(t'|z', \mathcal{D}) \propto e^{-\frac{(t' - y(z'; 0))^2}{2\sigma^2}} \quad (23)$$

where $\sigma^2 = \sigma_t^2 + g_{\alpha\gamma}(\beta, \mathcal{D})^{-1} \left|_{\theta=0} \frac{\partial y(z'; \theta)}{\partial \theta_\alpha} \right|_{\theta=0} \frac{\partial y(z'; \theta)}{\partial \theta_\gamma} \left|_{\theta=0} \right|$. In noise injection (Matsuoka, May-June/1992; Wright, 1999), β is not a parameter, but is implicitly set to 1.(Neal, 1993) However, in sufficient training, the variance equation generalizes to any positive non-zero inverse temperature β . The variance σ^2 of the prediction uncertainty distribution has two terms: the variance introduced by uncertainty in data target measurements, σ_t^2 , and a variance-like quantity defined by the sensitivity of the free energy and the model predictions to changes in θ . The overall effect of temperature on the second variance term is to reduce the FIM, $g_{\alpha\gamma}$, thereby increasing the second variance term contribution, but the exact way in which changing $g_{\alpha\gamma}(\beta, \mathcal{D})$ affects $\partial_\theta y$ is problem-dependent.

We hope that this change clarifies how sufficient training relates to noise injection in certain training problems.

Next, we address the suggestion to compare sufficient training to a collection of early stopped models. We thank the Referee for this suggestion, and have added a comparison between ab initio simmering and both a singular early stopped model and a combination of independently trained early stopped models, as well as with dropout, trained on the CIFAR-10 dataset and the Portuguese-English TED talk transcript translation dataset.(Ye *et al.*, 2018) The revised manuscript, in the Results section, now reads:

Method	CIFAR-10 Test Accuracy (%)	PT-EN Translation Test Accuracy (%)
Early stopping (single)	71.90	39.62 (67 epochs)
Early stopping (ensemble)	73.81	41.42 (53.1 epochs–average)
Dropout	75.53	39.86 (60 epochs)
Ab initio simmering	82.05	43.46 (21 epochs)

Table 1: **Simmering outperforms early stopping, ensembled early stopping and dropout on the CIFAR-10 (Krizhevsky & Hinton, 2009) and Portuguese-English (PT-EN) TED talk transcript translation (Ye *et al.*, 2018) datasets.** For training on the CIFAR-10 dataset, all four learning cases employed the ConvNet architecture (Krizhevsky, Sutskever, *et al.*, 2012), the standard train/test split. For the dropout case, we applied the dropout rates reported in Ref. (Srivastava *et al.*, 2014). We used TensorFlow’s implementations of early stopping and dropout were, paired with stochastic gradient descent (Martín Abadi *et al.*, 2015) to optimize the early stopped and dropout-equipped models. The early stopping ensemble consists of 100 independently trained early stopped models, where the average training time was 14.56 epochs. We chose the single early stopped model by its proximity to the average accuracy of the ensemble members (71.60%). Dropout and ab initio simmering each trained for 20 epochs, and we included the models corresponding to the last 2000 weight updates in the simmering ensemble. At the testing stage, since it has no explicit ensemble to aggregate, we took dropout’s “inference mode” prediction to be its ensemble prediction. (Hara *et al.*, 2016) We used majority voting to aggregate the early stopping and simmering ensembles. For learning the translation task, we trained a reduced version (described in Supplementary Methods) of the Transformer architecture presented in Ref. (Vaswani *et al.*, 2017) with a pre-trained BERT tokenizer from TensorFlow’s model database, and assessed translation accuracy via teacher-forced token prediction accuracy. We used the same batch size and fixed learning rate for all cases, and used the Adam optimizer (Kingma & Ba, 2017) for all the non-simmering training scenarios. The early stopped ensemble consists of 10 independently trained models, with an average training time of 53.1 epochs, aggregated with majority voting as for CIFAR-10. The single early stopped model is presented as a characteristic example of the ensemble members. Based on the average early stopped training time, we trained a model with dropout for 60 epochs and determined its “ensemble test accuracy” in the same way as for CIFAR-10. We found that the simmering ensemble exceeded the test accuracy of all other cases in only 21 training epochs, with a majority-voted ensemble prediction derived from its last 200 sampled models. Thus, for equivalent training time, ab initio simmering produces more accurate predictions than both single-model and ensembled overfitting mitigation techniques for the CIFAR-10 dataset. However, simmering can also both accelerate and exceed the accuracy of other overfitting techniques in a natural language processing task like language translation.

Employing a minimally biased model of training data error allows ab initio simmering to outperform other commonly used overfitting avoidance methods in two different learning tasks: image classification with a convolutional neural network and language translation with a Transformer. Table 1 shows that, for both learning tasks, initio simmering achieves a higher test data accuracy than standard implementations of early stopping and dropout. Consistent with ensemble learning approaches, ab initio simmering outperforms a singular optimized model treated with early stopping. For CIFAR-10, given an equal training time of 20 epochs, ab initio simmering also outperforms two other ensemble learning methods: ensembled early stopping, an ensemble of independently optimized neural networks, and dropout, an ensemble of simultaneously-trained thinned neural networks. We find that simmering yields a test accuracy of over 82% in 20 epochs, whereas no other method exceeded 76% test accuracy on the same basis. For the translation problem, simmering outperforms all three overfitting mitigation methods with a shorter training time. Using fixed, equivalent learning rates, simmering exceeded the accuracy of both dropout and ensembled early stopping in a small fraction of the training time (21 epochs for simmering, versus 60 epochs for dropout and an average of 53.1 epochs for ensembled early stopping). These comparisons highlight the role of simmering’s distinct minimally-biased ensemble generation in efficiently producing generalizable predictions.

Beyond generalizable prediction, simmering also yields prediction uncertainty estimates that reflect data

difficulty. Figure 3 shows how the three ensemble learning approaches predict on two different CIFAR-10 test samples (Fig. 3a–b). Though ensemble uncertainty estimates can be extracted either directly or indirectly (see Supplementary Methods) from all three ensemble learning approaches, the ensemble generated by *ab initio* simmering allows for uncertainty estimates that are proportional to image difficulty as shown in Fig. 3c–d. Thus, the maximum entropy distribution sampling that supports simmering provides both generalizable predictions and additional insight into the neural network’s learning process.

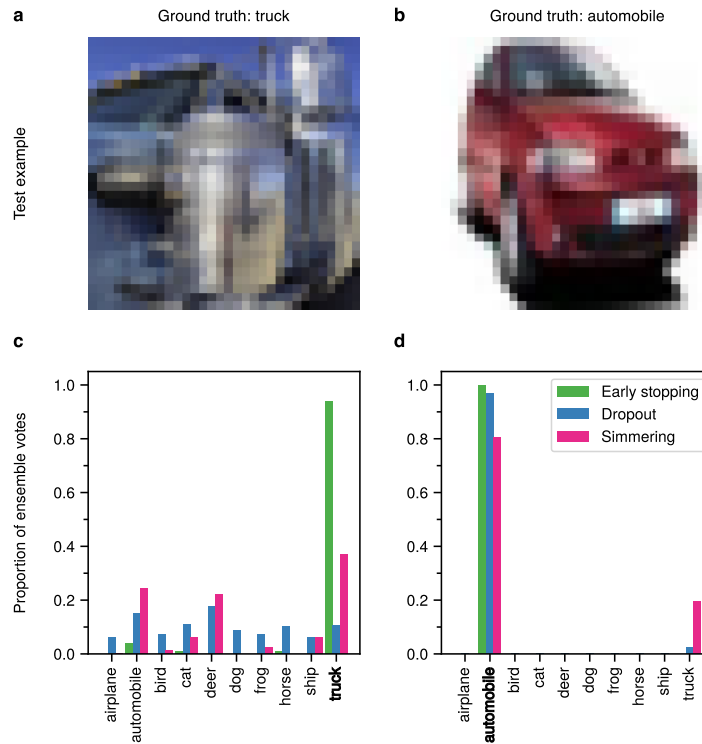


Figure 3: *Ab initio* simmering correctly classifies CIFAR-10 test images correctly while also yielding confidence estimates that correspond to image difficulty. Early stopping (green bars), dropout (blue bars) and *ab initio* simmering (pink bars) ensembles predict an “easy” image of an automobile (panel b), with a clear foreground and with the automobile in full view, with similarly high confidence (high proportion of ensemble votes). However, when tasked with classifying a “difficult” test image of a truck (panel a), which presents the truck at close range and with low contrast with the background, early stopping remains nearly equally confident (panel c, green bar). In contrast, simmering predicts the correct class, but with lower confidence, reflecting the difficulty of identifying the truck in the image (see Supplementary Methods for image difficulty quantification). Simmering’s second-most voted prediction bears relation to the correct class (“automobile” and “truck” have similar image features). Though dropout also predicts uncertainly, it predicts the correct label with near-equivalent confidence to most other classes. Thus, *ab initio* simmering predicts images of varying difficulty accurately with confidence proportional to image difficulty.

We thank the Referee for this suggestion and hope that our revision clarifies the difference in performance between the mentioned overfitting avoidance schemes and sufficient training.

From the Report:

To make significant contribution in machine learning, it may benefit from closely investigating the problem and identifying the key cause of overfitting instead of only proposing a large physics-based formulation and claim that it helps mitigating overfitting.

This may require simplifying your system and identifying minimal components that make it work, which will provide more insight to machine learning researchers who do not share a physics background/interest.

We thank the Referee for the opportunity to clarify the presentation of sufficient training in our manuscript.

First, we address the question raised by the Referee about how our approach mitigates overfitting.

In the introduction of the manuscript, we highlight that one of the main advantages of neural networks is their universal estimation capability, which is supported by their high-dimensional (over-parameterized) parameter spaces. In problems with noisy data, a universal estimator like a neural network can produce either a generalizable representation, that has higher training error but predicts unseen data more accurately, and an overspecified one, which has lower training error but poor test performance. An optimization algorithm will always seek the lower-error representation of the data, and will always be overspecified if sufficiently effective at traversing the loss landscape. Other works referenced in the introduction ((Livni *et al.*, 2014; Ruder, 2017; Ghojogh & Crowley, 2023)) also identify overfitting as a symptom of treating over-parameterized neural networks with optimization. Thus, as described in the Results section “Sufficient training by simmering”, to avoid overfitting entirely, we can seek non-optimal neural network parameters from the outset.

The mechanism of overfitting avoidance can be shown in two ways: via the training dynamics and via loss landscape curvature analysis. We have expanded both descriptions in Methods. We highlight the key features of sufficient training that support overfitting avoidance in these extended descriptions.

The extended description of the training dynamics in Main Methods now reads:

The particles’ equations of motion are derived from the NHC Hamiltonian.(Martyna *et al.*, 1996; Tuckerman *et al.*, 1992) For simplicity of notation, we will henceforth set $m_i = 1, Q_k = 1 \forall i, k$ and describe the equations of motion in terms of the positions and velocities, rather than the positions and momenta. The real particles’ positions evolve according to(Jang & Voth, 1997)

$$\dot{x}_i(t) = v_i(t) \quad (24)$$

$$\dot{v}_i(t) = a_i(t) - v_{s_1}(t)v_i(t), \quad (25)$$

where the i subscript denotes real particle quantities, and the s_k subscript denotes a quantity affiliated with the k^{th} virtual particle. The $v_{s_1}(t)v_i(t)$ term from the NHC in Equation (25) distinguishes simmering from gradient-based optimization. For example, in momentum gradient descent, Equation (25) would have $v_{s_1}(t) = \text{const.}$ (França *et al.*, 2020) In contrast, $v_{s_1}(t)$ in NHC dynamics can take on both positive and negative values over time.

The virtual particle dynamics are given by

$$\dot{s}_k(t) = v_{s_k}(t) \quad (26)$$

$$\dot{v}_{s_1}(t) = a_{s_1}(t) - v_{s_2}(t)v_{s_1}(t) \quad (27)$$

$$\dot{v}_{s_k}(t) = a_{s_k}(t) - v_{s_{k+1}}(t)v_{s_k}(t), \quad (28)$$

with the accelerations for all particles $a_i(t)$, $a_{s_1}(t)$, $a_{s_k}(t)$ defined as

$$a_i(t) = -\frac{1}{m_i} \nabla_{x_i} L(x_1(t), \dots, x_N(t)) \quad (29)$$

$$a_{s_1}(t) = \frac{1}{Q_1} \left(\sum_i m_i v_i^2(t) - NT_{target} \right) \quad (30)$$

$$a_{s_k}(t) = \frac{1}{Q_k} \left(Q_{k-1} v_{k-1}^2(t) - T_{target} \right). \quad (31)$$

The key takeaway of this implementation of sufficient training is the introduction of a dynamical coefficient of $v_i(t)$, $v_{s_1}(t)$, that varies over time. This way, the accelerations of the neural network weights and biases are never entirely determined by the gradient of the loss and are instead continuously driven to non-optimal loss by the dynamics in $\dot{v}_{s_1}$, which depend on the temperature T .

Another interpretation of simmering’s overfitting avoidance is through loss landscape curvature analysis analogous to mathematical arguments that support L_2 -norm regularization. We have extended the existing comparison in the Main Methods, which now reads,

The FIM describes the relative “proximity” of θ to the free energy minimum $\theta = 0$ in parameter space as a function of $\beta = 1/T$. The squared displacement ds^2 between θ and $\theta = 0$ is

$$ds^2 \propto \frac{1}{T} \sum_j \lambda_j d\lambda_j^2, \quad (16)$$

where λ_j are the eigenvalues of the Hessian of $F(\theta, \mathcal{D})$ that describe the curvature of F along different modes in parameter space. Distances (Equation (16)) in a finite-temperature system diverge for small T , except along directions corresponding to “sloppy modes” for which the corresponding eigenvalues $\lambda_j \rightarrow 0$. Simmering and other sufficient training implementations exploit this effect of temperature on distances in parameter space to systematically sample families of loss-equivalent networks located along these sloppy modes. ...

... This information geometric framing presented here also motivates the effectiveness of both ab initio simmering and retrofitting at reducing overfitting in a neural network. The effect of T on the Hessian of $F(\theta, \mathcal{D})$ (shown in Equation (16)) is analogous to the effect of the regularization strength parameter, typically denoted λ , on the eigenvalues of the Hessian of the training loss in L_2 -norm regularization. (Hastie *et al.*, 2009) Using finite-temperature parameter sampling, both ab initio simmering and retrofitting rescale distances in parameter space in the same manner as L_2 -norm regularization, which has already been shown to reduce overfitting. (Hastie *et al.*, 2009)

We hope that the dynamical angle and comparison with the mathematical arguments supporting another well-known overfitting mitigation technique clarify the key components of sufficient training that distinguish it from optimization and thus generically avoid overfitting.

From the Report:

** Does the work support the conclusions and claims, or is additional evidence needed?

As discussed above, I believe the comparison with similar techniques or simpler alternative is lacking. It is not yet demonstrated that this physics-motivated formulation is really necessary. Not enough simpler alternatives are discussed and compared.

We have revised the manuscript to include a comparison with a singular optimized model, as well as two ensemble approaches, on the CIFAR-10 dataset and the Portuguese-English TED talk transcript translation dataset.(Ye *et al.*, 2018). We appreciate the Referee’s prompt for comparison, and agree that this comparison with other overfitting mitigation methods better motivates the necessity of sufficient training approaches. The revised Results now read:

Method	CIFAR-10 Test Accuracy (%)	PT-EN Translation Test Accuracy (%)
Early stopping (single)	71.90	39.62 (67 epochs)
Early stopping (ensemble)	73.81	41.42 (53.1 epochs–average)
Dropout	75.53	39.86 (60 epochs)
Ab initio simmering	82.05	43.46 (21 epochs)

Table 1: Simmering outperforms early stopping, ensembled early stopping and dropout on the CIFAR-10 (Krizhevsky & Hinton, 2009) and Portuguese-English (PT-EN) TED talk transcript translation(Ye *et al.*, 2018) datasets. For training on the CIFAR-10 dataset, all four learning cases employed the ConvNet architecture(Krizhevsky, Sutskever, *et al.*, 2012), the standard train/test split. For the dropout case, we applied the dropout rates reported in Ref. (Srivastava *et al.*, 2014). We used TensorFlow’s implementations of early stopping and dropout were, paired with stochastic gradient descent(Martín Abadi *et al.*, 2015) to optimize the early stopped and dropout-equipped models. The early stopping ensemble consists of 100 independently trained early stopped models, where the average training time was 14.56 epochs. We chose the single early stopped model by its proximity to the average accuracy of the ensemble members (71.60%). Dropout and ab initio simmering each trained for 20 epochs, and we included the models corresponding to the last 2000 weight updates in the simmering ensemble. At the testing stage, since it has no explicit ensemble to aggregate, we took dropout’s “inference mode” prediction to be its ensemble prediction. (Hara *et al.*, 2016) We used majority voting to aggregate the early stopping and simmering ensembles. For learning the translation task, we trained a reduced version (described in Supplementary Methods) of the Transformer architecture presented in Ref. (Vaswani *et al.*, 2017) with a pre-trained BERT tokenizer from TensorFlow’s model database, and assessed translation accuracy via teacher-forced token prediction accuracy. We used the same batch size and fixed learning rate for all cases, and used the Adam optimizer(Kingma & Ba, 2017) for all the non-simmering training scenarios. The early stopped ensemble consists of 10 independently trained models, with an average training time of 53.1 epochs, aggregated with majority voting as for CIFAR-10. The single early stopped model is presented as a characteristic example of the ensemble members. Based on the average early stopped training time, we trained a model with dropout for 60 epochs and determined its “ensemble test accuracy” in the same way as for CIFAR-10. We found that the simmering ensemble exceeded the test accuracy of all other cases in only 21 training epochs, with a majority-voted ensemble prediction derived from its last 200 sampled models. Thus, for equivalent training time, ab initio simmering produces more accurate predictions than both single-model and ensembled overfitting mitigation techniques for the CIFAR-10 dataset. However, simmering can also both accelerate and exceed the accuracy of other overfitting techniques in a natural language processing task like language translation.

Employing a minimally biased model of training data error allows ab initio simmering to outperform other commonly used overfitting avoidance methods in two different learning tasks: image classification with a convolutional neural network and language translation with a Transformer. Table 1 shows that, for both learning tasks, initio simmering achieves a higher test data accuracy than standard implementations of early stopping and dropout. Consistent with ensemble learning approaches, ab initio simmering outperforms a singular optimized model treated with early stopping. For CIFAR-10, given an equal training time of 20 epochs, ab initio simmering also outperforms two other ensemble learning methods: ensembled early stopping, an ensemble of independently optimized neural networks, and dropout, an ensemble of simultaneously-trained thinned neural networks. We find that simmering yields a test accuracy of over 82% in 20 epochs, whereas no other method exceeded 76% test accuracy on the same basis. For the translation problem, simmering outperforms all three overfitting mitigation methods with a shorter training time. Using fixed, equivalent learning

rates, simmering exceeded the accuracy of both dropout and ensembled early stopping in a small fraction of the training time (21 epochs for simmering, versus 60 epochs for dropout and an average of 53.1 epochs for ensembled early stopping). These comparisons highlight the role of simmering’s distinct minimally-biased ensemble generation in efficiently producing generalizable predictions.

Beyond generalizable prediction, simmering also yields prediction uncertainty estimates that reflect data difficulty. Figure 3 shows how the three ensemble learning approaches predict on two different CIFAR-10 test samples (Fig. 3a–b). Though ensemble uncertainty estimates can be extracted either directly or indirectly (see Supplementary Methods) from all three ensemble learning approaches, the ensemble generated by ab initio simmering allows for uncertainty estimates that are proportional to image difficulty as shown in Fig. 3c–d. Thus, the maximum entropy distribution sampling that supports simmering provides both generalizable predictions and additional insight into the neural network’s learning process.

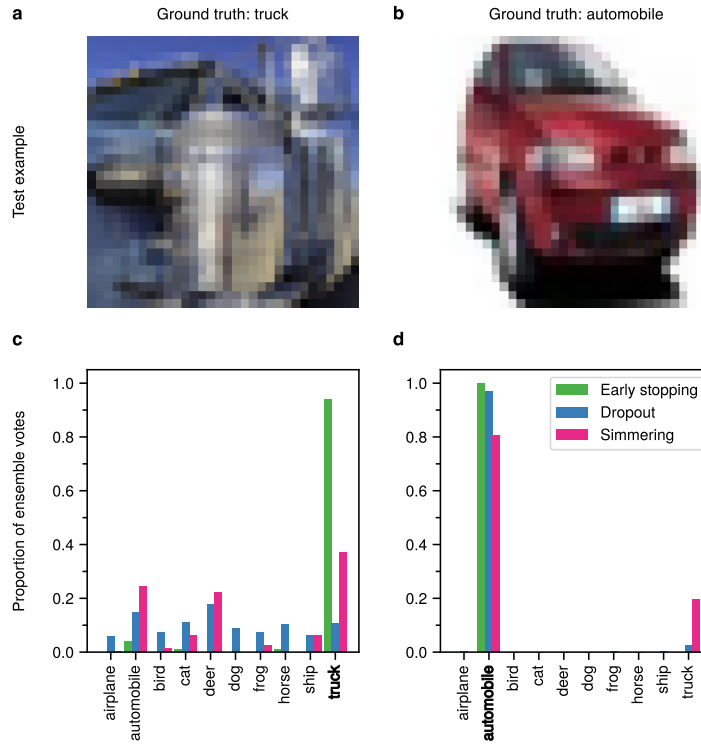


Figure 3: Ab initio simmering correctly classifies CIFAR-10 test images correctly while also yielding confidence estimates that correspond to image difficulty. Early stopping (green bars), dropout (blue bars) and ab initio simmering (pink bars) ensembles predict an “easy” image of an automobile (panel b), with a clear foreground and with the automobile in full view, with similarly high confidence (high proportion of ensemble votes). However, when tasked with classifying a “difficult” test image of a truck (panel a), which presents the truck at close range and with low contrast with the background, early stopping remains nearly equally confident (panel c, green bar). In contrast, simmering predicts the correct class, but with lower confidence, reflecting the difficulty of identifying the truck in the image (see Supplementary Methods for image difficulty quantification). Simmering’s second-most voted prediction bears relation to the correct class (“automobile” and “truck” have similar image features). Though dropout also predicts uncertainly, it predicts the correct label with near-equivalent confidence to most other classes. Thus, ab initio simmering predicts images of varying difficulty accurately with confidence proportional to image difficulty.

Table 1 shows that simmering achieves the highest test accuracy on CIFAR-10 given a common training duration, and achieves the highest test accuracy in a fraction of the training time, given equal and fixed learning rates, for the language translation task. Additionally, Fig. 3 shows that, beyond making generalizable predictions, the ensemble prediction distribution reflects qualitative similarities between classes and the image difficulty. We hope that this revised comparison highlights that sufficient training is both accurate and informative in comparison to what the Referee describes as simpler approaches.

The Referee raises an interesting question regarding the relationship between our approach and optimization. This issue is raised in three locations in the review. We believe it is most straightforward to discuss these remarks together as a group.

The Referee states:

Some claims in the main text and abstract are a bit exaggerated. The author seems to suggest that "sufficient training" is different from optimizing. A more reasonable view is that "sufficient training" is just another way of optimizing, since your approach is essentially a different type of optimizer, which rely on gradients computed by automatic differentiation via backpropagation.

Further, the Referee remarks:

The way particles are pushed by forces of gradients in your system is essentially optimizing.

Finally the Referee notes:

The tone of "creating a new paradigm different from optimization" needs to be softened to avoid misleading if you still rely on backpropagation/automatic differentiation.

Through these remarks, the Referee raises an important distinction. We appreciate the opportunity to clarify.

As Goodfellow, Bengio and Courville state in Chapter 6 of their textbook *Deep Learning*: “The term back-propagation is often misunderstood as meaning the whole learning algorithm for multi layer neural networks. Actually, back-propagation refers only to the method for computing the gradient, while another algorithm, such as stochastic gradient descent, is used to perform learning using this gradient.” The referee is correct that we use backpropagation/automatic differentiation to compute gradients but instead of using them for optimization, we use gradients for filtration.

In our filter-based training approach, the fact that backpropagation/automatic differentiation for neural networks has already been implemented in TensorFlow makes it particularly convenient for computing the forces we use to implement the filter.

One way to identify the distinction between our approach and optimization is by referring to the integral transform we present in main text Equation (1):

$$Z(\beta, \mathcal{D}) = \int d^N x e^{-\beta L(x, \mathcal{D})} \quad (1)$$

Mathematically, Z plays the role of a generating function that acts as a low-pass filter for loss (L) rather than as an optimizer. From this generating function, one could inspect the effective forces on model parameters, e.g., in terms of collective coordinates θ , which have the form

$$\nabla L(\theta, \mathcal{D}) - \frac{1}{\beta} \nabla S(\theta, \mathcal{D}),$$

where the second term is an effective entropic force. Simulating the effect of these effective forces on model parameters allows sufficient training to sample the sets of model parameters that contribute to Z .

These entropic forces cause sufficient training to exhibit distinct training dynamics from optimization. To help clarify this distinction, we have revised the corresponding Methods section, which now reads

The particles' equations of motion are derived from the NHC Hamiltonian.(Martyna *et al.*, 1996; Tuckerman *et al.*, 1992) For simplicity of notation, we will henceforth set $m_i = 1, Q_k = 1 \forall i, k$ and describe the equations of motion in terms of the positions and velocities, rather than the positions and momenta. The real particles' positions evolve according to (Jang & Voth, 1997)

$$\dot{x}_i(t) = v_i(t) \quad (24)$$

$$\dot{v}_i(t) = a_i(t) - v_{s_1}(t)v_i(t), \quad (25)$$

where the i subscript denotes real particle quantities, and the s_k subscript denotes a quantity affiliated with the k^{th} virtual particle. The $v_{s_1}(t)v_i(t)$ term from the NHC in Equation (25) distinguishes simmering from gradient-based optimization. For example, in momentum gradient descent, Equation (25) would have $v_{s_1}(t) = \text{const.}$ (França *et al.*, 2020) In contrast, $v_{s_1}(t)$ in NHC dynamics can take on both positive and negative values over time.

The virtual particle dynamics are given by

$$\dot{s}_k(t) = v_{s_k}(t) \quad (26)$$

$$\dot{v}_{s_1}(t) = a_{s_1}(t) - v_{s_2}(t)v_{s_1}(t) \quad (27)$$

$$\dot{v}_{s_k}(t) = a_{s_k}(t) - v_{s_{k+1}}(t)v_{s_k}(t), \quad (28)$$

with the accelerations for all particles $a_i(t), a_{s_1}(t), a_{s_k}(t)$ defined as

$$a_i(t) = -\frac{1}{m_i} \nabla_{x_i} L(x_1(t), \dots, x_N(t)) \quad (29)$$

$$a_{s_1}(t) = \frac{1}{Q_1} \left(\sum_i m_i v_i^2(t) - NT_{target} \right) \quad (30)$$

$$a_{s_k}(t) = \frac{1}{Q_k} (Q_{k-1} v_{k-1}^2(t) - T_{target}). \quad (31)$$

The acceleration of the real particles in Equation (29) is proportional to the negative gradient of the loss with respect to the weights and biases in the neural network.

Note that the factor of v_{s_1} in the expression for $\dot{v}_i(t)$ marks a crucial distinction from optimization. In gradient-based optimization, e.g., momentum gradient descent, the v_{s_1} factor is a positive constant. Over training time, such a constant has a diminishing effect on the dynamics of the optimizer. However, for the

NHC thermostat, this $v_{s1}(t)$ factor is a dynamic quantity that can be both positive and negative. The presence of this factor prevents our approach from optimizing.

It is also possible to illustrate the distinction between optimization and our approach (Equations (24)–(31)) by examining training dynamics. We give an example of the dynamics of our algorithm in the loss landscape in a new Supplementary Fig. 1. The revised Supplementary Information now reads:

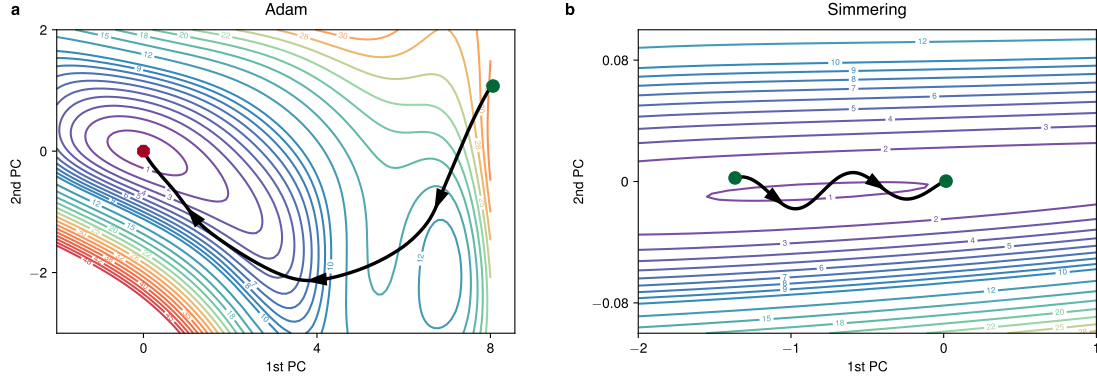


Figure 1: Simmering (panel b), an implementation of sufficient training, produces a distinct training trajectory (black curve, panel b) from the Adam optimizer (panel a). The Adam optimizer (black curve, panel a) traverses the training loss landscape (coloured contours in panel a) perpendicular to the loss contours, as expected in gradient-based optimization. In contrast, simmering travels along equivalent-loss (coloured contours in panel b) directions in parameter space, aligning with the information-geometric interpretation of sufficient training presented in Main Methods. The Adam trajectory (panel a) corresponds to the Adam portion of Main Fig. 1a, and the simmering trajectory is a 100-epoch slice of the simmering sampling shown in Fig. 1a. The loss landscape and training trajectories have been projected onto the principal components (PCs) of the Adam and simmering training trajectories in panels a and b respectively.

Supplementary Figure 1, generated using the loss visualization method of Ref. (Li *et al.*, 2018), shows the difference in training dynamics between Adam, (Kingma & Ba, 2017) a gradient-based optimizer, and simmering, an implementation of sufficient training. The Adam optimizer travels perpendicular to the loss contours, whereas simmering travels parallel to the direction of near-equivalent loss. The first principal component (PC) of Adam describes 90.44% of its trajectory variance in parameter space, and the second PC describes another 6.12%. For simmering, the first and second PCs describe 99.9428% and 0.0324% of the parameter space trajectory respectively. In the latter case, a larger proportion of the training trajectory is described by the first PC. Since this PC points along the direction of equivalent loss in parameter space, Supplementary Fig. 1b shows that sufficient training automatically identifies and samples along sloppy (equivalent-loss) modes in model parameter space as described in Main Methods.

This visualization shows that our approach traverses the loss landscape along the direction of the principal component, while undergoing transverse oscillation. These transverse oscillations are a clear signal that distinguishes the behaviour of our approach from an optimization algorithm that is constructed to migrate to a training loss minimum and halt.

One could recover a form of optimization from our approach by moving to the $\beta \rightarrow \infty$ limit of Equation (1). We do not do that. Instead, we specifically maintain finite $\beta = 1/T$ during sampling. There is also a sense in which one could construct a distribution of parameters $\rho(x)$ that optimally represents the action of the filter at finite β . However, it is important to note that that approach represents constructing a sufficient

representation of a non-optimal quantity (in this context, non-minimal training loss), which differs strongly in spirit from optimizing (which seeks the minimal training loss representation). Even so, instead of doing that, we simply sample from the filter in Equation (1), employing gradients as a convenient way to generate a sample of a set of parameters that contributes to Z . Although our implementation of sufficient training employs gradients, other implementations could use other methods to sample from the filter, e.g., by using random walks.

In addition, we further discuss the motivation for the differences between our approach and optimization in the revised Discussion.

** Are there any flaws in the data analysis, interpretation and conclusions? Do these prohibit publication or require revision?

One thing I notice is that the results of your approach are all obtained with ensemble models. It is expected that ensemble models can benefit generalization, it may be fairer to compare with baselines in a non-ensemble fashion if possible. Or you can establish an ensemble baseline using popular ensemble techniques.

This ensemble vs. single model comparison issue need to be addressed in order to support the claim that your approach is better than baseline.

We thank the Referee for this suggestion. It is true that ensembled prediction benefits generalization, and we agree that both a non-ensemble and ensemble baseline can clarify the generalizability advantage of sufficient training as an ensemble generation method. The revised manuscript provides the following comparison to a singular, early stopped model and two ensemble approaches for two learning problems (CIFAR-10 image classification with a convolutional neural network and language translation with a Transformer):

Method	CIFAR-10 Test Accuracy (%)	PT-EN Translation Test Accuracy (%)
Early stopping (single)	71.90	39.62 (67 epochs)
Early stopping (ensemble)	73.81	41.42 (53.1 epochs–average)
Dropout	75.53	39.86 (60 epochs)
Ab initio simmering	82.05	43.46 (21 epochs)

Table 1: **Simmering outperforms early stopping, ensembled early stopping and dropout on the CIFAR-10 (Krizhevsky & Hinton, 2009) and Portuguese-English (PT-EN) TED talk transcript translation (Ye *et al.*, 2018) datasets.** For training on the CIFAR-10 dataset, all four learning cases employed the ConvNet architecture (Krizhevsky, Sutskever, *et al.*, 2012), the standard train/test split. For the dropout case, we applied the dropout rates reported in Ref. (Srivastava *et al.*, 2014). We used TensorFlow’s implementations of early stopping and dropout were, paired with stochastic gradient descent (Martín Abadi *et al.*, 2015) to optimize the early stopped and dropout-equipped models. The early stopping ensemble consists of 100 independently trained early stopped models, where the average training time was 14.56 epochs. We chose the single early stopped model by its proximity to the average accuracy of the ensemble members (71.60%). Dropout and ab initio simmering each trained for 20 epochs, and we included the models corresponding to the last 2000 weight updates in the simmering ensemble. At the testing stage, since it has no explicit ensemble to aggregate, we took dropout’s “inference mode” prediction to be its ensemble prediction. (Hara *et al.*, 2016) We used majority voting to aggregate the early stopping and simmering ensembles. For learning the translation task, we trained a reduced version (described in Supplementary Methods) of the Transformer architecture presented in Ref. (Vaswani *et al.*, 2017) with a pre-trained BERT tokenizer from TensorFlow’s model database, and assessed translation accuracy via teacher-forced token prediction accuracy. We used the same batch size and fixed learning rate for all cases, and used the Adam optimizer (Kingma & Ba, 2017) for all the non-simmering training scenarios. The early stopped ensemble consists of 10 independently trained models, with an average training time of 53.1 epochs, aggregated with majority voting as for CIFAR-10. The single early stopped model is presented as a characteristic example of the ensemble members. Based on the average early stopped training time, we trained a model with dropout for 60 epochs and determined its “ensemble test accuracy” in the same way as for CIFAR-10. We found that the simmering ensemble exceeded the test accuracy of all other cases in only 21 training epochs, with a majority-voted ensemble prediction derived from its last 200 sampled models. Thus, for equivalent training time, ab initio simmering produces more accurate predictions than both single-model and ensembled overfitting mitigation techniques for the CIFAR-10 dataset. However, simmering can also both accelerate and exceed the accuracy of other overfitting techniques in a natural language processing task like language translation.

Employing a minimally biased model of training data error allows ab initio simmering to outperform other commonly used overfitting avoidance methods in two different learning tasks: image classification with a convolutional neural network and language translation with a Transformer. Table 1 shows that, for both learning tasks, initio simmering achieves a higher test data accuracy than standard implementations of early stopping and dropout. Consistent with ensemble learning approaches, ab initio simmering outperforms a singular optimized model treated with early stopping. For CIFAR-10, given an equal training time of 20 epochs, ab initio simmering also outperforms two other ensemble learning methods: ensembled early stopping, an ensemble of independently optimized neural networks, and dropout, an ensemble of simultaneously-trained thinned neural networks. We find that simmering yields a test accuracy of over 82% in 20 epochs, whereas no other method exceeded 76% test accuracy on the same basis. For the translation problem, simmering outperforms all three overfitting mitigation methods with a shorter training time. Using fixed, equivalent learning rates, simmering exceeded the accuracy of both dropout and ensembled early stopping in a small fraction of the training time (21 epochs for simmering, versus 60 epochs for dropout and an average of 53.1 epochs for ensembled early stopping). These comparisons highlight the role of simmering’s distinct minimally-biased ensemble generation in efficiently producing generalizable predictions.

Beyond generalizable prediction, simmering also yields prediction uncertainty estimates that reflect data

difficulty. Figure 3 shows how the three ensemble learning approaches predict on two different CIFAR-10 test samples (Fig. 3a–b). Though ensemble uncertainty estimates can be extracted either directly or indirectly (see Supplementary Methods) from all three ensemble learning approaches, the ensemble generated by *ab initio* simmering allows for uncertainty estimates that are proportional to image difficulty as shown in Fig. 3c–d. Thus, the maximum entropy distribution sampling that supports simmering provides both generalizable predictions and additional insight into the neural network’s learning process.

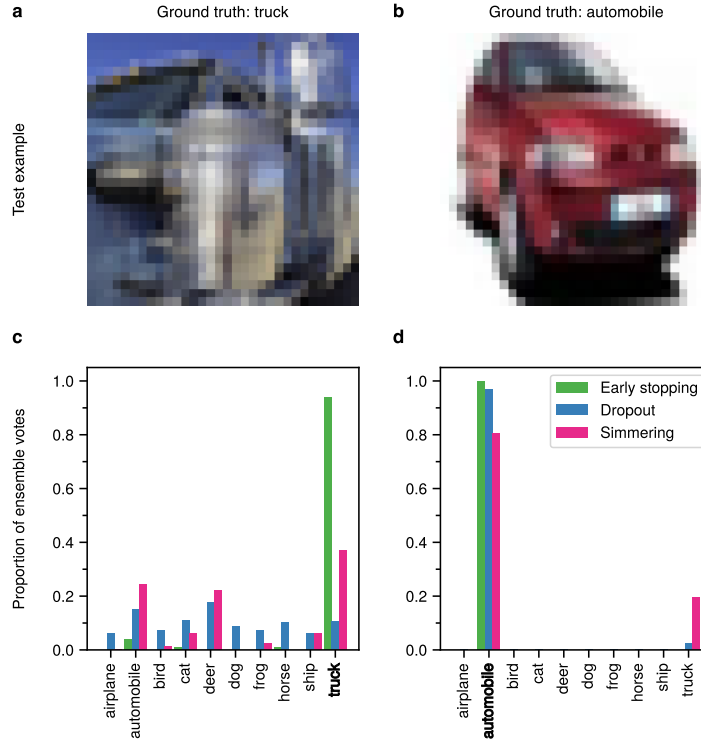


Figure 3: *Ab initio* simmering correctly classifies CIFAR-10 test images correctly while also yielding confidence estimates that correspond to image difficulty. Early stopping (green bars), dropout (blue bars) and *ab initio* simmering (pink bars) ensembles predict an “easy” image of an automobile (panel b), with a clear foreground and with the automobile in full view, with similarly high confidence (high proportion of ensemble votes). However, when tasked with classifying a “difficult” test image of a truck (panel a), which presents the truck at close range and with low contrast with the background, early stopping remains nearly equally confident (panel c, green bar). In contrast, simmering predicts the correct class, but with lower confidence, reflecting the difficulty of identifying the truck in the image (see Supplementary Methods for image difficulty quantification). Simmering’s second-most voted prediction bears relation to the correct class (“automobile” and “truck” have similar image features). Though dropout also predicts uncertainly, it predicts the correct label with near-equivalent confidence to most other classes. Thus, *ab initio* simmering predicts images of varying difficulty accurately with confidence proportional to image difficulty.

Table 1 shows that simmering outperforms both a singular, optimized model and two other ensemble prediction approaches. This comparison motivates the use of minimally-biased ensemble generation in sufficient training over both single-model prediction and other implicit and explicit ensemble collection methods.

** Is the methodology sound? Does the work meet the expected standards in your field

Some general comments about the direction/approach are given in the above section about significance.

Regarding details and execution, I feel that the datasets used are on a simple/toy or uncommon side. MNIST, sine fitting, IRIS, etc. are all relatively toy datasets in machine learning nowadays. CIFAR10 and CIFAR100 have the same format as MNIST and are less "toy" — it is pretty standard in the field to try on them. Optionally, if you can obtain some results on advanced tasks like modern NLP and ImageNet, it would make your work much more solid.

The referee raises an interesting point about the set of datasets chosen to illustrate the approach. We agree that testing the method on one of the CIFAR datasets is useful for contextualizing the performance of the method against a standard benchmark, and have thus revised our manuscript to include the performance of ab initio simmering on the CIFAR-10 dataset. We have also included an example of ab initio simmering performance on a natural language processing task with a Transformer. The revised Results section now reads:

Method	CIFAR-10 Test Accuracy (%)	PT-EN Translation Test Accuracy (%)
Early stopping (single)	71.90	39.62 (67 epochs)
Early stopping (ensemble)	73.81	41.42 (53.1 epochs-average)
Dropout	75.53	39.86 (60 epochs)
Ab initio simmering	82.05	43.46 (21 epochs)

Table 1: Simmering outperforms early stopping, ensembled early stopping and dropout on the CIFAR-10 (Krizhevsky & Hinton, 2009) and Portuguese-English (PT-EN) TED talk transcript translation(Ye *et al.*, 2018) datasets. For training on the CIFAR-10 dataset, all four learning cases employed the ConvNet architecture(Krizhevsky, Sutskever, *et al.*, 2012), the standard train/test split. For the dropout case, we applied the dropout rates reported in Ref. (Srivastava *et al.*, 2014). We used TensorFlow’s implementations of early stopping and dropout were, paired with stochastic gradient descent(Martín Abadi *et al.*, 2015) to optimize the early stopped and dropout-equipped models. The early stopping ensemble consists of 100 independently trained early stopped models, where the average training time was 14.56 epochs. We chose the single early stopped model by its proximity to the average accuracy of the ensemble members (71.60%). Dropout and ab initio simmering each trained for 20 epochs, and we included the models corresponding to the last 2000 weight updates in the simmering ensemble. At the testing stage, since it has no explicit ensemble to aggregate, we took dropout’s “inference mode” prediction to be its ensemble prediction. (Hara *et al.*, 2016) We used majority voting to aggregate the early stopping and simmering ensembles. For learning the translation task, we trained a reduced version (described in Supplementary Methods) of the Transformer architecture presented in Ref. (Vaswani *et al.*, 2017) with a pre-trained BERT tokenizer from TensorFlow’s model database, and assessed translation accuracy via teacher-forced token prediction accuracy. We used the same batch size and fixed learning rate for all cases, and used the Adam optimizer(Kingma & Ba, 2017) for all the non-simmering training scenarios. The early stopped ensemble consists of 10 independently trained models, with an average training time of 53.1 epochs, aggregated with majority voting as for CIFAR-10. The single early stopped model is presented as a characteristic example of the ensemble members. Based on the average early stopped training time, we trained a model with dropout for 60 epochs and determined its “ensemble test accuracy” in the same way as for CIFAR-10. We found that the simmering ensemble exceeded the test accuracy of all other cases in only 21 training epochs, with a majority-voted ensemble prediction derived from its last 200 sampled models. Thus, for equivalent training time, ab initio simmering produces more accurate predictions than both single-model and ensembled overfitting mitigation techniques for the CIFAR-10 dataset. However, simmering can also both accelerate and exceed the accuracy of other overfitting techniques in a natural language processing task like language translation.

Employing a minimally biased model of training data error allows ab initio simmering to outperform other commonly used overfitting avoidance methods in two different learning tasks: image classification with a convolutional neural network and language translation with a Transformer. Table 1 shows that, for both learning tasks, initio simmering achieves a higher test data accuracy than standard implementations of early stopping and dropout. Consistent with ensemble learning approaches, ab initio simmering outperforms a singular optimized model treated with early stopping. For CIFAR-10, given an equal training time of 20 epochs, ab initio simmering also outperforms two other ensemble learning methods: ensembled early stopping, an ensemble of independently optimized neural networks, and dropout, an ensemble of simultaneously-trained thinned neural networks. We find that simmering yields a test accuracy of over 82% in 20 epochs, whereas no other method exceeded 76% test accuracy on the same basis. For the translation problem, simmering outperforms all three overfitting mitigation methods with a shorter training time. Using fixed, equivalent learning rates, simmering exceeded the accuracy of both dropout and ensembled early stopping in a small fraction of the training time (21 epochs for simmering, versus 60 epochs for dropout and an average of 53.1 epochs for ensembled early stopping). These comparisons highlight the role of simmering’s distinct minimally-biased ensemble generation in efficiently producing generalizable predictions.

**** Is there enough detail provided in the methods for the work to be reproduced?**

I believe there is enough detail. Also the code is provided.

referee #1 (Remarks on code availability):

The code has a README file and runs on default settings via one command line without any problem. The code consists of a demo of applying the proposed approach on learning a noisy sine curve.

We thank the Referee for taking the time to verify the reproducibility of our work by testing the code we provided. We have revised the open-source implementation to be a demonstration of ab initio simmering on the CIFAR-10 dataset, which this referee noted is a standard benchmark in machine learning.

We thank the Referee for their careful examination of the work. We believe that Referee’s suggestions have stimulated changes in the revised manuscript that substantially improve its presentation.

Report of Referee 2

In this manuscript, the authors present a novel algorithm, termed "simmering," for training neural networks. By leveraging Nose-Hoover chain thermostats, a concept originating from molecular dynamics, the proposed approach aims to address the persistent challenge of overfitting in deep learning. While the idea is intriguing and has potential, the current work lacks sufficient experimental evidence to fully support the claims made.

The experiments conducted primarily utilize synthetic data or relatively simple datasets, including MNIST, HIGGS, and IRIS, in conjunction with baseline architectures such as LeNet-5. While these initial results provide some insight into the potential of the proposed method, they are insufficient to establish its robustness and generalizability. To strengthen the conclusions, I strongly recommend that the authors extend their evaluation to more complex and realistic datasets, such as ImageNet, and incorporate benchmark neural network architectures, such as Transformer or ResNet-101.

We appreciate the Referee’s careful examination of our work. We agree that evaluating the model performance on more complex architectures and datasets helps contextualize the utility of our approach for contemporary use. We have extended our Results to include a comparison between ab initio simmering and other overfitting mitigation techniques. The two learning tasks presented are: training on the CIFAR-10 dataset with the ConvNet architecture, and performing Portuguese-English translation of TED talk transcripts (Ye *et al.*, 2018) with a Transformer. The setup of the comparisons is described in Supplementary Methods, and the Results section now reads

Method	CIFAR-10 Test Accuracy (%)	PT-EN Translation Test Accuracy (%)
Early stopping (single)	71.90	39.62 (67 epochs)
Early stopping (ensemble)	73.81	41.42 (53.1 epochs–average)
Dropout	75.53	39.86 (60 epochs)
Ab initio simmering	82.05	43.46 (21 epochs)

Table 1: Simmering outperforms early stopping, ensembled early stopping and dropout on the CIFAR-10 (Krizhevsky & Hinton, 2009) and Portuguese-English (PT-EN) TED talk transcript translation(Ye *et al.*, 2018) datasets. For training on the CIFAR-10 dataset, all four learning cases employed the ConvNet architecture(Krizhevsky, Sutskever, *et al.*, 2012), the standard train/test split. For the dropout case, we applied the dropout rates reported in Ref. (Srivastava *et al.*, 2014). We used TensorFlow’s implementations of early stopping and dropout were, paired with stochastic gradient descent(Martín Abadi *et al.*, 2015) to optimize the early stopped and dropout-equipped models. The early stopping ensemble consists of 100 independently trained early stopped models, where the average training time was 14.56 epochs. We chose the single early stopped model by its proximity to the average accuracy of the ensemble members (71.60%). Dropout and ab initio simmering each trained for 20 epochs, and we included the models corresponding to the last 2000 weight updates in the simmering ensemble. At the testing stage, since it has no explicit ensemble to aggregate, we took dropout’s “inference mode” prediction to be its ensemble prediction. (Hara *et al.*, 2016) We used majority voting to aggregate the early stopping and simmering ensembles. For learning the translation task, we trained a reduced version (described in Supplementary Methods) of the Transformer architecture presented in Ref. (Vaswani *et al.*, 2017) with a pre-trained BERT tokenizer from TensorFlow’s model database, and assessed translation accuracy via teacher-forced token prediction accuracy. We used the same batch size and fixed learning rate for all cases, and used the Adam optimizer(Kingma & Ba, 2017) for all the non-simmering training scenarios. The early stopped ensemble consists of 10 independently trained models, with an average training time of 53.1 epochs, aggregated with majority voting as for CIFAR-10. The single early stopped model is presented as a characteristic example of the ensemble members. Based on the average early stopped training time, we trained a model with dropout for 60 epochs and determined its “ensemble test accuracy” in the same way as for CIFAR-10. We found that the simmering ensemble exceeded the test accuracy of all other cases in only 21 training epochs, with a majority-voted ensemble prediction derived from its last 200 sampled models. Thus, for equivalent training time, ab initio simmering produces more accurate predictions than both single-model and ensembled overfitting mitigation techniques for the CIFAR-10 dataset. However, simmering can also both accelerate and exceed the accuracy of other overfitting techniques in a natural language processing task like language translation.

Employing a minimally biased model of training data error allows ab initio simmering to outperform other commonly used overfitting avoidance methods in two different learning tasks: image classification with a convolutional neural network and language translation with a Transformer. Table 1 shows that, for both learning tasks, initio simmering achieves a higher test data accuracy than standard implementations of early stopping and dropout. Consistent with ensemble learning approaches, ab initio simmering outperforms a singular optimized model treated with early stopping. For CIFAR-10, given an equal training time of 20 epochs, ab initio simmering also outperforms two other ensemble learning methods: ensembled early stopping, an ensemble of independently optimized neural networks, and dropout, an ensemble of simultaneously-trained

thinned neural networks. We find that simmering yields a test accuracy of over 82% in 20 epochs, whereas no other method exceeded 76% test accuracy on the same basis. For the translation problem, simmering outperforms all three overfitting mitigation methods with a shorter training time. Using fixed, equivalent learning rates, simmering exceeded the accuracy of both dropout and ensembled early stopping in a small fraction of the training time (21 epochs for simmering, versus 60 epochs for dropout and an average of 53.1 epochs for ensembled early stopping). These comparisons highlight the role of simmering’s distinct minimally-biased ensemble generation in efficiently producing generalizable predictions.

Beyond generalizable prediction, simmering also yields prediction uncertainty estimates that reflect data difficulty. Figure 3 shows how the three ensemble learning approaches predict on two different CIFAR-10 test samples (Fig. 3a–b). Though ensemble uncertainty estimates can be extracted either directly or indirectly (see Supplementary Methods) from all three ensemble learning approaches, the ensemble generated by *ab initio* simmering allows for uncertainty estimates that are proportional to image difficulty as shown in Fig. 3c–d. Thus, the maximum entropy distribution sampling that supports simmering provides both generalizable predictions and additional insight into the neural network’s learning process.

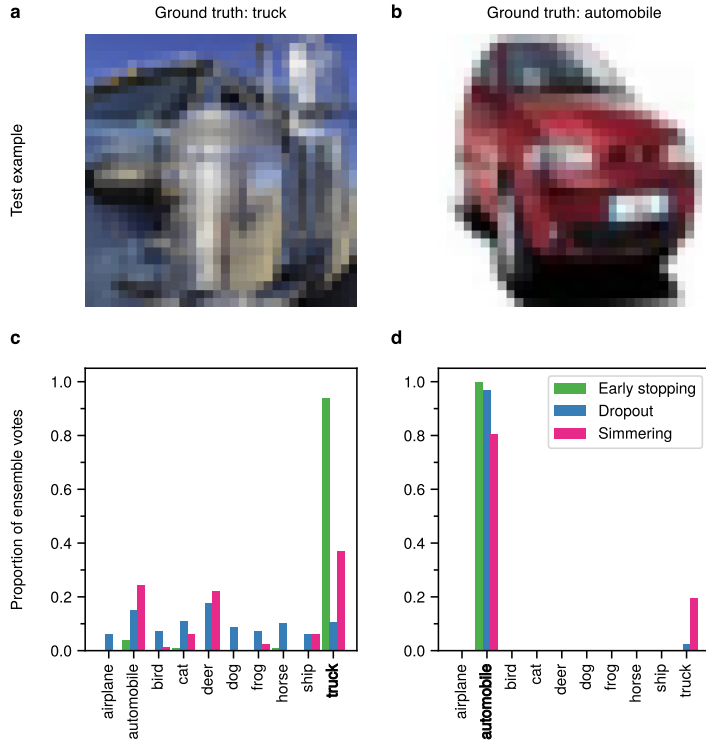


Figure 3: **Ab initio simmering correctly classifies CIFAR-10 test images correctly while also yielding confidence estimates that correspond to image difficulty.** Early stopping (green bars), dropout (blue bars) and ab initio simmering (pink bars) ensembles predict an “easy” image of an automobile (panel b), with a clear foreground and with the automobile in full view, with similarly high confidence (high proportion of ensemble votes). However, when tasked with classifying a “difficult” test image of a truck (panel a), which presents the truck at close range and with low contrast with the background, early stopping remains nearly equally confident (panel c, green bar). In contrast, simmering predicts the correct class, but with lower confidence, reflecting the difficulty of identifying the truck in the image (see Supplementary Methods for image difficulty quantification). Simmering’s second-most voted prediction bears relation to the correct class (“automobile” and “truck” have similar image features). Though dropout also predicts uncertainly, it predicts the correct label with near-equivalent confidence to most other classes. Thus, ab initio simmering predicts images of varying difficulty accurately with confidence proportional to image difficulty.

For both training tasks, which employ different complex architectures and train on distinct data types, ab initio simmering can achieve a higher test data accuracy in either equivalent or less training time.

We note that the for the purposes of a constructing a direct, head-to-head comparison of Transformer training, we have used fixed, identical learning rates as well as identical batch sizes, and train/test splits for all learning methods.

Furthermore, the manuscript would benefit from a more thorough comparison of the proposed method against existing regularization techniques commonly used in deep learning, particularly

in mitigating overfitting. Examples include weight decay, dropout, and early stopping. Such comparisons are essential to position the proposed approach within the existing body of work and to demonstrate its advantages or limitations.

We thank the Referee for this suggestion. The revised Results now includes comparisons to dropout and early stopping for training on CIFAR-10 and a language translation task:

Method	CIFAR-10 Test Accuracy (%)	PT-EN Translation Test Accuracy (%)
Early stopping (single)	71.90	39.62 (67 epochs)
Early stopping (ensemble)	73.81	41.42 (53.1 epochs-average)
Dropout	75.53	39.86 (60 epochs)
Ab initio simmering	82.05	43.46 (21 epochs)

Table 1: Simmering outperforms early stopping, ensembled early stopping and dropout on the CIFAR-10 (Krizhevsky & Hinton, 2009) and Portuguese-English (PT-EN) TED talk transcript translation(Ye *et al.*, 2018) datasets. For training on the CIFAR-10 dataset, all four learning cases employed the ConvNet architecture(Krizhevsky, Sutskever, *et al.*, 2012), the standard train/test split. For the dropout case, we applied the dropout rates reported in Ref. (Srivastava *et al.*, 2014). We used TensorFlow’s implementations of early stopping and dropout were, paired with stochastic gradient descent(Martín Abadi *et al.*, 2015) to optimize the early stopped and dropout-equipped models. The early stopping ensemble consists of 100 independently trained early stopped models, where the average training time was 14.56 epochs. We chose the single early stopped model by its proximity to the average accuracy of the ensemble members (71.60%). Dropout and ab initio simmering each trained for 20 epochs, and we included the models corresponding to the last 2000 weight updates in the simmering ensemble. At the testing stage, since it has no explicit ensemble to aggregate, we took dropout’s “inference mode” prediction to be its ensemble prediction. (Hara *et al.*, 2016) We used majority voting to aggregate the early stopping and simmering ensembles. For learning the translation task, we trained a reduced version (described in Supplementary Methods) of the Transformer architecture presented in Ref. (Vaswani *et al.*, 2017) with a pre-trained BERT tokenizer from TensorFlow’s model database, and assessed translation accuracy via teacher-forced token prediction accuracy. We used the same batch size and fixed learning rate for all cases, and used the Adam optimizer(Kingma & Ba, 2017) for all the non-simmering training scenarios. The early stopped ensemble consists of 10 independently trained models, with an average training time of 53.1 epochs, aggregated with majority voting as for CIFAR-10. The single early stopped model is presented as a characteristic example of the ensemble members. Based on the average early stopped training time, we trained a model with dropout for 60 epochs and determined its “ensemble test accuracy” in the same way as for CIFAR-10. We found that the simmering ensemble exceeded the test accuracy of all other cases in only 21 training epochs, with a majority-voted ensemble prediction derived from its last 200 sampled models. Thus, for equivalent training time, ab initio simmering produces more accurate predictions than both single-model and ensembled overfitting mitigation techniques for the CIFAR-10 dataset. However, simmering can also both accelerate and exceed the accuracy of other overfitting techniques in a natural language processing task like language translation.

Employing a minimally biased model of training data error allows ab initio simmering to outperform other commonly used overfitting avoidance methods in two different learning tasks: image classification with a convolutional neural network and language translation with a Transformer. Table 1 shows that, for both learning tasks, initio simmering achieves a higher test data accuracy than standard implementations of early stopping and dropout. Consistent with ensemble learning approaches, ab initio simmering outperforms a singular optimized model treated with early stopping. For CIFAR-10, given an equal training time of 20 epochs, ab initio simmering also outperforms two other ensemble learning methods: ensembled early stopping, an ensemble of independently optimized neural networks, and dropout, an ensemble of simultaneously-trained

thinned neural networks. We find that simmering yields a test accuracy of over 82% in 20 epochs, whereas no other method exceeded 76% test accuracy on the same basis. For the translation problem, simmering outperforms all three overfitting mitigation methods with a shorter training time. Using fixed, equivalent learning rates, simmering exceeded the accuracy of both dropout and ensembled early stopping in a small fraction of the training time (21 epochs for simmering, versus 60 epochs for dropout and an average of 53.1 epochs for ensembled early stopping). These comparisons highlight the role of simmering’s distinct minimally-biased ensemble generation in efficiently producing generalizable predictions.

We have also extended this analysis to include a comparison of ensemble prediction distributions generated by ensembled early stopping, dropout and ab initio simmering:

Beyond generalizable prediction, simmering also yields prediction uncertainty estimates that reflect data difficulty. Figure 3 shows how the three ensemble learning approaches predict on two different CIFAR-10 test samples (Fig. 3a–b). Though ensemble uncertainty estimates can be extracted either directly or indirectly (see Supplementary Methods) from all three ensemble learning approaches, the ensemble generated by ab initio simmering allows for uncertainty estimates that are proportional to image difficulty as shown in Fig. 3c–d. Thus, the maximum entropy distribution sampling that supports simmering provides both generalizable predictions and additional insight into the neural network’s learning process.

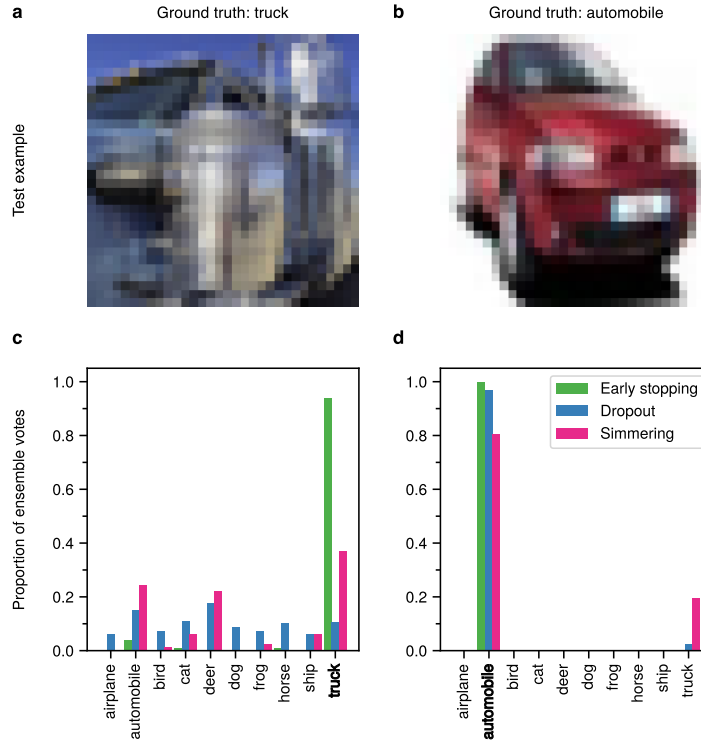


Figure 3: **Ab initio simmering correctly classifies CIFAR-10 test images correctly while also yielding confidence estimates that correspond to image difficulty.** Early stopping (green bars), dropout (blue bars) and ab initio simmering (pink bars) ensembles predict an “easy” image of an automobile (panel b), with a clear foreground and with the automobile in full view, with similarly high confidence (high proportion of ensemble votes). However, when tasked with classifying a “difficult” test image of a truck (panel a), which presents the truck at close range and with low contrast with the background, early stopping remains nearly equally confident (panel c, green bar). In contrast, simmering predicts the correct class, but with lower confidence, reflecting the difficulty of identifying the truck in the image (see Supplementary Methods for image difficulty quantification). Simmering’s second-most voted prediction bears relation to the correct class (“automobile” and “truck” have similar image features). Though dropout also predicts uncertainly, it predicts the correct label with near-equivalent confidence to most other classes. Thus, ab initio simmering predicts images of varying difficulty accurately with confidence proportional to image difficulty.

We believe that the comparison presented in Table 1 and Fig. 3 demonstrates the accuracy and prediction uncertainty quantification advantages of simmering over both singular-model and ensembled predictions in two different training settings.

Finally, the clarity of the methodological description should be improved. As it stands, the explanation of the proposed approach is not sufficiently detailed to enable reproducibility. I recommend the authors provide a more comprehensive and precise account of the algorithm, including its implementation details and overall pipeline.

We thank the Referee for this suggestion. We have revised our description of the implementation in Methods, which now reads:

The particles' equations of motion are derived from the NHC Hamiltonian.(Martyna *et al.*, 1996; Tuckerman *et al.*, 1992) For simplicity of notation, we will henceforth set $m_i = 1, Q_k = 1 \forall i, k$ and describe the equations of motion in terms of the positions and velocities, rather than the positions and momenta. The real particles' positions evolve according to(Jang & Voth, 1997)

$$\dot{x}_i(t) = v_i(t) \quad (24)$$

$$\dot{v}_i(t) = a_i(t) - v_{s_1}(t)v_i(t), \quad (25)$$

where the i subscript denotes real particle quantities, and the s_k subscript denotes a quantity affiliated with the k^{th} virtual particle. The $v_{s_1}(t)v_i(t)$ term from the NHC in Equation (25) distinguishes simmering from gradient-based optimization. For example, in momentum gradient descent, Equation (25) would have $v_{s_1}(t) = \text{const.}$ (França *et al.*, 2020) In contrast, $v_{s_1}(t)$ in NHC dynamics can take on both positive and negative values over time.

The virtual particle dynamics are given by

$$\dot{s}_k(t) = v_{s_k}(t) \quad (26)$$

$$\dot{v}_{s_1}(t) = a_{s_1}(t) - v_{s_2}(t)v_{s_1}(t) \quad (27)$$

$$\dot{v}_{s_k}(t) = a_{s_k}(t) - v_{s_{k+1}}(t)v_{s_k}(t), \quad (28)$$

with the accelerations for all particles $a_i(t), a_{s_1}(t), a_{s_k}(t)$ defined as

$$a_i(t) = -\frac{1}{m_i} \nabla_{x_i} L(x_1(t), \dots, x_N(t)) \quad (29)$$

$$a_{s_1}(t) = \frac{1}{Q_1} \left(\sum_i m_i v_i^2(t) - NT_{target} \right) \quad (30)$$

$$a_{s_k}(t) = \frac{1}{Q_k} (Q_{k-1} v_{k-1}^2(t) - T_{target}). \quad (31)$$

The acceleration of the real particles in Equation (29) is proportional to the negative gradient of the loss with respect to the weights and biases in the neural network.

We present the discretization scheme we used to numerically integrate Equations (26)–(28) in Supplementary Methods. We have revised this description, which reads:

To numerically integrate the particle positions over time, we discretize Main Equations 24–28 using a Verlet integrator for position and velocity for both the real and virtual particles. Verlet integration is used here because it preserves Hamilton's equations.(Jang & Voth, 1997; Yamamoto, 2006) The resulting discretized

equations of motion are

$$x_i(t + \Delta t/2) = x_i(t) + \frac{\Delta t}{2} v_i(t) \quad (32)$$

$$s_{2k}(t + \Delta t/2) = s_{2k}(t) + \frac{\Delta t}{2} v_{s_{2k}}(t) \quad (33)$$

$$v_{s_{2k-1}}(t + \Delta t/2) = v_{s_{2k-1}}(t) e^{\frac{-\Delta t}{2} v_{s_{2k}}} + \frac{\Delta t}{2} a_{s_{2k-1}}(t) e^{\frac{-\Delta t}{4} v_{s_{2k}}}, \quad (34)$$

$$v_i(t + \Delta t) = v_i(t) e^{-\Delta t v_{s_1}(t + \Delta t/2)} + \Delta t a_i(t + \Delta t/2) e^{-\frac{1}{2} \Delta t v_{s_1}(t + \Delta t/2)} \quad (35)$$

$$s_{2k-1}(t + \Delta t) = s_{2k-1}(t) + \Delta t v_{s_{2k-1}}(t + \Delta t/2) \quad (36)$$

$$v_{s_{2k}}(t + \Delta t) = v_{s_{2k}}(t) e^{-\Delta t v_{s_{2k+1}}(t + \Delta t/2)} + \Delta t a_{s_{2k}}(t + \Delta t/2) e^{-\frac{1}{2} \Delta t v_{s_{2k+1}}(t + \Delta t/2)}, \quad (37)$$

$$x_i(t + \Delta t) = x_i(t + \Delta t/2) + \frac{\Delta t}{2} v_i(t + \Delta t) \quad (38)$$

$$s_{2k}(t + \Delta t) = s_{2k}(t + \Delta t/2) + \frac{\Delta t}{2} v_{s_{2k}}(t + \Delta t) \quad (39)$$

$$v_{s_{2k-1}}(t + \Delta t) = v_{s_{2k-1}}(t + \Delta t/2) e^{-\frac{1}{2} \Delta t v_{s_{2k}}(t + \Delta t)} + \frac{\Delta t}{2} a_{s_{2k-1}}(t + \Delta t) e^{-\frac{1}{4} \Delta t v_{s_{2k}}(t + \Delta t)}, \quad (40)$$

where $a_i, a_1, a_{s_k, k \neq 1}$ are defined in Main Equations 29–31. Given this numerical integration scheme, an ensemble of neural networks can be generated by selecting an appropriate learning rate Δt and iteratively computing $x(t + \Delta t)$.

We also note that the experimental conditions corresponding to all test cases and comparisons presented in the manuscript are documented for reproduction in the Supplementary Methods.

We have also updated the open-source implementation provided at Ref. (Babayan *et al.*, 2025) to present a demonstration of training on the CIFAR-10 dataset with ab initio simmering. We made this modification because CIFAR-10 is a standard benchmark dataset for testing machine learning model performance, and its familiarity to the machine learning community facilitates understanding of our training approach. We hope that the extended implementation details in Methods and Supplementary Methods, as well as the updated open-source implementation, clarify the implementation of simmering used in this manuscript.

In summary, while the manuscript introduces a potentially promising approach, significant revisions are needed to substantiate the claims and enhance the clarity and reproducibility of the work. With these improvements, the study could make a more meaningful contribution to the field.

We thank the Referee for the suggestions to better contextualize and describe our work. We believe that the addition of larger convolutional neural network and Transformer training problems, the comparison to dropout, early stopping, and other ensemble methods, and the improved articulation of our methods, substantially improve the manuscript.

Manuscript ID: **NCOMMS-25-24798-A**
Title: **Simmering: sufficient is better than optimal for training neural networks**
Authors: Irina Babayan, Hazhir Aliahmadi, Greg van Anders

Response to Referee Reports

Thank you for your communication and efforts in editing and collating these reviews. We also thank both Referees for their constructive feedback on the manuscript. The Referees made a number of suggestions that improve the clarity and reproducibility of our results. We believe the addition of training performance curves, computational complexity analysis, algorithm pseudocode and additional training procedure details significantly improve the manuscript's presentation.

The reports are reproduced below in their entirety. The quotations are verbatim and indented in **blue, sans-serif font**. In some places we quote text directly from the revised manuscript using **red, serif font**; text is reproduced verbatim except in places where it involves reference numbers, which, since this report does not appear with the context of the full reference list, we have given author-date pairs instead.

In addition, we corrected some minor spelling and grammatical errors.

Following the response to the Referees, we append a highlighted version of the manuscript that indicates the changes we have made in response to the Referees' suggestions and formatting issues.

We hope that you will find the revised manuscript suitable for publication in *Nature Communications*.

Report of Referee 1

Most of my questions are reasonably well addressed by the authors' response. The newly added experiments on CIFAR-10 with ConvNets and NLP with Transformers make the work more solid. Yet some more details are needed for those experiments:

1. for all settings, both training and test performances should be shown, preferably in curves, as functions of training epochs.

We thank the Referee for this suggestion. We agree that the addition of training and test performance curves clarifies the difference in training dynamics between the compared methods. We have added Supplementary Figures 1 and 2, to complement Table 1 in Main Results, which depict training performance as a function of training epochs. The Supplementary Figures section now reads:

Supplementary Figure 1 shows the training performance curves that accompany the CIFAR-10 test accuracy results presented in Main Table 1. Simmering (Supplementary Fig. 1a) and early stopping (Supplementary Fig. 1b) converge to similar individual-model training and validation data accuracies, and their ensemble predictions are aggregated in the same manner (majority

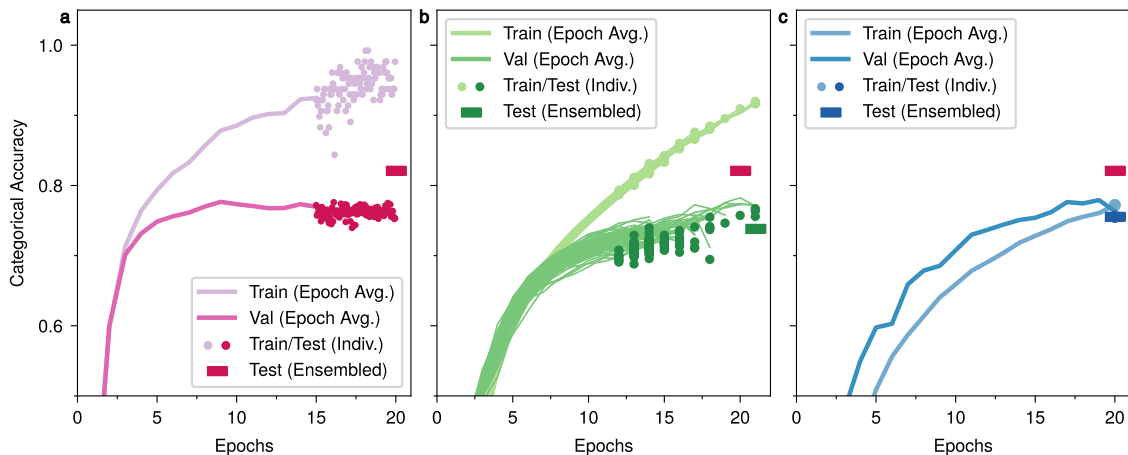


Figure 1: The training convergence curves corresponding to the CIFAR-10 training results, shown in Main Table 1, show that simmering (panel a) produces both the highest ensemble accuracy (red rectangular marker, all panels, corresponding to CIFAR-10 results in Main Table 1) and the most significant ensembling advantage (round vs. rectangular markers in panel a) compared to early stopping (round vs. rectangular markers in panel b), and dropout (round vs. rectangular markers in panel c). In all panels, lines denote the individual model accuracy averaged over all weight updates of a given epoch, and round markers denote the performance of individual models that are included in the ultimate ensemble.

voting). However, the early stopped ensemble aggregate prediction accuracy is approximately the average accuracy of its constituent models (round vs. rectangular markers in Supplementary Fig. 1b), whereas the simmering ensemble significantly improves on its constituent models’ accuracies (round vs. rectangular markers in Supplementary Fig. 1a). The model trained with dropout (Supplementary Fig. 1c) outperforms the early stopping ensemble but does not exceed the ensembled test accuracy of simmering for equivalent training time. Though all three model ensembles have similar individual-model epoch-averaged validation and test accuracies, the ensemble collected via simmering yields the highest ensembled prediction accuracy.

Supplementary Figure 2 shows the accuracy convergence curves that accompany the PT-EN translation test accuracy results (rectangular markers in Supplementary Fig. 2) shown in Main Table 1. Given an identical, fixed learning rate, simmering converges to a higher individual-model training and validation accuracy than models trained with early stopping and dropout in less than half of the number of epochs. As for CIFAR-10, simmering also displays the greatest ensembling advantage relative to the test accuracy of ensemble members: the test accuracy of the early stopped ensemble is lower than its most accurate constituent model (green round marker vs. green rectangular marker in Supplementary Fig. 2), whereas the simmering ensemble accuracy is several percentage points higher than the most accurate constituent model. Dropout converges the most slowly out of the three methods tested (red vs. blue rectangular markers in Supplementary Fig. 2). Overall, given identical learning rates, simmering converges to a higher ensembled test accuracy than both an ensemble of early stopped models and a model trained with dropout in a fraction of the number of training epochs.

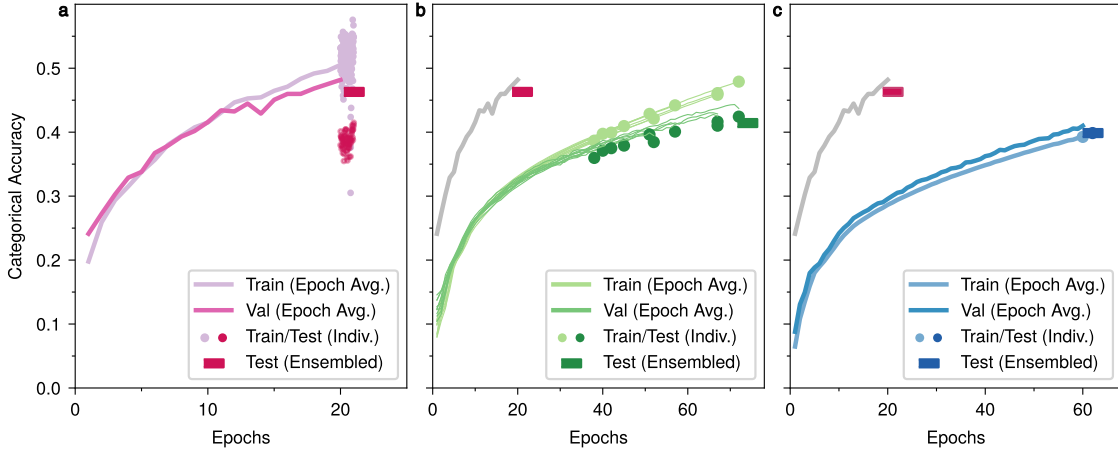


Figure 2: The training convergence curves corresponding to the PT-EN translation training results, shown in Main Table 1, show that simmering (panel a) produces both the highest ensemble accuracy (red rectangular marker, all panels, corresponding to Main Table 1) and the most significant ensembling advantage (round vs. rectangular markers in panel a) in less than half of the training time (grey vs. coloured line, and red vs. green or blue rectangular marker in panels b and c) compared to training with an equivalent learning rate with early stopping (panel b), and dropout (panel c). In all panels, lines denote individual-model accuracy averaged over all weight updates of a given epoch, and round markers denote the performance of individual models that are included in the ultimate ensemble. The simmering epoch-averaged validation curve and ensembled test accuracy (grey line and red rectangular marker, respectively, in panels b and c) are provided for convergence timescale and performance comparison.

2. did you use any data augmentation with CIFAR-10?

We appreciate the opportunity to clarify this aspect of the comparison presented in Table 1. We did not use data augmentation with any of the methods shown in this table. All of the details, e.g., batch size, model architecture, hyperparameter choices, pertaining to the setup of each training experiment are described in Supplementary Methods.

Regarding the new Figure 3 of model confidence. It would be more convincing if some statistics are shown with a larger sample size, instead of a single example. One example is good for illustration purpose [sic] but it is not sufficient to draw a conclusion on such a property.

The Referee raises an important point about the model confidence example we show in Main Figure 3. We agree that statistics for a larger sample size would substantiate the presence of the property we illustrate in the figure. The challenge with collecting statistics to measure model confidence dependence on image difficulty is that there is currently no consensus on a single measure, computed from a data sample, that effectively and universally encapsulates “image difficulty”. For example, there is poor agreement between “easy” and “difficult” images based on pixel-intensity standard deviation, Shannon entropy, and intra-class cosine similarity. Defining an appropriate measure, and comparing the performance of different training methods using this measure, is beyond the scope of this investigation and we intend to report on this in future work.

To ensure that Figure 3 is interpreted as an illustrative example, we have edited the description of the figure in the main text. The Main Results section now reads

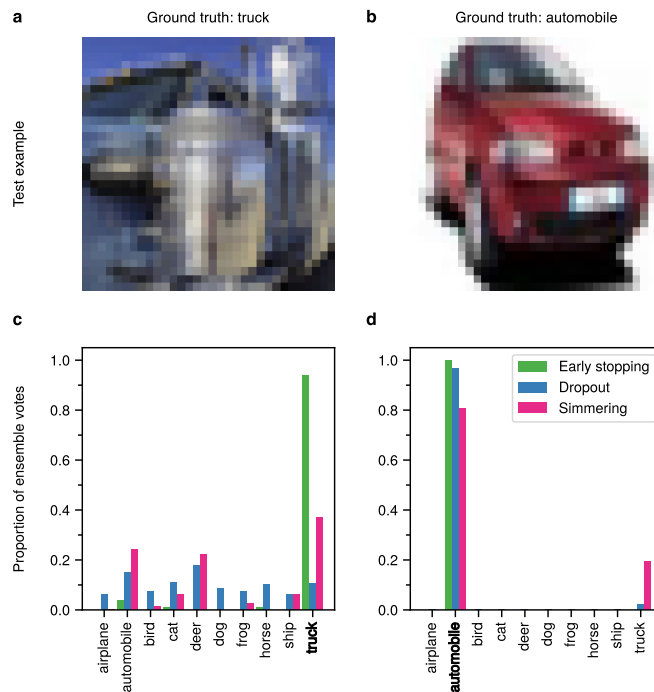


Figure 3: Ab initio simmering correctly classifies CIFAR-10 test images. Panel b shows an image with high pixel-intensity standard deviation and low Shannon entropy, which depicts an automobile, with a clear foreground and with the automobile in full view. Early stopping (green bars), dropout (blue bars) and ab initio simmering (pink bars) ensembles yield predictions with similarly high confidence (high proportion of ensemble votes). In contrast, panel a shows an image with low pixel-intensity standard deviation and high Shannon entropy, which depicts a truck at close range with low background contrast. Early stopping remains nearly equally confident (panel c, green bar) whereas simmering predicts the correct class with lower confidence, aligning with the difficulty of identifying the truck in the image. Simmering’s second-most voted prediction bears relation to the correct class (“automobile” and “truck” have similar image features). Though dropout also predicts uncertainly, it predicts the correct label with near-equivalent confidence to most other classes. Thus, ab initio simmering predicts images of varying difficulty accurately with confidence that can reflect image difficulty.

...

Beyond generalizable prediction, simmering also yields prediction uncertainty estimates that can reflect data difficulty. Figure 3 shows how the three ensemble learning approaches predict two different CIFAR-10 test samples (Fig. 3a–b) chosen based on relative image difficulty (pixel-intensity standard deviation and Shannon entropy). Though ensemble uncertainty estimates can be extracted either directly or indirectly (see Supplementary Methods) from all three ensemble learning approaches, the ensemble generated by ab initio simmering allows for uncertainty estimates that reflect relative image difficulty as shown in Fig. 3c–d. Thus, the maximum entropy distribution sampling that supports simmering provides generalizable predictions and may offer additional insight into the neural network’s learning process.

We thank the Referee for their careful examination of the work. We believe that Referee’s suggestions have stimulated changes in the revised manuscript that substantially improve its presentation.

Report of Referee 2

The paper proposes a “sufficient training” method called Simmering to mitigate overfitting in deep learning. It uses a Nosé–Hoover Chain (NHC) thermostat to sample parameter configurations and aggregates predictions via ensemble [sic]. While applying NHC to deep learning is novel, the core gains against overfitting largely stem from ensembling, which is already well studied.

We thank the Referee for noting that our approach provides gains against overfitting. The Referee is also correct that our approach uses ensembles, but that is not the primary origin of the advantage of simmering.

The manuscript describes several ways in which simmering differs from optimization-based training. One key difference is in the way it generates an ensemble. As a sufficient training method, simmering does not attempt to generate optimal weights and biases. Instead it systematically generates non-optimal weights and biases. This contrasts with, e.g., ensembles of early-stopped models where, prior to stopping, the algorithm seeks to identify optimal weights and biases. The distinction between systematically generating non-optimal weights and biases and searching for optimal weights and biases is a fundamental difference between the approach we present here and all other ensemble-based methods we are aware of.

We also have added two figures to the SI which, in addition to depicting the accuracy convergence, show the ensembling advantage relative to individual model accuracy of each ensembling method. The Supplementary Figures section now reads

Supplementary Figure 1 shows the training performance curves that accompany the CIFAR-10 test accuracy presented in Main Table 1. Simmering (Supplementary Fig. 1a) and early stopping (Supplementary Fig. 1b) converge to similar individual-model training and validation data accuracies, and their ensemble predictions are aggregated in the same manner (majority voting). However, the early stopped ensemble aggregate prediction accuracy is approximately the average accuracy of its constituent models (round vs. rectangular markers in Supplementary Fig. 1b), whereas the simmering ensemble significantly improves on its constituent models’ accuracies (round vs. rectangular markers in Supplementary Fig. 1a). The model trained with dropout (Supplementary Fig. 1c) outperforms the early stopping ensemble but does not exceed the ensembled test accuracy of simmering for equivalent training time. Though all three model ensembles have similar individual-model epoch-averaged validation and test accuracies, the ensemble collected via simmering yields the highest ensembled prediction accuracy.

Supplementary Figure 2 shows the accuracy convergence curves that accompany the PT-EN translation test accuracy (rectangular markers in Supplementary Fig. 2) shown in Main Table 1. Given an identical, fixed learning rate, simmering converges to a higher individual-model training and validation accuracy than models trained with early stopping and dropout in less than half of the number of epochs. As for CIFAR-10, simmering also displays the greatest ensembling advantage relative to the test accuracy of ensemble members: the test accuracy of the early stopped ensemble is lower than its most accurate constituent model (green round marker vs. green rectangular marker in Supplementary Fig. 2), whereas the simmering ensemble accuracy is several

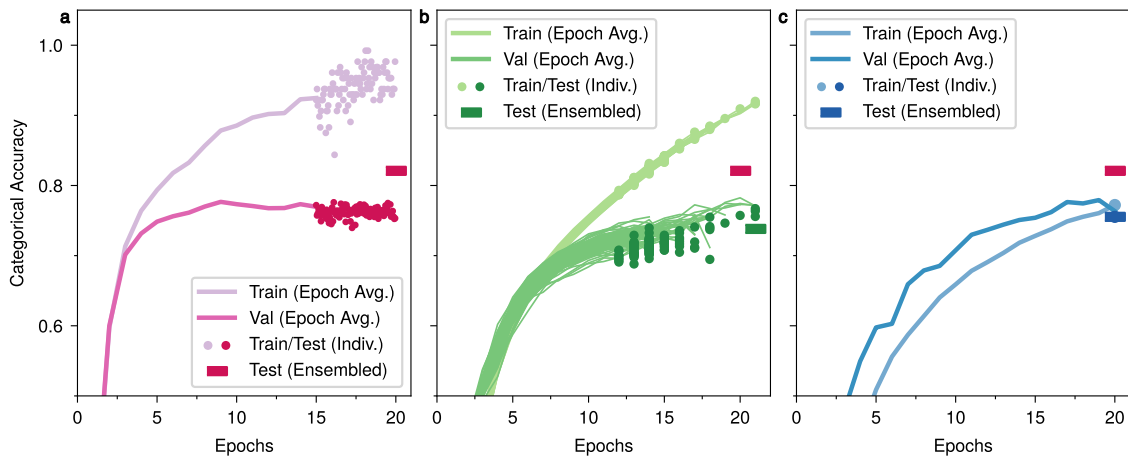


Figure 1: The training convergence curves corresponding to the CIFAR-10 training , shown in Main Table 1, show that simmering (panel a) produces both the highest ensemble accuracy (red rectangular marker, all panels, corresponding to CIFAR-10 in Main Table 1) and the most significant ensembling advantage (round vs. rectangular markers in panel a) compared to early stopping (round vs. rectangular markers in panel b), and dropout (round vs. rectangular markers in panel c). In all panels, lines denote the individual model accuracy averaged over all weight updates of a given epoch, and round markers denote the performance of individual models that are included in the ultimate ensemble.

percentage points higher than the most accurate constituent model. Dropout converges the most slowly out of the three methods tested (red vs. blue rectangular markers in Supplementary Fig. 2). Overall, given identical learning rates, simmering converges to a higher ensembled test accuracy than both an ensemble of early stopped models and a model trained with dropout in a fraction of the number of training epochs.

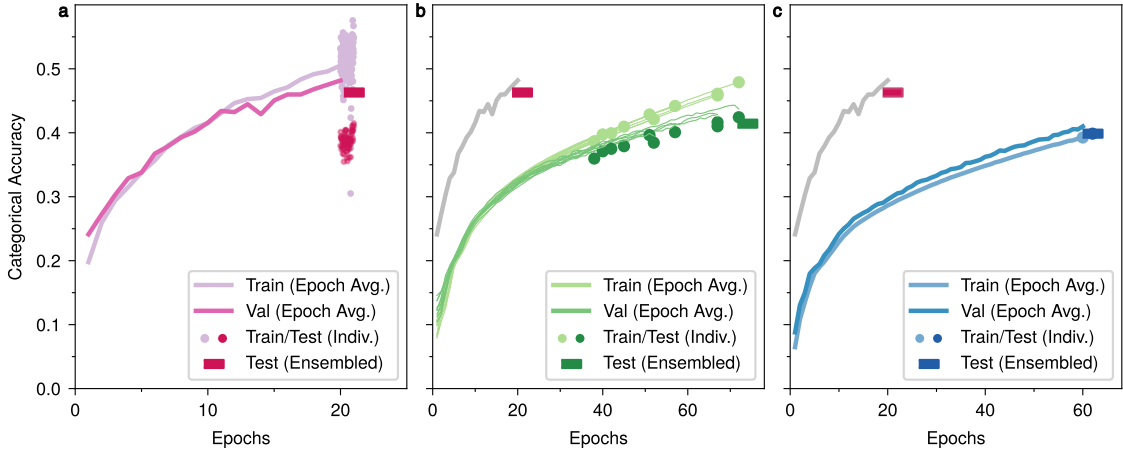


Figure 2: The training convergence curves corresponding to the PT-EN translation training , shown in Main Table 1, show that simmering (panel a) produces both the highest ensemble accuracy (red rectangular marker, all panels, corresponding to Main Table 1) and the most significant ensembling advantage (round vs. rectangular markers in panel a) in less than half of the training time (grey vs. coloured line, and red vs. green or blue rectangular marker in panels b and c) compared to training with an equivalent learning rate with early stopping (panel b), and dropout (panel c). In all panels, lines denote individual-model accuracy averaged over all weight updates of a given epoch, and round markers denote the performance of individual models that are included in the ultimate ensemble. The simmering epoch-averaged validation curve and ensembled test accuracy (grey line and red rectangular marker, respectively, in panels b and c) are provided for convergence timescale and performance comparison.

If the advantage of simmering stemmed solely from the act of ensembling models, then simmering would achieve a similar ensembled accuracy to the early stopped ensemble and dropout given similar individual model test accuracy. Instead, we find that the simmering ensemble yields both the highest ensemble test accuracy and the largest individual model vs. ensemble performance improvement, highlighting the difference between ensembling with simmering and with optimization-based methods.

Simmering requires forward and backward passes over many parameter configurations, incurring substantial computational and time costs that scale poorly for large models. However, the paper provides no quantitative analysis or comparison of computational complexity, training time, or resource usage, and experiments are mostly limited to small-scale settings, making it difficult to assess real-world performance and scalability.

We thank the Referee for suggesting this clarification. We have now added the following computational complexity analysis to SI:

Simmering has a per-iteration complexity comparable to standard first-order methods (e.g., SGD with momentum), with time complexity $O(N)$. The parameter updates involve only elementary operations (exponentiation and multiplication) on a handful of variables and a single N -length dot product. The algorithm stores one momentum value per neural network parameter (similar to momentum-based optimizers) and $2 \times N_{NHC}$ variables for the thermostat chain (e.g., 2×10 variables for the tested problems). This is comparable to algorithms like SGD with momentum or Adam (which store one or two extra tensors of size N). Therefore, memory usage scales as $O(N)$, which is of essentially the same order as standard training. Moreover, the extra computations for thermostat variable integration are a fixed small cost, independent of model size. No large auxiliary data structures are needed beyond what typical optimizers use.

In practice, training time depends not only on per-iteration cost (discussed above) but also on the number of iterations required to reach a target performance. Traditional ensemble methods like ensembled early stopping, bagging, or boosting, require an independent training run per ensemble member prior to aggregation. In contrast, since simmering samples near-optimal solutions rather than seeking a single loss minimum, it can generate an entire ensemble implicitly within a single training run. This reduces the overall training time significantly compared to traditional ensemble methods, particularly when convergence is time-consuming, e.g., for large Transformers.

In addition, we note that simmering requires the same number of forward/backward passes as standard gradient-based optimization training. For example, for the default parameters in the simmering open-source code (Babayan *et al.*, 2025), initialization and training to achieve a test accuracy of 78.5% on CIFAR-10 requires 59 seconds of wall time on a workstation with an NVIDIA GeForce RTX 4060 GPU, whereas obtaining a similar level of accuracy using Adam and dropout on the same hardware requires 89 seconds of wall time. We also note that our simmering implementation is in pure Python, and we expect that its performance could be further improved if the integration steps were implemented in C++/CUDA.

However, we also note that our aim here is not to identify or produce the most optimal implementation of simmering. Instead our aim is to demonstrate that sufficient training is a viable approach for generating generalizable neural networks and that even a naive implementation of simmering is comparable to other leading methods.

The authors claim that “ab initio” Simmering from random initialization outperforms baselines, but do not provide corresponding training/validation loss curves, leaving the evidence insufficient.

We thank the Referee for this suggestion. We agree that the addition of the training/validation performance curves clarifies the difference in training dynamics between ab initio simmering and the other methods shown in Main Table 1. We have added two new figures to SI, Supplementary Figures 1 and 2, to complement Main Table 1 in Main Results. The Supplementary Figures section now reads

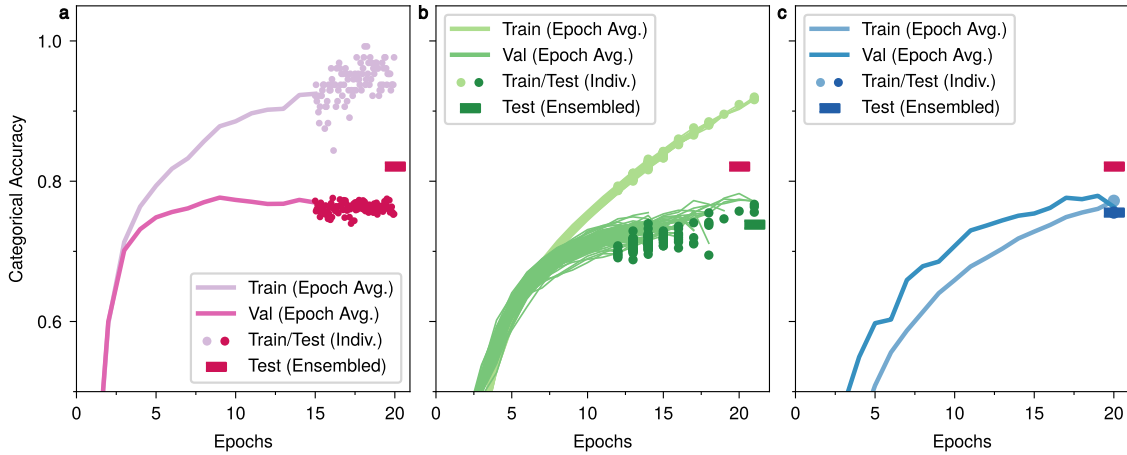


Figure 1: The training convergence curves corresponding to the CIFAR-10 training , shown in Main Table 1, show that simmering (panel a) produces both the highest ensemble accuracy (red rectangular marker, all panels, corresponding to CIFAR-10 results in Main Table 1) and the most significant ensembling advantage (round vs. rectangular markers in panel a) compared to early stopping (round vs. rectangular markers in panel b), and dropout (round vs. rectangular markers in panel c). In all panels, lines denote the individual model accuracy averaged over all weight updates of a given epoch, and round markers denote the performance of individual models that are included in the ultimate ensemble.

Supplementary Figure 1 shows the training performance curves that accompany the CIFAR-10 test accuracy results presented in Main Table 1. Simmering (Supplementary Fig. 1a) and early stopping (Supplementary Fig. 1b) converge to similar individual-model training and validation data accuracies, and their ensemble predictions are aggregated in the same manner (majority voting). However, the early stopped ensemble aggregate prediction accuracy is approximately the average accuracy of its constituent models (round vs. rectangular markers in Supplementary Fig. 1b), whereas the simmering ensemble significantly improves on its constituent models’ accuracies (round vs. rectangular markers in Supplementary Fig. 1a). The model trained with dropout (Supplementary Fig. 1c) outperforms the early stopping ensemble but does not exceed the ensembled test accuracy of simmering for equivalent training time. Though all three model ensembles have similar individual-model epoch-averaged validation and test accuracies, the ensemble collected via simmering yields the highest ensembled prediction accuracy.

Supplementary Figure 2 shows the accuracy convergence curves that accompany the PT-EN translation test accuracy results (rectangular markers in Supplementary Fig. 2) shown in Main Table 1. Given an identical, fixed learning rate, simmering converges to a higher individual-model training and validation accuracy than models trained with early stopping and dropout in

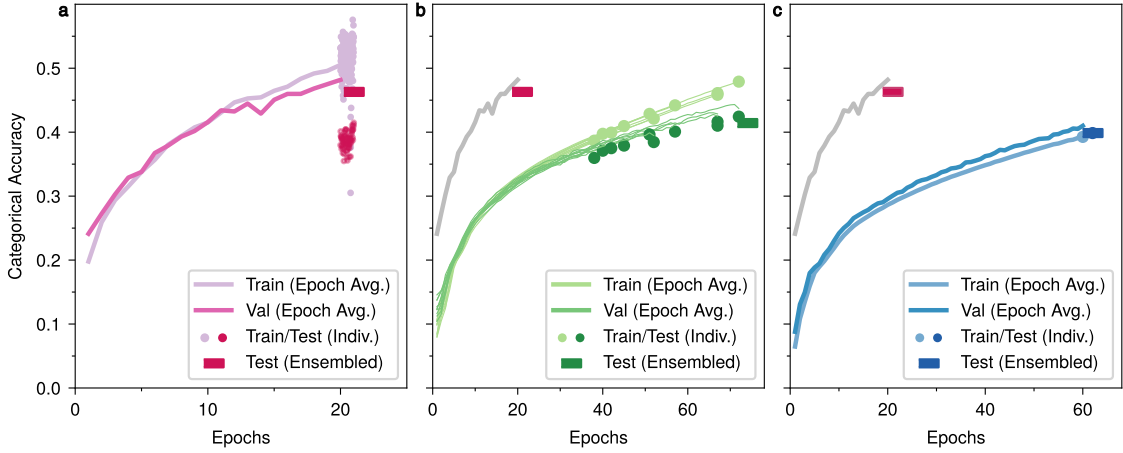


Figure 2: The training convergence curves corresponding to the PT-EN translation training results, shown in Main Table 1, show that simmering (panel a) produces both the highest ensemble accuracy (red rectangular marker, all panels, corresponding to Main Table 1) and the most significant ensembling advantage (round vs. rectangular markers in panel a) in less than half of the training time (grey vs. coloured line, and red vs. green or blue rectangular marker in panels b and c) compared to training with an equivalent learning rate with early stopping (panel b), and dropout (panel c). In all panels, lines denote individual-model accuracy averaged over all weight updates of a given epoch, and round markers denote the performance of individual models that are included in the ultimate ensemble. The simmering epoch-averaged validation curve and ensembled test accuracy (grey line and red rectangular marker, respectively, in panels b and c) are provided for convergence timescale and performance comparison.

less than half of the number of epochs. As for CIFAR-10, simmering also displays the greatest ensembling advantage relative to the test accuracy of ensemble members: the test accuracy of the early stopped ensemble is lower than its most accurate constituent model (green round marker vs. green rectangular marker in Supplementary Fig. 2), whereas the simmering ensemble accuracy is several percentage points higher than the most accurate constituent model. Dropout converges the most slowly out of the three methods tested (red vs. blue rectangular markers in Supplementary Fig. 2). Overall, given identical learning rates, simmering converges to a higher ensembled test accuracy than both an ensemble of early stopped models and a model trained with dropout in a fraction of the number of training epochs.

The assertion in Supplementary Fig. 1 that Adam and Simmering follow distinct training trajectories is also questionable: the Adam trajectory reflects training from scratch and is expected to move normal to loss contours; the Simmering trajectory is a 100-epoch slice taken from the Adam run after near-convergence, where oscillations around a local minimum are typical. Thus, differences between Fig. 1a and Fig. 1b are likely due to different training phases rather than algorithmic mechanics.

The Referee is correct that the Adam and simmering trajectories are taken from different training regimes. However, that is by necessity, since optimization-based training approaches like Adam do not exhibit the form of oscillations that occur in simmering. Hence there exists no comparable regime to show.

We thank the Referee for pointing out this issue of potential confusion. In Adam and similar algorithms, the oscillations that the Referee describe occur because finite differencing prevents algorithms from reaching exact convergence. The oscillations that occur in simmering are not a finite differencing artifact. Despite the apparent similarity, the oscillations in simmering are the manifestation of the physical analogue of thermal oscillations. Thermal oscillations occur in nature where there is no finite time step (in nature these oscillations determine heat capacity), and they would occur in simmering even the time integration was exact. This same effect is present in tens of thousands of papers that employ molecular dynamics simulation in conventional physics, chemistry, biological, and materials science contexts.

The thermal nature of the fluctuations in simmering is also readily apparent in the retrofitting results shown in main text Fig. 1. At the scale of the plots, fluctuations are not visible in the initial simmering period at low temperature but are clearly discernible at higher temperature. If the oscillations were the same numerical artifact that occurs in optimization, they would not exhibit this characteristic temperature-dependent behaviour that is present in reams of molecular dynamics literature.

We clarified this in the SI text associated with Supplementary Figure 3 as follows:

Figure 3, generated using the loss visualization method of Ref. (Li *et al.*, 2018), shows the difference in training dynamics between Adam, (Kingma & Ba, 2017) a gradient-based optimizer, and simmering, an implementation of sufficient training. The Adam optimizer travels perpendicular to the loss contours, whereas simmering travels parallel to the direction of near-equivalent loss. The first principal component (PC) of Adam describes 90.44% of its trajectory variance in parameter space, and the second PC describes another 6.12%. For simmering, the first and second PCs describe 99.9428% and 0.0324% of the parameter space trajectory respectively. In the latter case, a larger proportion of the training trajectory is described by the first PC. Since this PC points along the direction of equivalent loss in parameter space, Supplementary Fig. 3b shows that sufficient training automatically identifies and samples along sloppy (equivalent-loss) modes in model parameter space as described in Main Methods. Note that optimization-based training can also exhibit a form of oscillation that results from finite differencing artifacts, whereas simmering is designed to exhibit thermal oscillations instead. These thermal oscillations are a critical feature that simmering leverages to sample an ensemble of models.

Moreover, the images in Fig. 3 are very low resolution, making human recognition difficult and weakening the persuasive power of the comparison.

We appreciate the opportunity to motivate our choice of illustrative example presented in Main Figure 3. The resolution of the visualization we show exactly matches the resolution of the images in the CIFAR-10 dataset, which consists of 32x32x3-pixel images. (Krizhevsky & Hinton, 2009) We opted to generate this illustrative example with this dataset because it was recommended by both Referees in the first set of revisions as a familiar and effective benchmark for learning method performance.

The writing also needs improvement: the paper contains many mathematical symbols and formulas, but the exposition is not well organized and the notation is unclear, failing to present an intuitive algorithmic pipeline and hindering reproducibility.

Given the interdisciplinary nature of this work, to maximize reproducibility we presented our approach in terms and notation that most directly facilitate comparison with similar algorithms that appear in well-known literature on thermostats in other contexts. Choosing notation that facilitates this form of reproducibility has the secondary advantage that it provides the most straightforward way of representing algorithms of this type. (Which is not coincidental because the notation was developed in other literature to do just that.) We also note that molecular dynamics algorithms were some of the first algorithms that were developed for digital computers in the 1950s, and our presentation of the algorithm follows disciplinary conventions that evolved to ensure the reproducibility of scientific algorithms over the course of several decades.

In addition, we note that it is possible to resolve any questions that readers may encounter with unfamiliar notation by referring to/comparing with the source code we provide. For example, as noted by one of the Referees in the first round of review, it is possible to reproduce results from this investigation using a one-line command with the code we provide. Our most recent code release provides this reproducibility capability for the CIFAR-10 dataset. We describe the simmering hyperparameters and other training details, e.g., model architecture, data pre-processing, loss function, batch size, and number of epochs, for each training scenario we presented, in the Supplementary Methods. These descriptions, paired with the open-source code we provided, would enable readers to directly reproduce any of the specific training problems we present in this work.

To further facilitate reproducibility we have added a pseudocode description of the algorithm that is analogous to representation of Adam in Ref. (Kingma & Ba, 2017), accompanied by a training procedure description.

The revised Supplementary Methods section now reads

Algorithm 1 describes the implementation of simmering used to generate the results presented in this work. There are two main “phases” of simmering: the equilibration phase, and the sampling phase. Both phases use the same algorithm (Algorithm 1), but during sampling, model parameters and/or predictions are saved for offline ensemble prediction. The main simmering hyperparameters are the thermostat temperature T , the particle mass m , and the learning rate Δt . The thermostat temperature T can be either a constant or a schedule that depends on training time t , but note that the system must be equilibrated during sampling to sample from the canonical ensemble. For our training experiments, we used the same mass for all real and virtual particles, i.e., $m_i = Q_k \forall i, k$, based on Ref. (Martyna *et al.*, 1992). Generally, decreasing the mass makes the algorithm more sensitive to the loss gradient and thus can accelerate equilibration. The learning rate Δt can be a schedule, i.e., depend on time t , but must be the same for all particles (real and virtual). We also note that we used a NHC length of 10 for all experiments. To commence equilibration, provide the initial weights and biases of the neural network as $x_{all,0}$. Then, during training, before executing line 20 of the for-loop in Algorithm 1, set the neural network parameters to $x_{all,t+1/2}$, and use backpropagation to compute the gradient of the loss $\nabla_x L$. The system is equilibrated once T_{real} is fluctuating around T . The precision with which the thermostat algorithm will fluctuate about T depends on the neural network parameter space dimensionality, so expect that T_{real} will fluctuate with a larger amplitude for smaller neural networks.

Once equilibration is complete, the same algorithm is reused for sampling. The only difference in execution is that the model parameters and/or predictions (depending on the neural network and dataset sizes) generated at the loss gradient computation stage (line 18 of Algorithm 1) are saved for offline ensemble analysis. There is no prescription for sampling frequency. However, some options for sampling frequency include saving every iteration’s predictions, saving the predictions at a fixed iteration interval, or saving randomly. If sampling at some fixed interval,

check that the samples are uncorrelated at that interval. After the desired number of samples have been collected, the set of models and/or predictions can be aggregated or analyzed as a distribution of solutions in the ways described in the main text.

Algorithm 1 Simmering algorithm for a generic learning problem described by a differentiable objective function L which has parameters x , based on the NHC thermostat dynamics integration in Ref. (Jang & Voth, 1997). The first subscript of each variable indexes the particles (i for real particles and k for virtual), and the second subscript indicates the timestep of the associated variable. Note that since the NHC particles are integrated differently based on their chain position (even or odd), the virtual particle index takes on values $k = 1, \dots, N_{NHC}/2$. Unless otherwise indicated (e.g., with a particle index of *all*), operations are performed element-wise for all elements in the given vector(s). If no initialization is specified, assume that a vector's initial value is 0 for all elements.

Require: $T > 0$: Thermostat temperature

Require: Δt : Learning rate

Require: $L(x)$: Objective function L with parameters x

Require: $x_{all,0}$: Initial parameter vector/initial real particle positions, of dimension N

Require: m : Real particle mass vector, of dimension N

Require: t_{max} : Number of iterations

Require: Q : NHC particle mass vector, of dimension N_{NHC}

```

1: function POSITION_FORWARD( $a, b, dt$ )                                ▷ Integrator for positions ( $x_i, s_{2k}, s_{2k-1}$ ).
2:   return  $a + dt \cdot b$ 
3: function VELOCITY_FORWARD( $a, b, c, dt$ )                            ▷ Integrator for velocities ( $v_i, v_{s_{2k}}, v_{s_{2k-1}}$ ).
4:   return  $a \exp\{-dt \cdot c\} + dt \cdot b \cdot \exp\{-dt/2 \cdot c\}$ 
5: function ACCELERATION_S1( $v, m, T, N, Q_1$ )                        ▷ Acceleration of the first NHC particle ( $a_{s_1}$ ).
6:   return  $(\sum_j m_j v_j^2 - N \cdot T)/Q_1$ 
7: function ACCELERATION_SK( $v, T, b, c$ )                            ▷ Acceleration of all other NHC particles ( $a_{s_{2k}}, a_{s_{2k-1} \neq 1}$ ).
8:   return  $(c \cdot v^2 - T)/b$ 
9: function REAL_T( $v, m$ )                                            ▷ For computing real particle system temperature.
10:  return  $(\sum_j m_j v_j^2)/N$ 
11:  $v_{i,0} \leftarrow \sqrt{\frac{T}{m}}$                                           ▷ Initialize velocity vector elements.
12:  $T_{real} \leftarrow T$                                               ▷ Define current real system temperature.
13:  $t \leftarrow 0$                                                     ▷ Initialize timestep.
14: while  $t < t_{max}$  do
15:    $x_{i,t+1/2} \leftarrow \text{POSITION\_FORWARD}(x_{i,t}, v_{i,t}, \Delta t/2)$     ▷ First half-step update to  $x_i$ .
16:    $s_{2k,t+1/2} \leftarrow \text{POSITION\_FORWARD}(s_{2k,t}, v_{s_{2k},t}, \Delta t/2)$     ▷ First half-step update to  $s_{2k}$ .
17:    $a_{s_1,t} \leftarrow \text{ACCELERATION\_S1}(v_{all,t}, m_{all}, T, N, Q_1)$ 
18:    $a_{s_{2k-1} \neq 1,t} \leftarrow \text{ACCELERATION\_SK}(v_{s_{2(k-1)},t}, T, Q_{2k-1}, Q_{2(k-1)})$ 
19:    $v_{s_{2k-1},t+1/2} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{2k-1},t}, a_{s_{2k-1},t}, v_{s_{2k},t}, \Delta t/2)$     ▷ First half-step update to
     $v_{s_{2k-1}}$ .
20:    $a_{i,t+1/2} \leftarrow -\nabla_{x_i} L(x_{all,t+1/2})/m_i$                 ▷ Use loss gradient here!
21:    $v_{i,t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{i,t}, a_{i,t+1/2}, v_{s_1,t+1/2}, \Delta t)$     ▷ Full-step update to  $v_i$ .
22:    $s_{2k-1,t+1} \leftarrow \text{POSITION\_FORWARD}(s_{2k-1,t}, v_{s_{2k-1},t+1/2}, \Delta t)$     ▷ Full-step update to  $s_{2k-1}$ .
23:    $a_{s_{2k},t+1/2} \leftarrow \text{ACCELERATION\_SK}(v_{s_{2k-1},t+1/2}, T, Q_{2k}, Q_{2k-1})$ 
24:    $v_{s_{2k \neq N_c},t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{2k},t}, a_{s_{2k},t+1/2}, v_{s_{2k+1},t+1/2}, \Delta t)$     ▷ Full-step update to
     $v_{s_{2k}}$ .
25:    $v_{s_{N_c},t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{N_c},t}, a_{s_{N_c},t+1/2}, 0, \Delta t)$     ▷ Full-step update to  $v_{s_{N_c}}$ .
26:    $x_{i,t+1} \leftarrow \text{POSITION\_FORWARD}(x_{i,t+1/2}, v_{i,t+1}, \Delta t/2)$     ▷ Last half-step update to  $x_i$ .
27:    $s_{2k,t+1} \leftarrow \text{POSITION\_FORWARD}(s_{2k,t+1/2}, v_{s_{2k},t+1}, \Delta t/2)$     ▷ Last half-step update to  $s_{2k}$ .
28:    $a_{s_1,t+1} \leftarrow \text{ACCELERATION\_S1}(v_{all,t+1}, m_{all}, T, N, Q_1)$ 
29:    $a_{s_{2k-1} \neq 1,t+1} \leftarrow \text{ACCELERATION\_SK}(v_{s_{2(k-1)},t+1}, T, Q_{2k-1 \neq 1}, Q_{2(k-1)})$ 
30:    $v_{s_{2k-1},t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{2k-1},t+1/2}, a_{s_{2k-1},t+1}, v_{s_{2k},t+1}, \Delta t/2)$     ▷ Last half-step
    update to  $v_{s_{2k-1}}$ .
31:    $T_{real} \leftarrow \text{REAL\_T}(v_{all,t+1}, m_{all})$                 ▷ Optional: Track real system T throughout sampling.
32:    $t \leftarrow t + 1$ 

```

I recommend adding a complete training pipeline description (including hyperparameters, temperature/learning-rate schedules, sampling and ensembling strategy), providing complexity and resource-cost comparisons, releasing convergence curves for from-scratch training, and supplying higher-quality visualizations to substantiate the claims.

We thank the Referee for the suggestions to better present our work. We believe that the addition of convergence curves, complexity analysis and additional implementation details promote reproducibility and better contextualize our method's performance.

Manuscript ID: **NCOMMS-25-24798-C**
Title: **Simmering: sufficient is better than optimal for training neural networks**
Authors: Irina Babayan, Hazhir Aliahmadi, Greg van Anders

Response to Referee Reports

We thank both Referees for their feedback on the manuscript. The Referees made a number of suggestions regarding results and methods presentation that improve the impact and interpretability of our results.

The reports are reproduced below in their entirety. The quotations are verbatim and indented in **blue, sans-serif font**. In some places we quote text directly from the revised manuscript using **red, serif font**; text is reproduced verbatim except in places where it involves reference numbers, for which, since this report does not appear with the context of the full reference list, we have given author-date pairs instead.

Report of Referee 1

The results seem overall interesting. A few things to improve:

1. I feel that Supplementary Figure 1 and 2 are much more informative than Table 1. Perhaps replace table 1 with those figures will make the main paper more convincing.

We thank the Referee for this suggestion, and agree that the contents of Supplementary Figures 1 and 2 better display the unique advantages of simmering.

We have replaced Table 1 with a new figure, Main Figure 3, that compares the ensemble and individual model performances on test data for each of the three overfitting mitigation methods, and for both training tasks tested. As indicated by this Referee’s next comment, these are the features of the convergence curves that most clearly highlight the difference between simmering and other methods. We have left Supplementary Figures 1 and 2 as they are in Supplementary Information, so the full accuracy convergence curves are still available in the manuscript.

The Main Results section now reads,

Employing a minimally biased model of training data error allows ab initio simmering to outperform other commonly used overfitting avoidance methods in two different learning tasks: image classification with a convolutional neural network and language translation with a Transformer. Fig. 3 shows that, for both learning tasks, ab initio simmering achieves a higher test data accuracy than standard implementations of early stopping and dropout. Consistent with ensemble learning approaches, ab initio simmering outperforms singular optimized models treated with early stopping (rectangular maroon marker vs. round green markers in Fig. 3). For CIFAR-10, given an equal training time of 20 epochs, ab initio simmering also outperforms two other ensemble learning methods: ensembled early stopping, an ensemble of independently optimized neural networks, and dropout, an ensemble of simultaneously-trained thinned neural networks. We find

that simmering yields a test accuracy of over 82% in 20 epochs, whereas no other method exceeded 76% test accuracy on the same basis. In addition to yielding the most accurate ensemble predictions, we find that the simmering ensemble yields the most pronounced improvement in accuracy after ensemble aggregation, despite its ensemble members having similar accuracies to those of the early stopped ensemble. This difference in ensembling effectiveness indicates that the simmering ensemble, consisting of sufficient rather than optimal representations, uniquely ensures generalizable aggregate predictions. In Supplementary Information, we also present examples of prediction uncertainty analysis on CIFAR-10 images, analogous to Figure ??c-d, that display the difference in ensemble prediction distribution between the three overfitting mitigation methods in Fig. 3a.

For the translation problem, simmering outperforms all three overfitting mitigation methods with a shorter training time. Using fixed, equivalent learning rates, simmering exceeded the accuracy of both dropout and ensembled early stopping in a small fraction of the training time (21 epochs for simmering, versus 60 epochs for dropout and an average of 53.1 epochs for ensembled early stopping). As for CIFAR-10, simmering had both the most accurate ensemble prediction and the most significant ensembling improvement. These comparisons highlight the role of simmering’s distinct minimally-biased ensemble generation in efficiently producing generalizable predictions.

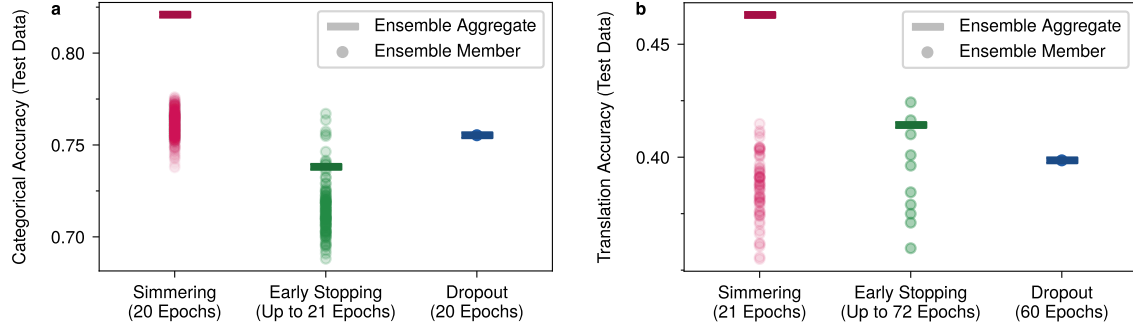


Figure 3: Simmering outperforms early stopping, ensembled early stopping and dropout on the CIFAR-10 (Krizhevsky & Hinton, 2009) (panel a) and Portuguese-English (PT-EN) TED talk transcript translation (Ye *et al.*, 2018) (panel b) datasets. For both problems, simmering’s ensemble prediction (rectangular marker) achieves both the highest accuracy and the most significant ensembling improvement (rectangular marker vs. round markers), with the latter indicating that the advantage of simmering extends beyond just the act of ensembling. In contrast, the early stopped ensemble accuracy (rectangular marker) does not exceed that of its ensemble members (round markers) for both training tasks. For training on the CIFAR-10 dataset, all four learning cases employed the ConvNet architecture (Krizhevsky, Sutskever, *et al.*, 2012), and the standard train/test split. For the dropout case, we applied the dropout rates reported in Ref. (Srivastava *et al.*, 2014). We used TensorFlow’s implementations of early stopping and dropout, paired with stochastic gradient descent (Martín Abadi *et al.*, 2015) to optimize the early stopped and dropout-equipped models. The early stopping ensemble consists of 100 independently trained early stopped models, where the average training time was 14.56 epochs. Dropout and ab initio simmering each trained for 20 epochs, and we included the models corresponding to the last 2000 weight updates in the simmering ensemble. At the testing stage, since it has no explicit ensemble to aggregate, we took dropout’s “inference mode” prediction to be its ensemble prediction. (Hara *et al.*, 2016) We used majority voting to aggregate the early stopping and simmering ensembles. For learning the translation task, we trained a reduced version (described in Supplementary Methods) of the Transformer architecture presented in Ref. (Vaswani *et al.*, 2017) with a pre-trained BERT tokenizer from TensorFlow’s model database, and assessed translation accuracy via teacher-forced token prediction accuracy. We used the same batch size and fixed learning rate for all cases, and used the Adam optimizer (Kingma & Ba, 2017) for all the non-simmering training scenarios. The early stopping ensemble consists of 10 independently trained models, with an average training time of 53.1 epochs, aggregated with majority voting as for CIFAR-10. Based on the average early stopped training time, we trained a model with dropout for 60 epochs and determined its “ensemble test accuracy” in the same way as for CIFAR-10. We found that the simmering ensemble exceeded the test accuracy of all other cases in only 21 training epochs, with a majority-voted ensemble prediction derived from 200 models sampled during the last epoch. Full accuracy convergence curves for both training tasks are shown in Supplementary Figures 1–2, and additional details on comparison implementation are described in Supplementary Methods. For equivalent training time, ab initio simmering produces more accurate predictions than other ensembled overfitting mitigation techniques for the CIFAR-10 dataset. However, simmering can also both accelerate training and exceed the accuracy of other overfitting techniques in a natural language processing task like language translation.

2. From Supplementary Figure 1, it seems that Simmering produces more "diverse" models that works better in an ensemble than traditional early stopping. Although this may be beyond the scope of current paper, it may be interesting to explore more along this direction.

The Referee has drawn attention to a very important aspect of our work, and we appreciate the opportunity to elaborate. The fact that simmering produces a more “diverse” ensemble than achievable by independently optimized, early-stopped models is not a coincidence – simmering leverages information theory to generate the most representative ensemble possible at a given temperature.

The aim of this work is to present a generically generalizable neural network training method. Other methods for improving generalizability are case-dependent because they impose additional restrictions on the parameter distribution, e.g., L_2 -norm regularization encourages smaller-magnitude parameters, which may not be a generically suitable restriction on parameter magnitude for all contexts. In contrast, the principle of maximum entropy (Jaynes, 1957) generates the minimally biased parameter distribution, i.e. is most uncertain about the parameter distribution, subject to all available knowledge about the parameters. Since this distribution is the most uncertain way to model the neural network parameters, it is also the least restrictive on what parameter values are sampled, and thus yields the most diverse representation of the available knowledge about the process being modelled.

To ensure that this important property of simmering is conveyed strongly in the paper, in addition to the existing description of entropy maximization in Main Methods, we have replaced Main Table 1 with Main Figure 3 which displays the effect of this diverse ensemble generation. See response to the previous comment for the extracts from the manuscript that address this.

3. In Supplementary Figure 2, Simmering converges much faster than the baseline method on Transformers, where it only uses 20 epochs where the baseline takes 60 epochs to reach the same level of performance. If this result is true, it could be more interesting the rest of the paper combined. Why is this convergence speedup not explored more in the paper? I find it hard to believe that it converges so much faster.

The Referee raises an interesting point about our Transformer training comparison.

The goal of this work is to present a generically generalizable training method. As such, the comparison between different overfitting mitigation methods is intended to highlight the distinctive features of simmering that facilitate generalizability.

Since accelerating convergence was not the aim of this work, we elected not to focus on the effect of simmering on the rate of convergence when presenting these results. However, we see that temperature can positively affect the rate of convergence, and we expect to explore this further in subsequent work.

4. Main text Figure 3 about classification confidence, on the other hand, is much less interesting since it contains merely one illustrative example. It is hard to conclude anything from one example. If a more statistical measure is beyond the scope the paper, I suggest reducing the emphasis on this, by perhaps moving this one to the supplementary materials and add the Supplementary Figure 1 and 2 to the main text, together with related analyses.

We thank the Referee for their suggestion regarding the presentation of the classification confidence example. We have moved the figure to the Supplementary Information (now Supplementary Figure 3).

(Remarks on code availability):

I looked at the original code in the initial review and it ran without problem. I have NOT yet checked the latest code about CIFAR10 and transformers (if any).

We appreciate the Referee’s initial review of our code. We would like to note that since that initial review, the only changes to the code were substitutions of the neural network architecture, dataset, and simmering hyperparameters to correspond to the CIFAR-10 training problem (Figure 3) rather than the noisy sine curve problem (Figure 1). The current version of the code can be run without modification to train a neural network on CIFAR-10.

We thank the Referee for their careful examination of the work. We believe that Referee’s suggestions have promoted changes in the revised manuscript that substantially improve its presentation.

Report of Referee 2

My main concerns have been addressed, but I have one further suggestion. Although this is an interdisciplinary work, the proposed method is ultimately aimed at training neural network models. Therefore, the notations and pseudocode in the paper should follow the conventions of neural network training algorithms, rather than those of thermodynamics. This will enable researchers in the artificial intelligence field to better understand and utilize the proposed method.

We thank the Referee for their suggestion for our method presentation. The Referee is correct that some of the notation we used may be unfamiliar to some members of the artificial intelligence research community. However, in our reading, the equivalent terminology (in entirely artificial intelligence convention) is not employed consistently in analogous contexts. Using physics notation allows us to present our method in a cohesive way. However, to aid in the interpretability of our methodology, we have prepared a table that describes each physics symbol we employ in both Main Methods and Supplementary Methods, alongside one or more machine learning (ML) analogues.

We have also modified the “Training Procedure” of the Supplementary Information to provide additional context for the physics-motivated aspects of the training procedure.

We refer to the addition of the table in Main Methods:

...Starting from entropy maximization, we show how sufficient training generically uses the geometry of the loss landscape to deviate from optimal training loss during sampling, and how this deviation is controlled by the temperature T . For additional information about our notation, see Supplementary Methods.

Symbol	Physical Terminology	Equivalent ML Terminology
$\mathbf{x} \in \mathbb{R}^N$	Full parameter vector (weights and biases)	θ (trainable parameters)
$L(\mathbf{x}, \mathcal{D})$	Training loss evaluated on data set $\mathcal{D}$	$\mathcal{L}(\theta; \mathcal{D})$
$T = 1/\beta$	Temperature	Temperature (Hinton <i>et al.</i> , 2015; Wenzel <i>et al.</i> , 2020)
$p(\mathbf{x} \mid \mathcal{D})$	Canonical distribution over parameters	Tempered posterior (Wenzel <i>et al.</i> , 2020) over weights
$\theta(\mathbf{x}, \mathcal{D}) \in \mathbb{R}^n$	Collective variables / collective coordinates	Latent parameter; low-dimensional embedding of models (Kingma & Welling, 2022)
$e^{-\beta F(\theta, \mathcal{D})}$	Effective free energy	Energy in latent parameter space
$p(\theta \mid \mathcal{D})$	Distribution over collective variables	Tempered posterior (Wenzel <i>et al.</i> , 2020) over latent parameters
x_i, v_i, m_i	Real particle position, velocity, mass (particle index i)	Parameter value θ_t , momentum m_t (Kingma & Ba, 2017), per-param scale
s_k, v_{s_k}	Virtual thermostat particle position, velocity (chain index k)	Auxiliary thermostat states ξ , momentum $\mathbf{P}$ (Ding <i>et al.</i> , 2014)
$a_i(t)$	Real-particle acceleration	Preconditioned / scaled gradient
$a_{s_1}(t)$	First thermostat acceleration	Temperature control term
$a_{s_k}(t)$	Chain thermostat accelerations ($k \geq 2$)	Coupled control terms
Q_k	Thermostat masses	Hyperparameters controlling response time
N_{NHC}	Length of the Nosé–Hoover chain	Thermostat depth
T_{target}	Target thermostat temperature	Set-point for temperature
T_{real}	Measured kinetic temperature	Empirical Temperature / Scaled Momentum Variance
Δt	Integration time step / learning rate	Step size η
$v_{i,0} \leftarrow \sqrt{T/m_i}$	Velocity initialization	Momentum initialization consistent with T (Ding <i>et al.</i> , 2014)
t, t_{max}	Iteration index and maximum iterations	Training step k / total steps K

Table 1: Symbols used to describe simmering and its implementation in Methods and Supplementary Methods, and their analogues in machine learning (ML) contexts.

The Supplementary Methods itself now reads,

Table 1 describes the notation used in Main Methods and Supplementary Methods. For each symbol, we describe its physical meaning (in statistical physics literature) and its equivalent or analogous notation in machine learning (ML) literature.

...

Algorithm 1 describes the implementation of simmering used to generate the results presented in this work. There are two main “stages” of simmering: the equilibration stage, and the model sampling stage. The purpose of this algorithm is to sample models from the canonical distribution of models at a given temperature, and the equilibration stage ensures that the model parameter distribution reaches thermal equilibrium, i.e., is distributed according to the canonical distribution, before models are sampled. The main simmering hyperparameters are the thermostat temperature T , the particle mass m , and the learning rate Δt . The thermostat temperature T can be either a constant or a schedule that depends on training time t . Model sampling can commence once the average parameter temperature T_{real} over time is approximately equal to the thermostat temperature T . Note that the precision with which the neural network parameter temperature will fluctuate about T depends on the neural network parameter space dimensionality, so expect that T_{real} will fluctuate with a larger amplitude for neural networks with less parameters. For our training experiments, all real and virtual particles had the same mass, i.e., $m_i = Q_k \forall i, k$, based on Ref. (Martyna *et al.*, 1992). Generally, decreasing the mass makes the parameter updates more sensitive to the loss gradient (see m dependence in line 20) and thus can accelerate equilibration. The learning rate Δt is fixed and identical for all particles (neural network parameters and NHC particles). We also note that we used a NHC length of 10 for all experiments.

To commence equilibration, provide the initial weights and biases of the neural network as $x_{all,0}$, and proceed with Algorithm 1. During training, before executing line 20 of the for-loop in Algorithm 1, set the neural network parameters to $x_{all,t+1/2}$, and use backpropagation to compute the gradient of the loss $\nabla_x L$. Once equilibration is complete, reuse the same algorithm for sampling models. The only difference in execution is that the model parameters and/or predictions generated at the loss gradient computation stage (line 18 of Algorithm 1) are saved for offline ensemble analysis. There is no prescription for sampling frequency. However, some options for sampling frequency include saving every iteration’s predictions, saving the predictions at a fixed iteration interval, or saving randomly. If sampling at some fixed interval, check that the samples are uncorrelated at that interval. After the desired number of samples have been collected, the set of models and/or predictions can be aggregated or analyzed as a distribution of solutions in the ways described in the main text.

Algorithm 1 Simmering algorithm for a generic learning problem described by a differentiable objective function L which has parameters x , based on the NHC thermostat dynamics integration in Ref. (Jang & Voth, 1997). In this integration scheme, particle positions are integrated over one half-step, i.e., half of the learning rate, at a time, and corresponding velocities are integrated forward by a full time-step at once. The objective function parameters (e.g., neural network parameters) are treated as the positions of N one-dimensional particles which evolve over time according to Supplementary Equations 1–9. In this algorithm description, the first subscript of each variable indexes the parameters (i for the i^{th} parameter, k for the k^{th} even or odd NHC variable), and the second subscript indicates the timestep of the associated variable. Note that since the NHC particles are integrated differently based on their chain position (even or odd), the virtual particle index takes on values $k = 1, \dots, N_{NHC}/2$. Unless otherwise indicated (e.g., with a particle index of *all*), operations are performed element-wise for all elements in the given vector(s). If no initialization is specified, assume that a vector’s initial value is 0 for all elements.

Require: $T > 0$: Thermostat temperature

Require: Δt : Learning rate

Require: $L(x)$: Objective function L with parameters x

Require: $x_{all,0}$: Initial neural network parameter vector, of dimension N

Require: m : Neural network parameter mass vector, of dimension N

Require: t_{max} : Number of iterations

Require: Q : NHC particle mass vector, of dimension N_{NHC}

```

1: function POSITION_FORWARD( $a, b, dt$ )           ▷ Integrator for parameters  $x_i$  and positions  $s_{2k}, s_{2k-1}$ .
2:   return  $a + dt \cdot b$ 
3: function VELOCITY_FORWARD( $a, b, c, dt$ )       ▷ Integrator for velocities ( $v_i, v_{s_{2k}}, v_{s_{2k-1}}$ ).
4:   return  $a \exp\{-dt \cdot c\} + dt \cdot b \cdot \exp\{-dt/2 \cdot c\}$ 
5: function ACCELERATION_S1( $v, m, T, N, Q_1$ )    ▷ Acceleration of the first NHC particle ( $a_{s_1}$ ).
6:   return  $(\sum_j m_j v_j^2 - N \cdot T)/Q_1$ 
7: function ACCELERATION_SK( $v, T, b, c$ )        ▷ Acceleration of all other NHC particles ( $a_{s_{2k}}, a_{s_{2k-1} \neq 1}$ ).
8:   return  $(c \cdot v^2 - T)/b$ 
9: function REAL_T( $v, m$ )                       ▷ For computing the parameter temperature.
10:  return  $(\sum_j m_j v_j^2)/N$ 
11:  $v_{i,0} \leftarrow \sqrt{\frac{T}{m}}$                      ▷ Initialize velocity vector elements.
12:  $T_{real} \leftarrow T$                          ▷ Define initial parameter temperature.
13:  $t \leftarrow 0$                                 ▷ Initialize timestep.
14: while  $t < t_{max}$  do
15:    $x_{i,t+1/2} \leftarrow \text{POSITION\_FORWARD}(x_{i,t}, v_{i,t}, \Delta t/2)$            ▷ First half-step update to  $x_i$ .
16:    $s_{2k,t+1/2} \leftarrow \text{POSITION\_FORWARD}(s_{2k,t}, v_{s_{2k},t}, \Delta t/2)$        ▷ First half-step update to  $s_{2k}$ .
17:    $a_{s_1,t} \leftarrow \text{ACCELERATION\_S1}(v_{all,t}, m_{all}, T, N, Q_1)$ 
18:    $a_{s_{2k-1} \neq 1,t} \leftarrow \text{ACCELERATION\_SK}(v_{s_{2(k-1)},t}, T, Q_{2k-1}, Q_{2(k-1)})$ 
19:    $v_{s_{2k-1},t+1/2} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{2k-1},t}, a_{s_{2k-1},t}, v_{s_{2k},t}, \Delta t/2)$    ▷ First half-step update to
    $v_{s_{2k-1}}$ .
20:    $a_{i,t+1/2} \leftarrow -\nabla_{x_i} L(x_{all,t+1/2})/m_i$                        ▷ Use loss gradient here!
21:    $v_{i,t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{i,t}, a_{i,t+1/2}, v_{s_1,t+1/2}, \Delta t)$    ▷ Full-step update to  $v_i$ .
22:    $s_{2k-1,t+1} \leftarrow \text{POSITION\_FORWARD}(s_{2k-1,t}, v_{s_{2k-1},t+1/2}, \Delta t)$    ▷ Full-step update to  $s_{2k-1}$ .
23:    $a_{s_{2k},t+1/2} \leftarrow \text{ACCELERATION\_SK}(v_{s_{2k-1},t+1/2}, T, Q_{2k}, Q_{2k-1})$ 
24:    $v_{s_{2k \neq N_c},t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{2k},t}, a_{s_{2k},t+1/2}, v_{s_{2k+1},t+1/2}, \Delta t)$    ▷ Full-step update to
    $v_{s_{2k}}$ .
25:    $v_{s_{N_c},t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{N_c},t}, a_{s_{N_c},t+1/2}, 0, \Delta t)$    ▷ Full-step update to  $v_{s_{N_c}}$ .
26:    $x_{i,t+1} \leftarrow \text{POSITION\_FORWARD}(x_{i,t+1/2}, v_{i,t+1}, \Delta t/2)$            ▷ Last half-step update to  $x_i$ .
27:    $s_{2k,t+1} \leftarrow \text{POSITION\_FORWARD}(s_{2k,t+1/2}, v_{s_{2k},t+1}, \Delta t/2)$        ▷ Last half-step update to  $s_{2k}$ .
28:    $a_{s_1,t+1} \leftarrow \text{ACCELERATION\_S1}(v_{all,t+1}, m_{all}, T, N, Q_1)$ 
29:    $a_{s_{2k-1} \neq 1,t+1} \leftarrow \text{ACCELERATION\_SK}(v_{s_{2(k-1)},t+1}, T, Q_{2k-1 \neq 1}, Q_{2(k-1)})$ 
30:    $v_{s_{2k-1},t+1} \leftarrow \text{VELOCITY\_FORWARD}(v_{s_{2k-1},t+1/2}, a_{s_{2k-1},t+1}, v_{s_{2k},t+1}, \Delta t/2)$    ▷ Last half-step
   update to  $v_{s_{2k-1}}$ .
31:    $T_{real} \leftarrow \text{REAL\_T}(v_{all,t+1}, m_{all})$    ▷ Optional: Track parameter temperature throughout sampling.
32:    $t \leftarrow t + 1$ 

```

(Remarks on code availability):

NA

We thank the Referee for their suggestion regarding method presentation. We believe the resulting changes allow our work to be more easily interpreted by members of the artificial intelligence community.

Manuscript ID: **NCOMMS-25-24798-C**
Title: **Simmering: sufficient is better than optimal for training neural networks**
Authors: Irina Babayan, Hazhir Aliahmadi, Greg van Anders

Response to Referee Reports

Thank you for your communication and efforts in editing and collating these reviews. We also thank both Referees for their feedback on the manuscript.

The reports are reproduced below in their entirety. The quotations are verbatim and indented in blue, sans-serif font.

We hope that you will find the manuscript suitable for publication in *Nature Communications*.

Report of Referee 1

My suggestions and concerns have been addressed in the latest manuscript. The results presented in the paper are interesting and convincing to me.

(Remarks on code availability):

It would be nice if the author can provide the source code for the transformer based NLP experiment.

We thank the Referee for their careful examination of the work, and appreciate their interest in the transformer-based NLP experiment.

For the open-source implementation of simmering, we opted to present the CIFAR-10 training problem as the use case, because it is feasible to run on hardware that is generally accessible, whereas transformer training requires specialized equipment. Nonetheless, the code that we have published can be applied to NLP tasks by replacing the neural network architecture, simmering hyperparameter values and dataset used in the sample code. In Supplementary Methods (section Comparison With Other Overfitting Prevention Methods), we describe the transformer architecture, simmering hyperparameters, and training procedure (e.g., batching, number of epochs) needed to reproduce our NLP results. In this Supplementary Methods section, and in the main text, we also link to the open-source dataset we used for training. We believe that the description we provided is sufficient to reproduce the transformer-based experiment.

Report of Referee 2

My concerns have been addressed

We thank both Referees again for their feedback on current and previous versions of the manuscript.