Smart-Init of neural networks

1st David Denisov
dept. Computer Science
University of Haifa
Haifa, Isreal
daviddenisov14@gmail.com

2nd Dan Feldman
dept. Computer Science
University of Haifa
Haifa, Isreal
dannyf.post@gmail.com

3rd Shlomi Dolev
dept. Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Isreal
dolev@cs.bgu.ac.il

4th Michael Segal
dept. Communication Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, Isreal
segal@bgu.ac.il

***Abstract*—The initialization of neural networks is of significant importance for their performance. Currently, the prevalent initialization method is a random sample based on the network's structure. This work presents a general yet effective method to initialize neural networks. We provide repeated experiments of training variants of Mobile-Net over down-sampled variants of Image-Net, demonstrating accuracy gain and loss decrease across most of the test sets and validation sets. E.g., for Mobile-Net (v1) and Image-Net (32×32), we had 2.5% accuracy improvement over both the test and validation sets.**

***Index Terms*—Neural networks, Neural networks initialization, Optimization, Non-convex optimization**

I. Introduction

It would be an understatement to say that classification problems had a major interest over recent years, where neural networks (NNs) account for significant interest in themselves. The initialization of neural networks has a major effect on their performance and training time as previously demonstrated in [1]. Nonetheless, the current prevalent initialization method for NN is essentially a random sample based only on the network's structure as in [1].

We suggest a novel initialization method for NNs and demonstrate its practical improvement over the commonly used initialization method. For example, for Mobile-Net (Version 1) [2] and the 32×32 Down-Sampled variant of Image-Net [3], for both the validation and test set we had 2.5% accuracy improvement. Moreover, this initialization process took only 2% of the network's original training time, and even when accounting for this additional preprocessing time we had 2.5% accuracy improvement. Through this paper, we ran all the tests on a PC with 32 GB of RAM, 12400f CPU, and 1660 (Super) GPU.

A. Suggested method and motivation

Our suggested method for initializing NNs is as follows.

Method 1: (i). Initialize the network as commonly done in the literature, e.g. [1].
(ii). Train the network 10 epochs on a uniform sample (without repetitions) of 1.25% from the data.
(iii). Re-train the network 10 epochs on a different uniform sample (without repetitions) of 1.25% from the data.
We suggest resetting all the hyper-parameters after steps (ii) and (iii).
We suggest calibrating further the values stated.

This is motivated by the samples yielding a "simpler" loss function (e.g. cross-entropy), and thus their solution is less prone to local minima. Hence, gradient descent (or similar methods) from the solution over the sample can (hopefully) avoid local minima in the optimization process.

We illustrate our motivation by applying gradient descent via `Scipy.optimize.minimize` from the Scipy library [4] over the Ackley function [5]. In the following plots, where each surface is a triangulated mesh, the x and y values are vertices of a grid that covers the rectangular area $[-1.5, 1.5] \times [-1.5, 1.5]$ and for each point, $(x, y) \in \mathbb{R}^2$ on the grid the z-value (as per the Ackley-function) is mapped

$$f(x,y) := 20 + e - 20e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5\left(\cos(2\pi x)+\cos(2\pi x)\right)}.$$

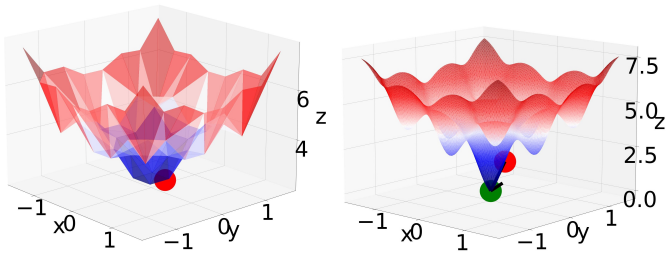

Fig. 1. **(Left - simplified function, 64 points)** The red dot is an approximation to the global minimum of the function defined by the mesh, which was found by using gradient descent from a uniformly at random sampled point on the mesh. **(Right - original function, 10,000 points)** The red dot is the projection of the previous solution corresponding to the left plot (only the z value changes), and the green point is an approximation to the global minimum, which is found by applying gradient descent from the red dot.

As seen from the right plot in Figure 1, applying gradient descent from a randomly sampled point would (most likely) perform poorly on the Ackley function due to being stuck in one of the various local minimums. On the other hand, applying gradient descent using the proposed method would (at least most likely) succeed in reaching the global minimum.

B. Related work and data used

Our method is similar to warm starting a network [6], that usually, to the best of our knowledge and as done in [6], entails training until convergence once over a subset of the data. Our method suggests training in two steps, and not until convergence. This yields an improvement in the result, unlike warm starting a network, which, as demonstrated in [6], usually results in worse performance over the validation and test set.

Another similar initialization method is curriculum learning [7], where the data is gradually revealed to the network.

To this end, the method requires a "teacher" (e.g., a neural network) to choose which data to reveal. Our method does not require such "teacher", and demonstrates noticeable improvement utilizing uniform samples from the data. Nonetheless, we believe that incorporating our method and curriculum learning can improve the results even further.

Through this paper, we will use Down-Sampled variants of Image-Net as proposed in [3] which observed accuracy decreases when decreasing the image sizes. Our results are no exception to this, and it should be emphasized that the neural networks used obtain significantly better accuracy on the original Image-Net [8], as noted in [2].

Nonetheless, we have used the Down-Sampled variants since they allowed preserving the complex nature of the classification task in Image-Net while reducing the large computational burden, unlike Cifar10 [9] or Cifar100 [9] that while computationally efficient result in tasks that are noticeably less complex as evidenced by the high accuracy of various methods following 2017th 3%-error obtained by [10].

Throughout the paper, besides Section IV, we have used [11] (specifically the TopImages option) as our test set; in Section IV we trained on the Cifar10 set [9], and as such used the corresponding test set.

II. Experimental results

In the following test, we train *Mobile-Net (version 1)* from [2] (implementation taken from https://github.com/jmjeon94/MobileNet-Pytorch) on Down-Sampled variants of Image-Net [3]; source code provided at [12]. We have optimized the model with AdamW [13] over the cross-entropy loss in batches of size 64, with an initial learning rate set to 0.001 with an exponential decay rate of 0.925 for all the data sets, besides 16×16 images, where, to improve performance, we have changed the decay rate to 0.9375. For $16\times16, 32\times32, 64\times64$ images, we have trained for $75, 50, 35$ epochs correspondingly, which suffices for convergence over the validation set as shown in Figure 2.

Our method, named in Table I *Smart-Init*, entails taking a uniform random sample (no repetitions) of 1.25% of the data and training over it 10 epochs (exactly as done for the entire data set). Then repeat the process for a sample of 12.5% of the data. To reduce noise, we have initialized our network (before our pre-training steps) with the same weights as the comparison *Original* network, which are taken with the values sampled as default by PyTorch [14]. To account for the additional running time of *Smart-Init* we set the first epoch of *Original* network as preprocessing, such that both of the methods would have equal preprocessing time.

Our results are summarised in Table I, where the results are averaged across 4 tests, along with the *s.t.d.* (standard deviation) over the 4 tests. Note that, as table 1 of [3] suggests, the results for the dataset used are lower than common for the original Image-Net [8]. Nonetheless, our results for the Original initialization method are in the range expected based on Table 1 of [3].

In Figure 2 we plot the results for the validation and test sets for 32×32 images. The lines are the median results and the error bands represent the range of the results across the 4 tests, we plot the results only after the fifth epoch to remove initial values that are far from the later values.

For 16×16 and 32×32 images, our results are considerably and consistently better, and the results (between the methods) form almost distinct sets, i.e., the sets are distinct besides few intersections where the worst result of our method is within the range of the results of the original method. For 64×64 images we had significant over-fitting (as suggested by the epochs of the best results over the validation set), as a consequence there was essentially no accuracy or loss improvement. Our results for all the validation sets with our method are above the ones for the smallest corresponding models from [3], that while smaller in size (see sizes in [3] and [2]) are more computationally expensive as mentioned in the comparison to ResNet (that WRN [15] uses) in [2].

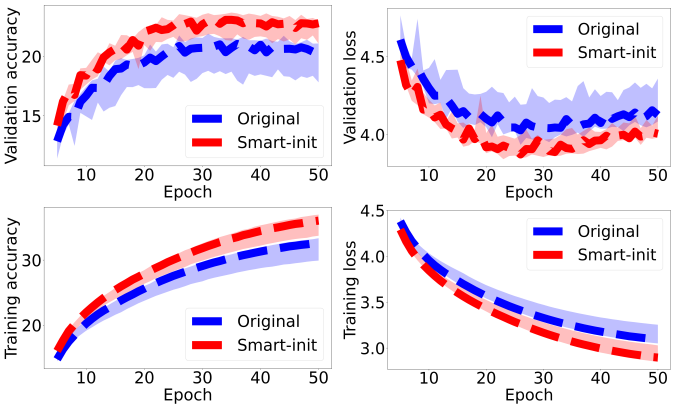

Fig. 2. Results for the validation (top row) and training (bottom row) set for 32×32 images in Section II.

TABLE I
RESULTS FOR THE TESTS AT SECTION II.

Test set	Image size 16x16		Image size 32x32		Image size 64x64	
Value	Original	Smart-Init	Original	Smart-Init	Original	Smart-Init
Loss (mean)	4.571	4.478	4.173	4.027	3.136	3.324
Loss s.t.d.	0.036	0.014	0.11	0.054	0.02	0.017
Accuracy (mean)	15.445	16.5	20.91	23.46	39.225	38.475
Accuracy s.t.d.	0.734	0.286	1.041	0.83	0.246	0.604

Validation set	Image size 16x16		Image size 32x32		Image size 64x64	
Value	Original	Smart-Init	Original	Smart-Init	Original	Smart-Init
Loss (mean)	4.567	4.474	4.029	3.88	2.887	2.894
Loss s.t.d.	0.036	0.013	0.101	0.05	0.005	0.011
Accuracy (mean)	14.844	15.954	20.798	23.045	38.868	38.994
Accuracy s.t.d.	0.585	0.227	1.175	0.718	0.082	0.114
Epoch taken (loss)	61.75	67	26.5	26	18.5	16.25

Training set	Image size 16x16		Image size 32x32		Image size 64x64	
Value	Original	Smart-Init	Original	Smart-Init	Original	Smart-Init
Loss	4.091	3.941	3.123	2.918	1.528	1.243
Loss s.t.d.	0.067	0.034	0.08	0.069	0.016	0.019
Accuracy	19.48	21.629	32.135	35.706	61.396	67.581
Accuracy s.t.d.	0.952	0.512	1.334	1.243	0.34	0.404
Epoch taken (loss)	74.75	74.75	50	50	35	35

A. Initialization repeat effect

In this section, inspired by the motivation at Section I-A, we test whether repeating the suggested initialization would improve the results by yielding better confidence to avoid the bad local minima. To this end, we repeat the training from Section II for 32×32 images, with the change that we initialize the network by 10 times taking a sample of 12.5% of the data and training the network 10 epochs over the sample. The results, over 4 test repeat as in Table I, are summarised in Table II. Due to the Original initialization method being already included in Table I, we present the results only for the current method.

TABLE II
RESULTS FOR SECTION II-A.

Value	Validation set	Training set
Loss (mean)	3.811	2.789
Loss s.t.d.	0.052	0.089
Accuracy (mean)	24.306	37.79
Accuracy s.t.d.	0.661	1.551
Epoch taken (loss)	26.5	50

The initialization repeats have indeed yielded significant improvement over our suggested initialization at Table I (which had outperformed the Original initialization method). However, this improvement came at the price of a significantly slower initialization process, which for this test took around 25% of the training time without the initialization, while previously the initialization took around 2%.

III. COMPARISON REGARDING WARM START

While our method is similar to the warm start, our initialization has improved the results. Unlike [6], where, for ResNet-18 [16], warm start yielded significantly worse results that were mitigated with the method proposed in the paper.

Inspired by [6], we hypothesize that warm start in [6], which entailed training until convergence over a 50% sample from the data, has yielded overfitting over the sample, which was not mitigated when training over the full data set. However, since we considered a more difficult problem compared to [6] the problem of overfitting was less prevalent, and instead, the original initialization method resulted in underfitting that our method could mitigate. This is consistent with our motivation in Figure 1, where the goal of our method is to find a better local minimum over the loss function. In the case of underfitting, a lower loss on the training set would likely translate to better results on the test and validation sets as well, however, for overfitting the opposite is true.

To validate our hypothesize we repeat a variant, which we call *Warm-start*, of the initialization in [6] where we run the test from Section II for 32×32 images, but with the difference that we sample 50% of the data and train 50 epochs. The results are summarised in Table III.

Note that while the results are similar to Table I the initialization process took 25 times longer, which is twice as long as in Table II that yielded significantly better results.

TABLE III
RESULTS THE VALIDATION SET OF SECTION III.

Value	Original	Warm-start
Loss (mean)	4.077	3.88
Loss s.t.d.	0.046	0.023
Accuracy (mean)	20.457	22.98
Accuracy s.t.d.	0.567	0.438
Epoch taken (loss)	26.25	28

A natural question is how the network behaves in our proposed initialization process. To this end, we repeat our initialization from Table I for 32×32 images, but with the difference that we sample 12.5% and 50% of the data and train 50 epochs, and repeat 40 times (to normalize the results) for 12.5% and 4 times for 50%. We save the accuracy and loss (cross-entropy) over the sampled set, the full training set, and the validation set. The lines are the median result and the error bands represent, for 12.5% the 25th and 75th percentile of the results, and for 50% the minimum and maximum.

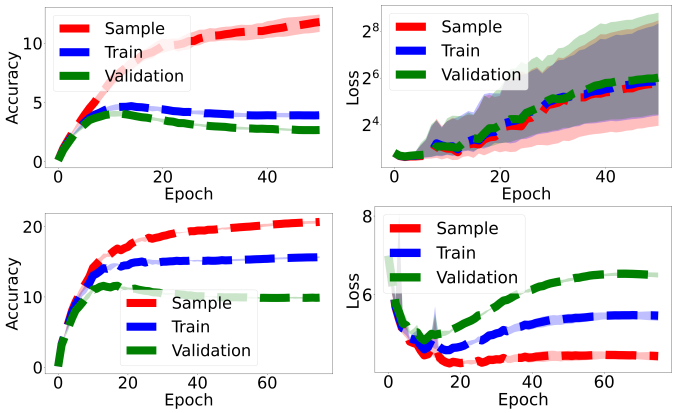

Fig. 3. The effect of training over the sample done in Section III. **Top:** Plots for a sample of 12.5% from the data for all the epochs, including epoch zero which is measured before any training. The loss is on a logarithmic scale due to its major increase across the epochs. **Bottom:** As the top plots but for a sample of 50% from the data, all the scales are linear.

In Figure 3, for a sample of 12.5% of the data the loss increased above 32 rather consistently, on all the sets, while the accuracy behaved more logically across the tests. For a sample of 50% from the data, all the sets initially improve and then begin to over-fit and the loss worsened over the validation and train set, to the point that the validation loss converged to only slightly lower value than before any training. This suggests, as we hypothesized, that the worsening of the results for warm start in [6] is due to over-fitting.

IV. Results for Cifar-10

In [6] it was shown that warm start for ResNet-18 [16], without the proposed mitigation proposed in the paper, yields around 8 percent accuracy decrees for the validation set. While we have demonstrated the effect of our initialization over a Down-Sampled variant of ImageNet [3], it is unclear if this effect is also presented with our initialization.

Therefore, in this test, we tested the effect of our initialization on Mobile-Net (version 1) over Cifar-10, where the data loading is as in https://gist.github.com/kevinzakka/d33bf8d6c7f06a9d8c76d97a7879f5cb. That is, we have normalized the RGB images to have means $(0.4914, 0.4822, 0.4465)$ and s.t.d. $(0.2023, 0.1994, 0.2010)$ across the channels consecutively, taken 10% of the training set as a validation set and augmented the training set via doing the following: (i). Random crop with padding equal 4 to size 32×32. (ii). Random horizontal flip that with probability half mirrors the images horizontally.

We optimized Mobile-Net (version 1) with AdamW [13] over the cross-entropy loss in batches of size 32, with an initial learning rate set to 0.001 with an exponential decay rate of 0.925 for 100 epochs; as demonstrated in Figure 4 this sufficed to obtain convergence of the networks.

Similarly to Section II, our initialization method, *Smart-Init*, entails taking a uniform random sample (no repetitions) of 1% of the data and training it over 10 epochs (exactly as done for the entire data set). Then repeat the process for a sample of 10% of the data. We also consider *Warm-start* initialization, where the network is initialized by training it 50 epochs over a uniform random sample (no repetitions) of 50% from the data. To reduce noise we have initialized all the networks (before the pre-training steps) with the same weights as the comparison *Original* network, which are taken with the values sampled as default by PyTorch [14].

We have run the test 40 times, and in Figure 4, we plot the results for the validation and train set. The lines are the median result and the error bands represent the 25th and 75th percentile of the results. We plot the results only after epoch 25 to remove initial values that are far from the later values.

TABLE IV
Results for the test set of the test at Section IV. We abbreviate Warm start to Warm.

Method	Original	Smart-Init	Warm
Loss (mean)	0.628	0.924	0.901
Loss (median)	0.589	0.6	0.71
Loss s.t.d.	0.113	0.795	0.493
Accuracy (mean)	80.12	79.754	79.108
Accuracy (median)	80.33	80.16	78.935
Accuracy s.t.d.	0.92	1.22	1.262

Note that while the colors are the same as in Figure 3, the colors denote unrelated values.

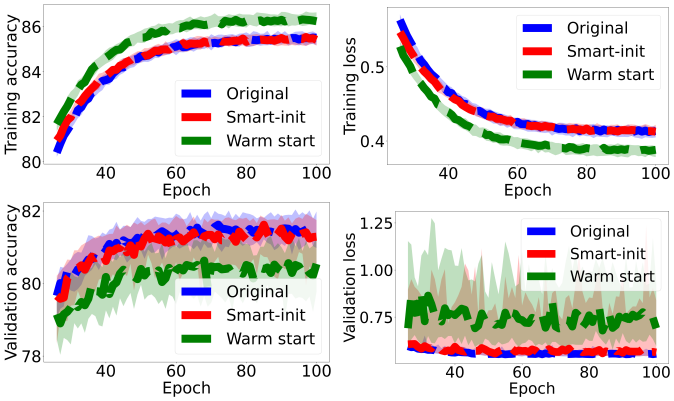

Fig. 4. Results for the validation and train set for Section IV.

Our method had essentially no effect on the median results but has caused an increase in the variance of the results, which in turn caused a slight worsening of the mean results as demonstrated in Table IV. Conversely, warm-start, while significantly less than in [6], yielded a worsening of the results over the validation and test set for both the mean and

median results. Moreover, our not-as-bad results for warm start compared to [6] are consistent with our previous hypothesis that the worsening of the results for warm start is due to over-fitting, which is less prevalent in our set-up due to the network being less computationally complex.

V. Results for other variants of Mobile-Net

Since the inception of *Mobile-Net (version 1)* at [2], a few variants of the model have been proposed. In this section, we will consider the following two notable variants: *Mobile-Net (version 2)* from [17], and *Mobile-Net (version 3-small)* from [18]. For both of those models, we use the official PyTorch [14] implementations from `torchvision.models` and repeat the tests from Section II for 32×32 images, where we increase (due to slower convergence of both models) the number of epochs to 75. Due to Mobile-Net (version 3-small) not converging even after the increase in the number of epochs, we have changed the exponential decay rate to 0.9; previously, it was 0.925. As in Section II, we refer to our suggested method as *Smart-init*, and the comparison one as *Original*, where both are as in Section II. We repeat the test 4 times, and present the results in Table V, and Figure 5. In Figure 5 the lines are the median result and the error bands represent the range of the results across the 4 repeats, and we plot the results only after epoch 25 to remove initial values that are far from the later values.

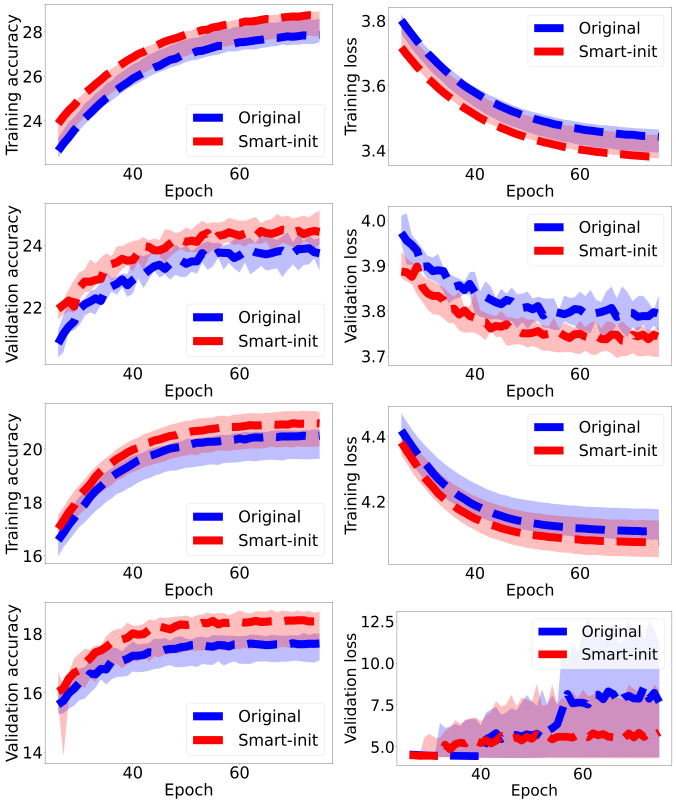

Fig. 5. Results for the train and validation sets, for the test in Section V. The top two rows correspond to Mobile-Net version 2 and the bottom two rows to Mobile-Net version 3-small.

TABLE V
Results for the test sets for Section V.
The top table corresponds to Mobile-Net version 2, and the bottom table to Mobile-Net version 3-small.

Version 2	Original	Smart-init
Loss (mean)	3.794	3.737
Loss s.t.d.	0.028	0.025
Accuracy (mean)	24.875	25.885
Accuracy s.t.d.	0.447	0.521

Version 3-small	Original	Smart-init
Loss (mean)	7.717	6.11
Loss s.t.d.	2.403	1.805
Accuracy (mean)	18.332	18.74
Accuracy s.t.d.	0.389	0.511

As demonstrated in Table V, for both models, our suggested method has yielded a significant improvement over the comparison initialization method that is commonly used.

The validation loss for Mobile-Net version 3-small at Figure 5 has increased noticeably across the epochs, thus we will not focus on this in our conclusion. For Figure 5 our method has yielded noticeable improvement as in Table V. This suggests that our method is not specific to a single network and a dataset, but generalizes (at the very least) to the Mobile-Net "family" of models and the Down-Sampled Image-Net "family" of datasets.

Conclusion and future work

We suggested a novel method to initialize neural networks and have demonstrated its practical use for various variants of the network Mobile-Net and the data-set Down-Sampled Image-Net. Our practical tests have demonstrated significant improvement over the commonly used initialization, without requiring a "teacher" as curriculum learning. We hope that our results will inspire better initialization of networks, which (to our knowledge) was less emphasized recently, and perhaps shed light on their training process.

Acknowledgment

David Denisov was (partially) funded by the Israeli Science Foundation (Grant No. 465/22). Shlomi Dolev was (partially) funded by the Israeli Science Foundation (Grant No. 465/22) and by Rita Altura trust chair in Computer Science. Michael Segal was (partially) funded by the Israeli Science Foundation (Grant No. 465/22), the Israeli Ministry of Science (Grant No. 0005355), and by the Army Research Office under Grant Number W911NF-22-1-0225. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the International Conference on Machine Learning*, vol. 28, no. 3, 2013.

[2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[3] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," 2017.

[4] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

[5] D. Ackley, *A connectionist machine for genetic hillclimbing*. Springer Science & Business Media, 2012, vol. 28.

[6] J. Ash and R. P. Adams, "On warm-starting neural network training," *Advances in neural information processing systems*, vol. 33, 2020.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *International conference on machine learnin*, 2009.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[9] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744

[10] X. Gastaldi, "Shake-shake regularization of 3-branch residual networks," *ICLR 2017, workshop submission*, 2017.

[11] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *International conference on machine learning*. PMLR, 2019.

[12] D. Denisov, "Open source code for this paper," 2025, https://github.com/DavidDenisov/Smart-Init.

[13] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

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

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[18] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.