A Convexity-Dependent Two-Phase Training Algorithm for Deep Neural Networks

Tomas Hrycej[1†], Bernhard Bermeitinger[2†][a], Massimo Pavone[1‡], Götz-Henrik Wiegand[1†][b] and Siegfried Handschuh[1†][c]

[1]*Institute of Computer Science, University of St.Gallen (HSG), St.Gallen, Switzerland*

[2]*Institute of Computer Science in Vorarlberg, University of St. Gallen (HSG), Dornbirn, Austria*

fi *fi*

Keywords: Conjugate Gradient, Convexity, Adam, Computer Vision, Vision Transformer.

Abstract: The key task of machine learning is to minimize the loss function that measures the model fit to the training data. The numerical methods to do this efficiently depend on the properties of the loss function. The most decisive among these properties is the *convexity* or *non-convexity* of the loss function. The fact that the loss function can have, and frequently has, non-convex regions has led to a widespread commitment to non-convex methods such as Adam. However, a local minimum implies that, in some environment around it, the function is convex. In this environment, second-order minimizing methods such as the Conjugate Gradient (CG) give a guaranteed superlinear convergence. We propose a novel framework grounded in the hypothesis that loss functions in real-world tasks swap from initial non-convexity to convexity towards the optimum — a property we leverage to design an innovative two-phase optimization algorithm. The presented algorithm detects the swap point by observing the gradient norm dependence on the loss. In these regions, non-convex (Adam) and convex (CG) algorithms are used, respectively. Computing experiments confirm the hypothesis that this simple convexity structure is frequent enough to be practically exploited to substantially improve convergence and accuracy.

1 INTRODUCTION

Fitting model parameters to training data is the fundamental task of Machine Learning (ML) with parameterized models. The sizes of the models have experienced extraordinary growth, recently reaching hundreds of billions. This makes clear that the efficiency of the optimization algorithm is of key importance. The optimization consists of minimizing an appropriate loss criterion such as *Categorical Cross-Entropy* (CCE), *Mean Squared Error* (MSE), or many other variants. These criteria are multidimensional functions of all model parameters. From the viewpoint of solvability, there are three basic classes of unconstrained minimization tasks according to the characteristics of the minimized function:

1. Convex functions
2. Non-convex functions with a single local minimum (which is also a global minimum)
3. Non-convex functions with multiple local minima

Non-convex functions are frequently referred to as a single group in the ML literature. This aggregation shadows a significant difference. In practical terms and for typical numbers of trainable parameters of current models, global minimization of a general function with multiple local minima is infeasible (see Section 2). By contrast, gradient descent can practically minimize non-convex functions with a single local minimum. Every descending path will reach the minimum with certainty if it is not trapped in singularities. For convex loss functions, the odds are even better. The classical theory of numerical optimization provides theoretically founded algorithms with a guaranteed convergence speed, also referenced in Section 2.

From the viewpoint of this problem classification, it is well known that loss functions with popular nonlinear models can possess multiple local minima, and thus count to the last class mentioned. Some of these minima are equivalent (such as those arising through permutations of hidden-layer units), but others may not. So, the paradoxical situation concern-

[a] https://orcid.org/0000-0002-2524-1850
[b] https://orcid.org/0009-0009-0392-056X
[c] https://orcid.org/0000-0002-6195-9034

Hrycej, T., Bermeitinger, B., Pavone, M., Wiegand, G.-H. and Handschuh, S.
A Convexity-Dependent Two-Phase Training Algorithm for Deep Neural Networks.
DOI: 10.5220/0013696100004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 1: KDIR*, pages 78-86

ing the training of nonlinear models is that methods are used that almost certainly cannot solve the problem of finding a global minimum. The implicit assumption is that the existence of multiple minima can be neglected in the hope that the concretely obtained local minimum is sufficiently suitable for the application. The positive experience with many excellent real models seems to justify this assumption. What remains is distinguishing between two former basic classes: convex functions and non-convex functions with a single minimum (further referred to simply as *non-convex*).

The fact that the loss functions of popular architectures are potentially non-convex has led to the widespread classification of these loss functions as non-convex. However, from a theoretical viewpoint, the loss function is certainly convex in some environment of the local minimum. This axiomatically results from the definition of a local minimum of any smooth function $L(x)$ by the gradient being zero:

$$\nabla L(x) = 0 \tag{1}$$

and the Hessian

$$H(x) = \nabla^2 L(x) \tag{2}$$

being positive definite, i.e., having positive eigenvalues. There, convex minimization algorithms are certainly worth using. This *guaranteed* convex region can optionally be — and frequently is — surrounded by a non-convex region.

From this point of view, the key question for algorithm choice is where the loss function is convex and where not. Although it is known that, in general, there may be an arbitrary patchwork of convex and non-convex subregions, a simpler, while not universally valid, assumption may exist that covers typical model architectures and application tasks. One such assumption is formulated in Section 3. In the next step, we will propose the appropriate optimization procedure accordingly (Section 4). If an assumption about a typical distribution of convexity is tentatively adopted, it is crucial to check how frequently this assumption applies in the spectrum of application problems. Although an extensive survey is not feasible due to resource limitations, experiments with a variety of typical architectures (with a focus on a Transformer and some of its simplified derivatives) are performed and reviewed to determine the validity of the assumption and the efficiency effect of optimization (Section 5).

2 RELATED WORK

The alleged infeasibility of minimizing functions with multiple local minima is based on algorithms available after decades of intensive research. Heuristics, such as momentum-based extensions of the gradient method, alleviate this problem by possibly surmounting barriers between individual attractors. Still, there is no guarantee (and also no acceptable probability) of reaching the global minimum in a finite time, since the number of attractors and boundaries between them is too large. Similarly, methods based on annealing or relaxation (Metropolis et al., 1953; Kirkpatrick et al., 1983) show asymptotical convergence in probability, but the time to reach some probabilistic bounds is by far unacceptable. Algorithms claiming complete coverage of the parameter space, like those based on Lipschitz constant bounds, or so-called clustering and Bayesian methods such as (Rinnooy Kan and Timmer, 1987; Mockus et al., 1997) are appropriate for small parameter set sizes less than ten.

By contrast, for non-convex functions with a single local minimum, every descending path will reach the minimum with certainty if not trapped in singularities. Today's algorithms, such as Adam (Kingma and Ba, 2015), focus on efficiency in following the descending path. There are convergence statements, for example, by (Fotopoulos et al., 2024; Chen et al., 2022). An interesting proposal for transforming a non-convex unconstrained loss function to a convex one with constraints is by (Ergen and Pilanci, 2023). However, this approach applies only to neural networks with one hidden layer and the ReLU activation function. A good option for covering both non-convex and convex regions would be second-order algorithms with adaptive reaction to local non-convexity, such as some variants of the *Levenberg-Marquardt* algorithm (Levenberg, 1944; Press et al., 1992). This algorithm is specific for least-squares minimization. It entertains a kind of "convexity weight" of deciding between a steepest gradient step and the step towards the estimated quadratic minimum. Unfortunately, the algorithm requires storing an estimate of the Hessian, which grows quadratically in the number of parameters, which makes it clearly infeasible for billions of parameters, even if using sparse Hessian concepts.

For convex loss functions, a numerical algorithm with a guaranteed convergence speed could be *nonlinear conjugate gradient method* (Fletcher and Reeves, 1964) and (Polak and Ribière, 1969). Both versions and their implementations are explained in (Press et al., 1992). They exploit the fact that convex functions can both be approximated quadratically. This quadratic approximation has an explicit minimum whose existence can be used to approach the non-quadratic but convex function minimum iteratively, with the guarantee of superlinear convergence.

3 CONVEX AND NON-CONVEX REGIONS OF LOSS FUNCTIONS

In this section, the hypothesis will be pursued that the following constellation characterizes the typical case: There is a convex region around the minimum, surrounded by a non-convex region. We are aware that this hypothesis will not apply to arbitrary tasks. However, if this were frequently the case in typical applications, it could be exploited for a dedicated use of first- and second-order algorithms, respectively.

A pictorial representation of the situation is given in Figure 1 showing the dependence of MSE on the scaling parameter p for a set of five random tasks with a single nonlinear layer tanh model (with 100 units)

$$y(x) = \sum_i \tanh(px) \tag{3}$$

and its square loss

$$L(x) = \left[y(x) - r\right]^2 \tag{4}$$

with reference values r of the output y randomly drawn from $(0,1)$. The set is generated for randomly selected input arguments x from $(-0.5, 0.5)$. Convexity around the minimum and non-convexity at margin areas can be observed.

A different view of the same five random tasks is the dependence of gradient norm on the loss, as depicted in Figure 2. The gradient norm is trivial in the one-dimensional case: it is the absolute value of the derivative. During optimization, the loss on the x-axis decreases (from the right to the left). The gradient norm (the y-axis) first increases (the non-convex region) and then decreases (the convex region) - this pattern can be observed for all five tasks. The two branches per task correspond to the different paths to the minimum (starting at the left or at the right margin, respectively, in Figure 1). It should be noted that there is no guarantee for this simple convexity pattern. Our hypothesis is that this pattern is frequently encountered and is not universally valid.

In the multidimensional parameter space, vertical cross-sections of a convex function are also convex so that the property of diminishing gradient norm is retained. This is also the case for steepest gradient paths, such as that given in the 2D plot of Figure 3; the level curves become successively less dense along the path. Of course, with an inappropriate step size, the optimization trajectory may contain segments with a temporarily increasing gradient norm if "climbing back the slope".

Real-world models are incomparably more complex. Theoretically, the patterns of non-convex regions may be alternating with intermediary convex segments, forming an arbitrary patchwork. This pitfall is analogous to those loss functions that can (and almost certainly) have multiple local minima, as mentioned in Section 1. Alternating convex and non-convex regions are, in fact, an *early stage of arising multiple local minima*. Observing a trivial two-layer network with the hidden layer

$$h(x) = \tanh(x) \tag{5}$$

and output layer

$$y(x) = \tanh(h(x)) + C\tanh(-2h(x)) \tag{6}$$

with a varying weight C, the loss function from Equation (4) will look like those in Figure 4. For $C = 0.40$, there is a single inner convex region. For $C = 0.45$ and $C = 0.50$, additional local convex regions (followed by a non-convex one) arise on the left slope. For $C = 0.55$ and $C = 0.60$, these convex regions convert to additional local minima.

However, the risk associated with an incorrect assumption about convexity is not as severe as in the case of one or multiple local minima. Using convex algorithms in a non-convex region is not disastrous: the only consequence is the loss of guarantee of superlinear convergence speed. A similarly moderate effect is using non-convex algorithms (e.g., Adam) in a convex setting. In this sense, it can nothing but be useful to commit to an optimistic assumption that

- the initial, usually random, parameter state is located in a non-convex region with a growing gradient norm and
- the boundary to the convex region is reached after the gradient norm decreases systematically

as in Figure 1. The expectation of a multidimensional loss function behaving approximately this way is not unreasonable, although not guaranteed. We will base our following considerations on this assumption and check how far they are encountered in real-world problems. Then, it is possible to approximately identify the extension of non-convex and convex regions in algorithmic terms. If the optimization algorithm is such that it produces a strictly decreasing loss (such as algorithms using line search), the entry to the convex region can be identified solely by detecting the point where the gradient norm starts its decrease. If loss fluctuations on the optimization path appear as in stochastic gradient methods, it is more reliable to observe the dependence of the gradient norm on the loss. In reality, both criteria may be disturbed by a zigzag optimization path in which the descent across loss-level curves does not always occur consistently. Then, some smoothing of the gradient norm curve has to be performed.

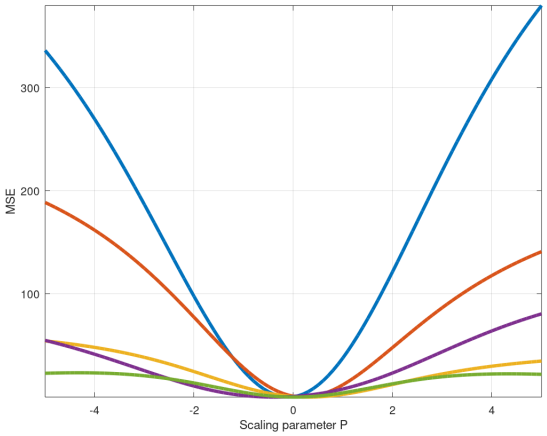

Figure 1: Loss functions of random trivial models.

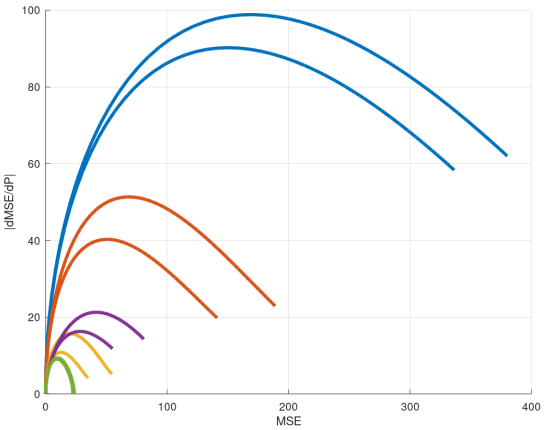

Figure 2: Dependence of the gradient norm on the loss for the random trivial models.

4 TWO-PHASE OPTIMIZATION

The basic hypothesis is as follows. Second-order numerical optimization methods, such as the *Conjugate Gradient* (CG) algorithm, can be assumed to be more efficient than first-order methods within the convex region. By contrast, the former methods offer no particular benefits in the non-convex regions. Then, sophisticated first-order methods (such as Adam) may be substantially more economical in their computational requirements because they use batch gradients. To do this, it is crucial to separate both regions during optimization. Following the principles presented in Section 3, the development of the gradient norm and its relationship with the loss currently attained can be used to detect the separating boundary.

The preceding ideas about gradient regions suggest a two-phase optimization formulated in Algorithm 1. Consistently with the hypothesis of non-convex and convex regions following the simple pattern depicted in Section 3, it is necessary to identify the point where the non-convex region transitions to the convex one. This point can be recognized with the help of an increasing or decreasing gradient norm. The swap point between the non-convex and convex regions is thus defined as the point where the increase changes to the decrease.

However, in practical terms, the computed gradient norm is contaminated by imprecision. In particular, the Adam algorithm with its batch-wise precessing delivers fluctuating values (as consecutive batches are different and thus show discontinuities). Gradient norms of the CG algorithm are nearly continuous, except for fluctuations caused by tolerances in the stopping rule of the line search. (This can be observed in Figure 5.)

This is why a practical rule to identify the swap point consists in setting a tolerance: a predefined gradient norm level below its peak value (here: 0.9).

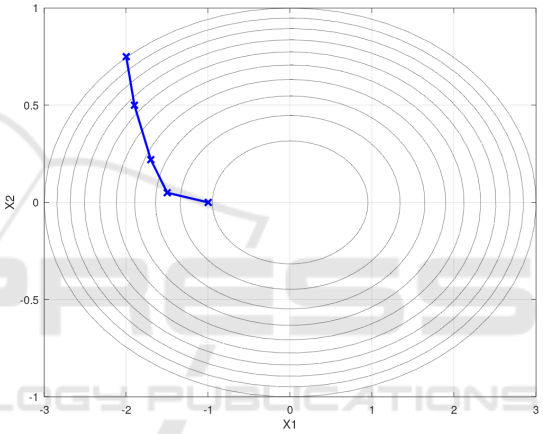

Figure 3: Gradient descent across level curves of a 2D parameter space.

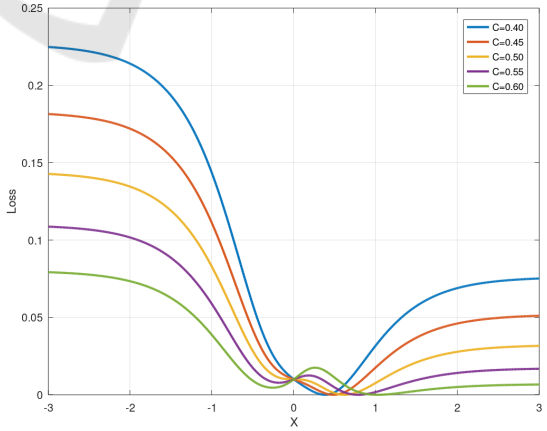

Figure 4: Loss function of a trivial model with two tanh layers, with various weights C.

The Adam algorithm was used for the first phase and CG with *golden line search* (Press et al., 1992) for the second phase.

Algorithm 1: Two-Phase Algorithm to switch from Adam to CG when the gradient norm peak has reached. Model and data are left out for brevity.

```
Data: nbEpochs > 1
adam ← true;
gnmax ← 0;
gnfact ← 0.9;
for epoch ← 1 to nbEpochs do
    if adam then
        ADAM();
        gn ← GETGRADIENTNORM();
        gnmax ← max(gn, gnmax);
        adam ← gn > (gnmax * gnfact);
    else
        CONJUGATEGRADIENT();
    end
end
```

CG has no meta-parameters except for defining a "zero" gradient norm and a tolerance for terminating the line search. In contrast, some tuning of Adam's meta-parameters is necessary to achieve good performance. The batch size is of particular importance. Some researchers argue that small batches exhibit lower losses for training and validation sets, e.g., (Keskar et al., 2017; Li et al., 2014; Chen et al., 2022). Consistent with this finding, in our experiments, batches greater than 512 elements have shown deteriorating performance (only integer powers of two have been tested). The convergence was very slow for batches exceeding 2048 elements (for even larger batches, even hardly discernible). However, batches smaller than 512 were also inferior. The performance of a batch size of 512 was good and robust for various variants of the models and has been used in further experiments. This size has, of course, only an experimental validity for the given datasets and models.

Whether this two-phase optimization is superior to conventional algorithms depends on the extension of the convex region. In general, this extension is not known. Theoretically, it might be too small for switching the algorithm to be profitable. In contrast, optimally converging algorithms may bring essential benefits in optimum quality and convergence speed. The alternative that prevails can only be investigated empirically.

5 COMPUTING EXPERIMENTS

Empirical support for a hypothesis must always be viewed with skepticism. Nevertheless, many statements about nonlinear models cannot be made in an ultimate theoretical way, making the resort to empirical investigation inevitable. Doubts about the validity will arise if the experimental settings do not represent the application domain. In today's world of very large models, scaling is difficult to cover, as most single experiments are not feasible with the means of many research institutions. We have focused on another aspect of particular relevance to the shape of the loss function and, thus, to the relationship between convex and non-convex regions: the variety of model architectures. As the most relevant model family based on transformers, a set of reduced transformer architectures, in addition to the full transformer, has been investigated. Furthermore, a different architecture has been used: the convolutional network *VGG5* (analogous to VGG architectures but with only five weight layers (Simonyan and Zisserman, 2015)). If the results are consistent with this set of architectures, the expectation that this will frequently be the case in practice is justified. The loss criterion has been the mean squared error (MSE) in all cases.

The first series of experiments examined small variants of the *Vision Transformer (ViT)* architecture (Dosovitskiy et al., 2021). These reduced variants consist of 3 consecutive transformer encoder layers with each 4 attention heads and a model size (embedding size) of 64, in the reduced forms investigated in (Bermeitinger et al., 2024):

- *vit-mlp*: a complete ViT variant with multi-head attention and multi-layer perceptron (MLP) The MLP is the typical two-layer neural network with one nonlinear layer with the number of units set to 4 times the model size (here: 256 units) and the activation function *gelu*, followed by a linear layer to reduce the dimensions back to 64.
- *vit-nomlp*: a variant without the MLP, thus saving many of the original model's parameters
- *vit-nomlp-wkewq*: a variant without the MLP and additionally using a symmetric similarity measure, using the same matrix for keys and queries
- *vit-nomlp-wkewq-wvwo*: a minimal variant additionally omitting value processing matrices W_v and W_o

All experiments were performed with well-known datasets CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and MNIST (LeCun et al., 1998). Every experiment consists of comparing

1. the baseline loss optimization with Adam over 1000 epochs (700 for the MLP variants);
2. an initial optimization with Adam for 300 epochs (210 with MLP); followed by a further optimization with CG over 700 epochs (490 with MLP), using the result of the preceding Adam optimization as an initial parameter state.

All variants have shown a qualitatively similar course of the epoch-wise gradient norm. The full ViT version, including the MLP, is shown for illustration. Figure 5 shows the gradient norm in dependence on the loss (analogy to Figure 2). The x-axis contains the loss values, the y-axis the gradient norm. Since the loss decreases during optimization, the training progresses from right to left along this axis. The gradient norm values are growing from high loss values (right margin of the x-axis) towards lower ones. This corresponds to the non-convex region, over which optimization takes place with the help of the Adam algorithm. A turning point can be observed at the loss value of around 0.04: the gradient norm starts to decrease. This is qualitatively analogous to the artificial example of Figure 2 and demonstrates the entry into a convex region. Because of this convexity, the second-order CG is used after this turning point. This phase corresponds to the magenta curve in Figure 5.

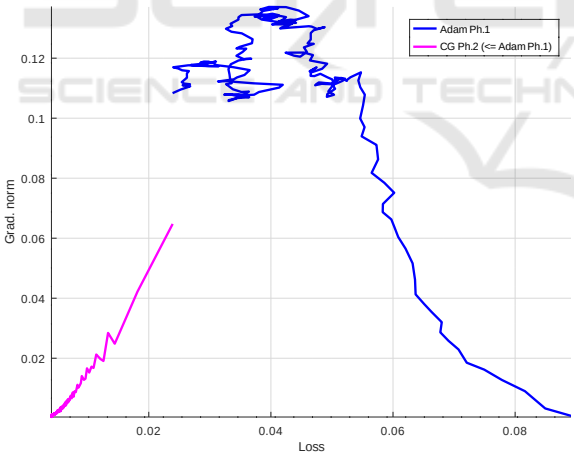

Figure 5: Empirical dependence of the gradient norm on the loss, indicated here on the dataset CIFAR-10 and a ViT architecture. The training starts at the right side with a larger loss, decreases to the left, and decreases quickly after switching from the Adam optimizer to CG.

In Figure 6, the convergence of the loss along a magnitude approximately proportional to MFLOPS is depicted. The blue curve shows the first phase of using Adam and the green curve shows its continuation (corresponding to using Adam in a typical way). The second phase loss of CG (magenta curve) decreases

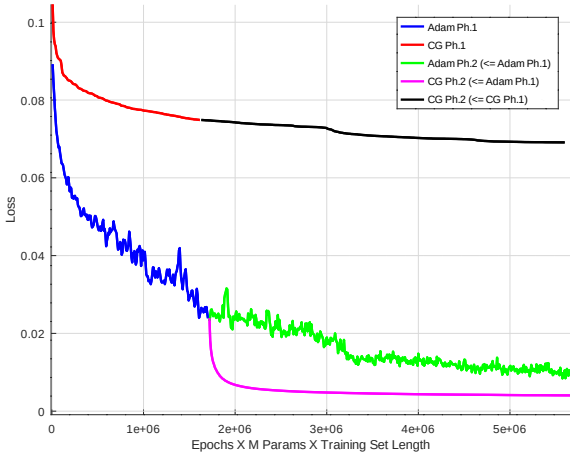

Figure 6: Empirical loss function development with alternative algorithm sequences on the dataset CIFAR-10 and a ViT architecture. The most effective strategy is the two-phase training with Adam (blue) and CG (magenta). For comparison, the green line shows continuation of the Adam phase, while the red and black lines show the training purely done with CG.

considerably faster than its Adam counterpart (green curve). The traditional Adam optimization over all 700 epochs (the blue curve and its continuation by the green curve) is visibly inferior to the convergence of the two-phase algorithm (blue and magenta curve).

The advantage of the two-phase algorithm remains substantial, even considering additional forward passes per epoch spent by line search of CG. For comparison, using CG in both phases, the loss is depicted by the red and black curves.

This pattern occurred for all investigated model variants and datasets (ViT variants and VGG5 with CIFAR-10, CIFAR-100, and MNIST). The sustained simplicity of this pattern was striking and somewhat unexpected. There were no indicators for saddle points or spurious minima, which would become apparent as regions of a very small gradient norm. Once the gradient norm peak passed, the second-order optimization path became straightforward. The final results comparing a pure Adam training run and a two-phase Adam+CG are presented in Table 1.

Furthermore, in terms of performance metrics loss and accuracy, the overdetermination ratio of each benchmark candidate has been evaluated (Hrycej et al., 2023):

$$Q = \frac{KM}{P} \tag{7}$$

with K being the number of training examples, M being the length of the output vector (usually equal to the number of classes) and P being the number of trainable model parameters.

This formula justifies itself by ensuring that the

Table 1: Final results (loss and accuracy for the training and validation split) from the experiments on the three datasets MNIST, CIFAR-10, and CIFAR-100 for different variants of ViT and VGG5. The algorithm column indicates the conventional training with *Adam* or the proposed second-phase training *Adam+CG* using the conjugate gradient optimization method.

	Model variant	Algorithm	Train loss	Train acc.	Val. loss	Val. acc.	Q
MNIST	vit-mlp	Adam	0.0008	0.995	0.0061	0.965	3.9
	vit-mlp	Adam+CG	0.0001	1.000	0.0044	0.974	3.9
	vit-nomlp	Adam	0.0003	0.998	0.0064	0.963	11.0
	vit-nomlp	Adam+CG	0.0002	0.999	0.0053	0.969	11.0
	vit-nomlp-wkewq	Adam	0.0004	0.998	0.0057	0.967	14.1
	vit-nomlp-wkewq	Adam+CG	0.0002	0.999	0.0048	0.971	14.1
	vit-nomlp-wkewq-wvwo1	Adam	0.0016	0.990	0.0073	0.955	33.5
	vit-nomlp-wkewq-wvwo1	Adam+CG	0.0006	0.996	0.0063	0.962	33.5
	vgg5-max-relu	Adam	0.0001	1.000	0.0014	0.993	4.9
	vgg5-max-relu	Adam+CG	0.0001	1.000	0.0011	0.994	4.9
CIFAR-10	vit-mlp	Adam	0.0091	0.943	0.0997	0.428	3.1
	vit-mlp	Adam+CG	0.0041	0.970	0.0991	0.435	3.1
	vit-nomlp	Adam	0.0290	0.819	0.0981	0.428	7.9
	vit-nomlp	Adam+CG	0.0175	0.891	0.0982	0.444	7.9
	vit-nomlp-wkewq	Adam	0.0386	0.744	0.0889	0.441	9.9
	vit-nomlp-wkewq	Adam+CG	0.0270	0.833	0.0881	0.461	9.9
	vit-nomlp-wkewq-wvwo1	Adam	0.0567	0.575	0.0775	0.414	19.1
	vit-nomlp-wkewq-wvwo1	Adam+CG	0.0527	0.612	0.0738	0.436	19.1
	vgg5-max-relu	Adam	0.0059	0.967	0.0531	0.710	4.1
	vgg5-max-relu	Adam+CG	0.0047	0.969	0.0491	0.719	4.1
CIFAR-100	vit-mlp	Adam	0.0041	0.706	0.0128	0.155	29.7
	vit-mlp	Adam+CG	0.0028	0.758	0.0134	0.151	29.7
	vit-nomlp	Adam	0.0062	0.478	0.0112	0.166	72.6
	vit-nomlp	Adam+CG	0.0053	0.534	0.0116	0.165	72.6
	vit-nomlp-wkewq	Adam	0.0069	0.425	0.0108	0.174	88.4
	vit-nomlp-wkewq	Adam+CG	0.0059	0.487	0.0109	0.176	88.4
	vit-nomlp-wkewq-wvwo1	Adam	0.0082	0.291	0.0099	0.157	156.4
	vit-nomlp-wkewq-wvwo1	Adam+CG	0.0078	0.326	0.0097	0.164	156.4
	vgg5-max-relu	Adam	0.0032	0.755	0.0108	0.300	38.9
	vgg5-max-relu	Adam+CG	0.0032	0.737	0.0102	0.321	38.9

numerator KM is equal to the number of constraints to be satisfied (the reference values for all training examples). This product must be larger than the number of trainable parameters for the system to be sufficiently determined. Otherwise, there are infinite solutions, most of which do not generalize. This is equivalent to the requirement for the overdetermination ratio Q to be larger than unity. On the other hand, too large Q values may explain a poor attainable performance — the model does not have enough parameters to represent the input/output relationship. This is the case for CIFAR-100.

For the evaluation of the hypothesis formulated in Section 3, only the loss values (that is, MSE) on the training set are significant since this magnitude is what is directly minimized and thus tests the efficiency of the minimization algorithm. There, sustained superiority of the two-phase concept can be observed.

Nevertheless, the superiority can also be extended to the accuracies and the validation set measures. The extent of the generalization gap (the performance difference between the training and the validation sets) varies greatly. In most cases, they can be explained by the overdetermination ratio: its large values coincide with a small training gap. This does not apply across model groups; VGG5 generalizes better than ViT for given model architectures.

Most models used here do not reach peak performances reached by optimally tuned models for image classification. They are typically substantially smaller to allow for the experiment series with a sufficient number of epochs. Low epoch numbers would bring about the risk of staying in the initial non-convex re-

gion without approaching the genuine minimum.

6 CONCLUSION

Our empirical results strongly support the hypothesis that loss functions exhibit a predictable convexity structure proceeding from the initial non-convexity towards final convexity, enabling targeted optimization strategies that outperform conventional methods. Initial weight parameters (small random values) fall into the non-convex region, while a broad environment of loss minimum is convex. The validity of this hypothesis can be observed in the development of the gradient norm in dependence on the instantaneous loss: a norm growing with decreasing loss indicates non-convexity, while a shrinking norm suggests convexity.

This can be exploited to identify the swap point (gradient norm peak) between both. Then, an efficient non-convex algorithm such as Adam can be applied in the initial non-convex phase, and a fast second-order algorithm such as CG with guaranteed superlinear convergence can be used in the second phase. A set of benchmarks has been used to test the validity of the hypothesis and the subsequent efficiency of this optimization scheme. Although they are relatively small to remain feasible with given computing resources, they cover relevant variants of the *ViT* architecture that can be expected to impact convexity properties: using or not using an MLP, defining the similarity in the attention mechanism symmetrically or asymmetrically, and putting the value vectors of embeddings in a compressed or uncompressed form (matrices W_v and W_o). A completely different architecture, the convolutional network *VGG5*, has also been tested.

The results have been surprisingly unambiguous. All variants exhibited the same pattern of the gradient norm increasing towards a swap point and decreasing after it. The final losses with a two-phase algorithm have always been better than those with a single algorithm (Adam). CG alone did not perform well in the initial non-convex phase, which caused a considerable lag so that the convex region was not attained. The same is true with a single exception for CIFAR-100. An analogical behavior can be observed for the performance of the validation set, which has been admittedly relatively poor for CIFAR-100 because of the excessive overdetermination with given models — the parameter sets seem to have been insufficient for image classification with 100 classes. The top-5 accuracy on this dataset was more acceptable, over 50 %.

Of course, it must be questioned how far this empirical finding can be generalized to arbitrary architectures, mainly to large models. One of the very difficult questions is the convexity structure of loss functions with arbitrary models or even with a model class relevant to practice. However, it is essential to note that there is no particular risk when using the two-phase method. Gradient norms can be automatically monitored and deviations from the hypothesis can be identified. If there is evidence against a single gradient norm peak corresponding to the swap point, a non-convex method can be used to continue as a safe resort. If the hypothesis can be confirmed, there is an almost certain reward in convergence speed and accuracy.

Nevertheless, the next goal of our work is to verify the hypothesis on a large text-based model.

REFERENCES

Bermeitinger, B., Hrycej, T., Pavone, M., Kath, J., and Handschuh, S. (2024). Reducing the Transformer Architecture to a Minimum. In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 234–241, Porto, Portugal. SCITEPRESS.

Chen, C., Shen, L., Zou, F., and Liu, W. (2022). Towards practical Adam: Non-convexity, convergence theory, and mini-batch acceleration. *J. Mach. Learn. Res.*, 23(1):229:10411–229:10457.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, page 21, Vienna, Austria.

Ergen, T. and Pilanci, M. (2023). The Convex Landscape of Neural Networks: Characterizing Global Optima and Stationary Points via Lasso Models.

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154.

Fotopoulos, G. B., Popovich, P., and Papadopoulos, N. H. (2024). Review Non-convex Optimization Method for Machine Learning.

Hrycej, T., Bermeitinger, B., Cetto, M., and Handschuh, S. (2023). *Mathematical Foundations of Data Science*. Texts in Computer Science. Springer International Publishing, Cham.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations*.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Dataset, University of Toronto.

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

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Kdd '14, pages 661–670, New York, NY, USA. Association for Computing Machinery.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Mockus, J., Eddy, W., and Reklaitis, G. (1997). *Bayesian Heuristic Approach to Discrete and Global Optimization: Algorithms, Visualization, Software, and Applications*. Nonconvex Optimization and Its Applications. Springer US.

Polak, E. and Ribière, G. (1969). Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle. Série rouge*, 3(16):35–43.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*. Cambridge University Press, USA.

Rinnooy Kan, A. H. G. and Timmer, G. T. (1987). Stochastic global optimization methods part II: Multi level methods. *Mathematical Programming*, 39(1):57–78.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*.