ISSN 1064-5624, Doklady Mathematics, 2022, Vol. 106, Suppl. 1, pp. S43–S62. © The Author(s), 2022. This article is an open access publication.
ISSN 1064-5624, Doklady Mathematics, 2022. © The Author(s), 2022. This article is an open access publication.
Russian Text © The Author(s), 2022, published in Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya, 2022, Vol. 508, pp. 50–69.

## ADVANCED STUDIES IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

# Loss Function Dynamics and Landscape for Deep Neural Networks Trained with Quadratic Loss

**M. S. Nakhodnov[a], M. S. Kodryan[b], E. M. Lobacheva[b], and D. S. Vetrov[a,b,*]**

Presented by Academician of the RAS A.L. Semenov

Received October 28, 2022; revised October 28, 2022; accepted November 1, 2022

**Abstract**—Knowledge of the loss landscape geometry makes it possible to successfully explain the behavior of neural networks, the dynamics of their training, and the relationship between resulting solutions and hyperparameters, such as the regularization method, neural network architecture, or learning rate schedule. In this paper, the dynamics of learning and the surface of the standard cross-entropy loss function and the currently popular mean squared error (MSE) loss function for scale-invariant networks with normalization are studied. Symmetries are eliminated via the transition to optimization on a sphere. As a result, three learning phases with fundamentally different properties are revealed depending on the learning step on the sphere, namely, convergence phase, phase of chaotic equilibrium, and phase of destabilized learning. These phases are observed for both loss functions, but larger networks and longer learning for the transition to the convergence phase are required in the case of MSE loss.

**Keywords:** scale invariance, batch normalization, training of neural networks, optimization, MSE loss function

**DOI:** 10.1134/S1064562422060187

## 1. INTRODUCTION

A major task that can be successfully addressed using deep neural networks is multiclass classification. An important component of problem solving is the proper choice of the loss function. In most cases, the cross-entropy loss function is used in classification problems. Other options are also possible, and there is evidence that alternative loss functions can lead to better performance for a large class of tasks and architectures [1].

On the other hand, the solution of modern problems in machine learning relies heavily on empirical techniques for obtaining the best results. For example, the choice of an optimizer or a learning rate schedule has long been based on empirical results for particular architectures [2]. Based on the study of the loss function landscape, engineering techniques, such as batch normalization [3] and residual connections [4], were justified and new methods for improving model generalization [5] were proposed.

It is well known that the optimum width has a strong correlation with generalization [6]. That is why the width at the current point and its dynamics in the course of training are of primary interest in analyzing the loss function landscape.

The loss function surface is difficult to study, since optimization occurs in a multidimensional space and the function specified by a deep neural network is nonconvex. Normalization layers in networks makes the analysis even more complicated because of the emergence of scale-invariant symmetries. To avoid these symmetries, we pass to optimization on a sphere. By varying the resolution on the sphere, we find three regimes of neural network training corresponding to different domains of the loss function surface. In this paper, these phases are analyzed from the point of view of model generalization and the width of the resulting solutions. Additionally, various loss functions are compared in terms of the influence exerted on the found phases and the dynamics of training.

## 2. EXPERIMENTAL SETUP

### *2.1. Symmetries in Neural Networks*

The study of neural networks is complicated in the case of a significant level of overparameterization [7] and in the presence of internal symmetries. The simplest examples of such symmetries are consistent rearrangements of neurons in adjacent layers and consistent rescaling of weights in networks with the ReLU

[a] *AIRI, Moscow, Russia*
[b] *HSE University, Moscow, Russia*
**e-mail: dvetrov@hse.ru*

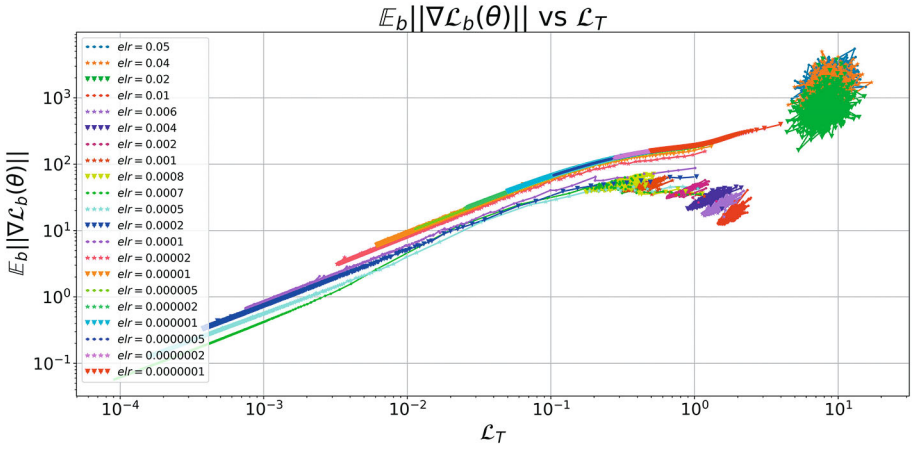


**Fig. 1.** Phase diagram for the curvature and the cross-entropy loss function for various values of *elr* for ConvNet. Three different regimes of the trajectory behavior are observed.

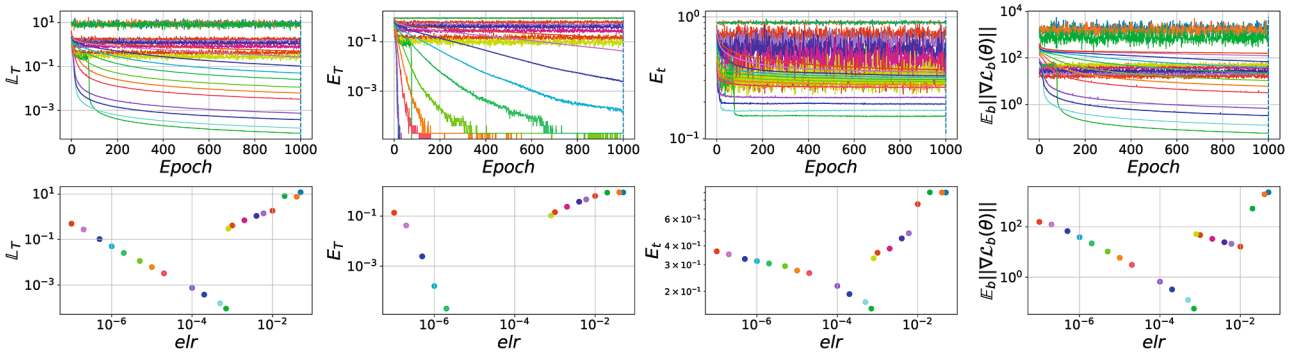


**Fig. 2.** Basic metrics for various *elr*. The rightmost diagram demonstrates jumps between the phases for varying *elr*.

activation function [8]. Such transformations usually leave the network functionally unchanged, although, in the weight space, the model can change significantly. Another important symmetry is scale invariance in networks with batch normalization. Due to the use of batch normalization after a convolutional layer, the multiplication of weights preceding the normalization layer does not change the network as a function of its input. Consider a neural network $f(\theta)$ with weights $\theta \in R^d$. Parameters the multiplication of which by an arbitrary positive coefficient $\alpha$ does not change the functional form of the network will be called scale-invariant (SI). In this case, for SI parameters, we have

$$f(\alpha\theta) = f(\theta), \forall\ \theta, \alpha > 0, \quad (1)$$

$$\nabla f(\alpha\theta) = \frac{1}{\alpha}\nabla f(\theta). \quad (2)$$

The presence of SI parameters in a network leads to ambiguity in the determination of the optimum width, since depending on the normalization of the weights, functionally identical models will have different gradients and second derivatives according to Eq. (2). To get rid of the invariance, we consider networks involving only scale-invariant parameters on a sphere of fixed radius. To be definite, we assume by default that the network is specified on a unit sphere, i.e., $\theta \in B\_\{1\} = \{\theta | \|\theta\| = 1\}$. The learning rate of a network with fully scale-invariant (FSI) parameters on a unit sphere will be called the effective learning rate (*elr*). When such a model is trained with gradient methods, it may happen that, after the current step, the norm of the weights becomes different from 1. In this case, we propose normalizing the weights at the current step. The difference of this approach from Riemannian optimization on a sphere is discussed in Appendix 3.

It should be noted that fixing the general norm of parameters eliminates only part of the symmetry in a neural network, while individual filters of convolutional layers remain invariant under renormalization.

To study the effects associated with the dynamics of optimization along the loss function surface, we need an experimental setup in which the training features are isolated from extraneous effects, such as overfitting and symmetries in the neural network. For this purpose, we employ the following controllable, yet close

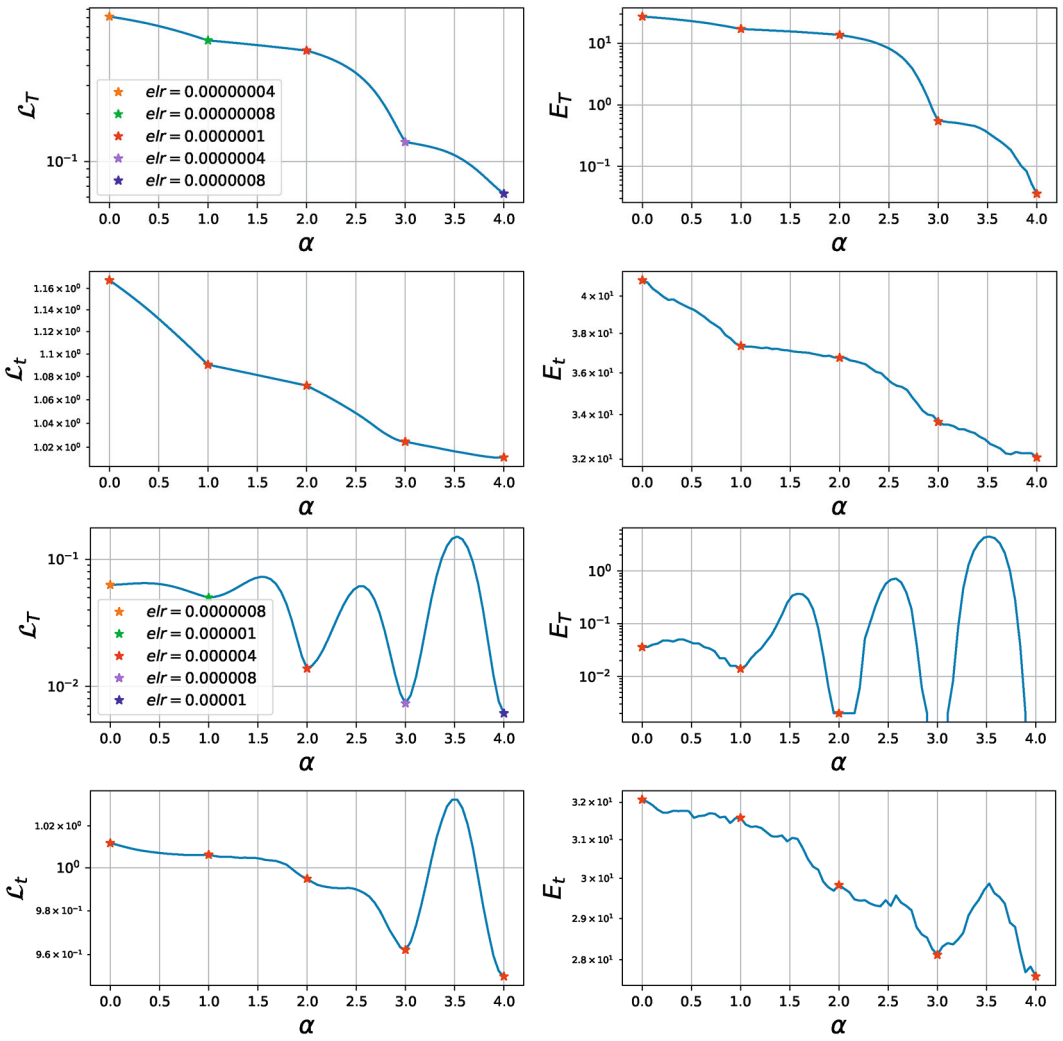


**Fig. 3.** Mode connectivity for various models from the first phase: models from the unconverged first phase (top panels) and the converged first phase (bottom panels). After convergence is achieved, the domains of optima for various *elr* are linearly nonconnected.

to actual, training conditions. First, as a training set, we use the CIFAR10 dataset without augmentation. Second, as a neural network architecture, we use the convolutional neural network with batch normalization ConvNet with FSI parameters. An FSI architecture is achieved by fixing affine layers of batch normalization and fixing the last linear layer of the network. A detailed description of the architecture is given in Appendix 1. It is well known that such constraints on the parameters do not affect the test performance of the model. Third, training is based on stochastic gradient descent (SGD) without using momentum or $L_2$ regularization. All models are trained from the same initialization with the same order of batches during optimization.

As a loss function, we consider two alternatives. The standard choice for the optimized error in a $C$-class classification task is the cross-entropy

$$L(\hat{y}, y) = -\log \frac{\exp \widehat{y_y}}{\sum_{i=1}^{C} \exp \widehat{y_i}}, \tag{3}$$

where $y, \hat{y} \in R^C$ are the correct class label and the network output, respectively.

As an alternative, we consider the mean squared error (MSE) loss function

$$L(\hat{y}, y) = \sum_{i=1}^{C} (\widehat{y_i} - 1[y = i])^2. \tag{4}$$

It is not clear from available works which of these functions is preferable. On the one hand, it has been long believed that the MSE loss function converges more slowly and leads to worse test performance with the use of SGD [9, 10].

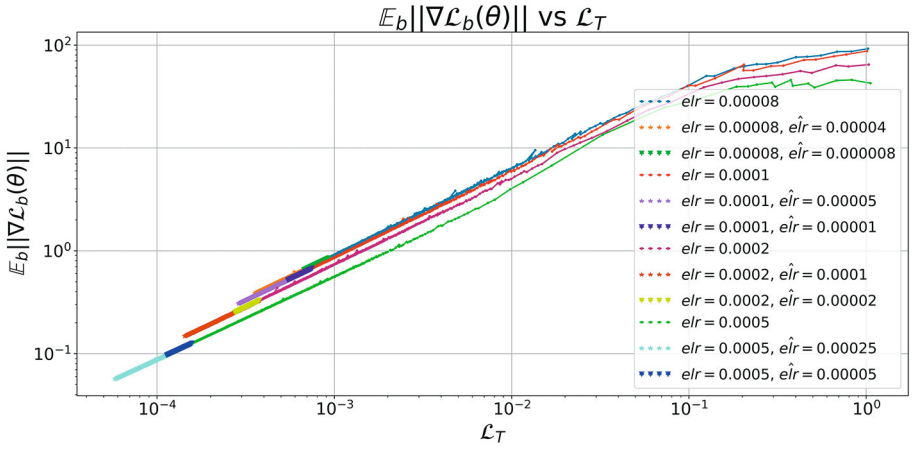


**Fig. 4.** Phase diagram for decreasing *elr* for ConvNet. The final *elr* is denoted by $\widehat{elr}$. The trajectories continue the initial trend, which suggests that they get stuck in a fixed domain near a local minimum.

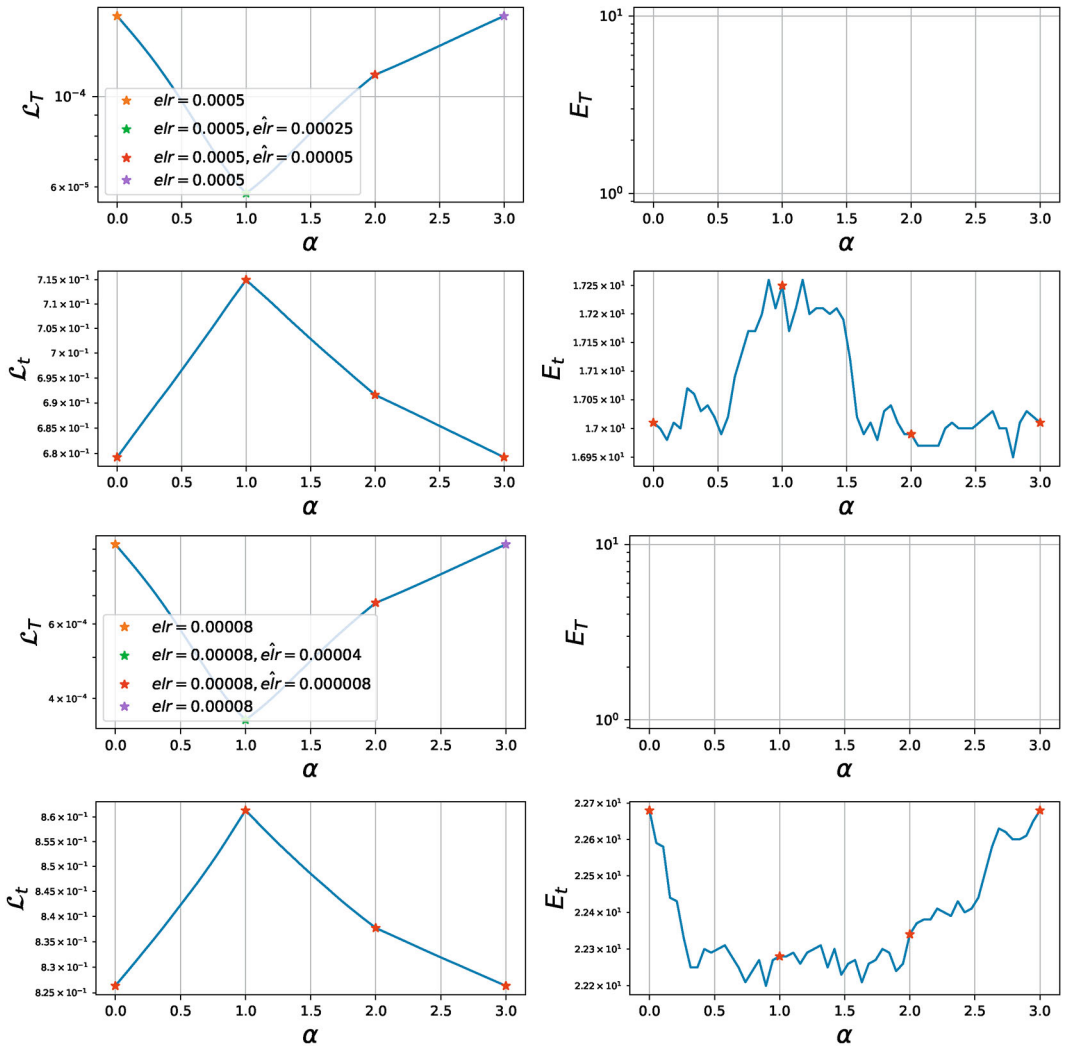


**Fig. 5.** Mode connectivity for models with lower *elr*. Since the models achieved a zero train error, the corresponding points on the plots for $E_T$ are not presented. The models remain linearly connected.

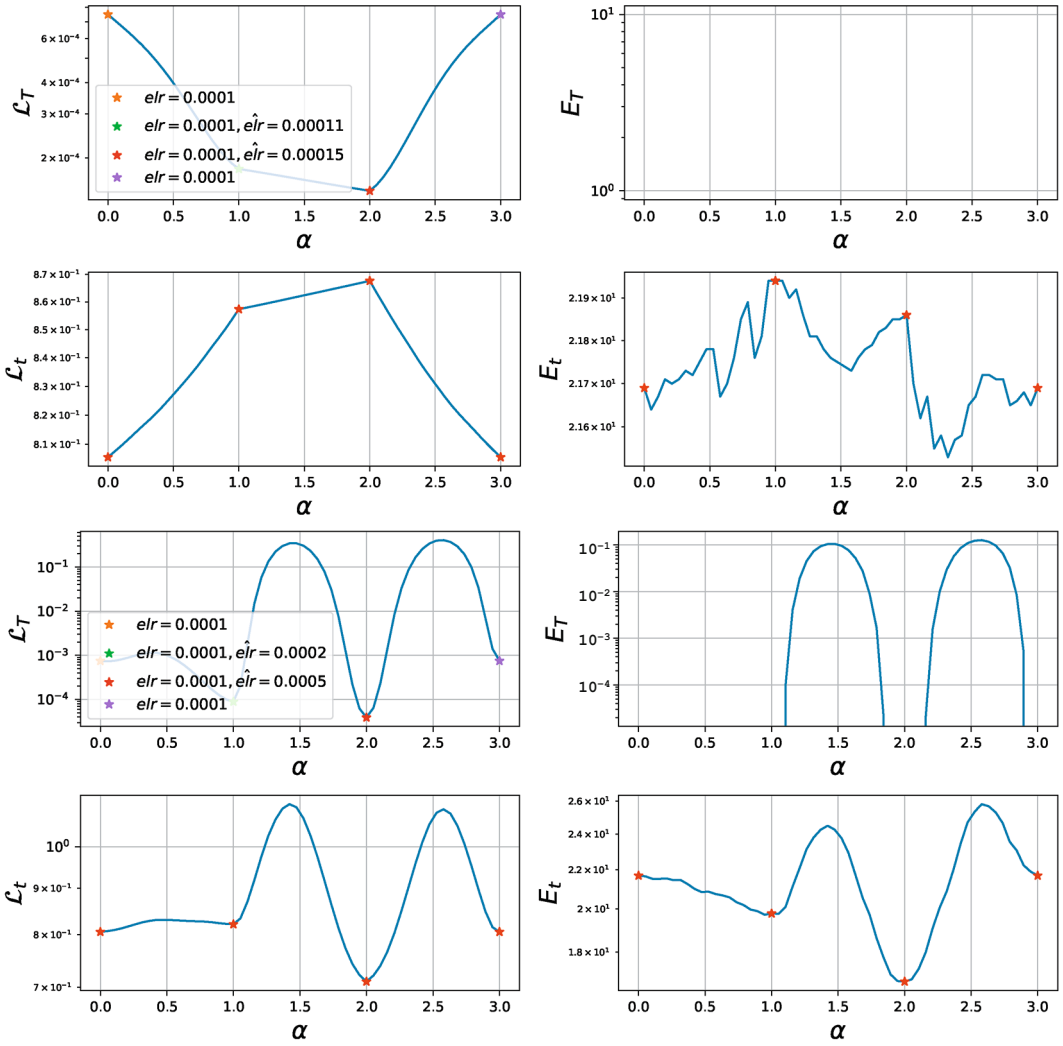


**Fig. 6.** Mode connectivity for $elr = 0.0001$ for finetuning with a higher learning rate. For small increases (top panels), the points remain linearly connected. A stronger increase in $\widehat{elr}$ leads to the transition to a neighborhood of another optimum (bottom panels). Models that have achieved a zero train error are not presented in the plots for $E_T$.

However, more recent studies demonstrate an opposite situation: a detailed analysis in [1] on a broad class of architectures and problems showed that they are equivalent in terms of quality with the MSE loss function having a slightly slower convergence rate.

From a theoretical point of view, there is no definitive answer either. With a high degree of reparametrization typical of neural networks and under sufficiently strong conditions, it has been shown that solutions based on the cross-entropy and MSE loss functions coincide functionally [11]. However, available works do not say whether the results can be extended to modern architectures of deep neural networks.

As basic objects of study, we use the mean value $L_T$ of the loss function on a training set, the fractions $E_T, E_t$ of incorrectly classified objects on training and test sets, and the curvature metric $GM = E_b \|\nabla L_b(\theta)\|$.

### *2.2. Curvature Metrics*

As a baseline metric of the width, we use the mean norm of the gradients over individual batches of the training set *GM*. Intuitively, this metric shows how large the scatter of the gradients over individual objects at a point of the weight space is. This metric must be small in wide plane domains and large in narrow ones.

In practice, this metric well correlates with second-order curvature metrics, such as the trace of the Fisher information matrix or the maximum eigenvalue of the Hessian matrix. A theoretical analysis also confirms high correlations between these metrics [12]. Note that the computation of *GM* requires only one back-

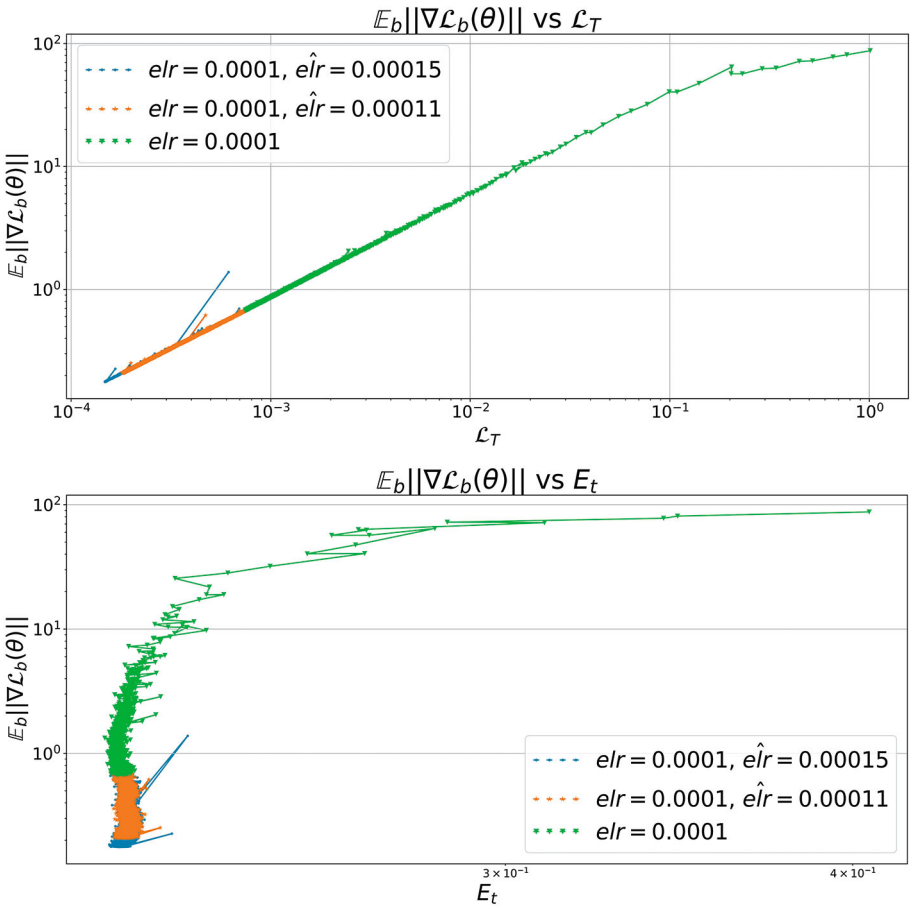


**Fig. 7.** Phase diagrams for $elr = 0.0001$ for fine-tuning with a higher learning rate. Under a small increases in $elr$, the model stays within the initial optimum.

ward pass, in contrast to two backward passes in the computation of estimates on statistics of the Hessian and the Fisher information matrix. Moreover, since the averaging is performed over batches, rather than over individual objects, the curvature estimate can be computed much faster without degrading the quality of the approximations. A comparison of these metrics is made in Appendix 2.

## 3. TRAINING WITH CROSS-ENTROPY LOSS FUNCTION

To analyze the dynamics of neural network training, FSI models were trained with the cross-entropy loss function for various values of *elr*.

Figure 1 shows that the models can be formally divided into three groups. In the first group, the models converge to a domain with wide minima and a low loss function value. In the second group, the loss and curvature models oscillate about some value, as can be seen in the upper panels in Fig. 2. The third group contains models with the largest curvature.

Below, each group is analyzed separately.

### *3.1. First Phase*

Models for which $elr \le 7\times10^{-4}$ are assigned to the first phase. In these models, with an increase in *elr*, all metrics decrease consistently at the end of training. The upper boundary of the phase is determined by a sharp change in all metrics (lower panels in Fig. 2), which suggests that the behavior changes qualitatively in crossing this boundary. Moreover, the first phase can be formally divided into two subphases, namely, models achieving a zero error on the training set at the end of training (converged first phase) and the other models with smaller values of *elr* (unconverged first phase).

To analyze the first phase, we investigate the linear connectivity of models (mode connectivity). For this purpose, we consider two models of $f(\theta)$ parameterized by weights $\theta_1, \theta_2$. By the mode connectivity, we mean the values of the metrics for the models on the interval between two starting points $f(\alpha\theta_1 + (1-\alpha)\theta_2)$, $\alpha \in [0, 1]$. The models are called linearly connected or lying in the same domain if the graph of the mode connectivity does not contain sharp extrema at intermedi-

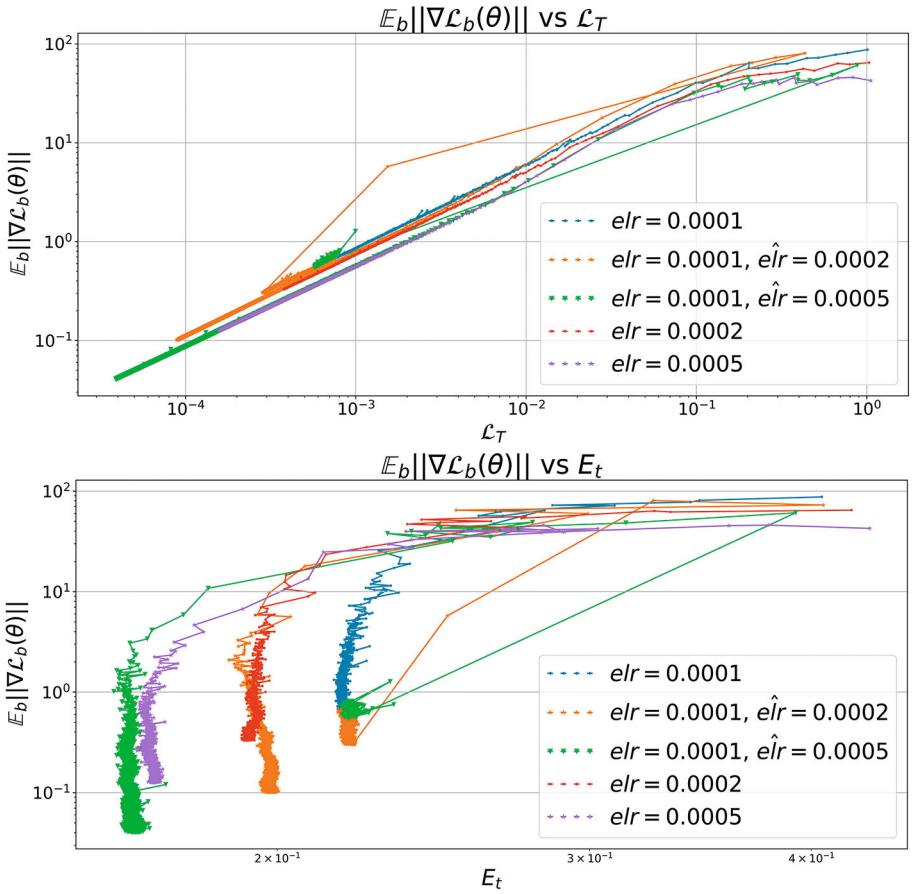


**Fig. 8.** Phase diagrams for $elr = 0.0001$ for finetuning with a higher learning rate. A jump from one trajectory to another in the transition to another, flatter optimum can be seen in both diagrams.

ate points $\alpha \in (0,1)$. Otherwise, the models are called linearly nonconnected.

Models that do not achieve a zero error due to the low learning rate converge to a single linearly connected domain. At high learning rates, models manage to diverge to different optima, which leads to a peak in the loss function value (see the plot of the mode connectivity in Fig. 3). It should be noted that the exact boundary between the subphases corresponds not to a zero train error, but rather to an error of order of 10–15 objects.

Thus, for $elr \le 8\times10^{-7}$, the models converge to a single domain, but at different rates, which is also confirmed by their coinciding trajectories in the phase diagram in Fig. 1.

After a rather low train error is achieved, the models begin to converge along different trajectories. An increase in the learning rate leads to generalization growing monotonically. This agrees well with the fact that, for large values of *elr*, the resolution of the network decreases, thus leading to convergence to progressively wider plane domains, which, in turn, determine a better quality on the test set. With a further increase in *elr*, the network sharply ceases to converge to a zero train error and its behavior changes qualitatively. Thus, this experiment confirms that the best generalization is achieved in the widest optimum, assuming that the network converges.

Additionally, the absence of mode connectivity in the first phase for sufficiently large *elr* shows that the models converge in different domains of the weight space. Let us show that, in each of these domains, there are no minima of smaller width. For this purpose, the models are additionally trained for 5000 iterations with a sharp decrease in the learning rate by a factor of 2 and 10.

The diagram in Fig. 4 shows that a decrease in *elr* does not change the dynamics of model-training. This indirectly confirms that the global properties of the optimum remain stable and, inside the optimum of given width, there is no minimum of smaller width.

The mode connectivity presented in Fig. 5 also confirms that a decrease in *elr* does not lead to convergence to linearly nonconnected optima.

Now, we examine the behavior of networks with increasing *elr*. Depending on the final *elr*, several basic

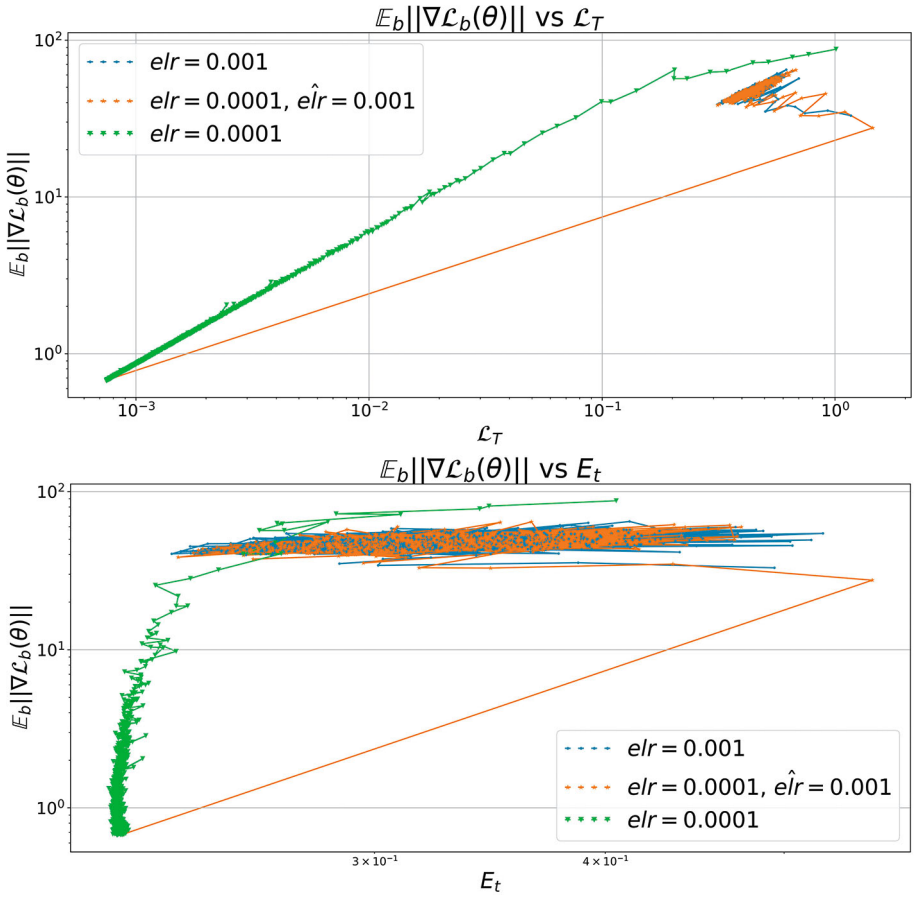


**Fig. 9.** Phase diagrams for $elr = 0.0001$ for finetuning with a higher learning rate. An instantaneous transition to a chaotic regime is observed in this case.

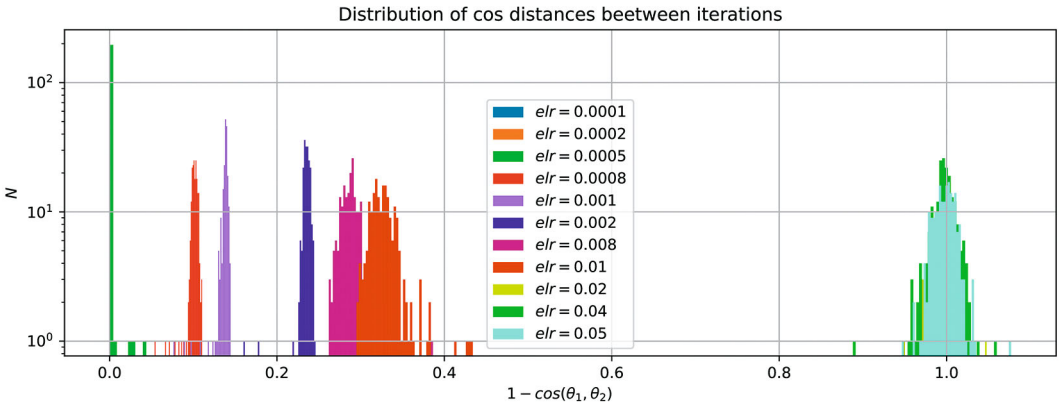


**Fig. 10.** Distribution of cosine distances between all neighboring epochs for various learning rates shows the separation into training phases.

situations are possible. For small growth factors, the dynamics of the network changes weakly. Figures 6 and 7 show that the model stays in the same optimum in terms of generalization and the curvature metric.

As the final *elr* increases further, the network is trained less steadily and, at some time, there is a "jump" and transition to a new trajectory. The values of $E_t$ show that the model can converge to minima slightly different in quality. The results demonstrate that, in the presence of jumps during additional training, the training prehistory has a weak effect, i.e., leaving the instability region, the model returns to the tra-

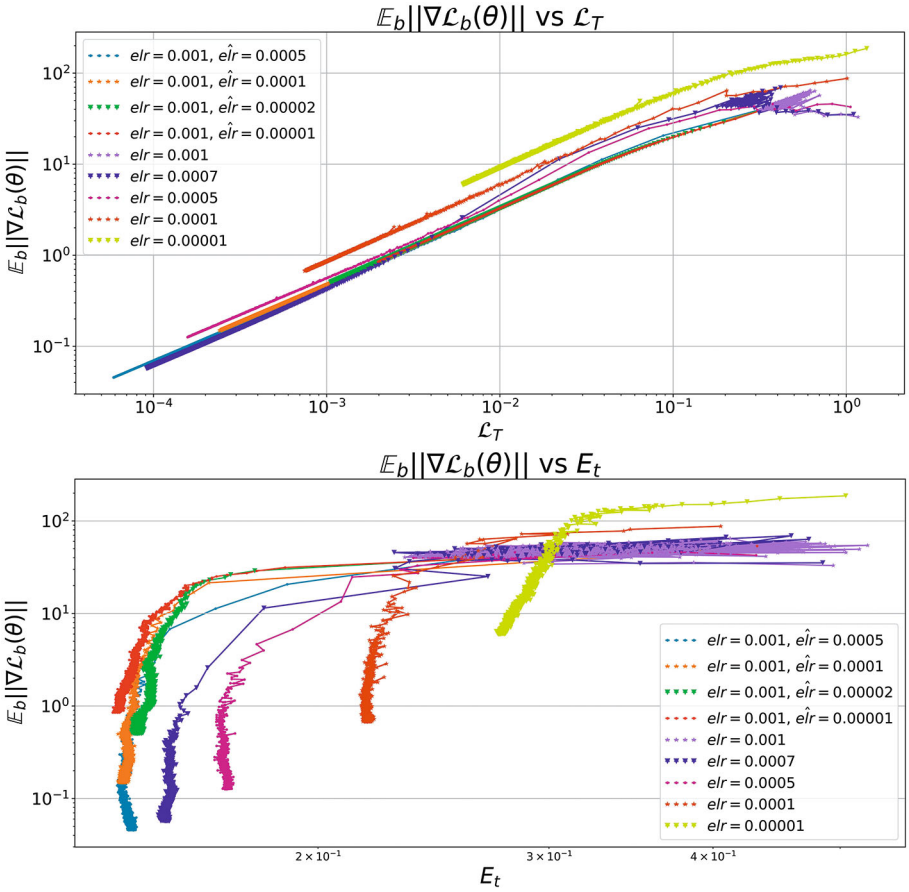


**Fig. 11.** Phase diagrams for $elr = 10^{-3}$ for finetuning from the second phase with a lower learning rate. The models converge to the widest optimum.

jectory corresponds to the initial training with the final $\widehat{elr}$.

Finally, if the final *elr* exceeds the upper boundary of the first phase, then the model quickly diverges and passes into a domain corresponding to training from scratch with a learning rate in the second phase.

### *3.2. Second Phase*

When the learning rate increases to $8\times10^{-4}$, the transition in the next regime occurs: all the metrics quickly (during the first 5–10 iterations) reach a fixed average level and then do not vary in the course of training. The transition to the new phase is accompanied by a sharp jump in all the metrics. Moreover, in the second phase, the model converges to a considerably better quality on the training set, rather than to random predictions. With a further increase in *elr*, the convergence of the model progressively degrades until it passes to the third phase. Interestingly, in this phase, an increase in *elr* leads to a reduced curvature.

It can be supposed that the difference between the first and second phases is caused by the presence of a high-curvature domain in the weight space, which has to be overcome to achieve a low-value loss function. In the phase diagram, we can see a typical turn on the trajectories of the models from the first phase in the domain $L_T \sim 10^{-1}$–$2\times10^{0}$, where a local peak of curvature is observed.

In the second phase, the weights of the network do not converge and the weight vectors at neighboring iterations are less correlated than in the first phase; moreover, the correlation reduces with increasing *elr*, which can be seen in Fig. 10.

As *elr* increases further, the weights at neighboring iterations cease to correlate with each other, which corresponds to the onset of the third phase.

Let us examine the properties of the models in the second phase. The mean values of the metrics do not vary significantly over all 1000 iterations. Due to the high learning rate, the models get stuck at a fixed level and the values of the metrics can reduce only after passing through the curvature "bottleneck." However, since the gradients are extremely high in this zone, this passage is possible only if the SGD step is decreased. Accordingly, we consider an experiment in which a

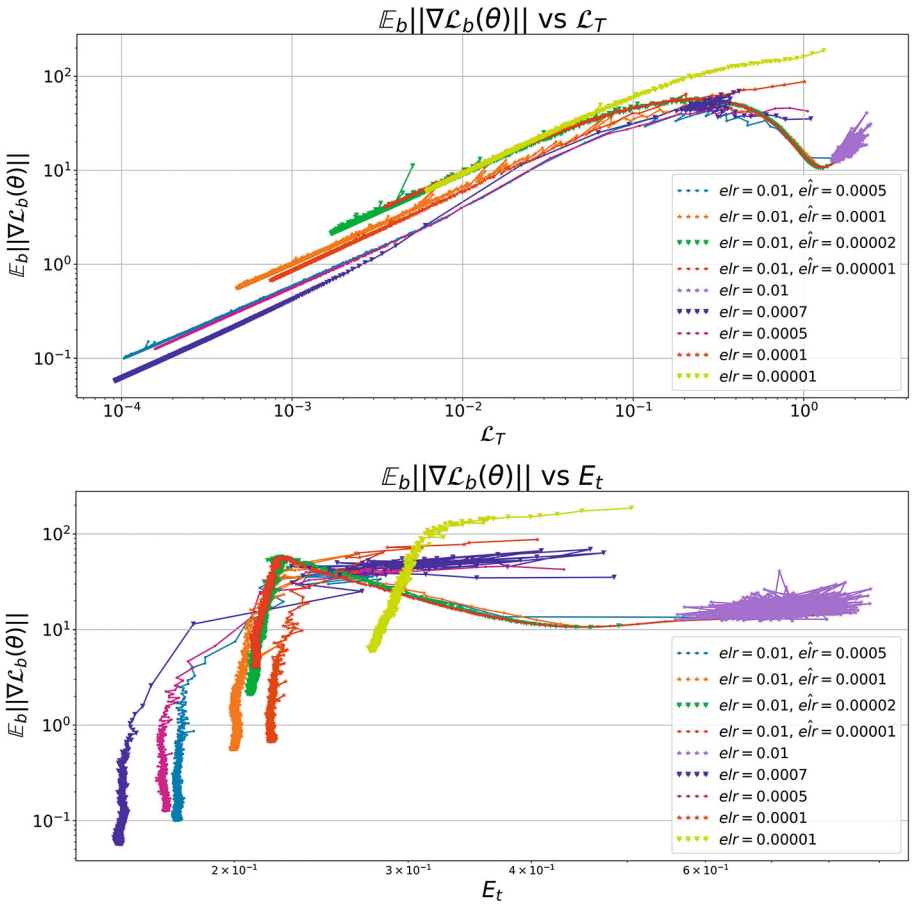


**Fig. 12.** Phase diagrams for $elr = 10^{-2}$ for finetuning from the second phase with a lower learning rate. The models converge to optima of different widths.

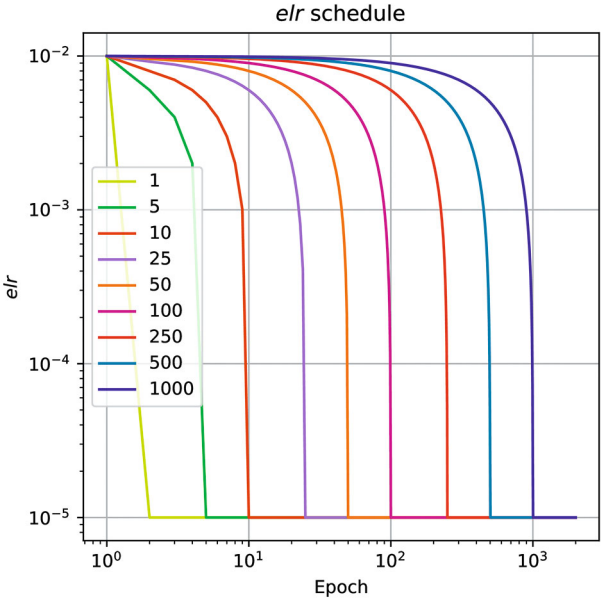


**Fig. 13.** Linear *elr* schedules with various transition lengths.

model from the second phase is additionally trained with a sharply reduced value of *elr*.

Figure 11 shows that the models when launched with a small *elr* from the second phase have a phase diagram similar to the one in Fig. 4: the trajectory continues the trend of points from the second phase, converging near the line corresponding to the highest learning rate in the first phase ($elr = 7\times10^{-4}$), regardless of the final *elr*. Moreover, the generalization of such models exceeds the best generalization among all networks obtained in the first phase. The absence of the correlation between *GM* and $E_t$ in this example once again emphasizes the shortcoming of local curvature metrics when they are used as proxy for generalization.

Fine-tuning with the higher $elr = 10^{-2}$ leads to opposite results. First, according to the top panel in Fig. 12, the trajectories of such models converge to the same curves as those corresponding to models trained with the initially given *elr*. The bottom panel shows that the resulting quality improves only at low final learning rates ($elr \le 10^{-4}$).

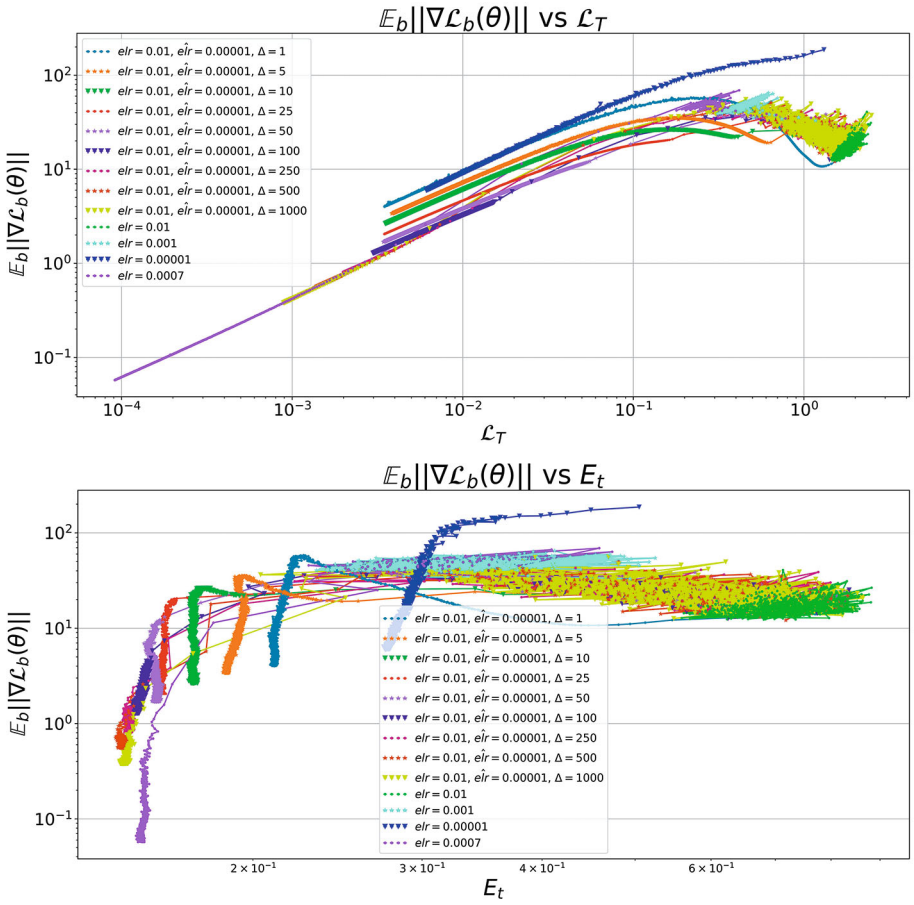


**Fig. 14.** Phase diagrams for $elr = 10^{-2}$ for finetuning up to $\widehat{elr} = 10^{-5}$ with various schedules; $\Delta$ is the transition length.

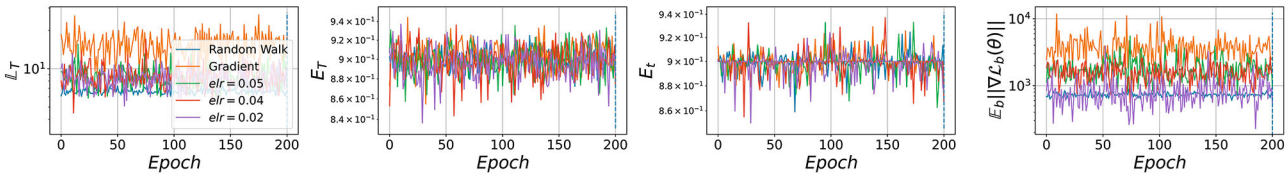


**Fig. 15.** Basic metrics in the third phase: random walk (blue) and gradient descent (orange). Learning in the third phase is similar to random walk.

A comparison of the results for large and small values of *elr* in the second phase shows that pre-training in the second phase has a larger effect on the resulting quality $E_t$ when the starting *elr* is lower.

Additionally, relying on the independence of the resulting quality and the trajectory for finetuning with small *elr* in the second phase, we can hypothesize that the optimum in the first phase is chosen in the transition from the second to the first phase in the early stage of training. To check this hypothesis, the following experiment is considered: for the fixed starting value $elr = 10^{-2}$, we launch finetuning with the piecewise linear learning rate schedule presented in Fig. 13.

Figure 14 shows that, by varying the reduction rate of *elr*, it is possible to obtain a variety of trajectories with one end exhibiting convergence along the line associated with the highest *elr* of the first phase and with the other end (at the fastest variation in *elr*) corresponding to the trajectory of the model for $\widehat{elr}$ without pre-training. A slow decrease in *elr* in the second phase leads to the model gradually passing to trajectories corresponding to smaller *elr* in the second phase until the model jumps to a domain of smaller curvature.

The growth of $E_t$ also smoothly depends on the rate of decrease in *elr*. Thus, we can assume that the opti-

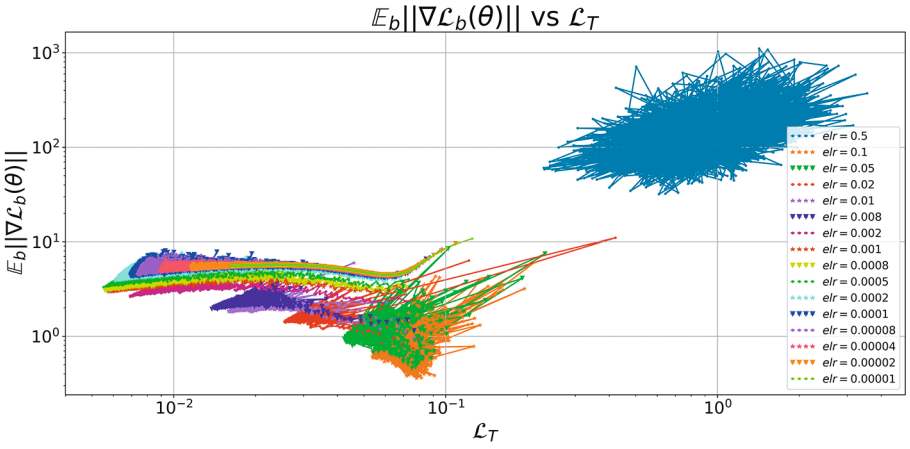


**Fig. 16.** Phase diagram for the curvature and the MSE loss function for various *elr*.

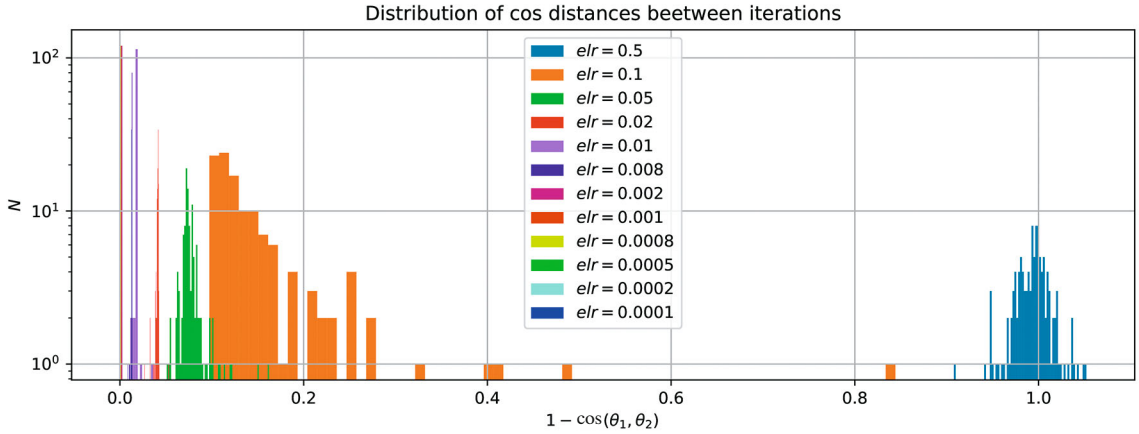


**Fig. 17.** Distribution of cosine distances between all neighboring epochs for various learning rates in the case of the MSE loss function.

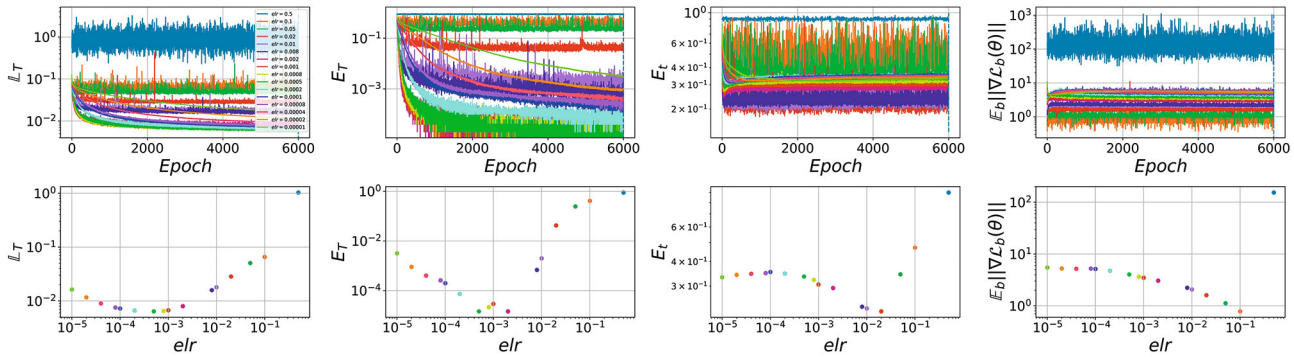


**Fig. 18.** Basic metrics for various *elr* for ConvNet with the MSE loss function. The plot of the test error indicates two phases of learning.

mal schedule strategy for *elr* is to stay at points of the second phase as long as possible before passing to the first phase, since, at the time of the transition, the model "fixes" the minimum to which it will converge. The minimum in the first phase can be changed only by significantly increasing *elr* and achieving instability in training, which makes it possible to "leap" into a domain with other functional characteristics.

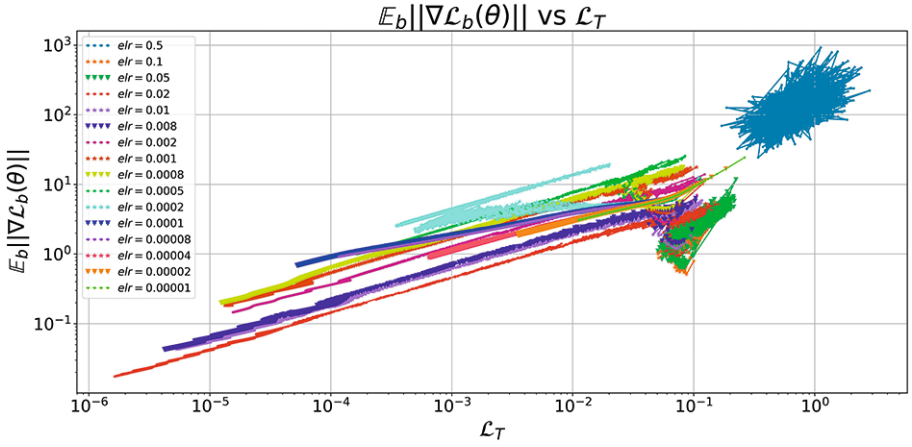


**Fig. 19.** Phase diagram for the curvature and the MSE loss function for various *elr*. ConvNet on a subset from CIFAR10.

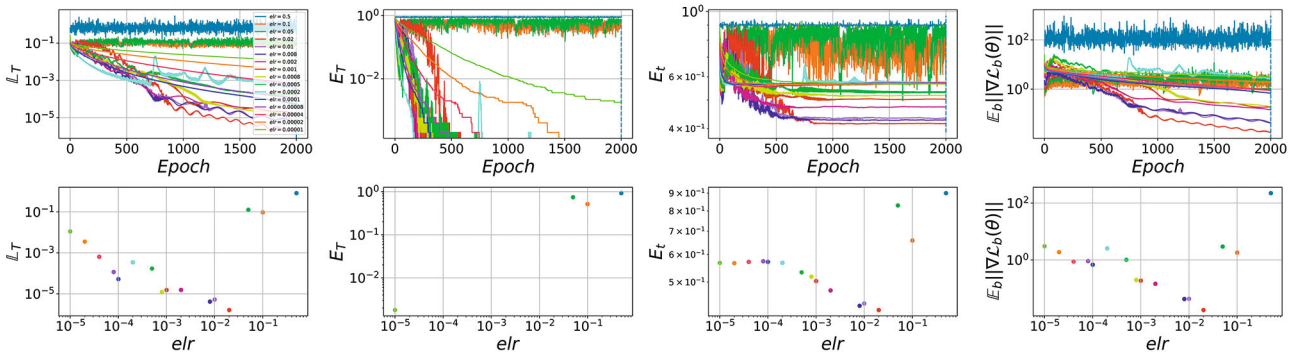


**Fig. 20.** Basic metrics for various *elr* ConvNet with the MSE loss function on a subsample from CIFAR10.

### *3.3. Third Phase*

Consider the remaining models corresponding to $elr \geq 2\times10^{-2}$. A further increase in *elr* in the second phase once again leads to a qualitative change in the behavior. Specifically, the model gradients grow sharply and the train error becomes equivalent to random guess (90% in the case of 10 classes). The transition to this regime corresponds to a switch to random prediction. To check this statement, the model was trained so that every gradient descent step replaced by a step of the same magnitude in a random direction. For comparison purposes, we conducted an experiment with the gradient substituted for the anti-gradient at the current step. The results show that random walking and gradient motion make it possible to bound the observed behavior of the curvature in the third phase from below and above.

## 4. TRAINING WITH THE MSE LOSS FUNCTION

Now we compare the phase diagrams for the MSE loss function. It is well known that the solution of a regression task is more complicated than the solution of a classification task [11], so training with MSE loss function requires a larger number of iterations to achieve convergence. For this reason, in the experiments with MSE loss, all models were trained over 6000 iterations.

An analysis of the phase diagram in Fig. 16 clearly reveals the third phase ($elr = 0.5$) and the unconverged part of the first phase ($elr \lesssim 0.0001$). The other models visually resemble both the first and the second phase for cross entropy. For a more accurate analysis, we consider the distribution of cosine distances for successive models along the learning trajectory. Figure 17 shows that the models with $elr \gtrsim 0.02$ can be assigned to the second phase, while the others, to the first phase.

Now, we examine how this separation into phases correlates with the behavior of the metrics in Fig. 18. It can be seen that none of the models converges to a zero error on the training set, which once again confirms that the models with MSE loss converge more slowly than the ones with the cross-entropy loss function. Another difference is that the boundary between the

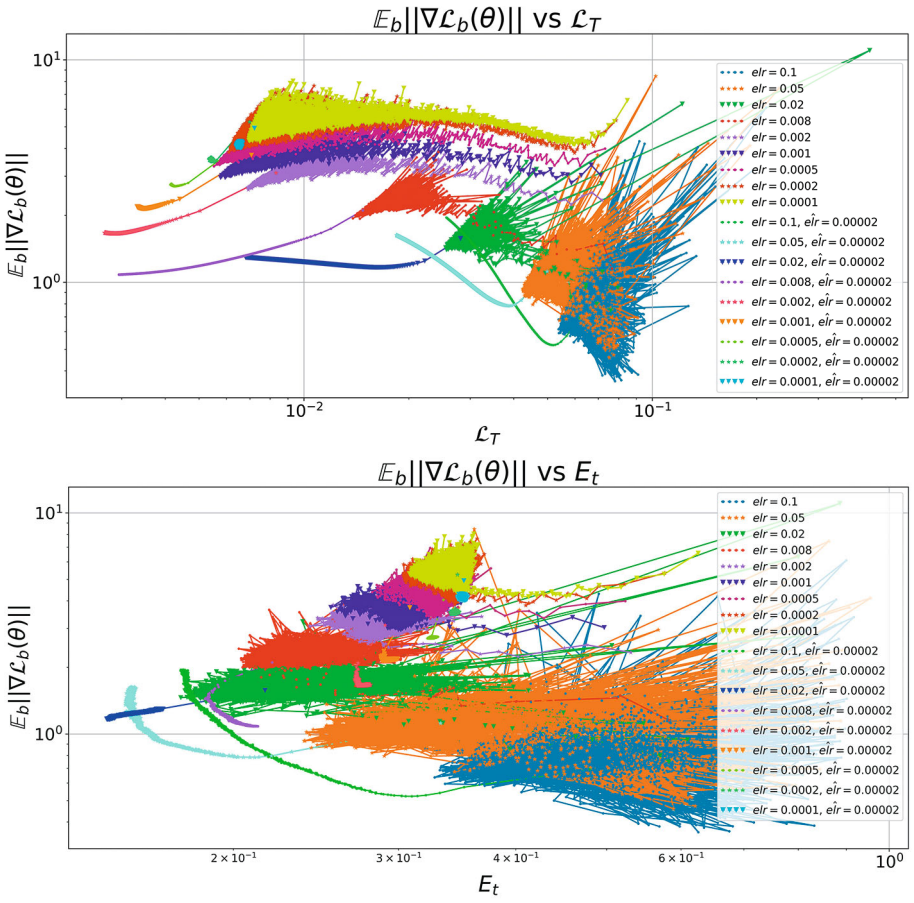


**Fig. 21.** Phase diagrams for finetuning with smaller values of *elr* for MSE.

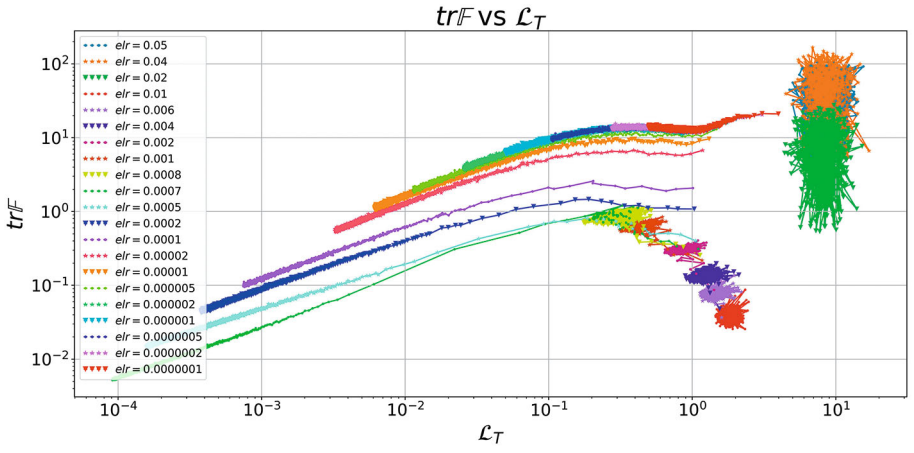


**Fig. 22.** Phase diagram for the trace of the Fisher matrix and the cross-entropy loss function.

first and second phases is fuzzier. Indeed, the models with *elr* ~ 0.008–0.02 monotonically reduce the number of incorrectly classified objects, while remaining in a domain of fixed width. This is in contrast to the first phase of cross-entropy, where the loss and the curvature decrease consistently along the entire trajectory.

Since the models with MSE loss do not achieve convergence on the whole set of 50000 objects, we train the networks on a subset of 10% in size.

In this case, the metrics in Fig. 20 and the phase diagram in Fig. 19 become similar to the case of cross-

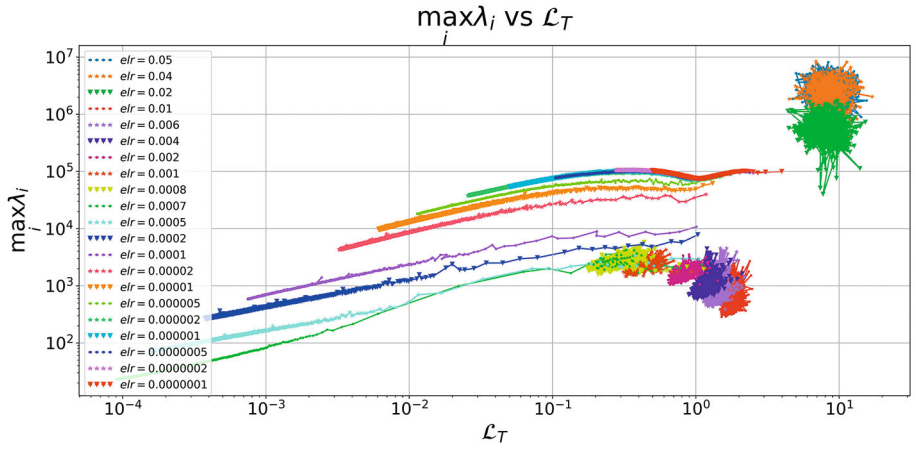


**Fig. 23.** Phase diagram for the maximum eigenvalue of the Hessian and the cross-entropy loss function.

entropy. All three training phases and two subphases of the first phase can be clearly seen in the phase diagram. The optimal value of *elr* in terms of the error on the test set corresponds to the highest learning rate in the first phase. The minimum curvature is attained at the same point.

By analogy with the cross-entropy loss function, we consider an experiment involving additional training with various *elr*.

An analysis of the results shows that a decrease in *elr* from trajectories corresponding to the first phase leads to behavior that is, on the one hand, consistent with the cross-entropy: the trajectories have the same trend as before a decrease in the learning rate. Moreover, the loss function decreases. However, in contrast to the cross-entropy, where the curvature decreases monotonically, for the MSE loss, *GM* either stabilizes ($elr \approx 10^{-2}$, the boundary between the first and second phases) or even begins to increase (for lower *elr*). This behavior suggests the beginning of overfitting, which is more clearly seen in the bottom panel in Fig. 21: $E_t$ first decreases, but then begins to grow sharply.

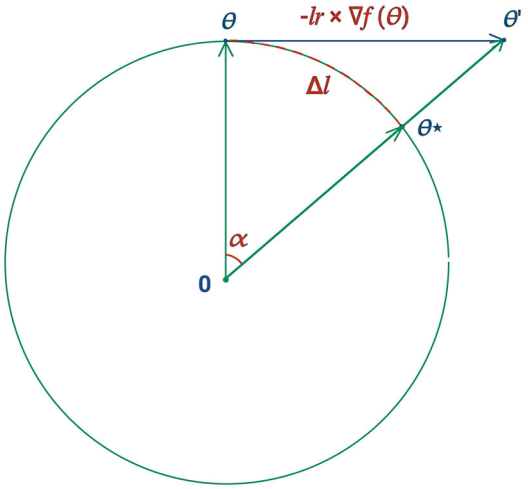


**Fig. 24.** Optimization on the sphere.

## 5. CONCLUSIONS

The conducted study has shown the existence of several phases of training fully scale-invariant networks on a sphere depending on the learning rate.

The first phase of training corresponds to weights converging to an optimum of fixed width.

The second phase is determined by chaotic equilibrium in which the metrics for the network stabilize near some level. The domain in which the neural network stabilizes in the second phase has a decisive impact on the resulting quality of the model.

The third phase is characterized by the transition to random walk on a sphere. There is no correlation between different models at neighboring iterations.

The revealed phases are manifested for various loss functions, namely, for both MSE loss and cross-entropy. The study performed in Appendix 4 shows that these conclusions also hold for networks with non-scale-invariant parameters and regularization.

Additionally, the results confirm the hypothesis that MSE requires a larger number of iterations or a larger number of network parameters than cross-entropy to achieve convergence of similar quality.

## *APPENDIX*

### 1. NETWORK ARCHITECTURE

As a baseline FSI model, we use a convolutional network consisting of four layers as described in Table 1.

Following the example in [13], the last linear layer in a random initialization is fixed so that its weight norm is equal to 10 in the case of cross-entropy loss

**Table 1.** Baseline architecture of the ConvNet network

| No. | Layer |
|---|---|
| 1 | Conv2d(3, 32, kernel_size=(3, 3), padding=(1, 1), bias=False) |
| 2 | BatchNorm2d(32, momentum=0.1, affine=False) |
| 3 | ReLU() |
| 4 | Conv2d(32, 64, kernel_size=(3, 3), padding=(1, 1), bias=False) |
| 5 | BatchNorm2d(64, momentum=0.1, affine=False) |
| 6 | ReLU() |
| 7 | MaxPool2d(kernel_size=2, stride=2) |
| 8 | Conv2d(64, 128, kernel_size=(3, 3), padding=(1, 1), bias=False) |
| 9 | BatchNorm2d(128, momentum=0.1, affine=False) |
| 10 | ReLU() |
| 11 | MaxPool2d(kernel_size=2, stride=2) |
| 12 | Conv2d(128, 256, kernel_size=(3, 3), padding=(1, 1), bias=False) |
| 13 | BatchNorm2d(256, momentum=0.1, affine=False) |
| 14 | ReLU() |
| 15 | MaxPool2d(kernel_size=2, stride=2) |
| 16 | MaxPool2d(kernel_size=4, stride=4) |
| 17 | Linear(in_features=256, out_features=10, bias=True) |

and 1.5 in the case of MSE loss. This norm is close to the values taken by the norm of the last layer in training all network parameters. Norm initialization is necessary for achieving a low train error.

## 2. COMPARISON OF CURVATURE METRICS

To validate the chosen curvature metric $GM = E_b\|\nabla L_b(\theta)\|$ for the ConvNet model, we compared phase diagrams obtained other local methods for curvature estimation, namely, the trace of the Fisher information matrix $trF$ and the maximum eigenvalue of the Hessian $\max_i \lambda_i$.

Comparing Figs. 22 and 23 with the initial diagram in Fig. 1, we can see a clear similarity: first, all three phases of training are observed in all curvature metrics. Second, even more pronounced is the bottleneck effect, i.e., the high-curvature domains to be passed by the models from the first phase before going down to the domain of low loss function values.

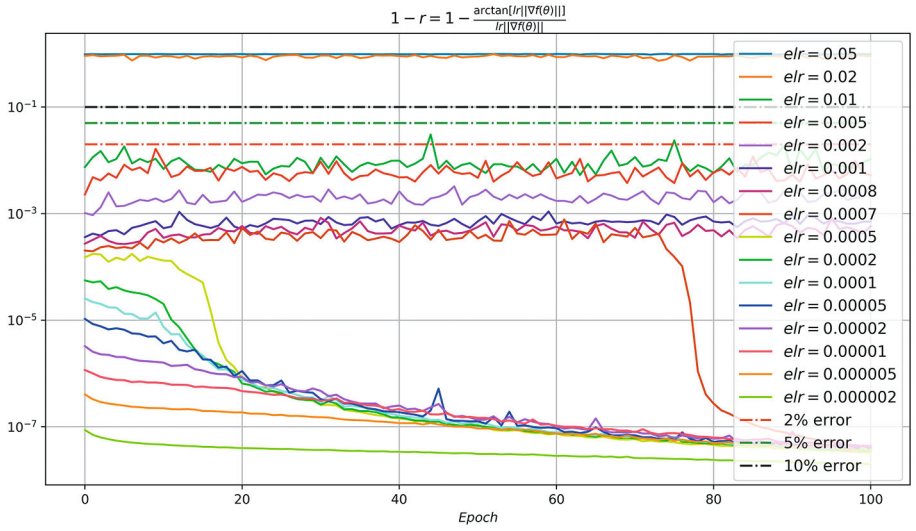


**Fig. 25.** Optimization error as a function of *lr*.

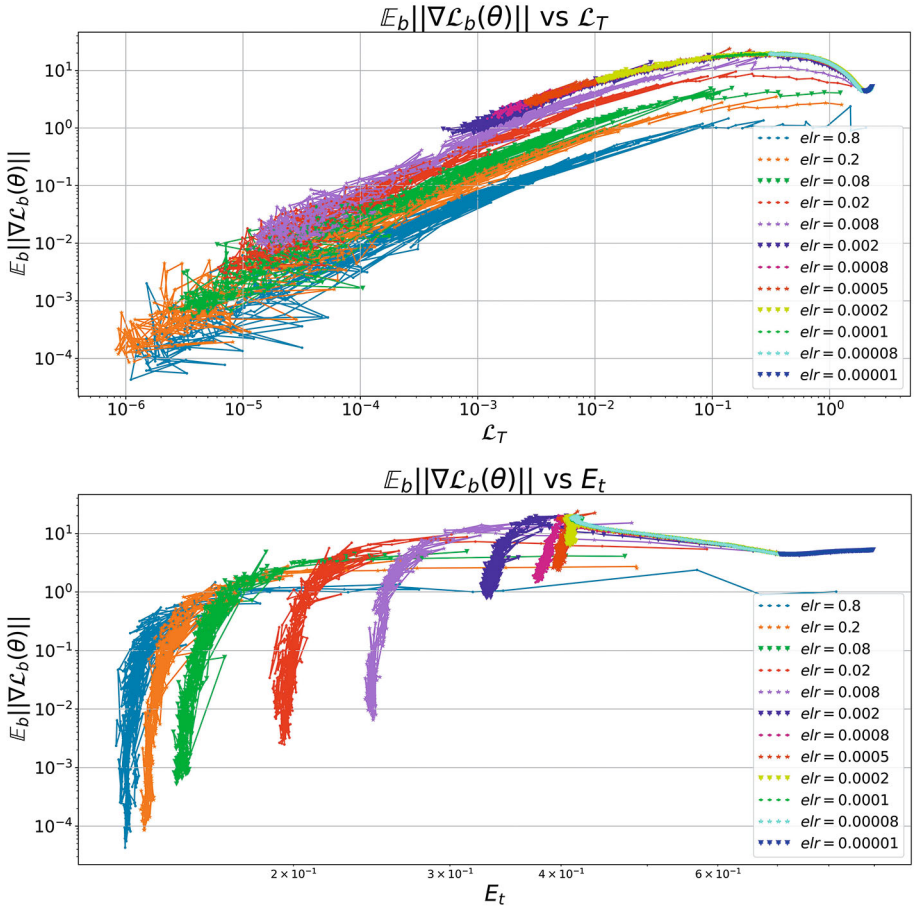


**Fig. 26.** Phase diagrams for VGG16BN, $wd = 0$.

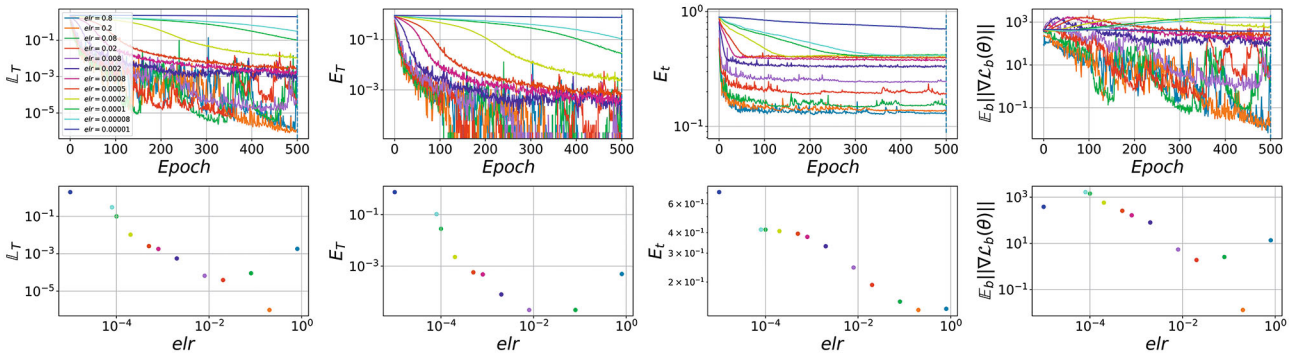


**Fig. 27.** Basic metrics for various elr for VGG16BN, $wd = 0.0$.

## 3. OPTIMIZATION ON A SPHERE

According to the experimental setup, the model is specified on a sphere of radius 1. Accordingly, a valid optimization method for this model is based on Riemannian optimization. However, the implementation of such algorithms in practice is redundant. Indeed, assume that the network $f(\theta)$ defined by Eq. (1) is trained with gradient descent in the space of weights $\theta \in R^d$ with some learning rate $lr$. Since the weight norm changes after performing the optimization step $\theta' = \theta - lr \times \nabla f(\theta)$, we project the weights back onto the sphere: $\theta^* = \dfrac{\theta'}{\|\theta'\|}$. Thus, the transition from $\theta$ to $\theta^*$ corresponds to the motion over the sphere along a great circle arc at a distance of $\Delta l$. The true value of the learning rate on the sphere is $elr = \dfrac{\Delta l}{\|\nabla f(\theta)\|}$. The error between $lr$ and $elr$ is defined as

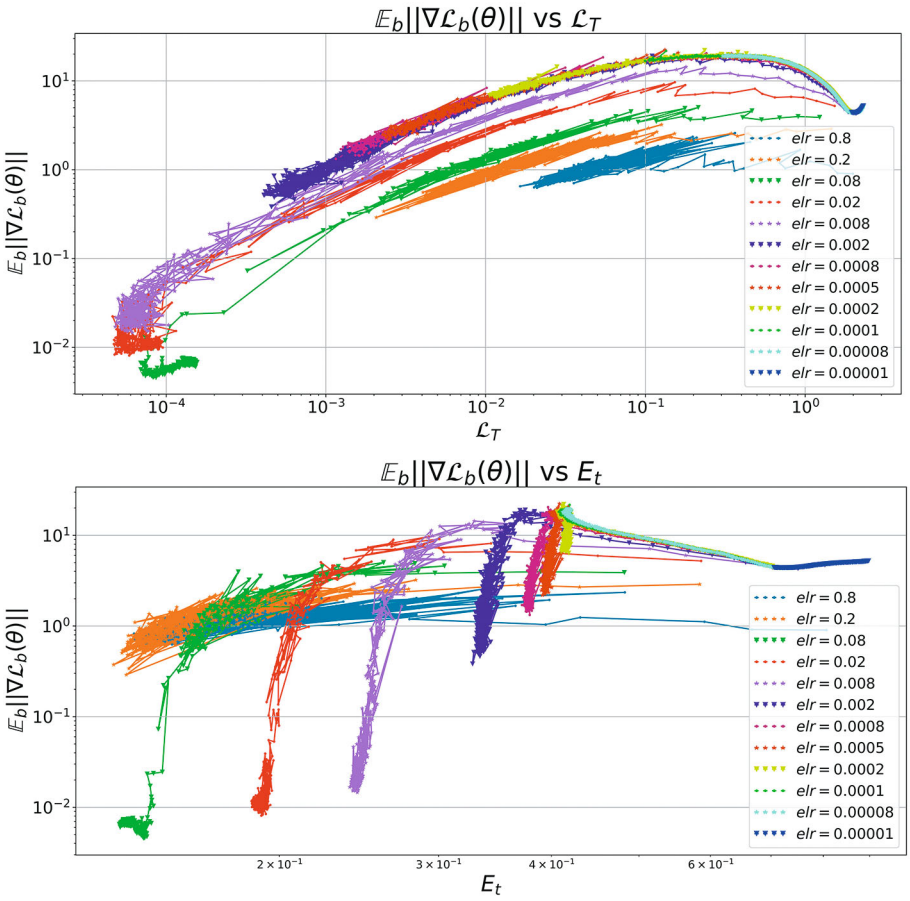


**Fig. 28.** Phase diagrams for VGG16BN, $wd = 0.0001$.

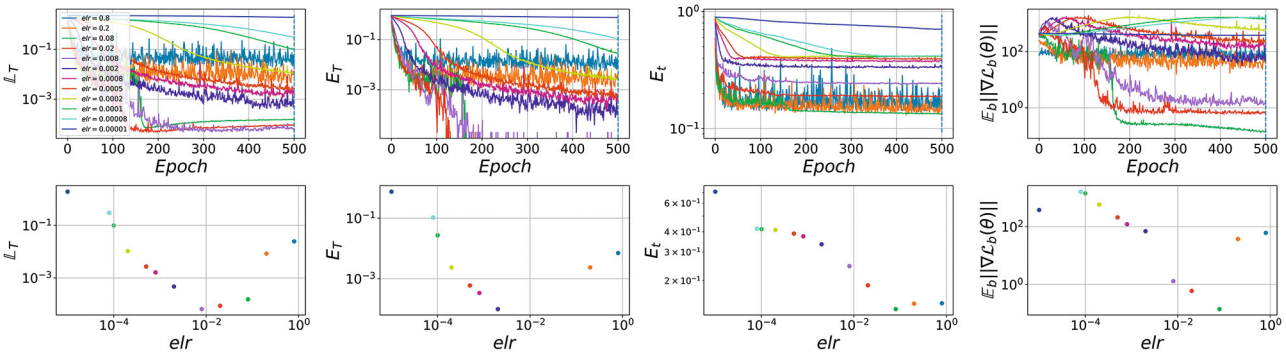


**Fig. 29.** Basic metrics for various *elr* for VGG16BN, $wd = 0.0001$.

$$r = \frac{elr}{lr} = \frac{\Delta l}{lr\|\nabla f(\theta)\|} = \frac{\arctan[lr\|\nabla f(\theta)\|]}{lr\|\nabla f(\theta)\|}.$$

Note that $r$ depends on the effective step size $lr\|\nabla f(\theta)\|$, which may potentially lead to a high error in the case of large gradient norms.

We examine how the relative error $1 - r$ varies in the course of training with the cross-entropy loss function.

Figure 25 shows that, for all models in the first and second phases, SGD-based optimization with weight renormalization corresponds to actual Riemannian optimization on the sphere with *elr* differing by at most

2% from the learning rate *lr* in the original Euclidean space. In the third phase, due to the growth of the gradient, this procedure leads to a significantly overestimated *elr*. Intuitively, in the third phase, every SGD step leads to the weight vector rotated by an angle $\alpha \approx 90°$. It should be noted that the procedure with weight normalization fails to examine the behavior of the networks for extremely large values of *elr*, which lead to the weight vector rotated by an angle larger than 90°.

## 4. RESULTS FOR VGG16BN

Consider a more practical experimental setup. As a neural network, we use VGG16 [14] with batch normalization. The linear layers and the affine parameters of batch normalization are not fixed. Additionally, the weights are not projected onto a sphere in training. Instead, the parameters are optimized in the initial space by applying SGD without momentum. Since the network is trained without constraints, the concept of *elr* loses meaning in this setting. However, for consistency, by *elr* in this section we mean the common learning rate (*lr*).

First, we consider training without $L_2$ regularization. In this case, the norm of the network weights increases in the optimization process, which leads to unstable training for large *elr*. In practice, training with large *elr* results in instantaneous destabilization and divergence of the weights in *NaN*. The phase diagrams in Fig. 26 for lower learning rates demonstrate behavior similar to the first phase, including its two subphases. By the end of training, the dynamics becomes much more noisy as compared with ConvNet, but the linear trend of the lines in the phase diagram still persists.

The metrics in Fig. 27 demonstrate a decreasing trend with an increase in *elr*, but, because of the large variance, the metrics for the training set are unstable. Nevertheless, it can be clearly seen that optimal generalization is achieved for the highest *elr* in the first phase in the same place where the curvature attains its minimum.

When $L_2$ regularization is added to the model, even for small values of weight decay ($wd = 10^{-4}$), the network becomes much more stable and the range of admissible *elr* expands.

Now, in the top panel in Fig. 28, we can see first phase and several models from the second phase, which do not pass into the domain of low gradients. Moreover, for models with large *elr* in the first phase, the loss function values exhibit a turn at the end of training. Most likely, this effect is associated with regularization, since, for small loss values, the weight decay begins to dominate in the optimization process. It can be seen that such regularization has a positive effect on the generalization: the model converges to a wider optimum with better test quality $E_t$. An analysis of the metrics depicted in the lower panels in Fig. 29 confirms the hypothesis that the best generalization is achieved for the maximum value of *elr* in the first phase. By analogy with the results for ConvNet, all the metrics in the first phase monotonically decrease with growing *elr*. Moreover, the metrics for models in the second phase reach a constant level not from the start of training, but, like in the case of MSE, first demonstrate a decrease.

## CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

## OPEN ACCESS



## REFERENCES

1. L. Hui and M. Belkin, "Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks" (2021). https://doi.org/10.48550/arXiv.2006.07322
2. L. N. Smith, "Cyclical learning rates for training neural networks" (2015). https://doi.org/10.48550/arXiv.1506.01186
3. S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" (2018). https://doi.org/10.48550/arXiv.1805.11604
4. F. He, T. Liu, and D. Tao, "Why ResNet works? Residuals generalize" (2019). https://doi.org/10.48550/arXiv.1904.01367
5. P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization" (2020). https://doi.org/10.48550/arXiv.2010.01412
6. Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio, "Fantastic generalization measures and where to find them" (2019). https://doi.org/10.48550/arXiv.1912.02178
7. Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers" (2018). https://doi.org/10.48550/arXiv.1811.04918

8. V. Badrinarayanan, B. Mishra, and R. Cipolla, "Understanding symmetries in deep networks" (2015). https://doi.org/10.48550/arXiv.1511.01029
9. A. S. Bosman, A. Engelbrecht, and M. Helbig, "Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions," Neurocomputing **400**, 113–136 (2020). https://doi.org/10.1016/j.neucom.2020.02.113
10. A. Demirkaya, J. Chen, and S. Oymak, "Exploring the role of loss functions in multiclass classification," *2020 54th Annual Conference on Information Sciences and Systems* (*CISS*), (2020).
11. V. Muthukumar, A. Narang, V. Subramanian, M. Belkin, D. Hsu, and A. Sahai, "Classification vs regression in overparameterized regimes: Does the loss function matter?" (2021). https://doi.org/10.48550/arXiv.2005.08054
12. V. Thomas, F. Pedregosa, B. van Merriënboer, P.-A. Manzagol, Y. Bengio, and N. L. Roux, "On the interplay between noise and curvature and its effect on optimization and generalization," *ConferenceProceedings of the 23rd International Conference on Artificial Intelligence and Statistics* (2020). https://doi.org/10.48550/arXiv.1906.07774
13. E. Lobacheva, M. Kodryan, N. Chirkova, A. Malinin, and D. Vetrov, "On the periodic behavior of neural network training with batch normalization and weight decay" (2021). https://doi.org/10.48550/arXiv.2106.15739
14. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition" (2014). https://doi.org/10.48550/arXiv.1409.1556
15. P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt" (2019). https://doi.org/10.48550/arXiv.1912.02292

*Translated by I. Ruzanova*