Exploring L2-Phase Transitions on Error Landscapes

Ibrahim Talha Ersoy, **Karoline Wiesner**
Complexity Science Group, Institute of Physics and Astronomy, University of Potsdam, Potsdam, Germany
talha.ersoy@uni-potsdam.de
karoline.wiesner@uni-potsdam.de

July 11, 2025

Abstract

When neural networks (NNs) are subject to L2 regularization, increasing the regularization strength beyond a certain threshold pushes the model into an under-parameterization regime. This transition manifests as a first-order phase transition in single-hidden-layer NNs and a second-order phase transition in NNs with two or more hidden layers. This paper establishes a unified framework for such transitions by integrating the Ricci curvature of the loss landscape with regularizer-driven deep learning. First, we show that a curvature change-point separates the model-accuracy regimes in the onset of learning and that it is identical to the critical point of the phase transition driven by regularization. Second, we show that for more complex data sets additional phase transitions exist between model accuracies, and that they are again identical to curvature change points in the error landscape. Third, by studying the MNIST data set using a Variational Autoencoder, we demonstrate that the curvature change points identify phase transitions in model accuracy outside the L2 setting. Our framework also offers practical insights for optimizing model performance across various architectures and datasets. By linking geometric features of the error landscape to observable phase transitions, our work paves the way for more informed regularization strategies and potentially new methods for probing the intrinsic structure of neural networks beyond the L2 context.

1 Introduction

L2 regularization is a common method used to avoid overfitting in different optimization tasks from simple linear regression [12] to the usage in neural networks (NNs) [11]. However the phenomena associated with L2 regularization in the context of training and understanding NNs goes beyond the usage as a tool for avoiding overfitting. A phenomenon occurring in L2 regularized NNs is the phase transition when the regularization strength is varied. A study investigating this by employing a Statistical Physics analogy is offered by Ziyin, Ueda et al. [5], who are using a loss function based on quadratic error and L2 regularization. They demonstrate that phase transitions occur when entering under-parameterization in simple linear neural networks, identifying second-order phase transitions in shallow networks and first-order transitions in deeper ones. These insights extend to non-linear networks only near the transition point, i.e. the onset of learning. Another approach studying phase transitions in similar set-ups is the information bottleneck (IB) [1].The training maximizes a mutual information-based accuracy term. The so called information bottleneck is defined by the mutual information of some latent representation and the input of the networks is minimized. Varying the bottleneck strength gives rise to phase transitions [3]. However, we will not pursue the IB approach further due to limited applicability of its theoretical predictions. Additionally, information geometric approaches, as discussed by Amari et al. [7] and Watanabe et al. [20], present powerful frameworks for analyzing the error landscape of neural networks. This perspective treats the parameter space of neural networks as defining a surface whose geometry reflects the models behavior throughout training, underscoring the significance of degeneracy and the non-convex nature of the loss landscape. Our work builds upon and extends these existing approaches by investigating the geometric interpretation of L2-regularization-induced phase transitions in neural networks. We unify the information geometric and regularization perspectives, offering a novel framework for understanding the dynamics of neural network training. Specifically, our contributions include the reproduction and extension of Ueda et al.'s [5] results, provid-

ing a detailed geometric interpretation of the phase transitions they observed and the identification of transitions to under-parameterized regimes corresponding to measurable geometric change-points in the error surface. We then conduct the observations of additional transitions for complex datasets, indicating the emergence of geometric substructures that represent learned features. We hypothesize, that by measuring the curvature of the error landscape along this path we can link the onset of learning to a change in accuracy regimes of the error landscape. Furthermore we predict existence of additional transitions when the data complexity is increased. Again this transition comes with a change in accuracy regimes and a geometric change-point in the error landscape. We confirm both hypotheses numerically using geometric quantities that we derive mathematically. Our findings provide a deeper understanding of phase transitions influenced by the interplay of regularization and data complexity. This work not only has potential practical implications for phenomena associated with over- and under-parametrization but also furthers our understanding of the geometry and structure of error landscapes and the notion of accuracy regimes.

1.1 Curvature Change-points at the onset of learning

We now examine the effects of L2 regularization on neural-network training, focusing on phase transitions at the onset of learning, i.e. at the transition from a trivial to a non-trivial NN model. Here, NNs with a single hidden layer exhibit different behavior from those with two or more hidden layers, i.e. Deep Neural Networks (DNNs): At the onset of learning, DNNs undergo first-order phase transitions, in contrast to the second-order transitions exhibited by single-hidden layer NNs [5]. We show in the following that these phase transitions can be understood as changes in Ricci curvatures of the error landscape. We, thus, link the observed phase transitions to the geometry of the error landscape irrespective of the regularization strength. As a consequence, they can be understood as the DNN model transitioning from one accuracy 'basin' to another through a curvature-change 'gate'.

The L2-regularized loss function was defined as: $\mathcal{L}_{reg}(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) = \mathcal{L}(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) + \beta\|\theta\|^2$, where $\mathcal{L}(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}))$ is the (unregularized) error function (here the mean-squared error) and β the regularization strength. Increasing β shifts the loss landscape's global minimum towards the origin, away from the minimum of the (unregularized) error landscape. As β become sufficiently large, the trained model suddenly transitions from a useful model to a "trivial model" with all parameters θ near zero.

This transition, termed the *onset of learning*, manifests as a first-order phase transition in DNNs and as a second-order transition in single-layer NNs [5]. Fig. 1 illustrates the change of the loss function as the NN passes through an onset of learning, showing how increasing β moves the error-minimizing model out of the "learnable" region. A look at the L2 term reveals its independence of the data, simply warping the loss landscape and smoothing out details. Our object of interest, however, is the error landscape. It remains the same for all regularizer strengths. Increasing the L2 can thus be seen as the model getting shifted on the error landscape, toward the origin. We hypothesize that the model, with increasing regularization, traverses through a set of basins in the error landscape as it approaches the origin. Specifically, we hypothesize that (i) as the model moves away from a minimum of the error landscape, the curvature gradually increases, and (ii) as the model leaves the basin of that minimum and enters the basin of another there is a sudden change in curvature of the error landscape at that point which manifests as phase transition in the model accuracy. We first present the experimental results for a one-hidden layer NN, trained on Gaussian data with 2D input $\mathbf{X}$ and 1D output data $\mathbf{Y}$ both sampled from a 3D Gaussian distribution $(\mathbf{X}, \mathbf{Y}) \sim \mathcal{N}(0, \Sigma_{xy})$ with Σ_{xy} the joint covariance. The model is trained with an MSE error function and an added L2 term with regularizer strength β that is successively increased. (see Appendix C.1 for details).

In Fig. 2a we can read of the onset of learning phase transition at β_0, as described by Ziyin and Ueda [5]. It shows a smooth transition as we would expect for a second order phase transition The Ricci Scalar in Fig. 2b shows a clear change as the model moves toward the transition point at the onset of learning β_0. After the transition the curvatures of course remain constant as the model doesn't move any more. We interpret the transition point as delineating two distinct accuracy regimes.

As we increase the data complexity, we expect the error landscape to develop additional substructures. In line with the information geometry literature [19], we hypothesize that new information is associated with the emergence of new minima, associated with new basins (ii). A consequence is that additional transitions may occur and be observable when a given model exits these. To test this, we utilize an L2 regularizer to shift a trained model out of the minimum toward the origin, proposing that a

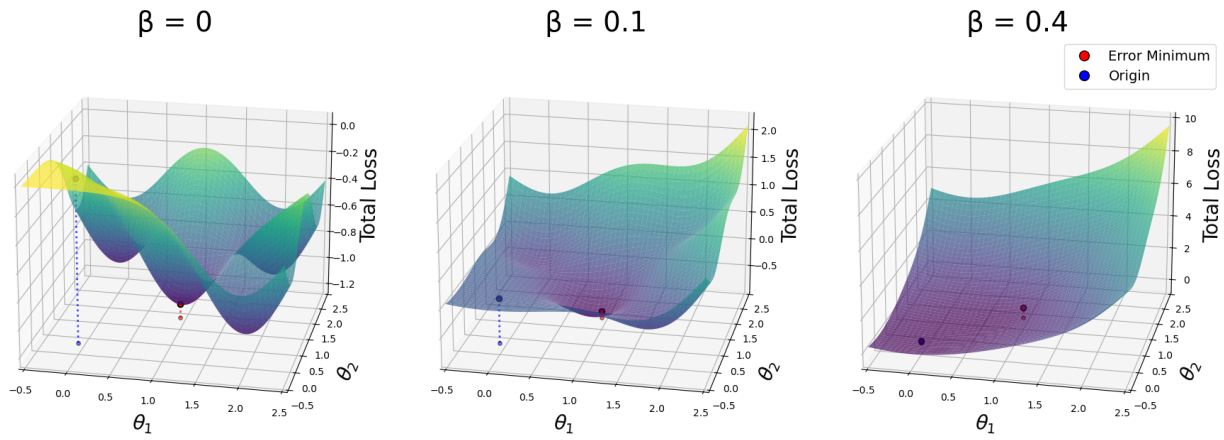
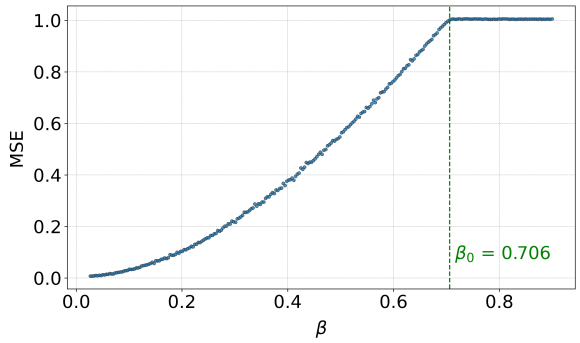
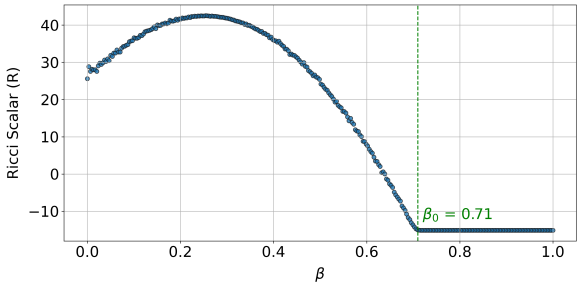


Figure 1: Sketch of a loss landscape with increasing L2 regularizer strength β. The minimum of the error term (red) and the minimum of the L2 term at the origin (blue) are marked. Increasing β progressively smooths the landscape, eliminating local minima while shifting the global minimum toward the origin.

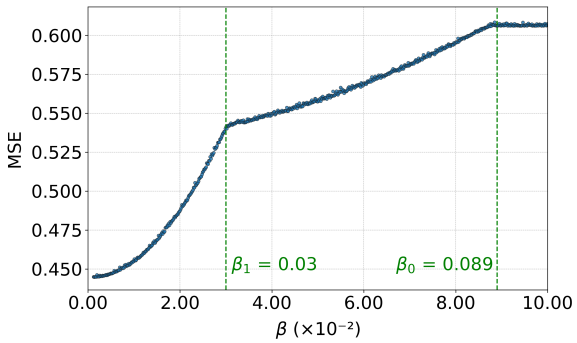


(a) MSE for increasing regularizer strength β

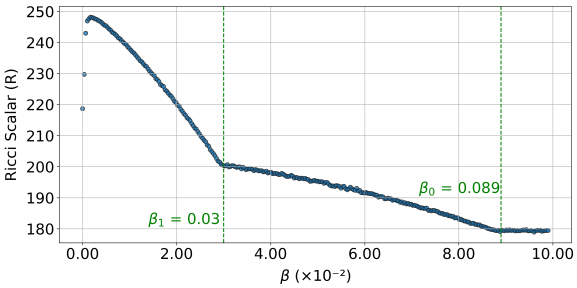


(b) Scalar Curvature (Ricci Scalar) of the error surface for increasing regularizer strength β

Figure 2: Metrics of a NN with one hidden layer, trained on 1D Gaussian data with increasing regularizer strength. Onset of learning determined with change-point detection and marked as β_0



(a) MSE for increasing regularizer strength β



(b) Scalar Curvature (Ricci Scalar) of the error surface for increasing regularizer strength β

Figure 3: Metrics of a NN with one hidden layer, trained on 2D Gaussian data with increasing regularizer strength. Onset of learning and second transition point determined with change-point detection and marked as β_0 and β_1

second transition point will be observable. We further propose that the second transition point will be accompanied by a curvature change-point, resulting from the model exiting a basin.

The experimental set-up is similar to the previous one, with the important difference that the output data $\mathbf{Y}$ is now two-dimensional, drawn from a 3-dimensional multivariate Gaussian distribution,

$(\mathbf{X}, \mathbf{Y}) \sim \mathcal{N}(0, \Sigma_{xy})$ (see Appendix C.1for further details). Again, we record, for each trained model, (a) the MSE, (b) the (Ricci) Scalar curvature at the model's position on the loss surface. The results are shown in Figs. 3 as a function of β. Fig. 3a depicts the MSE as a function of β for the one-hidden-layer and two-hidden-layer networks respectively. Again we see a smooth transition at the onset of learning β_0. But we now have an additional transition point at β_1. The curvature plots again display changes at the transition points. The Ricci Scalar displayed in Fig. 3b exhibits the same behavior at the onset of learning β_0 as well as the additional transition points β_1 as in the 1D case presented in the previous section.

We now repeat both experiments for the a Neural Network with two hidden layers.

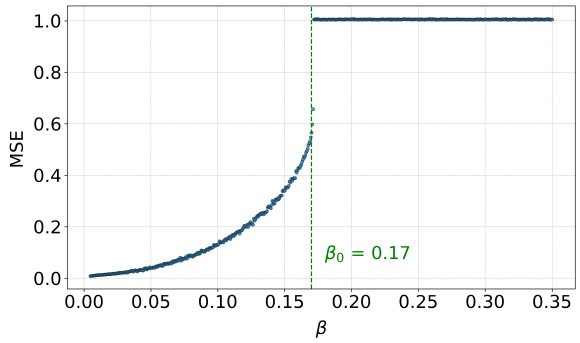
Again we see one transition point at onset of learning β_0 in Fig. 4a for the model trained on the 1D Gaussian as well as 2D case Fig. 5a. The main difference is the sharp transition that comes with a jump [1]. The observation are in accordance with a first order phase transition. In the 2D case Fig. 5a we again see an additional transition point β_1. The curvature plots in Figs. 4b and 5b show us a very similar phenomenology with change-points before the transition points β_i.

Conclusion

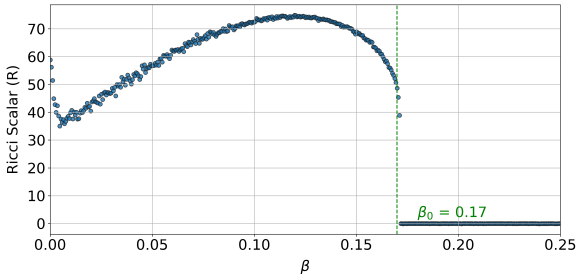
This study has provided insights into the geometric interpretation of phase transitions in neural networks, particularly focusing on the existence and exploration of accuracy regimes utilizing L2 regularization. Our investigation highlighted the critical role of the error landscape's geometric structure. We started with the known phenomenon of the 'onset-of-learning' transition and made a connection to the geometry of the error landscape. We showed that the transition is due to the model passing from a low-accuracy to a higher-accuracy regime. The two regimes are separated by a curvature change point. We interpreted the higher accuracy regime as a basin. We confirmed this and found further transitions for both shallow and deep architectures, identifying a mathematical link between regime transitions and the distinct curvature profile of the error landscape. To further demonstrate the geometric nature of the phenomena and to point out its independence from the specific L2 setup, we constructed a VAE type model where another mechanism is used to push the models around in the error landscape. Again we could clearly see that the model first enters a lower accuracy regime. When β is further increased the model also exists this regime and enter the trivial regime at the 'onset-of-learning'. An important note we need to make here is on the global error geometry. We have only investigated the local geometry along a specific path that start in the highest accuracy section we could find and ends at a parameter space section that corresponds to a trivial model. We can not deduce details on the global structure of the error landscape. Even though we have identified distinct accuracy regimes, we can not tell whether all regimes and change-points in the error submanifold are detected. The goal however is not a complete analysis but the establishment of a framework that allows us connect the transition phenomenon to accuracy regimes and geometric metrics. This can be build upon to come up with more elaborate schemes to explore the full implications. Also the notion of error basins linked to the accuracy regimes have to be taken with a grain of salt. We have merely demonstrated their existence. We can not make any predictions on their overall structure and the geometry of the basin boundaries. Due to the build-in degeneracy in NNs [2] we expect the basins to consist of valleys with complex and non-trivial boundary structures. Our framework gives a starting point for further investigations and establishes same basic facts.

[1] We expect a jump instead of a kink because of the learning algorithm. The path in parameter space is not continuous but comes in discrete steps that can lead to an earlier transition

[2] All NNs are degenerate and over-parametrized in the sense that they have permutation symmetries due to the architecture of the model. For every possible model there is always a number of node permutations, as well as symmetries of the chosen activation function, that will give the equivalent result

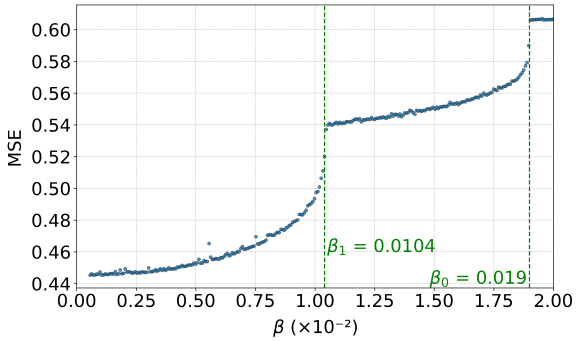


(a) MSE for increasing regularizer strength β

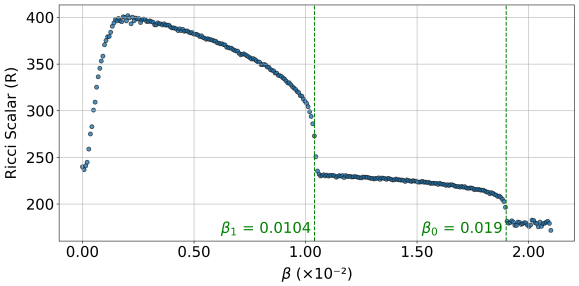


(b) Scalar Curvature (Ricci Scalar) of the error surface for increasing regularizer strength β

Figure 4: Metrics of a NN with one hidden layer, trained on 1D Gaussian data with increasing regularizer strength. Onset of learning determined with change-point detection and marked as β_0



(a) MSE for increasing regularizer strength β



(b) Scalar Curvature (Ricci Scalar) of the error surface for increasing regularizer strength β

Figure 5: Metrics of a NN with two hidden layers, trained on 2D Gaussian data with increasing regularizer strength. Onset of learning and second transition point determined with change-point detection and marked as β_0 and β_1

References

[1] N. Tishby, F. Pereira, and W. Bialek, "The Information Bottleneck Method," *arXiv preprint arXiv:0004057*, 2000.

[2] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss, "Information bottleneck for Gaussian variables," *Advances in Neural Information Processing Systems*, vol. 16, 2003.

[3] T. Wu, I. Fischer, I. L. Chuang, and M. Tegmark, "Learnability for the information bottleneck," in *Uncertainty in Artificial Intelligence*, pp. 1050–1060, PMLR, 2020.

[4] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[5] L. Ziyin and M. Ueda, "Zeroth, first, and second-order phase transitions in deep neural networks," *Physical Review Research*, vol. 5, no. 4, p. 043243, 2023.

[6] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[7] S. Amari, *Information Geometry and Its Applications*, vol. 194, Springer, 2016.

[8] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[9] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Artificial Intelligence and Statistics*, pp. 192–204, PMLR, 2015.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[12] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[13] E. Golikov, A. Jacot, C. Hongler, and F. Gabriel, "Feature Learning in L2-regularized DNNs: Attraction/Repulsion and Sparsity," in *NeurIPS 2022*, 2022.

[14] T. Wu and I. S. Fischer, "Phase Transitions for the Information Bottleneck in Representation Learning," *CoRR*, vol. abs/2001.01878, 2020.

[15] A. Montanari and P. R. Sgc, "Information-Driven Learning from High-Dimensional Data," *Annual Review of Statistics and Its Application*, vol. 6, pp. 121–146, 2019.

[16] L. Sagun, U. Evci, V. U. Güney, Y. N. Dauphin, and L. Bottou, "Empirical Analysis of the Hessian of Over-Parametrized Neural Networks," *CoRR*, vol. abs/1706.04454, 2017.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] L. Ziyin and M. Ueda, "Zeroth, first, and second-order phase transitions in deep neural networks," *Phys. Rev. Res.*, vol. 5, no. 4, p. 043243, 2023.

[19] S. Watanabe, *Algebraic Geometry and Statistical Learning Theory*, vol. 25, Cambridge University Press, 2009.

[20] S. Watanabe, "Review and Prospect of Algebraic Research in Equivalent Framework between Statistical Mechanics and Machine Learning Theory," *arXiv preprint arXiv:2406.10234*, 2024.

[21] R. Sun, D. Li, S. Liang, T. Ding, and R. Srikant, "The global landscape of neural networks: An overview," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 95–108, 2020.

[22] C. Stephenson, S. Padhy, A. Ganesh, Y. Hui, H. Tang, and S. Chung, "On the geometry of generalization and memorization in deep neural networks," *arXiv preprint arXiv:2105.14602*, 2021.

[23] S. Lang, *Fundamentals of Differential Geometry*, vol. 191, Springer, 2012.

[24] J. M. Lee, *Introduction to Riemannian Manifolds*, vol. 2, Springer, 2018.

[25] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, pp. 1885–1894, PMLR, 2017.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, Banff, Canada, 2013.

[27] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003.

A Geometry of the Model Space

A.0.1 Error and Loss Landscapes as Submanifolds in Model Space

Here, we derive and investigate the geometric properties of error and loss landscapes as d dimensional submanifolds of a $d+1$ Euclidean ambient space [23], defined by a smooth error or loss function as the embedding.

The *parameter space* $\Theta = \mathbb{R}^d$ is the high-dimensional space of all possible parameter configurations (weights and biases) for a given NN architecture. Each point $\boldsymbol{\theta} \in \Theta$ represents a specific set of weights and biases, i.e. a *model*. The parameter space is equipped with the Euclidean metric, with the previously defined L2 norm as the induced norm. We extend this parameter space to include an error dimension, resulting in $\Theta \times \mathbb{R} = \mathbb{R}^{d+1}$, which we

equip with a global coordinate map $(l, \theta_1, \ldots, \theta_d)$, with $l \in \mathbb{R}$ denoting the error coordinate and the Euclidean metric $e_{ab} = \delta_{ab}$ in index notation with $a, b = 0, ..., d$. We call this the *model space*. Given a smooth error function $\mathcal{L}(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}))$ evaluated on a dataset D, we can now define the error surface $\tilde{\mathcal{L}}$ as a Euclidean submanifold in the model space $\Theta \times \mathbb{R}$. The optimization (learning) process can be viewed as a trajectory on $\tilde{\mathcal{L}}$, where in practice some variation of gradient descent is used to locate the global minimum. In general, the surface is highly non-convex and contain saddle points [8] and degenerate global minima due to over-parametrization and the inherent symmetries [9] of the NNs. Despite the complexity of the geometry, we can assert some minimal assumptions, such as the existence of basins that correspond to higher accuracy in the model output and that are formed by the features present in the dataset. We now introduce some concepts describing basic differential geometry of the error landscape.

A.0.2 Fundamental Forms and Curvature of the Error Landscape

Denoting the error function as $\mathcal{L}(\mathbf{y}, \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) \equiv l(\boldsymbol{\theta})$, we express the error surface in parametric form as:

$$f(\theta_1, \ldots, \theta_d) = (l(\boldsymbol{\theta}), \theta_1, \ldots, \theta_d). \tag{1}$$

Alternatively, we can define it using an implicit function formulation:

$$F(l, \theta_1, \ldots, \theta_d) = l - l(\boldsymbol{\theta}) = 0 \tag{2}$$

We have the inclusion map for the submanifold in the ambient space 6:

$$\iota : \tilde{\mathcal{L}} \to \Theta \times \mathbb{R}. \tag{3}$$

Let T_pL be tangent space at some point $p \in \tilde{\mathcal{L}}$. For elements $v, w \in T_pL$, the scalar product $\langle v, w \rangle_p$ on $T_p\tilde{\mathcal{L}}$ is given by:

$$\langle v, w \rangle_p = (d\iota)_p(v)(d\iota)_p(w). \tag{4}$$

This is the *first fundamental form* in coordinate-free form[3], also known as the *pull-back* of the Euclidean metric in $d+1$-dimensional space to the submanifold. Given the coordinates $(l, \theta_1, ..., \theta_d)$ we get the basis of the tangent space $\left(\frac{\partial}{\partial l}, \frac{\partial}{\partial \theta_1}, ..., \frac{\partial}{\partial \theta_d}\right)$. With the Euclidean metric $e_{ij} = \delta_{ij}$ with $i, j = 1, ..., d$, [4] we get the components g_{ij} of the induced metric:

$$g_{ij} = \delta_{ij} + \frac{\partial l}{\partial \theta_i} \frac{\partial l}{\partial \theta_j} \ . \tag{5}$$

The first fundamental form gives us the scalar product and, hence, can be used to calculate lengths, angles and volumes on the submanifold. We now also derive the second fundamental form because it can be used to obtain the intrinsic curvature of the submanifold.

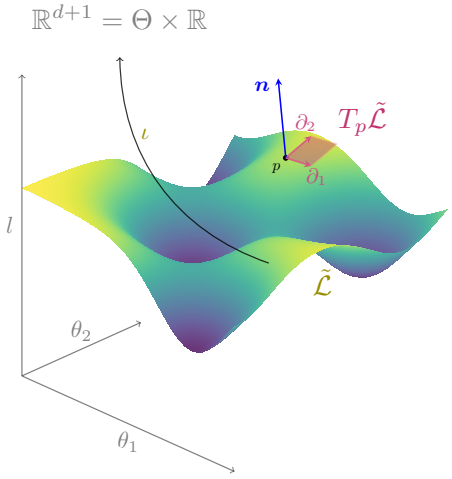


Figure 6: Sketch of an error surface $\tilde{\mathcal{L}}$ as a d dimensional submanifold of the Euclidean ambient space $\mathbb{R}^{d+1}$ consisting of the parameter space Θ and the error values. The tangent space $T_p\tilde{\mathcal{L}}$ at $p \in \tilde{\mathcal{L}}$ is spanned by $\partial_i = \frac{\partial}{\partial \theta_i}$.

To derive the second fundamental form, that measures the extrinsic curvature, we need to consider the normal space, which is one-dimensional, as we have a d-dimensional submanifold embedded in a (d + 1)-dimensional ambient space. With the implicit formulation given in Eq. 2, denoting $\partial_i = \frac{\partial}{\partial \theta_i}$ and $\|\nabla F\| = \sqrt{1 + \sum_{i=1}^{d} |\partial_i l|^2}$, the normal vector field $\mathbf{n}$ is given by:

$$\begin{aligned} \mathbf{n} &= \frac{-\nabla F}{\|\nabla F\|} = \frac{1}{\|\nabla F\|} (\partial_0 F, \partial_1 F, \cdots, \partial_d F) \\ &= \frac{1}{\|\nabla F\|} (1, -\partial_1 l, \cdots, -\partial_d l) \end{aligned} \tag{6}$$

With the parametrization f given in Eq. 1 and

[3] The first fundamental form is often denoted I_{ij}. As it gives us the induced metric, we use the notation g_{ij} which is more common in the physics literature.

[4] If not stated otherwise the indices run from 1 to d.

with $f_{ij} = \partial_i \partial_j f = (\frac{\partial^2 l}{\partial \theta_i \partial \theta_j}, 0, \cdots, 0)$ the second fundamental form II_{ij} is:

$$\mathrm{II}_{ij} = \mathbf{n} \cdot f_{ij} = \frac{1}{\|\nabla F\|} (1, -\partial_1 l, \cdots, -\partial_d l) \times \left(\frac{\partial^2 l}{\partial \theta_i \partial \theta_j}, 0, \cdots, 0 \right) = \frac{H_{ij}}{\|\nabla F\|} , \tag{7}$$

where H_{ij} is the Hessian of the loss function, scaled by the norm of the gradients plus one. This is a useful relation as the gradients are a key component in training and thus easily available for each trained NN. The Hessian is more expensive to obtain but cheap enough for the NNs we are investigating. As a symmetric bilinear form, the second fundamental form can be diagonalized at any point in the submanifold with eigenvectors and -values $v^{(i)}, \kappa^{(i)}$. The eigenvalues are referred to as principal curvatures. We can easily see that the eigenvectors of the second fundamental form and the Hessian are the same:

$$\kappa^{(i)} v^{(i)} = \mathrm{II} \cdot v^{(i)} = \frac{-1}{\|\nabla F\|} H \cdot v^{(i)} = \frac{-1}{\|\nabla F\|} \tilde{\kappa}^{(i)} v^{(i)} \tag{8}$$

We get the relation between the principal curvatures $\kappa^{(i)}$ and the Hessian eigenvalues $\tilde{\kappa}^{(i)}$:

$$\kappa^{(i)} = \frac{-1}{\|\nabla F\|} \tilde{\kappa}^{(i)} \tag{9}$$

Consequently, the principal curvatures can be readily obtained from the eigenvalues of the Hessian and the gradients of the error function, $\partial_i l$. With the determinant of the second fundamental form and the gradients, the Gauss-Kronecker curvature can be obtained:

$$K = \frac{\det(\mathrm{II})}{\det(g)} = \frac{1}{\|\nabla F\|^d} \frac{\prod_{i=1}^{d} \tilde{\kappa}^{(i)}}{\det(g)} . \tag{10}$$

The Problem with the Gauss-Kronecker curvature is the product over the large number of eigenvalues. Most eigenvalues are close to zero. The curvature can thus only be obtained by introducing a cutoff. Even then we get numerically unstable results. We thus derive a more stable curvature measure that scales better in higher dimensions. I.e. we use the second fundamental form to derive the Riemann-curvature. For Euclidean submanifolds the Gauss Equation [24] relating the second fundamental form (extrinsic curvature) to the Riemann tensor (intrinsic curvature) gives us the following expression for the Riemann-curvature. We start with a generalization of the Theorema Eggregium for arbitrary dimensions, also called Gauss-Equation. It related the Riemann Curvature to the second fundamental form. For Euclidean submanifolds it is given as:

$$R_{lijk} = \mathrm{II}_{ik} \mathrm{II}_{jl} - \mathrm{II}_{ij} \mathrm{II}_{kl} \tag{11}$$

The Riemann tensor can be contracted to get the Ricci tensor. In the Einstein summation notation Ricci curvature tensor is then given with the inverse of the metric g^{ij}:

$$R_{ij} = R^k_{ikj} = g^{kl} \Pi_{kl} \Pi_{ij} - \Pi_{ik} g^{km} \Pi_{mj} \tag{12}$$

Contracting the indices again we get the Ricci scalar:

$$\begin{aligned} R = g^{ij} R_{ij} &= g^{kl} \Pi_{kl} g^{ij} \Pi_{ij} - g^{ij} \Pi_{ik} g^{km} \Pi_{mj} \\ &= \tilde{H}^2 - |\Pi|^2, \end{aligned} \tag{13}$$

with the mean curvature $\tilde{H} = g^{ij} \Pi_{ij}$ and the squared norm of the second fundamental form $|\Pi|^2 = g^{ij} g^{kl} \Pi_{ik} \Pi_{jl}$. We use the the Sherman-Morrison Formula to give inverse of the metric:

$$g^{ij} = \delta^{ij} - \frac{\partial_i l \partial_j l}{\|\nabla F\|^2} \tag{14}$$

With $\Pi_{ij} = \frac{H_{ij}}{\|\nabla F\|}$ Plugging this into the mean curvature we get:

$$\begin{aligned} \tilde{H}^2 &= \frac{1}{\|\nabla F\|^2} \left[\left(\delta^{ij} - \frac{\partial_i l \partial_j l}{\|\nabla F\|^2} \right) H_{ij} \right]^2 \\ &= \frac{1}{\|\nabla F\|^2} \left[tr(H) - \frac{\nabla l^T H \nabla l}{\|\nabla F\|^2} \right]^2 \\ &= \frac{1}{\|\nabla F\|^2} \left(tr(H) \right)^2 - 2 \frac{tr(H) \nabla l^T H \nabla l}{\|\nabla F\|^4} \\ &\quad + \frac{\left(\nabla l^T H \nabla l \right)^2}{\|\nabla F\|^6} \end{aligned} \tag{15}$$

For the second term we get:

$$\begin{aligned} \frac{1}{\|\nabla F\|^2} g^{ij} g^{kl} \Pi_{ik} \Pi_{jl} &= \frac{1}{\|\nabla F\|^2} (\delta^{jk} \delta^{il} \\ -\delta^{jk} \frac{\partial_i l \partial_l l}{\|\nabla F\|^2} - \delta^{ij} \frac{\partial_j l \partial_k l}{\|\nabla F\|^2} &+ \frac{\partial_j l \partial_k l \partial_i l \partial_l l}{\|\nabla F\|^4}) H_{jk} H_{il} \\ &= \frac{tr(H)^2}{\|\nabla F\|^2} - \frac{2 \nabla l^T H^2 \nabla l}{\|\nabla F\|^4} + \frac{\left(\nabla l^T H \nabla l \right)^2}{\|\nabla F\|^6} \end{aligned} \tag{16}$$

With Eq.s 15 and 16 we get the final expression (for an alternative derivation see [22]):

$$\begin{aligned} R = &\frac{1}{\|\nabla F\|} (tr(H)^2 - tr(H^2)) \\ &+ \frac{2}{\|\nabla F\|^2} \nabla l^T (H^2 - tr(H) H) \nabla l \end{aligned} \tag{17}$$

The Ricci curvature in the experimental results is obtained using this expression.

A.1 Curvature and Fisher Information

A.1.1 Metric of the Error Submanifold and the Fisher Information

Consider a Deep Neural Network (DNN) with parameters θ, input $\mathbf{x}$, and output $\mathbf{f}(\mathbf{x};\boldsymbol{\theta})$ and the target variable $\mathbf{y}$ $\mathbf{x}$: We have a mean-squared error loss $\mathrm{MSE}(\mathbf{y},\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_i))^2$and Gaussian error with σ^2 is the variance. The likelihood of observing $\mathbf{y}$ given $\mathbf{x}$ and θ is [27]:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) &= \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{l}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y}-\mathbf{f}(\mathbf{x};\boldsymbol{\theta})\|^2\right), \end{aligned} \tag{18}$$

where n is the dimensionality of $\mathbf{y}$. The log-likelihood is:

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = &-\frac{n}{2}\log(2\pi\sigma^2) \\ &-\frac{1}{2\sigma^2}\frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 . \end{aligned} \tag{19}$$

A.1.2 Fisher Information Matrix

The Fisher Information Matrix (FIM) quantifies the amount of information that the observable random variable $\mathbf{y}$ carries about the parameters $\boldsymbol{\theta}$. The FIM is defined as the expected outer product of the score function:

$$\mathbf{I}(\boldsymbol{\theta}) = \mathbb{E}\left[\mathbf{S}(\boldsymbol{\theta})\mathbf{S}(\boldsymbol{\theta})^\top\right],$$

where the score function $\mathbf{S}(\boldsymbol{\theta})$ is the gradient of the log-likelihood with respect to $\boldsymbol{\theta}$:

$$\mathbf{S}(\boldsymbol{\theta}) = \frac{\partial \log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}.$$

For the likelihood (Eq. 18), the score function is:

$$\mathbf{S}(\boldsymbol{\theta}) = \frac{1}{\sigma^2}(\mathbf{y}-\mathbf{f}(\mathbf{x};\boldsymbol{\theta}))\frac{\partial\mathbf{f}(\mathbf{x};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} = \frac{1}{\sigma^2}\frac{\partial l}{\partial\boldsymbol{\theta}} \tag{20}$$

Substituting the score function into the definition of the FIM, we obtain:

$$\mathbf{I}(\boldsymbol{\theta}) = \mathbb{E}\left[\frac{1}{\sigma^4}(\mathbf{y}-\mathbf{f}(\mathbf{x};\boldsymbol{\theta}))^2\frac{\partial\mathbf{f}(\mathbf{x};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\frac{\partial\mathbf{f}(\mathbf{x};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}^\top\right]. \tag{21}$$

With l the loss over all of the dataset we can write this in components as:

$$\mathbf{I}_{ij} = \frac{1}{\sigma^2}\frac{\partial l}{\partial\theta_i}\frac{\partial l}{\partial\theta_j} \tag{22}$$

This is precisely the second term of the Eq. 5 under the assumptions of Gaussian noise and squared error function times the noise parameter. The FIM gives us the geometric properties of the log-likelihood surface, while our metric is the metric of the error submanifold. The relation holds for normal error and quadratic error terms. Both spaces are locally isometric as they give the same curvature values as can easily be checked.

B Classification and Variational Autoencoders

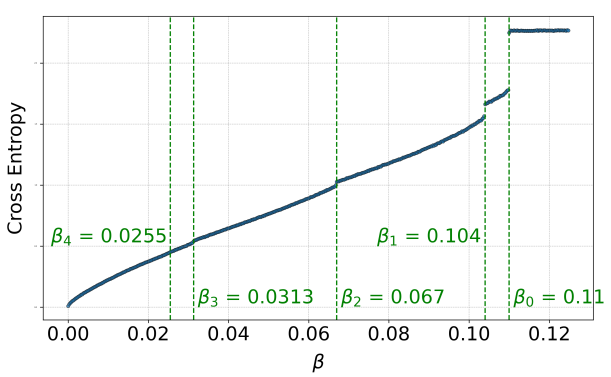
We also looked at a vastly higher dimensional case, i.e. a classifier. The setup is very similar, but instead of an MSE error we use a cross entropy error $\mathrm{CE}(\mathbf{y},\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}))$ and again add a regularizer with strength β train a model with a total loss $\mathcal{L}_{reg}(\mathbf{y},\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) = \mathrm{CE}(\mathbf{y},\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) + \beta\|\theta\|^2$, successively increasing the regularizer strength. The dataset is the MNIST dataset that consists of images of handwritten numerals from zero to ten and the corresponding labels. In Figs. 7a and 7b we can clearly see the onset of learning, again labeled β_0 and four additional transition points, in the error (Cross-Entropy) as well as the accuracy (percentage of correct classifications) against β.[5] The input has 784 nodes giving us a problem that is not only fundamentally different as it is a classification task, but also quite far from being a mere toy model as the previous 1- and 2- dimensional Gaussian datasets. We plot the Cross-Entropy, as well as the accuracy [6] against β and mark the change-points. We finally look at a different set-up without an L2 regularizer. We trained a VAE [26] like model (see Appendix C.2) with the loss of the form:

$$\mathcal{L} = \mathrm{MSE}(\mathbf{y},\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) + \beta D_{KL}(p|q) \tag{23}$$

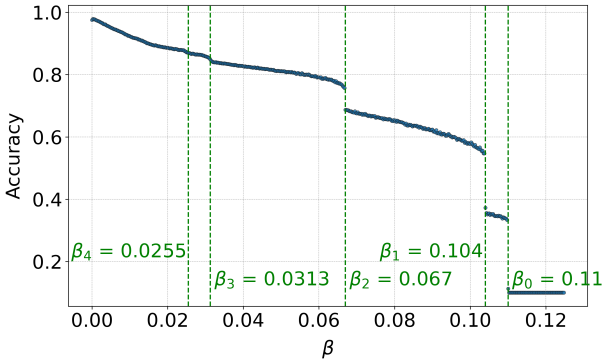
with $p_\theta(l|x)$ the encoding map [26] and $q \propto \mathcal{N}(0,\mathbf{1})$ white noise. The model is trained on the same

[5] We do not give a precise interpretation of what is happening at these transition points, as that would be beyond the scope of our investigations here, but demonstrate that they exist.

[6] The accuracy is measures as the fraction of the correctly classified images in the test dataset



(a) Cross-Entropy for increasing regularizer strength β



(b) Accuracy, i.e. the fraction of correctly classified datapoints, for increasing regularizer strength β

Figure 7: Cross-Entropy and Accuracy of a NN with two hidden layers, trained on the MNIST dataset for image classification. The metrics are evaluated for increasing L2 regularization strength β. The transition points are again determined with some change-point detection algorithm and marked as β_i

.

dataset $(\mathbf{X}, \mathbf{Y}) \propto \mathcal{N}(0, \Sigma_{xy})$, with the same error function as before. Increasing the β parameter pushes the model away from the error minimum toward the some trivial model that outputs noise. This gives us a mechanisms that is very different from the L2. It introduces a second term to the loss function that has a minimum at a model that has its minium defined by another model representation that is not defined by a location in the parameter space but by the neural network that has a latent representation that is closest to white noise. This gives us a more costly and less direct method of manipulating the model. However it gives a comparable manipulation of the model. We start at the MSE minimum determined by the data and gradually shift the effective minimum and thus the model toward a trivial model. The KL-divergence measures a sort of distance, but it is not the euclidean distance to the origin in parameter space but the distance to the trivial model [7]. In the error(MSE)-β plot (Fig. 8) we see very similar phenomenology to the L2 regularized models. This is to be expected as we argued that the transition is just a consequence of the model traversing the geometric change-points in the error landscape. Given that we have the same error function and the same dataset we expect the error landscape to have a very similar (not exactly the same as the parametrization and the precise model is a bit different) geometry.

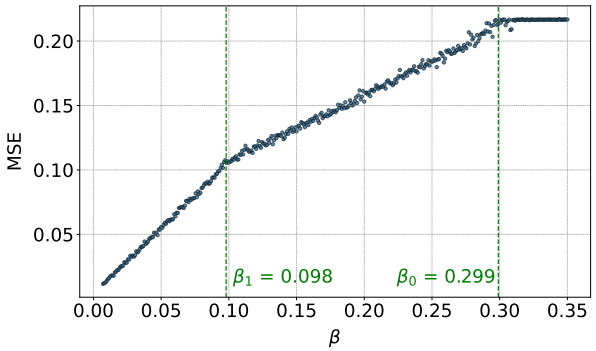


Figure 8: MSE for increasing regularizer strength β. The regularizer is the KL-divergence term (Eq. 23) of the VAE like model instead of the L2

C Experimental Setup

C.1 Setup 1: Simple Neural Network with L2 Regularization

In the first experimental setup, a simple feedforward neural network was implemented with a varying regularization parameter denoted by β. The main experiment consists in a loop where a network is trained tested and then trained again with an increasing β value. A specific β interval and the number of networks to be trained is set in the beginning. There are two possible modes. The first is the annealing mode where the parameters of the previous model are taken as the initialization of the following

[7] We need to note that the KL-divergence is not a proper measure of distance at it is not symmetric

model. Without annealing a new network is trained for each value.

C.1.1 Architecture

The network consists of an input layer, n hidden layers, and an output layer. The input layer has two or more dimensions corresponding to the features, and the output layer has one or more dimensions representing the target variable. For the main experiments the hidden layer contains 15 neurons and utilizes the Sigmoid activation function. The choice of width is just taken by running a number of trials. Above 15 there is no significant improvement of the performance. Decreasing it for the non-annealing mode has the effect that there is an overlap of phase. Close to transition point some of the models with lower than critical β converge to the highest possible accuracy while others get stuck at lower accuracy regimes.

C.1.2 Data Generation

We used Gaussian data with zero mean for all of the experiments. The covariance matrix used is of the form:

$$\Sigma_{xy} = \begin{bmatrix} 1 & \rho & \rho & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \rho \\ \rho & \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \cdots & 1 \end{bmatrix} \tag{24}$$

Where as long $\rho \in (0, 1)$ is fulfilled the matrix is positive definite. ρ determines the correlation strength between the different dimensions. For the 1D outputs we took the very simple cov matrix:

$$\Sigma_{xy} = \begin{bmatrix} 1 & 0.999 & 0.999 \\ 0.999 & 1 & 0.999 \\ 0.999 & 0.999 & 1 \end{bmatrix} \tag{25}$$

For generalizability of the setup to higher dimension we use the Cholesky decomposition for sampling. The sampling process begins with generating a set of raw data samples from a standard multivariate normal distribution, which includes generating random values with zero mean and unit variance. To shape these uncorrelated samples into a distribution with the desired properties, we apply the Cholesky decomposition of the specified covariance matrix. This transformation adjusts the samples so they reflect the variability and correlation defined by the covariance. Finally, we add the mean vector (zero in our experiments) to each sample, ensuring that the samples are centered around the specified mean values. This method ensures the resulting samples faithfully represent the intended multivariate Gaussian distribution characterized by the given mean and covariance. The first two dimensions are taken as input (x_1, x_2) and the last dimension y as the output. The sample size for training and testing is $N = 10^4$ each. This is chosen by trying a number of different orders of magnitude. Above this the performance does not improve significantly. To test the robustness of the findings we used the following scheme to generate other higher dimensional covariance matrices. We created a diagonal matrix of the desired shape and rotated in the given axis by random very small increments until we get some covariance matrix with a high enough correlation strength (close to one) between the x and y dimensions. This approach makes the scheme of choosing a distribution relatively random and ensures that the resulting distribution is positive definite. For the 2D and 3D experiments we picked the first of the randomly generated covariances.

C.1.3 Training Process

The model was trained for a specified number of epochs, with a varying β value in each iteration to examine the impact of regularization on performance. The training utilized an optimizer (configured as either SGD, Adam, or AdamW) to minimize the loss function, which combined the mean-squared error (MSE) and the L2 regularization term. The AdamW algorithm gives the best performance with faster convergence. The resulting epoch outputs are identical to the SGD ones. We thus argue that the choice of algorithm is mostly irrelevant to the results. We only look at the trained model and analyze the properties of the error landscape section and not the scheme to get to this point.
During the training, the model's performance was evaluated, and metrics such as MSE were recorded. After the last epochs the model parameters, the gradients and the eigenvalues of the Hessian matrix were evaluated and saved. Also the full hessian matrices were calculated for each trained model and saved in an array to be evaluated in curvature calculations.

C.1.4 Evaluation

The evaluation was done in a very straight forward way. There is no transformation or processing done to the data and its just plotted as viewed in chapters (3) and (4), except for the Gauss-Kronecker calculation. We introduce a cut-off at 10^{-10} for the hessian eigenvalues and throw out all values below to avoid a zero determinant. Besides this we calculate the

log-determinant and get the exponential afterwards for some numerical stability.

C.2 Setup 2: Variational Autoencoder (VAE)

The second experimental setup involved a Variational Autoencoder (VAE)-type setup. The main difference here is that, unlike the standard VAE our model is not trained to regenerate the input $\mathbf{x}$, but to produce the corresponding output $\mathbf{y}$. The latent representation is used to regularize the model and probe the range from no regularization to over-regularization, i.e. the point where the regularization is so strong it shifts the model into a trivial regime as in the L2 setup. The regularizer term in Eq. 23 takes the latent representation as the mean and variance of a Gaussian distribution and calculates Kullback-Leibler divergence with white noise. For high enough β the minimization of the regularizer term becomes the main objective leading to the latent representation becoming trivial.

C.2.1 Architecture

This VAE consists of an encoder and a decoder. The encoder takes the input data and transforms maps into a latent space with 2 dimensions. It outputs both the mean and log variance of the latent variables, allowing for sampling during training through the reparameterization trick. It interprets the latent space as a parametrization of a Gaussian map and calculates the KL-divergence with white noise. For high enough β the effective minimum shifts to the minimum of the KL-divergence term, forcing the model out of the error minimum and outputting a trivial model.

C.2.2 Data Generation

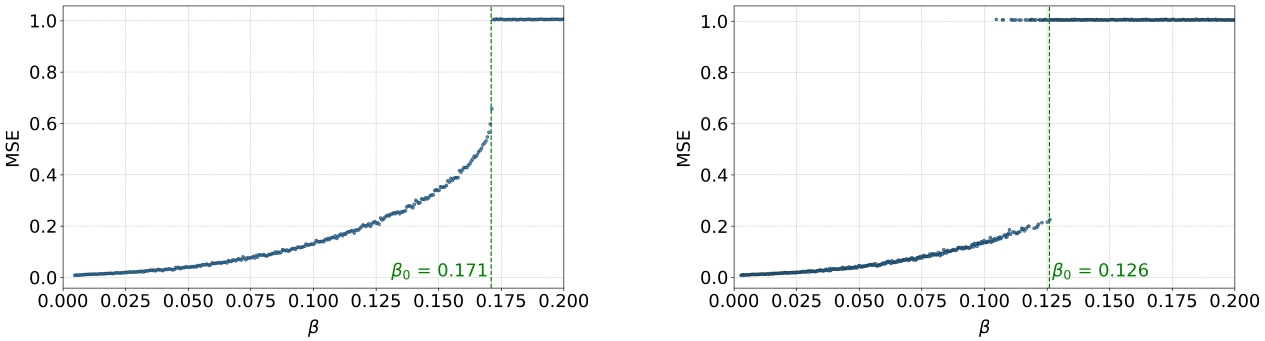
To train and test the VAE, synthetic Gaussian data was generated same as in the first setup.

C.2.3 Training Process

Each training session involved adjusting the hyperparameter β. The basic set-up is analogous to the L2 experiments. One can choose an annealing approach or a non-annealing one. The outputs here only consist of the MSE and the KL-divergence values and the model parameters for each β on the test data after each final epoch, as well as all epoch outputs. As in the L2 set-up we used sigmoid activations.

C.3 Hysteresis and role of Initialization

The two-layer case displays a sharp transition consistent with a first-order phase transition. Additionally we can see a hysteresis effect. The transition is delayed when we turn off the annealing an train independent models, i.e. when we initialize a new NN for each beta value. The precise initialization, as well as the step size of the learning algorithm will play a role for the transition point. There are two relevant minima in the critical regime. The effective minimum and the L2 minimum, i.e. the origin. When the effective minimum does not yet overlap with the L2 minimum, i.e. when the β value is still just fulfilling learnability conditions the model still 'jumps' or gets stuck in the L2 minimum. In the annealing approach the model remains in the effective minimum for longer than in the random initialization as one would expect. When we are close to the transition point β_{crit} and there are two close by minima. One is the less accurate local minimum and one the effective minimum that is still in the higher accuracy phase. Depending on the starting point of the model, i.e. the initialization the algorithm may not allow for the 'finding' of the higher accuracy phase. We can verify this by turning off annealing an observing that the transition point is clearly shifted to lower β values. Additionally we see a clear phase coexistence. A fraction of the models transitions to the trivial regime while some models still find the higher accuracy phase.



(a) MSE-Beta, 2hidden, 1D, Sigmoid, annealing

(b) MSE-Beta, 2hidden, 1D, Sigmoid, no-annealing

Figure 9: We see the clear shift as well as phase coexistence when the annealing is turned off and models are randomly initialized (initialization from uniformous distribution)