Noise Injection as a Probe of Deep Learning Dynamics

Noam Levi[§] [*] **, Itay M. Bloch**[†] [*] **, Marat Freytsis**[‡] **& Tomer Volansky**[§]

Abstract

We propose a new method to probe the learning mechanism of Deep Neural Networks (DNN) by perturbing the system using Noise Injection Nodes (NINs). These nodes inject uncorrelated noise via additional optimizable weights to existing feed-forward network architectures, without changing the optimization algorithm. We find that the system displays distinct phases during training, dictated by the scale of injected noise. We first derive expressions for the dynamics of the network and utilize a simple linear model as a test case. We find that in some cases, the evolution of the noise nodes is similar to that of the unperturbed loss, thus indicating the possibility of using NINs to learn more about the full system in the future.

1 Introduction

The training of DNNs is a highly opaque procedure. Beyond curvature evolution (Hochreiter and Schmidhuber, 1997; Sagun et al., 2016; Gur-Ari et al., 2018; Ghorbani et al., 2019; Yao et al., 2020; Papyan, 2018; Li et al., 2020), few metrics are available to describe how a network evolves as it trains. An interesting attempt at parameterizing the interplay between training dynamics and generalization was explored in the seminal work of Zhang et al. (2016), which demonstrated that when input data was corrupted by adding random noise, the generalization error deteriorated in correlation with its strength. Noise injection has gained further traction in recent years, both as a means of effective regularization (Graves, 2011; Ba and Frey, 2013; Goodfellow et al., 2013; Srivastava et al., 2014; Kang et al., 2016; Wager et al., 2013; Li and Liu, 2018) , as well as a route towards understanding DNN dynamics and generalization. For instance, label noise has been shown to affect the implicit bias of Stochastic Gradient Descent (SGD) (Blanc et al., 2020; Damian et al., 2021) .

Here, we take another step along this direction by allowing the network to actively regulate the effects of the injected noise during training. We define *Noise Injection Nodes* (NINs), whose output is a random variable, connected to a feed-forward DNN via trainable *Noise Injection Weights* (NIWs). The network is subsequently trained to perform a given task using vanilla SGD. Starting with a detailed analysis of a simple linear network, we study such systems both numerically and analytically. Our main results, partly summarized in Fig. 1, are as follows: (i) As a function of NIN variance, the system exhibits 4 distinct phases. (ii) In two phases, the NIWs evolve to small values, implying that a well-trained network can recognize that the noise contains no useful information within it. (iii) For these phases, the NIW dynamics is dictated by the local curvature of the training loss function.

Item (ii) may be expected if the NIN is re-randomized at each training epoch, yet we find essentially the same behavior repeated even when the NIN values are generated only once and fixed before training, putting them on equal footing with actual data inputs, as shown in Fig. 1 (center,right). It appears that while the system might in principle be able to memorize the specific noise samples, optimization dyanamics still prefer to suppress them. This implies a relation between the NIN reduction mechanism and the network's ability to generalize, to be explored further in future works.

2 Noise Injection Weight Evolution

Consider a DNN with parameters $\boldsymbol{\theta} = \{W^{(\ell)}, b^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell+1}}, \mathbb{R}^{d_{\ell+1}} | \ell = 0, \dots, N_L - 1\}$, corresponding to the weights and biases, and N_L layers, defined by its single sample loss function $\mathcal{L} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}$ and optimized under SGD to perform a supervised learning task. Here, at each SGD iteration, a mini-batch $\mathcal{B}$ consists of a set of labeled examples, $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{|\mathcal{B}|} \subseteq \mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\text{label}}}$.

[*]Equal contribution.

[†]BCTP, University of California and LBNL, CA, U.S.A. Email: `itayblochm@berkeley.edu`

[‡]NHETC, Rutgers University, NJ, U.S.A. Email: `marat.freytsis@rutgers.edu`

[§]. Department of Physics, Tel Aviv University, Tel Aviv, Israel. Email: `noam@mail.tau.ac.il`

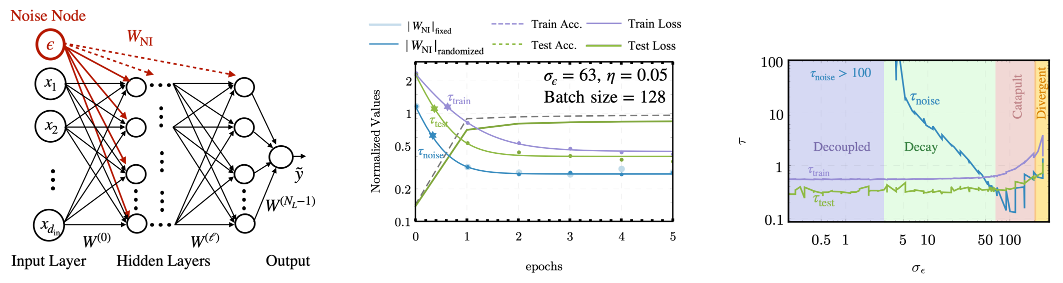

Figure 1: **Left:** Schematic of a generic DNN, with the addition of a single NIN connected via NIWs. **Center:** Example evolution displaying a decay behavior for both the NIWs and the losses within the decay phase of the system discussed below and with a fixed noise strength, σ_ϵ. Two cases are shown: re-initialized NIN at every epoch, and fixed value NIN. The similar behavior of the systems in both cases (**blue** and **light blue** points respectively) hint at a potent relation between the NIWs evolution and generalization. The **blue**, **green** and **violet** stars indicate the NIW, test loss and training loss decay time-scales. The three solid curves are fits to exponential decays, while the data is represented with points. **Right:** The different decay times as a function of the noise injection magnitude. The four shaded regions indicate the four phases of the system discussed in Sec. 3. The results are shown for a 3-hidden layer ReLU MLP with CE loss, trained on FMNIST to 100 % training accuracy.

We study the simple case of connecting a given NIN to a specific layer, denoted as $\ell_{\rm NI}$, via a NIW vector $W_{\rm NI} \in \mathbb{R}^{1\times d_{\ell_{\rm NI}+1}}$ (see Fig. 1, left). In this setup, the injected noise is taken as a random scalar variable, ϵ, sampled repeatedly at each SGD training epoch from a chosen distribution. The NIWs' evolution is best studied via their effect on preactivations, defined as $\boldsymbol{z}^{(\ell)} = W^{(\ell)} \cdot \boldsymbol{x}^{(\ell)} + b^{(\ell)}$. When a NIN is added, the preactivations at layer $\ell_{\rm NI}$ are subsequently shifted to $\boldsymbol{z}^{(\ell_{\rm NI})} \to \boldsymbol{z}^{(\ell_{\rm NI})} + W_{\rm NI}\epsilon$.

For a single NIN connected at layer $\ell_{\rm NI}$, the batch-averaged loss function can be written as a series expansion in the noise translation parameter[1]

$$L(\boldsymbol{\theta}, W_{\rm NI}) = \frac{1}{|\mathcal{B}|}\sum_{\substack{\{\boldsymbol{x},\epsilon,\boldsymbol{y}\}\\ \in\mathcal{B}}} \mathcal{L}(\tilde{\boldsymbol{\theta}}, \boldsymbol{z}^{(\ell_{\rm NI})} + W_{\rm NI}\epsilon, \boldsymbol{y}) = L(\boldsymbol{\theta}) + \frac{1}{|\mathcal{B}|}\sum_{\substack{\{\boldsymbol{x},\epsilon,\boldsymbol{y}\}\\ \in\mathcal{B}}}\sum_{k=1}^{\infty} \frac{(\epsilon W_{\rm NI}^T \cdot \nabla_{\boldsymbol{z}^{(\ell_{\rm NI})}})^k}{k!}\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \epsilon, \boldsymbol{y}), \quad (1)$$

where $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} \setminus \{W^{(\ell_{\rm NI})}\}$ and $L(\boldsymbol{\theta})$ is the loss function in the absence of a NIN. Focusing on a distribution with zero mean for the NIN (*e.g.*, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$) and performing the batch averaging on each term, we arrive at the update rule for the NIWs from the noisy loss expansion[2]

$$W_{\rm NI}^{(t+1)} = W_{\rm NI}^{(t)} - \eta\frac{\sigma_\epsilon \Phi}{\sqrt{|\mathcal{B}|}}\sqrt{\langle (g_{\ell_{\rm NI}}^{(t)})^2\rangle} - \frac{\eta\sigma_\epsilon^2}{2}\left\langle \mathcal{H}_{\ell_{\rm NI}}^{(t)}\right\rangle W_{\rm NI}^{(t)} + \cdots . \quad (2)$$

Here, batch averaging is denoted by $\langle\cdots\rangle$, Φ is a random variable with zero mean and unit variance, and σ_ϵ^2 is the variance of the injected noise. We denote the network-dependent *local* gradient and Hessian at the NIN layer as $g_{\ell_{\rm NI}} = \nabla_{\boldsymbol{z}^{(\ell_{\rm NI})}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ and $\mathcal{H}_{\ell_{\rm NI}} = \nabla_{\boldsymbol{z}^{(\ell_{\rm NI})}}\nabla^T_{\boldsymbol{z}^{(\ell_{\rm NI})}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$, respectively. A more complete derivation is given in App. C.

Terminating the expansion in Eq. (2) at 2nd order need not be valid for large σ_ϵ. We thus proceed by studying a linear test case for which the 2nd order expansion is precise. The persistence of analogous network behavior, and in particular its phases, for a more realistic setup is confirmed empirically.

3 Linear Toy Model

Consider a two-layer DNN with linear activations and layer widths ($d_{0,1} = 1$) and no biases ($b = 0$), tasked with univariate linear regression[3], and with a single NIN connected to the first layer ($\ell_{\rm NI} = 0$). The data consists of a set of training samples $\{(x_i, y_i) \in \mathbb{R}\times\mathbb{R}\}_{i=1}^m$, and we sample x_i and the noise ϵ_i from the normal distributions, $x_i, \epsilon_i \sim \mathcal{N}(0, \sigma_{x,\epsilon}^2)$. The corresponding data labels are

[1] In practice, it is often the case that one uses piece-wise analytic activation functions such as ReLU, and so if the noise causes the crossing of a non-analytic point, the above formal expansion is invalid. This subtlety does not change any of our conclusions and empirically we recognize the same phases when using ReLU activations.

[2] Additional $\sigma_\epsilon^2/\sqrt{|\mathcal{B}|}$ corrections coming from the variance of the 2nd order term emerge from batch-averaging, and are assumed to be negligible throughout this work.

[3] We use this toy model as a proxy for a diagonal linear network as in Gunasekar et al. (2018). A more general treatment including width and depth effects will be presented in future work.

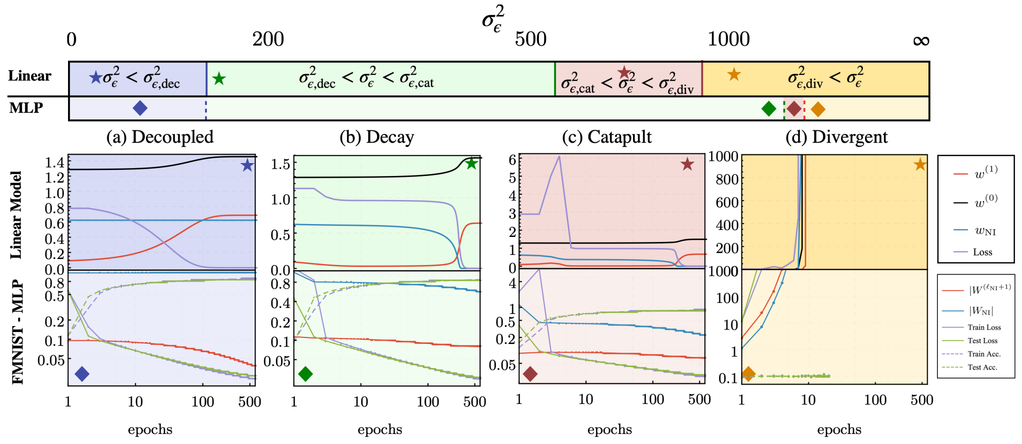

Figure 2: Evolution for a DNN with a NIN coupled to the first hidden layer, for two different models. **Top bar:** Regions for the phases as a function of the noise strength σ_ϵ. The **top darker shaded** and **bottom lighter shaded** bar regions correspond to the linear and Multi-Layer Perceptron (MLP) models respectively. The **stars** and **diamonds** indicate the values of σ_ϵ used in the bottom plots. **Top plots:** Numerical solutions for the various phases of Eq. (3), describing the linear model. The data weights (**black** and **red** lines), NIW (**blue**) and loss functions (**purple**) are shown. In the decoupled and decay phases, data weights approach optimum values while the NIW decays slowly, whereas a larger noise magnitude results in a longer relaxation time. In the initial stages of the catapult phase no learning is achieved until the NIW sufficiently decays, when standard learning is resumed. In the divergent phase the system fails to learn, with breakdown after a few epochs. **Bottom plots**: Various phases of NIW dynamics during training for a 1-hidden layer MLP with MSE loss and ReLU activations, trained on FMNIST. **Solid** (**dashed**) curves represent the loss (accuracy) values, for training (**purple**) and test (**green**) instances. The behavior displayed by the loss function, the NIWs (**blue**) and the subsequent layer weights (**red**) norms, verifies the predictions of the linear model of Sec. 2. For experimental details, see App. B.

given by a linear transformation of the inputs $y_i = M \cdot x_i$ with a fixed $M \in \mathbb{R}$. This regression problem is solved by minimizing the empirical loss, taken as the Mean Squared Error (MSE), $L_{\text{MSE}} = \frac{1}{2|\mathcal{B}|}\sum_{i\in\mathcal{B}}(w^{(1)}(w^{(0)}\cdot x_i + w_{\text{NI}}\epsilon_i) - y_i)^2$, with optimal solution $w_*^{(1)}w_*^{(0)} = M, w_{\text{NI},*} = 0$.

The evolution of the system can be studied by focusing on the coupled SGD equations for the hidden layer weight and the NIW, parameterized as

$$w_{t+1}^{(1)} = A_t\sigma_\epsilon + w_t^{(1)}(1 - B_t\sigma_\epsilon^2) - C_t, \qquad w_{\text{NI},t+1} = \tilde{A}_t\sigma_\epsilon + w_{\text{NI},t}(1 - \tilde{B}_t\sigma_\epsilon^2). \tag{3}$$

Here, the various terms are given explicitly by[4] $A_t = \eta\Phi_t\sigma_x/\sqrt{|\mathcal{B}|}w_{\text{NI},t}(2w_t^{(1)}w_t^{(0)} - M)$, $\tilde{A}_t = \eta\Phi_t\sigma_x/\sqrt{|\mathcal{B}|}w_t^{(1)}(w_t^{(1)}w_t^{(0)} - M)$, $B_t = \eta\, w_{\text{NI,t}}^2$, $\tilde{B}_t = \eta(w_t^{(1)})^2$, $C_t = \eta(w_t^{(1)}w_t^{(0)} - M)w_t^{(0)}\sigma_x^2$, and are functions of the NIW, the data weights, the learning rate η and the batch size $|\mathcal{B}|$.

While non-linear, the solution to Eq. (3) is rather simple and is dictated by σ_ϵ and the initial conditions. We identify four different phases, demonstrated in Fig. 1 (right) and Fig. 2:

Decoupled phase. When the scale of injected noise is sufficiently small, $\sigma_\epsilon \leq \sigma_{\epsilon,\text{dec}} \equiv 2\tilde{A}_0/\tilde{B}_0$, the original optimization trajectory, driven by the C_t term, is on average unaffected by the NIN, and the NIWs evolve according to a random walk, with step size dictated by the local gradient, $\tilde{A}_t$[5].

Decay phase. For $\sigma_{\epsilon,\text{dec}}^2 < \sigma_\epsilon^2 \ll \sigma_{\epsilon,\text{cat}}^2 \equiv \min[2/B_0, 2/\tilde{B}_0]$ the dynamics is initially dominated by the $\tilde{B}_t$ and C_t terms, and as a consequence w_{NI} exponentially falls. Once w_{NI} is sufficiently small, the A-terms dominate and the dynamics is decoupled from the noise as in the previous phase.

The above two phases occur for rather small noise injection and, as discussed above, the noise is a mere small perturbation to the dynamics. However, for $\sigma_\epsilon > \sigma_{\epsilon,\text{cat}}$, the equations become noise-dominated at initialization and the early dynamics becomes insensitive to the original learning objective. Early time evolution can then be understood by neglecting the A- and C-terms. Consequently, the equations describe a DNN, trained using completely random data with no labels or learning objectives. The network evolution can then be separated into the following two distinct phases:

[4] We match the local gradient $\sqrt{\langle g_0^2\rangle} = \sigma_x w_t^{(1)}(w_t^{(1)}w_t^{(0)} - M)$ and Hessian $\langle\mathcal{H}_0^{(t)}\rangle = 2(w_t^{(1)})^2$ to Eq. (2).

[5] If $w_{\text{NI,t}}$ sufficiently grows, this phase could potentially be exited, however it is quickly restored due to $\tilde{B}_t$.

Catapult phase. When $\sigma_{\epsilon,\text{cat}} < \sigma_\epsilon < \sigma_{\epsilon,\text{div}} \equiv \max(2/B_0, 2/\tilde{B}_0)$, some, but not all, of the network weights begin diverging, as a stiff equation regime ensues due to the discrete nature of the SGD algorithm. Generally, this divergence can be driven by either the data or the NIWs, though in realistic scenarios the latter is more common and we therefore discuss for concreteness the $2/\tilde{B}_0 < \sigma_\epsilon < 2/B_0$ case. While the NIW (and the loss function) diverges, $w_t^{(1)}$ decays fast enough for the Hessian to be reduced, allowing for the network to recover, and resulting in a catapult effect (Lewkowycz et al., 2020). At this point the dynamics of the system behave similarly to the Decay Phase, albeit typically at a slower rate since the Hessian is now significantly smaller. These results are visible in Fig. 2(c).

Divergent phase. Once $\sigma_\epsilon^2 \geq \sigma_{\epsilon,\text{div}}^2$, the NIN overwhelms the network, and all weights (and loss function) diverge resulting in a failed training process. In non-linear networks, this phase can also occur if the second order perturbative approximation for the loss function breaks down[6].

The bottom row of Fig. 2 demonstrates the persistence of our predicted phase diagram for an over-parameterized DNN trained on the FMNIST dataset (Xiao et al., 2017). While the decoupled and decay phases occur for similar σ_ϵ values as those in the linear model, the large noise behavior is altered by the choice of non-linearities, loss function, batch size and number of layers. In particular, the noise variance values dictating the phase boundaries are somewhat different than those predicted by the linear model, however the existence of the boundaries themselves persists throughout our experiments [7] . Some of these effects are partially explored in App. A.

4 Relating the Evolution of the NIWs and Loss

As discussed in Sec. 1, during the decay and catapult phases, the NIWs decay as the network attempts to learn, a phenomena which repeats for re-randomized as well as fixed value NINs. Since the latter can be interpreted as the suppression of uncorrelated data features, understanding the relationship between the NIWs and loss evolutions could provide future insights regarding generalization.

Here, we take a step towards connecting the NIW dynamics with the loss dynamics, for the simple linear model, hinting that more complicated constructions could allow a more direct relationship between the two. With no NINs and for small enough learning rates, $L^{(t+1)} - L^{(t)} \approx -\eta(\partial L^{(t)}/\partial\theta^{ij})^2$, which simplifies in the linear case to the trace of the global Hessian, multiplied by the loss, as shown in App. D. We demonstrate that a similar behavior holds for a realistic DNN in Fig. 1(center, right).

To show how this relates to the NIN, it is easier to take the continuous time limit ($\eta \to 0$) of Eq. (3), resulting in a set of coupled Langevin equations (Lemons and Gythiel, 1997). In App. D, we show that, neglecting the effect of the NIN on the loss, the solution of the full continuous time equations is

$$L_{\epsilon=0}(t) \sim L_{\epsilon=0}(0) e^{-2\sigma_x^2 \int_0^t \left[(w^{(0)}(t'))^2 + (w^{(1)}(t'))^2\right] dt'}, w_{\text{NI}}(t) \sim e^{-\sigma_\epsilon^2 \int_0^t (w^{(1)}(t'))^2 dt'}. \tag{4}$$

Defining $\tau_{\text{noise}} = -w_{\text{NI}}/\dot{w}_{\text{NI}}|_{t=0}$, $\tau_{\text{loss}} = -L_{\epsilon=0}/\dot{L}_{\epsilon=0}|_{t=0}$, the two time scales differ both by a multiplicative factor and by the magnitude of the weight connecting the input with the first layer. The first comes from the difference between data and noise distributions, while the second is due to the NINs being connected at the same layer as the input ($\ell_{\text{NI}} = 0$), such that the local Hessian which controls their evolution is insensitive to first layer contributions to the loss evolution. Adding multiple NINs in more complicated constructions which would be sensitive to the input layer's weights could provide a novel probe for how various local parts of the Hessian affect the evolution of loss functions.

5 Conclusions

Studying the relationship between training dynamics and learning capabilities of DNNs by injecting noise through optimizable connections can provide a novel path towards interpretable deep learning. In the future, NINs could have many possible technological applications. For instance, one could distinguish NIWs evolution from regular data weights to identify features which are uncorrelated with the labels, and thus build normalization schemes (Ioffe and Szegedy, 2015; Arpit et al., 2016; Xu et al., 2021) to suppress such weights in an effort to tackle internal covariate shift. Furthermore, it's possible that tracking the NIWs evolution could inform one about generalization, as the NIWs cannot be memorized by the network, only suppressed. Thus, landing in distinct points on the loss landscape due to noise mitigation could inform us on how loss curvature relates to noise resistant networks.

[6] In theory, there may be intermediate steps in unique scenarios (*e.g.*, the NIW decays, but without effect on the data weights' evolution), however we have not found these cases to empirically matter and ignore them here.

[7] We find that divergence is replaced by learning process failure due to neuron saturation for bounded activations.

Reproducibility Statement

In Sec. 2, we state our theoretical results, ensuring that we assert our assumptions and the limitations of the approximations we make at every step. In several instances, we rely on proofs, as well as supplement our analyses in Apps. C and D; The models and tools used for analysis in our experiments are provided in the following anonymous link: https://anonymous.4open.science/r/NoiseInjectionNodeCode-2A68, while explicit details regarding our experimental setup as well as a complete description of the data processing steps for the datasets we used, are given in App. B. All experiments were performed on a 20 node cluster, each consisting of 24-48 CPUs using Intel® Xeon® E5-2650 v4 CPU @ 2.20 GHz, with further implementation details given in App. B.

Acknowledgments

We thank Yasaman Bahri, Yohai Bar-Sinai, Kyle Cranmer, Guy Gur-Ari and Niv Haim for useful discussions and comments at various stages of this work. NL would like to thank the Milner Foundation for the award of a Milner Fellowship. MF is supported by the DOE under grant DE-SC0010008. NL and MF would like to thank Tel Aviv University, the Aspen Center for Physics (supported by the U.S. National Science Foundation grant PHY-1607611), and the Galileo Galilei Institute for their hospitality while this work was in progress. The work of TV is supported by the Israel Science Foundation (grant No. 1862/21), by the Binational Science Foundation (grant No. 2020220) and by the European Research Council (ERC) under the EU Horizon 2020 Programme (ERC-CoG-2015 - Proposal n. 682676 LDMThExp).

References

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace, 2018. URL https://arxiv.org/abs/1812.04754.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR, 2019.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE, 2020.

Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Xinyan Li, Qilong Gu, Yingxue Zhou, Tiancong Chen, and Arindam Banerjee. Hessian based analysis of sgd for deep nets: Dynamics and generalization. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 190–198. SIAM, 2020.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.

Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. *Advances in neural information processing systems*, 26:3084–3092, 2013.

Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pages 117–124. Springer, 2013.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Guoliang Kang, Jun Li, and Dacheng Tao. Shakeout: A new regularized deep neural network training scheme. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. *Advances in neural information processing systems*, 26:351–359, 2013.

Yinan Li and Fang Liu. Whiteout: Gaussian adaptive noise regularization in deep neural networks, 2018.

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an Ornstein–Uhlenbeck like process. In *Conference on learning theory*, pages 483–513. PMLR, 2020.

Alex Damian, Tengyu Ma, and Jason D Lee. Label noise SDG provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks, 2018. URL https://arxiv.org/abs/1806.00468.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism, 2020.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017.

Don S. Lemons and Anthony Gythiel. Paul langevin's 1908 paper "on the theory of brownian motion" ["sur la théorie du mouvement brownien," c. r. acad. sci. (paris) 146, 530–533 (1908)]. *American Journal of Physics*, 65(11):1079–1081, 1997. doi: 10.1119/1.18725. URL https://doi.org/10.1119/1.18725.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Devansh Arpit, Yingbo Zhou, Bhargava U. Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks, 2016. URL https://arxiv.org/abs/1603.01431.

Renzhe Xu, Peng Cui, Zheyan Shen, Xingxuan Zhang, and Tong Zhang. Why stable learning works? a theory of covariate shift generalization. *ArXiv*, abs/2111.02355, 2021.

A Additional Empirical Results

Here, we present additional results not included in the main text.

In Fig. 3 below, we show the different phases predicted in Sec. 2, for a 3 layer MLP with ReLU activations, trained on FMNIST using cross-entropy (CE) loss. These results are similar to the ones obtained for a network which was trained using the MSE loss, shown in Fig. 2(bottom), demonstrating the persistence of the phase structure regardless of the loss function.

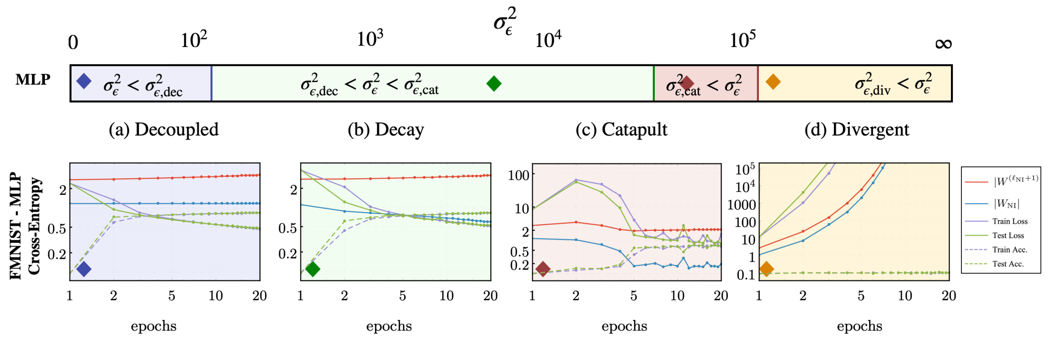

Figure 3: Dynamical evolution of a network with a NIN coupled to the first hidden layer of a DNN. **Top bar:** The **light shaded** regions separating the four phases as a function of the noise strength, σ_ϵ. The colored **diamonds** indicate the values of σ_ϵ used in the bottom plots. **Bottom plots**: Various phases of the NIW dynamics during training for a 1-hidden layer MLP with Cross-Entropy loss and ReLU activations, trained on the full FMNIST dataset. The data weights (**black** and **red lines**), NIW (**blue**) and loss functions (**purple**) are shown. In the decoupled and decay phases, data weights approach optimum values while the NIW decays slowly. Larger noise implies longer time for this process to end. In the initial stages of the catapult phase no learning is achieved until the NIW sufficiently decays, when standard learning is resumed. Lastly, in the divergent phase the system fails to learn, reaching a breakdown after only a few epochs. **Solid** (**dashed**) curves represent the loss (accuracy) values, for training (**violate**) and test (**green**) instances. The behavior displayed by the loss function, the norms of the NIWs (**blue**) and the weights connected to the subsequent layer (**red**), corroborates the predictions of the linear model, as well as the results obtained on the same network, trained using an MSE loss function, discussed in Sec. 2. For experimental details, see App. B.

B Experimental Details

In the main text we present results for models trained on the Fashion-MINST (FMNIST) dataset (Xiao et al., 2017). FMNIST contains 70,000 grayscale images in 10 categories. The images show individual articles of clothing at low resolution (28×28 pixels). We preprocess the data by subtracting the mean and dividing by the variance of the training data, and train using a $60/40$ training/validation split. All test accuracy evaluations are done with the NIN output set to 0, *i.e.*, $\epsilon = 0$.

Implementation Details : For all of our experiments, we utilize a Multi-Layer Perceptron (MLP).

We optimize using vanilla SGD with either MSE or Sparse Cross-Entropy loss. The model parameters $\boldsymbol{\theta}, \boldsymbol{w}_{\text{NI}}$ are initialized at iteration $t = 0$ using a normal distribution as $\boldsymbol{w}_0, \boldsymbol{w}_{\text{NI},0} = 1/\sqrt{N_{\text{fan-in}}}$ unless otherwise specified.

Here we describe experimental settings specific to a figure.

Figure 1. Fully connected, 3 hidden layers, $N_w = 1024$, ReLU non-linearity trained using SGD (no momentum) on FMNIST using a Sparse-Cross-Entropy loss function. Batch size = 128, with learning rate $\eta = 0.05$, using weight normalization $w^{(\ell)} \sim \mathcal{N}(0, 1/d_\ell), b = 0$. The hyperparameters are chosen to obtain good generalization performance without a NIN.

Figure 2. (top row): Numerical solution of Eq. (3). Learning rate set at $\eta = 0.01$. Intialization values for the weights were chosen to mimic a standard over-parameterized regime, i.e. $w^{(0)}, w_{\text{NI}}^{(0)} \sim \mathcal{N}(0, 1)$, while $w^{(0)}, w_{\text{NI}}^{(0)} \sim \mathcal{N}(0, 1/100)$. The target is the identity, i.e. $M = 1$, as the equations are invariant under reparameterization with respect to M. The only values of consequence are then

	Decoupled (σ_ϵ^2)	Decay (σ_ϵ^2)	Catapult (σ_ϵ^2)	Divergent (σ_ϵ^2)
MSE	$10^{-4} \cdot d_{\text{Input}}/(N_w\eta)$	$30 \cdot d_{\text{Input}}/(N_w\eta)$	$47 \cdot d_{\text{Input}}/(N_w\eta)$	$50 \cdot d_{\text{Input}}/(N_w\eta)$
CE	$10^{-4} \cdot d_{\text{Input}}/\eta$	$0.1 \cdot d_{\text{Input}}/\eta$	d_{Input}/η	$1.8 \cdot d_{\text{Input}}/\eta$

Table 1: Noise injection strength (σ_ϵ^2) used in Fig. 2(bottom), denoted as MSE, and Fig. 3, denoted as CE.

the ratios of weights at initialization and the learning rate. Therefore the actual values of the learning rate and noise variance are chosen to make the effects visible to the eye for a short training period but do not affect the final results in any way.

Figure 2 (bottom row), Fig. 3. Fully connected, one hidden layer $N_w = 1024$, ReLU non-linearity trained using SGD (no momentum) on FMNIST. Batch size = 1000, with learning rate $\eta = 0.01$, using weight normalization $W^{(\ell)} \sim \mathcal{N}(0, 1/d_\ell), b = 0$. The hyperparameters are chosen to obtain good generalization performance without a NIN.

In Fig. 2(bottom) and Fig. 3 we use different loss functions, namely, MSE and Cross-Entropy, respectively. We detail the amount of noise injection used in each of the two figures in Table 1.

C Further Details On the Noise Parameter Expansion

Here we provide additional details on the theoretical analysis of the general model in Sec. 2. Starting with the noise translated loss function

$$L(\boldsymbol{\theta}, W_{\text{NI}}) = \frac{1}{|\mathcal{B}|} \sum_{\{\boldsymbol{x},\epsilon,\boldsymbol{y}\}\in\mathcal{B}} \mathcal{L}(\boldsymbol{\theta}, W_{\text{NI}}; \boldsymbol{x}, \epsilon, \boldsymbol{y}) = \frac{1}{|\mathcal{B}|} \sum_{\{\boldsymbol{x},\epsilon,\boldsymbol{y}\}\in\mathcal{B}} e^{\epsilon W_{\text{NI}}^T \nabla_{\boldsymbol{z}^{(\ell_{\text{NI}})}}} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y})$$
$$= L(\boldsymbol{\theta}) + \frac{1}{|\mathcal{B}|} \sum_{\substack{\{\boldsymbol{x},\epsilon,\boldsymbol{y}\}\\ \in\mathcal{B}}} \sum_{k=1}^{\infty} \tfrac{1}{k!} (\epsilon W_{\text{NI}}^T \cdot \nabla_{\boldsymbol{z}^{(\ell_{\text{NI}})}})^k \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \epsilon, \boldsymbol{y}).$$

Expanding in powers of ϵW_{NI}, we obtain an infinite series given by

$$L(\boldsymbol{\theta}, W_{\text{NI}}) = L(\boldsymbol{\theta}) + \frac{1}{|\mathcal{B}|} \sum_{\{\boldsymbol{x},\epsilon,\boldsymbol{y}\}\in\mathcal{B}} \sum_{k=1}^{\infty} \frac{1}{k!} (\epsilon W_{\text{NI}}^T \cdot \nabla_{\boldsymbol{z}^{(\ell_{\text{NI}})}})^k \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \epsilon, \boldsymbol{y}). \tag{5}$$

Performing the batch averaging explicitly amounts to averaging over the statistics of the data and noise, resulting in the loss

$$\langle \mathcal{L}(\boldsymbol{\theta}, W_{\text{NI}}) \rangle = \langle \mathcal{L}(\boldsymbol{\theta}) \rangle + W_{\text{NI}}^T \cdot \langle \epsilon g_{\ell_{\text{NI}}} \rangle + \frac{1}{2} W_{\text{NI}}^T \langle \epsilon^2 \mathcal{H}_{\ell_{\text{NI}}} \rangle W_{\text{NI}} + \dots \tag{6}$$

Here, batch averaging is denoted by $\langle .. \rangle$, the *local* gradient is $g_{\ell_{\text{NI}}} = \nabla_{\boldsymbol{z}^{(\ell_{\text{NI}})}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ and Hessian $\mathcal{H}_{\ell_{\text{NI}}} = \nabla_{\boldsymbol{z}^{(\ell_{\text{NI}})}} \nabla_{\boldsymbol{z}^{(\ell_{\text{NI}})}}^T \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ are network-dependent functions, pertaining to the NIN layer.

Taking ϵ sampled from a distribution with zero mean, the local gradient and Hessian contributions simplify as $\langle \epsilon g_{\ell_{\text{NI}}} \rangle \sim \sqrt{\langle g_{\ell_{\text{NI}}}^2 \rangle} \Phi \sigma_\epsilon / \sqrt{|\mathcal{B}|}$, where $\langle g_{\ell_{\text{NI}}}^2 \rangle$ is the vector of the batch-averaged absolute value of the gradient and Φ is a random variable with mean 0 and variance 1, and $\langle \epsilon^2 \mathcal{H}_{\ell_{\text{NI}}} \rangle \sim \sigma_\epsilon^2 \langle \mathcal{H}_{\ell_{\text{NI}}} \rangle$, where we define $\sigma_\epsilon^2 \equiv \langle \epsilon^2 \rangle$ as the variance of the injected noise.

This result, as proven below, makes explicit that odd terms in the expansion are suppressed by $|\mathcal{B}|^{-1/2}$.

The last step in the derivation is performed by taking the SGD update step with respect to the NIWs, which is simply

$$W_{\text{NI}}^{(t+1)} = W_{\text{NI}}^{(t)} - \eta \frac{\partial L(\boldsymbol{\theta}^{(t)}, W_{\text{NI}}^{(t)})}{\partial W_{\text{NI}}^{(t)}}. \tag{7}$$

Finally, utilizing Eq. (6) we arrive at

$$W_{\mathrm{NI}}^{(t+1)} = W_{\mathrm{NI}}^{(t)} - \eta\frac{\sigma_\epsilon\Phi}{\sqrt{|\mathcal{B}|}}\sqrt{\langle (g_{\ell_{\mathrm{NI}}}^{(t)})^2\rangle} - \frac{\eta\sigma_\epsilon^2}{2}\left\langle \mathcal{H}_{\ell_{\mathrm{NI}}}^{(t)}\right\rangle W_{\mathrm{NI}}^{(t)} + \dots, \tag{8}$$

which is given in the main text as Eq. (2).

We now turn to discuss our estimation of batch-averaged terms which are proportional to powers of ϵ. Generically, such terms can be written as $q\epsilon^n$, where n is an integer, and q is some sample-dependent variable, possibly with other indices. The goal of this appendix is to prove the following theorem:

Theorem 1. *Let q be a sample-dependent variable, with a finite mean $\langle q\rangle$ and with $Q \equiv \sqrt{\langle q^2\rangle}$. And let ϵ be the output of some NIN, which has a PDF symmetric around zero (and consequentially of zero mean), and for some integer n, $\langle \epsilon^n\rangle = (\sigma_{n,\epsilon})^n$, and $\langle\epsilon^{2n}\rangle = (\sigma_{2n,\epsilon})^{2n}$. For such a case, the average of $q\epsilon^n$ over a batch $\mathcal{B}$ has a mean of $\langle q\rangle(\sigma_{n,\epsilon})^n$, and a variance of $\left(\sigma_{2n,\epsilon}^{2n}Q^2 - \langle q\rangle^2\sigma_{n,\epsilon}^{2n}\right)/\sqrt{|B|}$.*

For odd ns, this simplifies to having zero mean, and a standard deviation of $Q\sigma_{2n,\epsilon}^n/\sqrt{|\mathcal{B}|}$.

We note that this theorem is the reasoning for the estimate $|\langle q\epsilon^n\rangle_{\mathcal{B}}| \sim \mathcal{O}(1)(\sigma_{\epsilon,2n})^n Q/\sqrt{|\mathcal{B}|}$ for odd ns, as well as the estimate of $\langle q\epsilon^n\rangle_{\mathcal{B}} \approx \sigma_{n,\epsilon}^{2n}\langle q\rangle$, for an even n and a large enough $|\mathcal{B}|$. We also note that for a large enough $|\mathcal{B}|$, this theorem immediately follows from the law of large numbers, however we prove it for any $|\mathcal{B}|$.

Proof. We can see that for a single term, and the average over the entire distribution

$$\langle q\epsilon^n\rangle = \langle q\rangle\langle\epsilon^n\rangle\,. \tag{9}$$

This is due to the fact that ϵ is a random output that is independent of q, and thus $\langle q\epsilon^n\rangle = \langle q\rangle\langle\epsilon^n\rangle$. For an odd n, this is simply equal to 0, since ϵ's PDF is symmetric around zero, so for odd ns, $\langle\epsilon^n\rangle = 0$.

Similarly, we may note that

$$\mathrm{Var}(q\epsilon^n) = \left\langle (q\epsilon^n)^2 - \langle (q\epsilon^n)\rangle^2\right\rangle = Q^2\sigma_{\epsilon,n}^{2n} - \langle q\rangle\langle\epsilon^n\rangle. \tag{10}$$

Where we once again use the independence of q and ϵ, and this time also use the definitions of Q and $\sigma_{\epsilon,n}$.

We can now see that

$$\langle\langle q\epsilon^n\rangle_{\mathcal{B}}\rangle = \frac{1}{|\mathcal{B}|}\left\langle\sum_{i=1}^{|\mathcal{B}|} q_i\epsilon_i^n\right\rangle = \frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\langle q\epsilon^n\rangle = \langle q\rangle\langle\epsilon^n\rangle, \tag{11}$$

and similarly,

$$\mathrm{Var}\left(\langle q\epsilon^n\rangle_{\mathcal{B}}\right) = \left\langle \langle q\epsilon^n\rangle_{\mathcal{B}}^2 - \langle q\epsilon^n\rangle_{\mathcal{B}}^2\right\rangle. \tag{12}$$

We have already computed the second term, so now let us compute the first one,

$$\left\langle\langle q\epsilon^n\rangle_{\mathcal{B}}^2\right\rangle = \frac{1}{|\mathcal{B}|^2}\left\langle\sum_{j=1}^{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}(q_j\epsilon_j^n)(q_i\epsilon_i^n)\right\rangle = \frac{1}{|\mathcal{B}|}\left\langle q^2\epsilon^{2n}\right\rangle + \frac{1}{|\mathcal{B}|^2}\left\langle\sum_{j=1}^{|\mathcal{B}|}\sum_{i=1,i\neq j}^{|\mathcal{B}|}(q_j\epsilon_j^n)(q_i\epsilon_i^n)\right\rangle, \tag{13}$$

where we divided the two sums to the contribution from $i = j$ and the contribution from $i \neq j$. The second term in Eq. (13) is simply $|\mathcal{B}|^2 - |\mathcal{B}|$ times the same term, and is therfore equal to,

$$\frac{1}{|\mathcal{B}|^2}\left\langle\sum_{j=1}^{|\mathcal{B}|}\sum_{i=1,i\neq j}^{|\mathcal{B}|}(q_j\epsilon_j^n)(q_i\epsilon_i^n)\right\rangle = \frac{|\mathcal{B}|^2 - |\mathcal{B}|}{|\mathcal{B}|^2}\left\langle (q_j\epsilon_j^n)(q_i\epsilon_i^n)\right\rangle_{i\neq j}. \tag{14}$$

We assume there is an arbitrarily large space of samples, which we may name as $\mathcal{A}$. Let us write its size as $|\mathcal{A}|$, and assume that $|\mathcal{A}| \to \infty$ and therefore,

$$\left\langle (q_j\epsilon_j^n)(q_i\epsilon_i^n)\right\rangle_{i\neq j} = \frac{1}{|\mathcal{A}|}\sum_{i,j\in\mathcal{A},i\neq j}(q_j\epsilon_j^n)(q_i\epsilon_i^n) = \frac{1}{|\mathcal{A}|^2}\sum_{i,j\in\mathcal{A}}(q_j\epsilon_j^n)(q_i\epsilon_i^n) - \frac{1}{|\mathcal{A}|^2}\sum_{j\in\mathcal{A}}(q_j\epsilon_j^n)^2 \tag{15}$$
$$= \langle q\epsilon^n\rangle^2 - \frac{1}{|\mathcal{A}|}\langle (q\epsilon^n)\rangle = \langle q\epsilon^n\rangle^2\,,$$

where in the last equality we used $|\mathcal{A}|\to\infty$. We can now write the variance of the batch averaged quantity, by collecting all the different terms, and find tha

$$\mathrm{Var}\left(\langle q\epsilon^n\rangle_{\mathcal{B}}\right) = \frac{1}{|\mathcal{B}|}\left(\left\langle q^2\epsilon^{2n}\right\rangle - \langle q\epsilon^n\rangle^2\right), \tag{16}$$

as originally postulated. □

D Decay Time-scale Derivation

Here, we derive the timescale for the NIW to decay, while the loss converges to its minimum, for the linear example given in Sec. 2.

Assuming that the system is in the decay phase, i.e. the SGD equations are given by

$$w_{t+1}^{(1)} = w_t^{(1)} - \eta(w_t^{(1)}w_t^{(0)} - M)w_t^{(0)}\sigma_x^2, \tag{17}$$
$$w_{\mathrm{NI},t+1} = w_{\mathrm{NI},t}(1-\eta(w_t^{(1)})^2\sigma_\epsilon^2). \tag{18}$$

Since the noise injection does not cause the system to diverge at this stage, the continuous time limit ($\eta\to 0$) is expected to hold, simplifying the equations as

$$\dot{w}^{(1)} = -(w^{(1)}w^{(0)} - M)w^{(0)}\sigma_x^2, \tag{19}$$
$$\dot{w}_{\mathrm{NI}} = -(w^{(1)})^2\sigma_\epsilon^2 w_{\mathrm{NI}}, \tag{20}$$

where it is implied that all weights are functions of time, $w = w(t)$. Next, we can define the loss function in the absence of noise, which will be the quantity we wish to track. This function is simply

$$L = \frac{1}{2}\left((w^{(1)}w^{(0)} - M)\sigma_x\right)^2, \tag{21}$$

which results in the continuous time update equation for the loss and the data weights

$$\dot{L} = \sigma_x^2\left((w^{(1)}w^{(0)} - M)\right)\left(\dot{w}^{(1)}w^{(0)} + \dot{w}^{(0)}w^{(1)}\right) = \sigma_x\sqrt{2L}\left(\dot{w}^{(1)}w^{(0)} + \dot{w}^{(0)}w^{(1)}\right), \tag{22}$$
$$\dot{w}^{(1)} = -\mathrm{sign}(w^{(1)}w^{(0)} - M)\sqrt{2L}\sigma_x w^{(0)}, \quad \dot{w}^{(0)} = -\mathrm{sign}(w^{(1)}w^{(0)} - M)\sqrt{2L}\sigma_x w^{(1)}, \tag{23}$$

combining these equations we obtain for the loss function and the NIW we have

$$\dot{L} = -2\sigma_x^2\left((w^{(0)})^2 + (w^{(1)})^2\right)L, \tag{24}$$
$$\dot{w}_{\mathrm{NI}} = -\sigma_\epsilon^2(w^{(1)})^2 w_{\mathrm{NI}}, \tag{25}$$

where we identify that the loss function evolves according to the trace of the full Hessian $\mathrm{Tr}\left(H_{\boldsymbol{\theta}}\right) = \sigma_x^2\left((w^{(0)})^2 + (w^{(1)})^2\right)$, while the NIW evolves according to the trace of the local Hessian $\mathrm{Tr}\left(\mathcal{H}_z\right) = (w^{(1)})^2$.

These equations imply an exponential evolution for both the loss function and the NIW. The relevant timescales can be read by integrating the equations, hence

$$L(t) \sim e^{-2\sigma_x^2\int_0^t\left((w^{(0)}(t'))^2 + (w^{(1)}(t'))^2\right)dt'}, \tag{26}$$
$$w_{\mathrm{NI}}(t) \sim e^{-\sigma_\epsilon^2\int_0^t(w^{(1)}(t'))^2 dt'}. \tag{27}$$

as presented in the main text.