Neglected Hessian component explains mysteries in sharpness regularization

Yann N. Dauphin
Google DeepMind
ynd@google.com

Atish Agarwala
Google DeepMind
thetish@google.com

Hossein Mobahi
Google DeepMind
hmobahi@google.com

Abstract

Recent work has shown that methods that regularize second order information like SAM can improve generalization in deep learning. Seemingly similar methods like weight noise and gradient penalties often fail to provide such benefits. We investigate this inconsistency and reveal its connection to the the structure of the Hessian of the loss. Specifically, its decomposition into the positive semi-definite Gauss-Newton matrix and an indefinite matrix, which we call the Nonlinear Modeling Error (NME) matrix. Previous studies have largely overlooked the significance of the NME in their analysis for various reasons. However, we provide empirical and theoretical evidence that the NME is important to the performance of gradient penalties and explains their sensitivity to activation functions. We also provide evidence that the difference in regularization performance between gradient penalties and weight noise can be explained by the NME. Our findings emphasize the necessity of considering the NME in both experimental design and theoretical analysis for sharpness regularization.

1 Introduction

There is a long history in machine learning of trying to use information about loss landscape geometry to improve gradient-based learning. This has ranged from attempts to use the Fisher information matrix to improve optimization [1], to trying to regularize the Hessian to improve generalization [2]. More recently, first order methods which implicitly use or penalize second order quantities have been used successfully, including the *sharpness aware minimization* (`SAM`) algorithm [3]. On the other hand, there are many approaches to use second order information which once seemed promising but have had limited success [4]. These include methods like weight noise [5] and gradient norm penalties [6, 7, 8, 9, 10], which have shown mixed success.

Part of the difficulty of using second order information is the difficulty of working with the Hessian of the loss. With the large number of parameters in deep learning architectures, as well as the large number of datapoints, many algorithms use stochastic methods to approximate statistics of the Hessian [1, 11]. However, there is a *conceptual* difficulty as well which arises from the complicated structure of the Hessian itself. Methods often involve approximating the Hessian via the Gauss-Newton (GN) matrix - which is PSD for convex losses. This is beneficial for conditioners which try to maintain monotonicity of gradient flow via a PSD transformation of the gradient. Thus indefinite part of the Hessian is often neglected due to its complexity.

In this work we show that it is important to consider *both* parts of the Hessian to understand certain methods that use second order information for regularization. We study the non-PSD part of the Hessian, which we call the *Nonlinear Modeling Error* (NME). In contrast to commonly held assumptions, this work reveals the NME is key to understanding two previously unexplained phenomena in sharpness regularization:

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

- **Training with Gradient Penalties.** We show that the performance of gradient penalties is sensitive to the choice of activation functions. Our theoretical analysis reveals that the NME is particularly sensitive to the second derivative of the activation function, and we show that gradient penalties fail to improve performance when these derivatives have poor numerical properties. We also design an intervention that can mitigate this issue. To the best of our knowledge, this work is the first to show that methods using second order information are more sensitive to the choice of activation function.
- **Training with Hessian penalties.** Conventional analysis of weight noise casts it as a penalty on the GN part of the Hessian, but in reality it also penalizes the NME. Our experimental ablations show that the NME exerts a significant influence on generalization performance, and minimizing it is generally bad for training.

We conclude with a discussion about how these insights might be used to design activation functions not with an eye towards forward or backwards passes [12, 13], but for compatibility with methods that use second order information.

2 Understanding the structure of the Hessian

In this section, we lay the theoretical ground work for our experiments by explaining the structure of the Hessian. Given a model $\mathbf{z}(\boldsymbol{\theta}, \mathbf{x})$ defined on parameters $\boldsymbol{\theta}$ and input $\mathbf{x}$, and a loss function $\mathcal{L}(\mathbf{z}, \mathbf{y})$ on the model outputs and labels $\mathbf{y}$, we can write the gradient of the training loss with respect to $\boldsymbol{\theta}$ as

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \mathbf{J}^{\mathrm{T}}(\nabla_{\mathbf{z}}\mathcal{L}) \tag{1}$$

where the Jacobian $\mathbf{J} \equiv \nabla_{\boldsymbol{\theta}}\mathbf{z}$. The Hessian $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}$ can be decomposed as:

$$\nabla^2_{\boldsymbol{\theta}}\mathcal{L} = \underbrace{\mathbf{J}^{\mathrm{T}}\mathbf{H}_{\mathbf{z}}\mathbf{J}}_{\mathrm{GN}} + \underbrace{\nabla_{\mathbf{z}}\mathcal{L}\cdot\nabla^2_{\boldsymbol{\theta}}\mathbf{z}}_{\mathrm{NME}} \tag{2}$$

where $\mathbf{H}_{\mathbf{z}} \equiv \nabla^2_{\mathbf{z}}\mathcal{L}$. The first term, often referred to as the Gauss-Newton (GN) part of the Hessian, is a generalization of the classical Gauss-Newton matrix [14, 15]. If the loss function is convex with respect to the model outputs/logits (such as for MSE and CE losses), then the GN matrix is positive semi-definite. This term often contributes large eigenvalues. The second term has previously been studied theoretically where it is called the *functional Hessian* [16, 17]; in order to avoid confusion with the overall Hessian we call it the *Nonlinear Modeling Error* matrix (NME). It is in general indefinite and vanishes to zero at an interpolating minimum $\boldsymbol{\theta}^*$ where the model "fits"the data ($\nabla_z\mathcal{L}(\boldsymbol{\theta}^*) = \mathbf{0}$), as can happen in overparameterized settings. Due to this, it is quite common for studies to drop this term entirely when dealing with the Hessian. For example, many second order optimizers approximate the Hessian $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}$ with only the Gauss-Newton term [11, 18]. It is also common to drop this term in theoretical analysis of the Hessian $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}$ [19, 20]. However, we will show why this term should not be ignored.

While the NME term can become small late in training, it encodes significant information during training. More precisely, *it is the only part of Hessian that contains second order information from the model features* $\nabla^2_{\boldsymbol{\theta}}\mathbf{z}$. The GN matrix only contains second order information about the loss w.r.t. the logits with the term $\mathbf{H}_{\mathbf{z}}$. All the information about the model function in the GN matrix is first-order. In fact, the GN matrix can be seen as the Hessian of an approximation of the loss where a first-order approximation of the model $\mathbf{z}(\boldsymbol{\theta}', \mathbf{x}) \approx \mathbf{z}(\boldsymbol{\theta}, \mathbf{x}) + \mathbf{J}\boldsymbol{\delta}$ ($\boldsymbol{\delta} = \boldsymbol{\theta}' - \boldsymbol{\theta}$) is used [18]

$$\nabla^2_{\boldsymbol{\delta}}\mathcal{L}(\mathbf{z}(\theta, \mathbf{x}) + \mathbf{J}\boldsymbol{\delta},\ \mathbf{y})|_{\boldsymbol{\theta}'=\boldsymbol{\theta}} = \mathbf{J}^{\mathrm{T}}\mathbf{H}_{\mathbf{z}}\mathbf{J} \tag{3}$$

Thus we can see the GN matrix as the result of a linearization of the model and the NME as the part that takes into account the non-linear part of the model. The GN matrix exactly determines the linearized (NTK) dynamics of training, and therefore controls learning over small parameter changes when the features can be approximated as fixed (see Appendix A.1). In contrast, the NME encodes information about the *changes* in the NTK [21]. For additional intuition in the ReLU setting, see Appendix A.3.

2.1 Sharpness regularization and the NME: Case of the Gauss-Newton trace

Sharpness regularization often relies on geometric quantities of the loss landscape such as the largest eigenvalue [3] or combinations of the eigenvalues [22]. To illustrate the impact of the NME, let us consider the sharpness measure given by the trace of the Gauss-Newton tr$(\mathbf{G}) = \sum_i \lambda_i$, which gives information about the average eigenvalue λ_i of the Gauss-Newton. This measure, studied in Section 5, also shares some similarities to Section 4's gradient penalties but is simpler to analyze. Surprisingly, even though the quantity purposely ignores the NME, we will see that its gradient still crucially relies on it.

For GN matrix $\mathbf{G}$, if the loss can be written as the log-likelihood of an exponential family distribution, this measure can be expressed as a gradient penalty [23]:

$$\text{tr}\,(\mathbf{G}) = \mathrm{E}_{\hat{\mathbf{y}}\sim \mathrm{p}(\mathbf{z})}[||\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{y}})||^2] \tag{4}$$

Here, the expectation is taken over labels $\hat{\mathbf{y}}$ sampled from $\mathrm{p}(\mathbf{z})$ - that is, the probability distribution induced by the model logits $\mathbf{z}(\boldsymbol{\theta})$ via the log-likelihood. $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{y}})$ is the gradient of this loss defined with the sampled labels; derivatives are taken after the sampling process. For MSE loss, the gradient of $\mathrm{tr}(\mathbf{G})$ can be written as:

$$\nabla_{\boldsymbol{\theta}}\text{tr}\,(\mathbf{G}) = \mathrm{E}_{\hat{\mathbf{y}}\sim \mathrm{p}(\mathbf{z})}[\mathbf{N}_{\hat{\mathbf{y}}}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \hat{\mathbf{y}})] \tag{5}$$

where $\mathbf{N}_{\hat{\mathbf{y}}}$ is NME of the loss defined with the sampled labels. In this case if the NME is set to 0, gradient descent could not effectively minimize this sharpness measure. The rest of the work will explore the relationship between the structure of the Hessian and various curvature regularization techniques beyond this case through experimental and theoretical work.

3 Experimental Setup

Our analysis of the Hessian begs an immediate question: when does the NME affect learning algorithms? We conducted experimental studies to answer this question in the context of curvature regularization algorithms which seek to promote convergence to flat areas of the loss landscape. We use the following setups for the remainder of the paper:

Fashion MNIST We also include results on Fashion MNIST [24]. All experiments use the WideResnet 28-10 architecture with the same setup and hyperparameters as those used in our CIFAR-10 experiments.

CIFAR-10 We provide results on the CIFAR-10 dataset [25]. All experiments use the WideResnet 28-10 architecture with the same setup and hyperparameters as [26], except for the use of cosine learning rate decay. Batch size is 128. Models are trained on 8 Nvidia Volta GPUs.

Imagenet We conduct experiments on the popular Imagenet dataset [27]. All experiments use the Resnet-50 architecture with the same setup and hyperparameters as [28], except that we use cosine learning rate decay [29] over 350 epochs. Batch size is set to 1024. Models are trained on using on TPU V3 chips.

4 Explaining the pitfalls of gradient penalties

In this section we explore the sensitivity of gradient penalty regularizers to the activation function via the NME. We first establish gradient penalty regularizers as an approximation of the Sharpness Aware Minimization (`SAM`) learning rule, and then present a mystery: why do gradient penalties recover the benefits of `SAM` for GELU and not for ReLU? We resolve the mystery with a theoretical analysis of the NME, combined with a series of ablation studies and design a simple intervention which alleviates the issue.

4.1 `SAM` and gradient penalties

The `SAM` algorithm originates from seeking a minimum with a *uniformly low loss* in its neighborhood (hence flat). This is formulated in Foret et al. [3] as a minmax problem,

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\epsilon}} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \quad \text{s.t.} \quad \|\boldsymbol{\epsilon}\| \leq \rho\,. \tag{6}$$

For computational tractability, [3] approximates the inner optimization by linearizing $\mathcal{L}$ w.r.t. $\boldsymbol{\epsilon}$ around the origin. Plugging the optimal $\boldsymbol{\epsilon}$ into the objective function yields

$$\min_{\boldsymbol{\theta}} \mathcal{L}\Big(\boldsymbol{\theta} + \rho\,\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\|}\Big)\,. \tag{7}$$

The SAM algorithm approximately minimizes the objective with the following learning rule:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} + \rho \tilde{\mathbf{g}}), \ \tilde{\mathbf{g}} \equiv \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) / ||\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})|| \tag{8}$$

for some step-size parameter $\eta > 0$.

For small ρ, there is an alternative to the SAM rule. We may approximate $\mathcal{L}$ in (7) by its first order Taylor expansion around the point $\rho = 0$ as below.

$$\begin{aligned} \mathcal{L}_{\text{PSAM}}(\boldsymbol{\theta}) &\triangleq \mathcal{L}(\boldsymbol{\theta})_{\rho=0} + \rho \Big(\frac{\partial}{\partial \rho} \mathcal{L}\left(\boldsymbol{\theta} + \rho \frac{\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})\|}\right)\Big)_{\rho=0} + O(\rho^2) && (9) \\ &= \mathcal{L}(\boldsymbol{\theta}) + \rho \left\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}), \frac{\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})\|} \right\rangle + O(\rho^2) && (10) \\ &= \mathcal{L}(\boldsymbol{\theta}) + \rho \|\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})\| + O(\rho^2). && (11) \end{aligned}$$

Dropping terms of $O(\rho^2)$ we arrive at a *gradient penalty regularizer*. In general we can define gradient penalties as additive regularizers of the form

$$\mathcal{L}_{pen,p} = \rho ||\nabla \mathcal{L}_0||^p \tag{12}$$

for a base loss $\mathcal{L}_0$. Gradient penalties have recently gained popularity as regularizers [6, 7, 8, 9, 10]. We will focus on the $p = 1$ case in the remainder of the section which we will refer to as Penalty SAM (or PSAM for short).

4.2 Penalty SAM vs Original SAM

A natural question arises: when do SAM and PSAM have similar effects? We find, surprisingly, that the answer is highly dependent on the activation function of the architecture. On Resnet50 trained on Imagenet, networks trained with GELU activation show similar performance for SAM training and PSAM training (Figure 1a); in contrast, with ReLU activation PSAM performs significantly worse than SAM as ρ is increased - indeed, becoming worse than the baseline for the best values of ρ for SAM (Figure 1b).

In order to understand this difference, we must first look at the update rule for PSAM. Given the base, pre-regularized loss $\mathcal{L}_0$, the update to the parameters $\boldsymbol{\theta}_t$ under SGD can be written as:

$$\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t = -\eta \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_0 + \frac{1}{||\nabla_{\boldsymbol{\theta}} \mathcal{L}_0||} \mathbf{H} \nabla_{\boldsymbol{\theta}} \mathcal{L}_0 \right) \tag{13}$$

where $\mathbf{H} \equiv \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_0$. Therefore, PSAM captures curvature information via an explicit Hessian-gradient product. This is in contrast to SAM, which captures curvature information by evaluating the gradient at the alternative point in the $\tilde{\mathbf{g}}$ direction (Equation 8). This lets SAM effectively "integrate" over in this direction and benefit from higher order information - in contrast to PSAM which only has access to the Hessian. The natural followup question is: how does the activation function influence the Hessian? In the remainder of this section, we provide evidence that it is in fact the NME component that is sensitive to this choice - and in general, it is the most important term to the overall performance of PSAM.

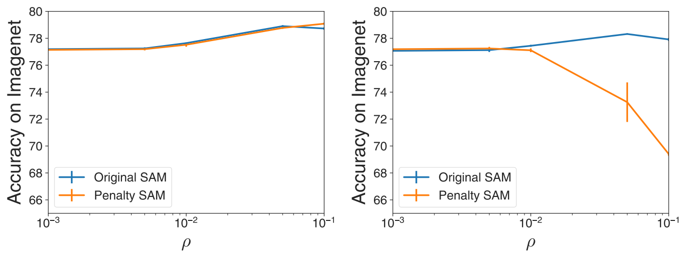
4.3 Effect of Activation functions on the NME

One important feature of the NME is that it depends on the *second derivative* of the activation function. We can demonstrate this most easily on a fully-connected network, but the general principle applies to most common architectures. Given an activation function ϕ, a feedforward network with L layers on an input $\mathbf{x}_0$ defined iteratively by

$$\mathbf{h}_l = \mathbf{W}_l \mathbf{x}_l, \ \mathbf{x}_{l+1} = \phi(\mathbf{h}_l) \tag{14}$$

The gradient of the model output $\mathbf{x}_L$ with respect to a weight matrix $\mathbf{W}_l$ is given by

$$\frac{\partial \mathbf{x}_L}{\partial \mathbf{W}_l} = \mathbf{J}_{L(l+1)} \circ \phi'(\mathbf{h}_l) \otimes \mathbf{x}_l, \ \mathbf{J}_{l'l} \equiv \prod_{m=l}^{l'-1} \phi'(\mathbf{h}_m) \circ \mathbf{W}_m, \ \phi'(x) \equiv \frac{d\phi}{dx}(x)$$



(a) Imagenet with GELU (b) Imagenet with ReLU

Figure 1: Test Accuracy as ρ increases across different datasets and activation functions averaged over 5 seeds. `PSAM` with GELU networks more closely follows the behavior of `SAM`. For ReLU networks and large ρ, there is a significant difference between `PSAM` and `SAM`.

where $\circ$ is the Hadamard (elementwise) product. The second derivative can be written as:

$$\frac{\partial^2 \mathbf{x}_L}{\partial \mathbf{W}_l \partial \mathbf{W}_m} = \left[\frac{\partial \mathbf{J}_{L(l+1)}}{\partial \mathbf{W}_m} \circ \phi'(\mathbf{h}_l) + \mathbf{J}_{L(l+1)} \circ \frac{\partial \phi'(\mathbf{h}_l)}{\partial \mathbf{W}_m}\right] \otimes \mathbf{x}_l \tag{15}$$

where without loss of generality $m \geq l$. The full analysis of this derivative can be found in Appendix A.4. The key feature is that the majority of the terms have a factor of the form

$$\frac{\partial \phi'(\mathbf{h}_o)}{\partial \mathbf{W}_m} = \phi''(\mathbf{h}_o) \circ \frac{\partial \mathbf{h}_o}{\partial \mathbf{W}_m}, \; \phi''(x) \equiv \frac{d^2\phi}{dx^2}(x) \tag{16}$$

via the product rule - a dependence on the second derivative ϕ'' of the activation function. On the diagonal $m = l$, all the terms depend on ϕ''. We note that a similar analysis can be found in Section 8.1.2 of [15].

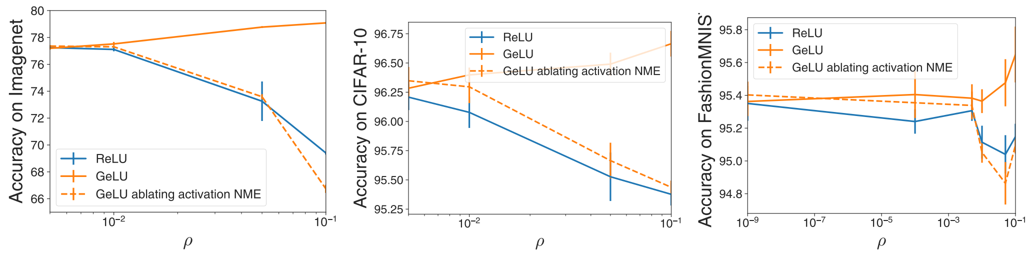
We immediately see how the role of the activation function differs in the NME compared to the gradient or even the GN: only the NME is sensitive to the second derivatives of the activation. GELU has a numerically stable second derivative function:

$$\frac{d^2}{dx^2}\mathrm{GELU}(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\left[2 - x^2\right] \tag{17}$$

Therefore, there are no issues computing the NME for GELU networks.

In contrast, the second derivative of ReLU is 0 everywhere except the origin, where it is undefined. In theoretical settings, the ReLU second derivative is defined as the Dirac delta "function" - more formally, the measure attained by a sequence of Gaussians centered at zero, as the standard deviation $\sigma \to 0$. It is integrable to 1 despite not attaining non-zero value anywhere. Therefore ReLU does not have a numerically well-posed second derivative, which affects computations involving the NME.

4.4 Ablating activation NME explains the gap



(a) Imagenet (b) CIFAR-10 (c) Fashion MNIST

Figure 2: Test accuracy as ρ increases for penalty SAM with parts of the NME ablated (average of 5 seeds). The removal of information from the NME controls the effectiveness of the gradient penalty.

In practical settings ReLU suffers from a "missing curvature" phenomenology: the second derivative is set to 0 in most autodifferentiating frameworks. This means that much of the information in the NME is inaccessible through the Hessian-vector product that at the heart of the update rule in Equation 13. Therefore `PSAM` suffers from a mismatch between the true NME information and the information available to the implemented algorithm. In contrast, `SAM` does not suffer from this issue; any curvature information is gained via differences in first derivatives - which by the fundamental theorem of calculus, are equivalent to integrals of second derivatives (and therefore of NME information).

We can confirm that the missing second derivative information in Equation 16 for ReLU networks is key for gradient penalties by removing it in GELU networks, and performing training with `PSAM`. This can be done by taking the second derivative of the GeLU to be zero:

$$\frac{d^2}{dx^2}\widehat{\mathrm{GELU}}(x) := 0 \tag{18}$$

and leaving the forward and backward propagation of the activation intact (see Appendix C.2 for implementation). This removes terms related to the activation in the same way ReLU does, so we denote as GeLU ablating activation NME for brevity. We see that across all 3 datasets in Figure 2 removing this portion of the NME degrades the performance of the GeLU with ρ similar to that of the ReLU, even though only the NME has been affected.

We can see similar results when we use a different method to approximate ReLU with GELU. The activation function $\mathrm{GELU}(\beta x)/\beta$ converges uniformly to ReLU as $\beta \to \infty$, while having well-posed derivatives for any finite β. We show in Appendix B.3 that gradient penalty performance degrades for large $\beta > 10^3$ - which we tie to high input sensitivity and increased sparsity of the second derivative. This suggests that even differentiable ReLU-like activation functions can fail to have compatibility with gradient penalties. We also confirm in Appendix C that ablating the full NME is also detrimental to generalization, though in a different way than ablating in the same manner as ReLU.

4.5 Adding synthetic activation NME improves results

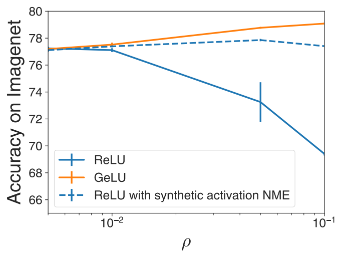


Figure 3: Test accuracy as ρ increases for penalty SAM with parts of the NME replaced synthetically (average of 5 seeds). The addition of information to the NME improves the performance as ρ increases.

Using this insight, we can modify the ReLU activation function to improve performance as follows. The issue with the ReLU is that the second derivative is a delta function - which can't be implemented numerically. However, we know that the delta function itself can be approximated by a Gaussian distribution. Therefore we replace the second derivative of ReLU with such a Gaussian

$$\frac{d^2}{dx^2}\widehat{\mathrm{ReLU}}(x) := \frac{\beta}{\sqrt{2\pi}}e^{-\beta^2x^2/2} \tag{19}$$

where β is the kernel width. We see that this prevents the drastic degradation at $\rho = 0.1$ of the original ReLU (Figure 3, $\beta = 100$). This suggests that it may be possible to design interventions to approximately access NME information in cases where second derivatives have poor numerical properties.

5 Explaining the pitfalls of Hessian penalties

In this section, we explore the impact of the NME on the effectiveness of Hessian penalties like weight noise. We first review the previously claimed link between gradient penalties and weight noise

[30], and then present a mystery: if this link is correct, why is there a difference in regularization boost between gradient penalties and weight noise? In contrast to the previous section where the NME solely featured in the update rule, weight noise implicitly minimizes the NME. We will show through ablations that minimizing the NME is detrimental, and explain why NME minimization is a poor strategy.

5.1 Weight noise equivalence assumes zero NME

We first review the analysis of training with noise established by [19]. Though the paper considers input noise, the same analysis can be applied to weight noise. Adding Gaussian $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$ noise with strength hyper-parameter σ to the parameters can be approximated to second order by

$$\begin{aligned} \mathrm{E}_{\boldsymbol{\epsilon}}[\mathcal{L}(\boldsymbol{\theta}+\boldsymbol{\epsilon})] &\approx \mathcal{L}(\boldsymbol{\theta}) + \cancel{\mathrm{E}_{\boldsymbol{\epsilon}}[\nabla_{\boldsymbol{\theta}}\mathcal{L}\cdot\boldsymbol{\epsilon}]} + \mathrm{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}^{\mathrm{T}}\mathbf{H}\boldsymbol{\epsilon}] \\ &= \mathcal{L}(\boldsymbol{\theta}) + \sigma^2 \mathrm{tr}(\mathbf{H}) \end{aligned} \tag{20}$$

where the second term has zero expectation since $\boldsymbol{\epsilon}$ is mean 0, and the third term is a variation of the Hutchison trace estimator [31]. (We note that though the second term vanishes in expectation, it still can have large effects on the training dynamics.) [19] argues that we can simplify the term related to the Hessian by dropping the NME in Equation 2 for the purposes of minimization, which in combination with 4 yields

$$\mathrm{tr}(\mathbf{H}) \approx \mathrm{tr}\left(\mathbf{G}\right) = \mathrm{E}_{\hat{\mathbf{y}}\sim\mathrm{Cat}(\mathbf{z})}[\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta},\hat{\mathbf{y}})\|^2]$$

The argument is that for the purposes of training neural networks, the NME can be dropped because it is zero at the global minimum we are trying to reach. However, the hypothesis that the NME has negligible impact in this setting has not been experimentally verified and we address this gap in the next section.

5.2 Minimizing the NME is detrimental

In order to study the impact of the NME in this setting, we evaluate ablations of weight noise to determine the impact of the different components. Recalling Equation 20, the methods we will consider are given by

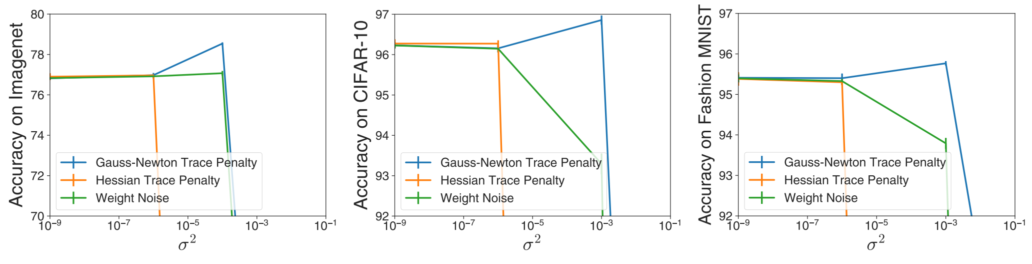
$$\underbrace{\mathrm{E}_{\boldsymbol{\epsilon}}[\mathcal{L}(\boldsymbol{\theta}+\boldsymbol{\epsilon})]}_{\text{Weight Noise}} = \overbrace{\underbrace{\mathcal{L}(\boldsymbol{\theta}) + \sigma^2\mathrm{tr}\left(\mathbf{G}\right)}_{\text{Gauss-Newton Trace Penalty}} + \sigma^2\mathrm{tr}\left(\mathbf{N}\right)}^{\text{Hessian Trace Penalty}} + \mathcal{O}(\|\boldsymbol{\epsilon}\|^2)$$

where $\mathbf{G} \equiv \mathbf{J}^{\mathrm{T}}\mathbf{H}_{\mathbf{z}}\mathbf{J}$ is the Gauss-Newton, $\mathbf{N} \equiv \nabla_{\mathbf{z}}\mathcal{L}\cdot\nabla^2_{\boldsymbol{\theta}}\mathbf{z}$ is the NME.

Hessian Trace penalty This ablation allows to us to single out the second order effect of weight noise, as it's possible the higher order terms from weight noise affect generalization. We implement this penalty with Hutchinson's trace estimator ($\mathrm{tr}(\mathbf{H}) = \mathrm{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}[\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon}]$).

Gauss-Newton Trace penalty This ablation removes the NME's contribution, enabling us to isolate and measure its specific influence on the model. We use the estimator from Equation 4, $\mathrm{tr}\left(\mathbf{G}\right) = \mathrm{E}_{\hat{\mathbf{y}}\sim\mathrm{Cat}(\mathbf{z})}[\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta},\hat{\mathbf{y}})\|^2]$, to compute the penalty. This is the norm of the gradients, with respect to labels $\hat{\mathbf{y}}$ sampled via the distribution $\mathrm{Cat}(\mathbf{z})$ induced by the model logits via the softmax [23]. This is an alternative gradient penalty, a point we will return to later. We do not pass gradients through the sampling of the labels $\hat{\mathbf{y}}$, but find similar results if we pass gradients using the straight-through estimator [32]. We also run experiments with the Hutchison's estimator $\mathrm{tr}(\mathbf{G}) = \mathrm{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}[\boldsymbol{\epsilon}^T\mathbf{G}\boldsymbol{\epsilon}]$ to control for the effect of the estimator.

Our experiments show that the methods perform quite differently for similar values of σ^2 (Figure 4) – confirming the influence of the NME. We can see that the generalization improvement of the Gauss-Newton Trace penalty is consistently greater than either weight noise or Hessian Trace penalty. Its improvement on Imagenet is a significant 1.6%. In contrast, the other methods provide little accuracy improvement. While these experiments use different estimators with a single sample, results are consistent even when compared with the same estimator and with 5 samples to get a better trace estimate (Figure 5).



(a) Imagenet (b) CIFAR-10 (c) Fashion MNIST

Figure 4: Test Accuracy as σ^2 increases across different datasets and activation functions averaged over 5 seeds. Large σ^2 reveals a stark contrast between the Gauss-Newton trace penalty, which excludes NME, and methods incorporating it, highlighting the NME's influence.

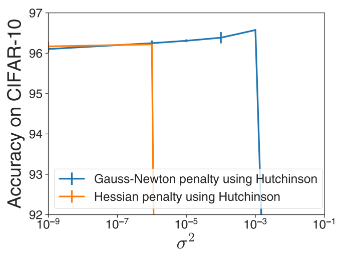


Figure 5: Test Accuracy as σ^2 increases with both penalties estimated using Hutchinson's estimator with 5 samples (curves are averaged over 2 seeds). Despite the larger number of samples, Hessian trace penalty remains unstable, while Gauss-Newton trace penalty is stable. This suggests the instability is not due the estimator.

In fact, both the weight noise and Hessian trace penalties show severe performance degradation for larger σ^2 - with the Hessian trace penalty in particular showing degradation as low as 10^{-6}, at least two orders of magnitude lower than the optimal GN trace regularizer values $\geq 10^{-4}$. This may be related to the fact that the full Hessian trace has no lower bound as it includes the indefinite NME, while the GN is a PSD matrix whose trace is bounded by 0. Measurements of the trace during training shows that indeed the trace grows large and negative superlinearly in the number of training iterations for the Hessian trace penalty (Figure 6, $\sigma^2 = 10^{-3}$).

Our experiments suggest that the Gauss-Newton trace is a better (and indeed, qualitatively different) regularizer than the full Hessian trace penalty. As computed in Section 2.1, this regularizer is a gradient penalty whose gradient relies crucially on the information in the NME. These results indicate that minimizing the NME in the loss is counter-productive. Similarly, Section 4.4 shows that zeroing out portions of the NME is also detrimental. This suggests that the NME is important in understanding the effects of curvature regularization.

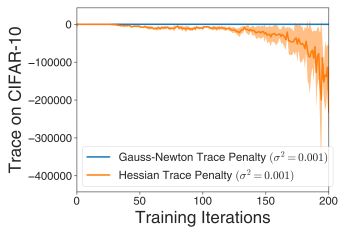


Figure 6: Trace penalty over training iterations for different methods averaged over 5 seeds. The trace of the Hessian can be negative due to the NME, while the trace of the Gauss-Newton cannot. Minimizing the Hessian trace causes it to become very negative, which is detrimental to training stability.

6 Discussion

Our theoretical analysis gives some understanding of the structure of the Hessian - in particular, the Nonlinear Modeling Error matrix. This piece of the Hessian is often neglected as it is generally indefinite and doesn't generate large eigenvalues, and is 0 at an interpolating minimum. However, the NME is the only part of the Hessian that encodes important second order information about the features, as it depends on $\nabla_{\boldsymbol{\theta}}^2 \mathbf{z}$ - the gradient of the Jacobian. Another intriguing observation was that the gradient of the trace of the Gauss Newton matrix $\nabla_{\boldsymbol{\theta}} \operatorname{tr}(\mathbf{G})$ can be written in terms of an NME-vector product. We also saw that the NME, particularly the diagonal, is sensitive to the second derivative of the activation function.

Our experiments suggest that these second derivative properties can be quite important when training with gradient penalty regularizers. ReLU has a poorly defined pointwise second derivative, and the resulting regularizer harms training. In contrast, GELU has a well defined one and gains benefits from modest values of the regularizer. Our ablation experiments showed that removing the second derivatives prevented gradient penalties from usefully using the NME information. We also found an alternative approximation for ReLU second derivatives which added NME information and improved training.

These results suggest that some second order methods may benefit from tuning the NME. This is especially true for methods which result in Hessian-vector products in update rules (like the gradient and Hessian trace penalties studied here). Another interesting avenue for research is to replace explicit second order methods with implicit second order methods which use first order information at discrete intervals - analogous to how `SAM` avoided sensitivity to bad second derivatives by "integrating" over a direction via differences, which accessed averaged quantities and higher order information.

Our experiments on Hessian trace penalties confirmed that the NME is important to understanding the successes and failures of those methods. It is intriguing that the variant which performed best, the GN trace penalty, can itself be written as an alternative gradient penalty. Exploring other gradient penalties is a promising research direction; they are non-negative, easy to compute, and generally contain NME information in their update rules.

7 Conclusion

Our work sheds light on the complexities of using second order information in deep learning. We have identified clear cases where it is important to consider the effects of *both* the Gauss-Newton and Nonlinear Modeling Error terms, and design algorithms and architectures with that in mind. This insight may unlock new classes of second order algorithms which use loss landscape geometry in qualitatively different ways.

References

[1] James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2408–2417. PMLR, June 2015.

[2] Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9481–9488, 2021.

[3] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[4] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[5] Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

[6] David G. T. Barrett and Benoit Dherin. Implicit gradient regularization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=3q5IqUrkcF.

[7] Samuel L. Smith, Benoit Dherin, David G. T. Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent, 2021.

[8] Jiawei Du, Zhou Daquan, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=xK6wRfL2mv7.

[9] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning, 2022.

[10] Patrik Reizinger and Ferenc Huszár. SAMBA: Regularized autoencoders perform sharpness-aware minimization. In *Fifth Symposium on Advances in Approximate Bayesian Inference*, 2023. URL https://openreview.net/forum?id=gk3PAmy_UNz.

[11] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training, 2023.

[12] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] James Martens, Andy Ballard, Guillaume Desjardins, Grzegorz Swirszcz, Valentin Dalibard, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Rapid training of deep neural networks without skip connections or normalization layers using Deep Kernel Shaping. *arXiv:2110.01765 [cs]*, October 2021.

[14] Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

[15] James Martens. New Insights and Perspectives on the Natural Gradient Method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. ISSN 1533-7928.

[16] Sidak Pal Singh, Gregor Bachmann, and Thomas Hofmann. Analytic Insights into Structure and Rank of Neural Network Hessian Maps, July 2021.

[17] Sidak Pal Singh, Thomas Hofmann, and Bernhard Schölkopf. The Hessian perspective into the Nature of Convolutional Neural Networks. In *Proceedings of the 40th International Conference on Machine Learning*, pages 31930–31968. PMLR, July 2023.

[18] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1033–1040, 2011.

[19] Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[20] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[21] Atish Agarwala, Fabian Pedregosa, and Jeffrey Pennington. Second-order regression models exhibit progressive sharpening to the edge of stability, October 2022.

[22] Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022.

[23] Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. In *International conference on machine learning*, pages 10181–10192. PMLR, 2020.

[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[28] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[30] Kevin Ho and John Sum. Note on weight noise injection during training a mlp. *Proc. TAAI'2009*, 2009.

[31] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[32] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[33] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems 31*, pages 8571–8580. Curran Associates, Inc., 2018.

[34] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In *Advances in Neural Information Processing Systems 32*, pages 8570–8581. Curran Associates, Inc., 2019.

[35] Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In *Advances in Neural Information Processing Systems 32*, pages 2937–2947. Curran Associates, Inc., 2019.

[36] Atish Agarwala, Jeffrey Pennington, Yann Dauphin, and Sam Schoenholz. Temperature check: Theory and practice for training models with softmax-cross-entropy losses, October 2020.

[37] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

[38] Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pages 639–668. PMLR, 2022.

[39] Atish Agarwala and Yann Dauphin. SAM operates far from home: Eigenvalue regularization as a dynamical phenomenon. In *Proceedings of the 40th International Conference on Machine Learning*, pages 152–168. PMLR, July 2023.

[40] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

A Hessian structure

A.1 Gauss-Newton and NTK learning

In the large width limit (width/channels/patches increasing while dataset is fixed), the learning dynamics of neural networks are well described by the *neural tangent kernel*, or NTK [33, 34]. Consider a dataset size D, with outputs $\mathbf{z}(\boldsymbol{\theta}, \mathbf{X})$ over the inputs $\mathbf{X}$ with parameters $\boldsymbol{\theta}$. The (empirical) NTK $\hat{\boldsymbol{\Theta}}$ is the $D \times D$ matrix given by

$$\hat{\boldsymbol{\Theta}} \equiv \frac{1}{D}\mathbf{J}\mathbf{J}^{\mathrm{T}}, \ \mathbf{J} \equiv \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}} \tag{21}$$

For wide enough networks, the learning dynamics can be written in terms of the model output $\mathbf{z}$ and the NTK $\hat{\Theta}$ alone. For small learning rates we can study the gradient flow dynamics. The gradient flow dynamics on the parameters $\boldsymbol{\theta}$ with loss function $\mathcal{L}$ (averaged over the dataset) is given by

$$\dot{\boldsymbol{\theta}} = -\frac{1}{D}\nabla_{\boldsymbol{\theta}}\mathcal{L} = -\frac{1}{D}\mathbf{J}^{\mathrm{T}}\nabla_{\mathbf{z}}\mathcal{L} \tag{22}$$

We can use the chain rule to write down the dynamics of $\mathbf{z}$:

$$\dot{\mathbf{z}} = \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}}\dot{\boldsymbol{\theta}} = -\frac{1}{D}\mathbf{J}\mathbf{J}^{\mathrm{T}}\nabla_{z}\mathcal{L} = -\hat{\boldsymbol{\Theta}}\nabla_{z}\mathcal{L} \tag{23}$$

In the limit of infinite width, the overall changes in individual parameters become small, and the $\hat{\Theta}$ is fixed during training. This corresponds to the *linearized* or *lazy* regime [35, 36]. The NTK encodes the linear response of $\mathbf{z}$ to small changes in $\boldsymbol{\theta}$, and the dynamics is closed in terms of $\mathbf{z}$. For finite width networks, this can well-approximate the dynamics for a number of steps related to the network width amongst other properties [34].

In order to understand the dynamics of Equation 23 at small times, or around minima, we can linearize with respect to $\mathbf{z}$. We have:

$$\frac{\partial \dot{\mathbf{z}}}{\partial \mathbf{z}} = -\frac{\partial \hat{\boldsymbol{\Theta}}}{\partial \mathbf{z}}\nabla_{\mathbf{z}}\mathcal{L} - \hat{\boldsymbol{\Theta}}\mathbf{H}_{\mathbf{z}} \tag{24}$$

where $\mathbf{H}_{\mathbf{z}} = \frac{\partial^2 \mathcal{L}}{\partial \mathbf{z}\partial \mathbf{z}'}$. In the limit of large width, the NTK is constant and the first term vanishes. The local dynamics depends on the spectrum of $\hat{\boldsymbol{\Theta}}\mathbf{H}_{\mathbf{z}}$. From the cyclic property of the trace, the non-zero part of the spectrum is equal to the non-zero spectrum of $\frac{1}{D}\mathbf{J}^{\mathrm{T}}\mathbf{H}_{\mathbf{z}}\mathbf{J}$ - which is the Gauss-Newton matrix.

Therefore the eigenvalues of the Gauss-Newton matrix control the short term, linearized dynamics of $\mathbf{z}$, for fixed NTK. It is in this sense that the Gauss-Newton encodes information about exploiting the local linear structure of the model.

A.2 GN and second order information

The GN part of the Hessian may *seem* like it must contain second order information about the model due to its equivalence to the Fisher information matrix for losses that can be written as negative log-likelihoods, like MSE and cross-entropy. For these, the Fisher information itself can be written as the Hessian of a slightly different loss [37]:

$$\mathbf{F} = \mathrm{E}_{\hat{\mathbf{y}}\sim\mathbf{p}_{\mathbf{z}}}\left[\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{z}, \hat{\mathbf{y}})\right] \tag{25}$$

where the only difference is that the labels $\hat{\mathbf{y}}$ are sampled from the model instead of the true labels. However, the NME is 0 for this loss. For example, in the case of MSE using Equation 2 we have

$$\begin{aligned}\mathrm{E}_{\hat{\mathbf{y}}}\left[\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{z}, \hat{\mathbf{y}})\right] &= \mathrm{E}_{\hat{\mathbf{y}}}\left[\mathbf{J}^{\mathrm{T}}\mathbf{H}_{\mathbf{z}}\mathbf{J} + \nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}, \hat{\mathbf{y}}) \cdot \nabla^2_{\boldsymbol{\theta}}\mathbf{z}\right] \\ &= \mathbf{J}^{\mathrm{T}}\mathbf{H}_{\mathbf{z}}\mathbf{J} + \cancel{\mathrm{E}_{\hat{\mathbf{y}}\sim\mathcal{N}(\mathbf{z},\mathbf{I})}[\mathbf{z} - \hat{\mathbf{y}}]} \cdot \nabla^2_{\boldsymbol{\theta}}\mathbf{z}\end{aligned}$$

The second term vanishes because we are at the global minimum for this loss. Therefore, as expected, the Fisher information (and therefore, the Gauss Newton matrix) carry no information about the model second derivatives $\nabla^2_{\boldsymbol{\theta}}\mathbf{z}$.

A.3 Intuitions about NME with ReLU

In a piecewise multilinear model like a ReLU network, we can think of the GN part of the Hessian as *exploiting* the linear (NTK) structure, while the NME gives information on *exploration* - namely, the benefits of switching to a different multilinear region where different neurons are active. The NME is made out of the sum of Dirac delta functions, whose "spikes" are given by the boundaries between the multilinear regions corresponding to ReLU reaching the saturated regime. The NME is small outside those regions.

We can gain additional intuition by constructing a differentiable approximation of ReLU. We define the β-GELU by

$$\beta\text{-GELU}(x) = x\Phi(\beta x) \tag{26}$$

where Φ is the standard Gaussian CDF. We can recover GELU by setting $\beta = 1$. β-GELU converges uniformly to ReLU in the limit $\beta \to \infty$. The second derivative is given by

$$\frac{d^2}{dx^2}\beta\text{-GELU}(x) = \frac{1}{\sqrt{2\pi\beta^{-2}}}e^{-x^2/2\beta^{-2}}\left[2 - (x/\beta^{-1})^2\right] \tag{27}$$

For large β, this function is exponentially small when $x \gg \beta^{-1}$, and $O(\beta)$ when $|x| = O(\beta^{-1})$. As β increases the non-zero region becomes smaller while the non-zero value becomes larger such that the integral is always 1.

The choice of β determines how much information the NME can convey in a practical setting. This second derivative is large only when the input to the activation is within distance $1/\beta$ of 0. In a deep network this corresponds to being near the boundary of the piecewise multilinear regions where the activations switch on and off in an equivalent ReLU newtork.

We can illustrate this using two parameters of an MLP in the same layer, where with ReLU activation the model is in fact piecewise linear with respect to those parameters (Figure 7). For GELU with large β, the second derivative serves as an "edge detector"[1] (more generally, hyperplane detector) of the corresponding multilinear boundaries for the equivalent ReLU network (denoted by the blue lines in the right panel of Figure 7). Therefore, the NME can be used to probe the usefulness of crossing these edges.

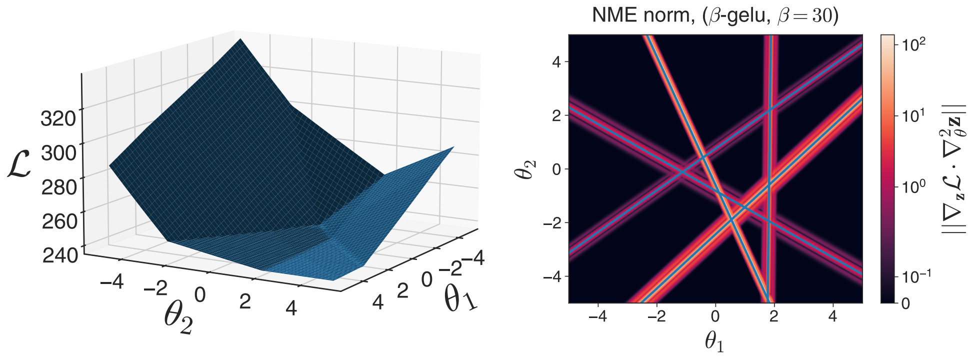


Figure 7: Loss (left) and Nonlinear Modeling Error matrix (NME) norm (right) as a function of 2 parameters in the same hidden layer of an MLP (MSE loss, one datapoint). For ReLU activation model is piecewise multilinear, and piecewise linear for parameters in same layer. Loss is piecewise quadratic for parameters in same layer (left). There is little NME information accessible pointwise and the main features are the boundaries of the piecewise linear regions (blue, right). For β-GELU, NME magnitude is high only within distance $1/\beta$ of those boundaries. Therefore the NME encodes information about the utility of switching between piecewise multilinear regions.

[1]In fact, the negative of the second order derivative of GELU is closely related to the Laplacian of Gaussian, which is a well-known edge-detector in image processing and computer vision.

A.4 Nonlinear Modeling Error and second derivatives of FCNs

We can explicitly compute the Jacobian and second derivative of the model for a fully connected network. We write a feedforward network as follows:

$$\mathbf{h}_l = \mathbf{W}_l \mathbf{x}_l,\ \mathbf{x}_{l+1} = \phi(\mathbf{h}_l) \tag{28}$$

The gradient of $\mathbf{x}_L$ with respect to $\mathbf{W}_l$ can be written as:

$$\frac{\partial \mathbf{x}_L}{\partial \mathbf{W}_l} = \frac{\partial \mathbf{x}_L}{\partial \mathbf{h}_l} \frac{\partial \mathbf{h}_l}{\partial \mathbf{W}_l} \tag{29}$$

which can be written in coordinate-free notation as

$$\frac{\partial \mathbf{x}_L}{\partial \mathbf{W}_l} = \frac{\partial \mathbf{x}_L}{\partial \mathbf{h}_l} \otimes \mathbf{x}_l \tag{30}$$

If we define the partial Jacobian $\mathbf{J}_{l'l} \equiv \frac{\partial \mathbf{x}_{l'}}{\partial \mathbf{x}_l}$, $l' > l$

$$\frac{\partial \mathbf{x}_L}{\partial \mathbf{W}_l} = \mathbf{J}_{L(l+1)} \circ \phi'(\mathbf{h}_l) \otimes \mathbf{x}_l \tag{31}$$

Here $\circ$ denotes the Hadamard product, in this case equivalent to matrix multiplication by $\mathrm{diag}(\phi'(\mathbf{h}_m))$.

The Jacobian can be explicitly written as

$$\mathbf{J}_{l'l} = \prod_{m=l}^{l'-1} \phi'(\mathbf{h}_m) \circ \mathbf{W}_m \tag{32}$$

Therefore, we can write:

$$\frac{\partial \mathbf{x}_L}{\partial \mathbf{W}_l} = \left[\prod_{m=l+1}^{L-1} \phi'(\mathbf{h}_m) \circ \mathbf{W}_m \right] \circ \phi'(\mathbf{h}_l) \otimes \mathbf{x}_l \tag{33}$$

The second derivative is more complicated. Consider

$$\frac{\partial^2 \mathbf{x}_L}{\partial \mathbf{W}_l \partial \mathbf{W}_m} = \frac{\partial}{\partial \mathbf{W}_m} \left[\mathbf{J}_{L(l+1)} \circ \phi'(\mathbf{h}_l) \otimes \mathbf{x}_l \right] \tag{34}$$

for weight matrices $\mathbf{W}_l$ and $\mathbf{W}_m$. Without loss of generality, assume $m \geq l$.

We first consider the case where $m > l$. In this case, we have

$$\frac{\partial \phi'(\mathbf{h}_l)}{\partial \mathbf{W}_m} = 0,\ \frac{\partial \mathbf{x}_l}{\partial \mathbf{W}_m} = 0 \tag{35}$$

since $\mathbf{W}_m$ comes after $\mathbf{h}_l$. If we write down the derivative of $\mathbf{J}_{L(l+1)}$, there are two types of terms. The first comes from the direct differentiation of $\mathbf{W}_m$; the others come from differentation of $\phi'(\mathbf{h}_n)$ for $n \geq m$. We have:

$$\frac{\partial \mathbf{J}_{L(l+1)}}{\partial \mathbf{W}_m} = \mathbf{J}_{L(m+1)} \phi'(\mathbf{h}_m) \frac{\partial \mathbf{W}_m}{\partial \mathbf{W}_m} \mathbf{J}_{(m-1)(l+1)} + \sum_{o=m}^{L-1} \mathbf{J}_{L(o+1)} \frac{\partial \phi'(\mathbf{h}_o)}{\partial \mathbf{W}_m} \mathbf{W}_o \mathbf{J}_{(o-1)(l+1)} \tag{36}$$

The $\mathbf{W}_m$ derivative projected into a direction $\mathbf{B}$ can be written as:

$$\begin{aligned} \frac{\partial \mathbf{J}_{L(l+1)}}{\partial \mathbf{W}_m} \cdot \mathbf{B} &= \mathbf{J}_{L(m+1)} \phi'(\mathbf{h}_m) \mathbf{B} \mathbf{J}_{(m-1)(l+1)} \\ &+ \sum_{o=m}^{L-1} \mathbf{J}_{L(o+1)} \left(\phi''(\mathbf{h}_o) \circ \mathbf{W}_o \frac{\partial \mathbf{x}_{o-1}}{\partial \mathbf{W}_m} \cdot \mathbf{B} \right) \mathbf{W}_o \mathbf{J}_{(o-1)(l+1)} \end{aligned} \tag{37}$$

From our previous analysis, we have:

$$\frac{\partial \mathbf{J}_{L(l+1)}}{\partial \mathbf{W}_m} \cdot \mathbf{B} = \mathbf{J}_{L(m+1)}\phi'(\mathbf{h}_m)\mathbf{B}\mathbf{J}_{(m-1)(l+1)} \\ + \sum_{o=m}^{L-1} \mathbf{J}_{L(o+1)} \left(\phi''(\mathbf{h}_o) \circ \left[\mathbf{W}_o\mathbf{J}_{o(m+1)} \circ \phi'(\mathbf{h}_{m+1}) \circ \mathbf{B}\mathbf{x}_m\right]\right) \frac{\partial \phi'(\mathbf{h}_o)}{\partial \mathbf{W}_m}\mathbf{W}_o\mathbf{J}_{(o-1)(l+1)} \tag{38}$$

In total, the second derivative projected into the $(\mathbf{A}, \mathbf{B})$ direction for $m > l$ is given by:

$$\frac{\partial^2 \mathbf{x}_L}{\partial \mathbf{W}_l \partial \mathbf{W}_m} \cdot (\mathbf{A} \otimes \mathbf{B}) = \Bigg[\mathbf{J}_{L(m+1)}\phi'(\mathbf{h}_m)\mathbf{B}\mathbf{J}_{(m-1)(l+1)} + \\ \sum_{o=m}^{L-1} \mathbf{J}_{L(o+1)} \left(\phi''(\mathbf{h}_o) \circ \left[\mathbf{W}_o\mathbf{J}_{o(m+1)} \circ \phi'(\mathbf{h}_{m+1}) \circ \mathbf{B}\mathbf{x}_m\right]\right) \frac{\partial \phi'(\mathbf{h}_o)}{\partial \mathbf{W}_m}\mathbf{W}_o\mathbf{J}_{(o-1)(l+1)}\Bigg] \\ \circ\, \phi'(\mathbf{h}_l)\mathbf{A}\mathbf{x}_l \tag{39}$$

Now consider the case $m = l$. Here there is no direct differentiation with respect to $\mathbf{W}_m$, but there is a derivative with respect to $\phi'(\mathbf{h}_m)$. The derivative is written as:

$$\frac{\partial^2 \mathbf{x}_L}{\partial \mathbf{W}_m \partial \mathbf{W}_m} \cdot (\mathbf{A} \otimes \mathbf{B}) = \mathbf{J}_{L(m+1)} \circ [\phi''(\mathbf{h}_m) \circ \mathbf{B}\mathbf{x}_l]\mathbf{A}\mathbf{x}_m + \\ \left[\sum_{o=m}^{L-1} \mathbf{J}_{L(o+1)} \left(\phi''(\mathbf{h}_o) \circ \left[\mathbf{W}_o\mathbf{J}_{o(m+1)} \circ \phi'(\mathbf{h}_{m+1}) \circ \mathbf{B}\mathbf{x}_m\right]\right) \frac{\partial \phi'(\mathbf{h}_o)}{\partial \mathbf{W}_m}\mathbf{W}_o\mathbf{J}_{(o-1)(m+1)}\right] \\ \circ\, \phi'(\mathbf{h}_m)\mathbf{A}\mathbf{x}_m \tag{40}$$

There are two key points: first, all but one of the terms in the off-diagonal second derivative depend on only first derivatives of the activation; for a deep network, the majority of the terms depend on ϕ''. Secondly, on the diagonal, all terms depend on ϕ''. Therefore if $\phi''(x) = 0$, the diagonal of the model second derivative is 0 as well.

B SAM and gradient penalties

The gradient penalties studied in Section 4 are related to the Sharpness Aware Minimization algorithm (SAM) developed to combat high curvature in deep learning [3]. In this appendix we review some additional facts about SAM and gradient penalties.

B.1 USAM

A related learning algorithm is unnormalized SAM (USAM) with update rule [38]

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta} + \rho \mathbf{g}\right), \; \mathbf{g} \equiv \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{41}$$

USAM has similar performance to SAM and is easier to analyze [39]. The unnormalized gradient penalty equivalent PUSAM is

$$\mathcal{L}_{\text{PUSAM}}(\boldsymbol{\theta}) \triangleq \mathcal{L}(\boldsymbol{\theta}) + \rho\,||\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})||^2 + O(\rho^2)\,. \tag{42}$$

which corresponds to the $p = 2$ case of the gradient penalty.

B.2 Penalty SAM vs. implicit regularization of SGD

The analysis of [7] suggested that SGD with learning rate η is similar to gradient flow (GF) with PUSAM with $\rho = \eta/4$. In this section we use a linear model to highlight some key differences between PUSAM and the discrete effects from finite stepsize SGD.

Consider a quadratic loss $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^{\mathrm{T}}\mathbf{H}\boldsymbol{\theta}$ for some parameters $\boldsymbol{\theta}$ and PSD Hessian $\mathbf{H}$. It is illustrative to consider gradient descent (GD) with learning rate η and (unnormalized) penalty `SAM` with radius ρ.

The gradient descent update rule is

$$\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t = -\eta(\mathbf{H} + \rho\mathbf{H}^2)\boldsymbol{\theta}_t \tag{43}$$

The "effective Hessian" is given by $\mathbf{H} + \rho\mathbf{H}^2$ (see [39] for more analysis). Solving the linear equation gives us

$$\boldsymbol{\theta}_t = \left(1 - \eta(\mathbf{H} + \rho\mathbf{H}^2)\right)^t \boldsymbol{\theta}_0 \tag{44}$$

This dynamics is well described by the eigenvalues of the effective Hessian - $\lambda + \rho\lambda^2$, where λ are the eigenvalues of $\mathbf{H}$. The effect of the regularizer is therefore to introduce eigenvalue-dependent modifications into the Hessian.

There is a special setting of ρ which can be derived from the calculations in [7]. Consider $\rho = \eta/2$, and consider the dynamics after $2t$ steps. We have:

$$\boldsymbol{\theta}_{2t} = \left(1 - \eta(\mathbf{H} + \frac{1}{2}\eta\mathbf{H}^2)\right)^{2t} \boldsymbol{\theta}_0 \tag{45}$$

which can be re-written as

$$\boldsymbol{\theta}_{2t} = \left(1 - 2\eta\mathbf{H} + \eta^3\mathbf{H}^3 + \frac{1}{4}\eta^4\mathbf{H}^4\right)^t \boldsymbol{\theta}_0 \tag{46}$$

To leading order in $\eta\mathbf{H}$, this is the same as the dynamics for learning rate 2η, $\rho = 0$ after t steps:

$$\boldsymbol{\theta}_t = (1 - 2\eta\mathbf{H})^t \boldsymbol{\theta}_0 \tag{47}$$

We note that these two are similar only if $\eta\mathbf{H} \ll 1$. Under this condition, $\eta\rho\mathbf{H}^2 = \frac{1}{2}\eta^2\mathbf{H}^2 \ll \eta\mathbf{H}$, and the gradient penalty only has a small effect on the overall dynamics. In many practical learning scenarios, including those involving `SAM`, $\eta\lambda$ can become $O(1)$ for many eigenvalues during training [39]. In these scenarios there will be qualitative differences between using penalty `SAM` and training with a different learning rate.

In addition, when ρ is set arbitrarily, the dynamics of η and 2η will no longer match to second order in $\eta\mathbf{H}$. This provides further theoretical evidence that combining SGD with penalty `SAM` is qualitatively and quantitatively different from training with a larger learning rate.

B.3 β-GELU experiments

We can understand the failure of NME information accessibility in ReLU via another method - constructing a sequence of ever closer approximations of ReLU which are still differentiable. We return to the β-GELU defined in Equation 26:

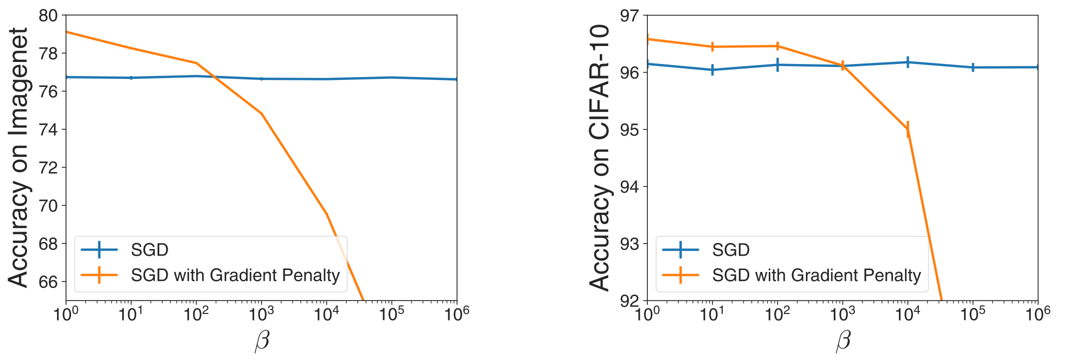
$$\beta\text{-GELU}(x) = x\Phi(\beta x)$$

We recall that $\lim_{\beta\to\infty} \beta\text{-GELU}(x) = \mathrm{ReLU}(x)$ - in fact, with uniform convergence. Note that this uniform convergence does not extend to second derivatives; β-GELU for large β is different from standard ReLU implementations at the origin, since it has second derivative β, and not 0, at the origin.

We can then ask: does β-GELU with large β behave similarly to ReLU? Training with SGD, we see that performance is close to invariant across 6 orders of magnitude of β (Figure 8, blue curves - average and standard deviation over 5 seeds). This is consistent with the uniform convergence which implies that forward and backwards passes of β-GELU become more similar to ReLU at large β.

In contrast, if we train with `PSAM` with $\rho = 0.1$ (the best setting we found for regular GELU with $\beta = 1$), we find that performance degrades with β, decreasing rapidly on a log scale for $\beta \geq 10^3$ (Figure 8, orange). This once again suggests that choice of activation function matters, and that ReLU-like activations combine poorly with gradient penalties.

This is interesting because for any finite β, β-GELU is in fact infinitely differentiable. Why are there issues with large β then? We can return to the form of $\nabla^2_{\boldsymbol{\theta}}\mathbf{z}$ for the answer. Recall from Equation 15



(a) Imagenet (b) CIFAR-10

Figure 8: Accuracy vs β for SGD and SGD with gradient penalty ($\rho = 0.1$) using β-GELU activations (average of 5 seeds). We observe that accuracy decreases with larger β with the gradient penalty but not without it. As our theory suggests that the sparsity of the NME increases with β, this is evidence that it has significant impact on gradient penalties.

that much of the NME depends on the second derivative of the activation function. From Equation 27, the second derivatives are exponentially small outside of a narrow band of width β^{-1}. Within this band, the derivatives are $O(\beta)$.

Therefore, for large β, the elements of the NME become "high frequency" functions - a small change in parameter values can lead to a massive change in the function value. Indeed, the statistics are characterized by sparse, large entries. For the Imagenet and CIFAR10 examples we can measure sparsity in the second derivative of the activation directly (Figure 9). We see that at initialization, the fraction of nonzero second derivatives drops with β (blue curve); this trend becomes much stronger after training, particularly in the Imagenet case (orange curve). This gives us a possible explanation for the failure at large β; even though the second derivatives exist, the NME becomes a sparse, high frequency estimate of local curvature information and does not provide value during training.

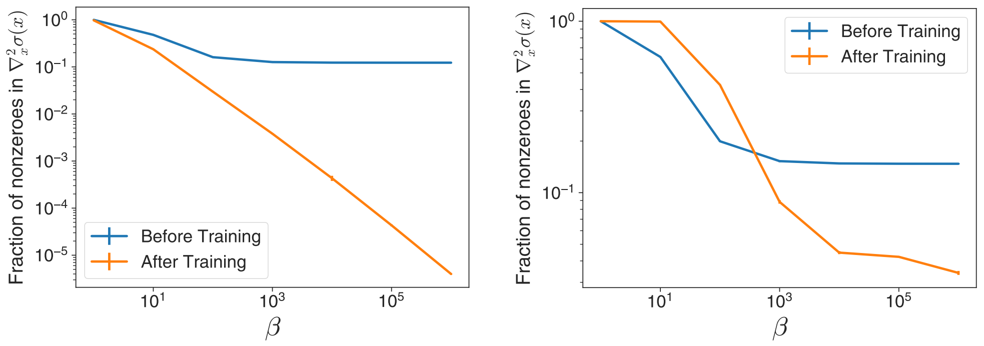


Figure 9: Fraction of non-zero activation second derivatives for β-GELU trained on Imagenet (left) and CIFAR10 (right). At initialization, fraction of non-zeros decreases somewhat with β (blue); after training, fraction of non-zeros depends strongly on β (orange).

Note that we are not claiming that the choice of the activation function is a sufficient condition for gradient penalties to work with larger ρ. There are many architectural changes that can affect the NME matrix and we have shown that the statistics of the activation function is a significant one.

C NME ablation details

In this section we go into more detail about how to implement various ablations of the NME shown in the experiments.

C.1 Gradient penalty, Gauss-Newton only

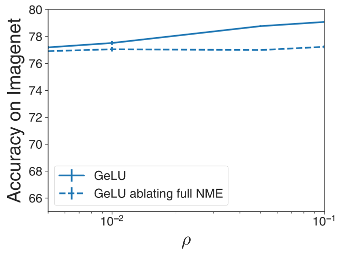


Figure 10: Test Accuracy as ρ increases ablating full NME from the update. We can see ablating the full NME is detrimental to performance.

For cross-entropy loss, we can completely remove the NME from the update rule in Equation 13 using a modified version of the penalty. Given model logits $\mathbf{z}(\boldsymbol{\theta})$, let $\mathbf{p}(\mathbf{z})$ be the probability induced by $\mathbf{z}$ (via the softmax). Given the true labels $\mathbf{y}$, consider the quantity:

$$g(\boldsymbol{\theta}, \mathbf{y}, \mathbf{t}) \equiv \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}) \cdot \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{t}) \tag{48}$$

for a probability vector $\mathbf{t}$. Differentiating with respect to $\boldsymbol{\theta}$, we have:

$$\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}, \mathbf{y}, \mathbf{t}) = \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{y})\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{t}) + \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{t})\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}) \tag{49}$$

This is simply the sum of two Hessian vector products.

Consider the special case where $\mathbf{t} = \mathbf{p}(\mathbf{z})$. In this case, $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{p}(\mathbf{z})) = 0$ - since the choice of logits $\mathbf{z}$ minimizes the loss with respect to true distribution $\mathbf{p}(\mathbf{z})$. The first term vanishes. This leaves us with the second term. Expanding the Hessian $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{t})$ into the GN and NME, we have:

$$\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{t}) = \mathbf{J}^{\mathrm{T}}\mathbf{H}_{\mathbf{z}}(\boldsymbol{\theta}, \mathbf{t})\mathbf{J} + \nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}(\boldsymbol{\theta}), \mathbf{t}) \cdot \nabla^2_{\boldsymbol{\theta}}\mathbf{z} \tag{50}$$

For cross-entropy loss, $H_{\mathbf{z}}$ is independent of the labels, and the first term is simply the GN matrix $\mathbf{G}$. For $\mathbf{t} = \mathbf{p}(\mathbf{z})$, we have $\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}(\boldsymbol{\theta}), \mathbf{p}(\mathbf{z})) = 0$ once again. Combining, we have:

$$\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}, \mathbf{y}, \mathbf{p}(\mathbf{z})) = \mathbf{G}\nabla_{\boldsymbol{\theta}}\mathcal{L} \tag{51}$$

Note that we are only differentiating with respect to the first coordinate of g. This is exactly the GN-vector product, rather than the Hessian-vector product found in Equation 13.

Therefore, if we define $\mathcal{L}_{GNpen,p}$ to be the version of $\mathcal{L}_{pen,p}$ with NME set to 0 in the update rule, we have:

$$\mathcal{L}_{GNPen,2}(\boldsymbol{\theta}) = \frac{1}{2}\rho g(\boldsymbol{\theta}, \mathbf{y}, \mathbf{p}(\mathbf{z})) \tag{52}$$

where we differentiate with respect to the first coordinate only (easily implemented in any AD framework). For arbitrary gradient penalty, recall that:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{pen,p}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}[\mathcal{L}_{pen,2}(\boldsymbol{\theta})^{p/2}] = \frac{p}{2}\mathcal{L}_{pen,2}(\boldsymbol{\theta})^{p/2-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{pen,2}(\boldsymbol{\theta}) \tag{53}$$

Therefore, we can implement $\mathcal{L}_{GNPen,p}(\boldsymbol{\theta})$ as

$$\mathcal{L}_{GNpen,p}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \frac{p}{4}\rho[g(\tilde{\boldsymbol{\theta}}, \mathbf{y}, \mathbf{p}(\mathbf{z}))]^{p/2-1}g(\boldsymbol{\theta}, \mathbf{y}, \mathbf{p}(\mathbf{z})) \tag{54}$$

where the derivative is taken with respect to the first coordiante, and $\tilde{\boldsymbol{\theta}}$ is set to the value $\boldsymbol{\theta}$ during evaluation. Alternatively, we can apply the `stop_gradient` operation (from the autodifferentiation framework) to the copy of g raised to the $p/2 - 1$ power, and maintain a function with a single input $\boldsymbol{\theta}$ only.

We note that this method works for any loss such that $\mathbf{H}_{\mathbf{z}}$ is independent of the labels.

An alternative approach is to use `vjp` and `stop_gradient` operations. Consider the vector valued function $\tilde{\mathbf{g}}$:

$$\tilde{\mathbf{g}}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \mathbf{J}^{\mathrm{T}}(\boldsymbol{\theta}_2)\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}(\boldsymbol{\theta}_1), \mathbf{y}) \tag{55}$$

This can be implemented by taking a `vjp` of the model at $\boldsymbol{\theta}_2$ with cotangent vector $\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}(\boldsymbol{\theta}_1), \mathbf{y})$.

We then have:

$$\mathcal{L}_{GNpen,p}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \rho||\tilde{\mathbf{g}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})||^p \tag{56}$$

Once again, derivatives are taken with respect to the first $\boldsymbol{\theta}$, and set $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}$ on evaluation (or alternatively, use the `stop_gradient`).

A third, orthogonal approach is to modify the update rule directly. That is, the second term in Equation 13 can be added explicitly to the update rule, but with the Hessian-vector product replaced with a GN-vector product. This is analogous to how for simple optimizers, ℓ^2 regularization of parameters can be added either to the loss or to the update rule as weight decay.

C.2 Implementing activation NME modifications

The experiments in Section 4.4 and 4.5 require us to define custom second derivatives for activation functions. The code below shows how to implement such a method in JAX [40], using the `custom_vjp` capability.

```python
from jax import grad, custom_vjp
import jax.numpy as jnp

def get_custom_act_fn(base_fn, second_deriv_fn):
  """Return version of base_fn replacing second derivative with second_deriv_fn."""

  dbase_fn_dx = jnp.vectorize(grad(base_fn)) # derivative function

  # Define custom derivative function, whose own derivative is second_deriv_fn.
  @custom_vjp
  def new_deriv(x):
    return dbase_fn_dx(x)
  def new_deriv_fwd(x):
    # Returns primal output and residuals to be used in backward pass.
    return dbase_fn_dx(x), second_deriv_fn(x)
  def new_deriv_bwd(res, g):
    return (res * g, )
  new_deriv.defvjp(new_deriv_fwd, new_deriv_bwd)

  # Define new version of base_fn, with custom derivatives

  @custom_vjp
  def mod_base_fn(x):
    return base_fn(x)
  def mod_base_fwd(x):
    return base_fn(x), new_deriv(x)
  def mod_base_bwd(res, g):
    return (res * g, )
  mod_base_fn.defvjp(mod_base_fwd, mod_base_bwd)

  return mod_base_fn
```

The GELU ablation was obtained by calling `get_custom_act_fn(gelu, lambda x: 0.)`, and the ReLU custom derivative example called `get_custom_act_fn(relu, gauss_pdf_b)` where

```python
import jax.numpy as jnp

beta = 100. # beta value, can be adjusted
def gauss_pdf_b(x):
  return (beta/jnp.sqrt(2*jnp.pi))*jnp.exp(-(beta*x)**2))
```

We used the setting $\beta = 100$ in our experiments. This framework can be used generally to define custom second derivatives for activations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims supported by theoretical and experimental work.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Claims are qualified.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Everything is included either in the main text or the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We use standard setups that are widely available and show the values for the additional hyper-parameters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Publicly available datasets are used, but the code is not open source.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Specified in experimental setup.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Errors reported for all results over different random seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See experimental setup.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and abide by the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This is an empirical study.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No model is released.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We used widely available public datasets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.