Estimating Implicit Regularization in Deep Learning

Joseph H. Rudoler **Kevin Tan** **Giles Hooker** **Konrad P. Kording**

University of Pennsylvania

Abstract

Deep learning systems are known to exhibit *implicit regularization* (alt. *implicit bias*), favoring simple solutions instead of merely minimizing the loss function. In some cases, we can analytically derive the implicit regularization – connecting it to an equivalent penalty that augments the learning objective. However, modern deep learning systems are complex, carrying modifications to the training procedure and architecture (e.g. early stopping, minibatching, dropout) whose effects are not always directly interpretable. Although estimating the resulting implicit regularization could aid theorists in algorithm design and practitioners in interpreting their hyperparameter choices, this problem has received little direct attention. It is also tractable: regularization makes weight updates deviate from loss gradients, promising a signal for identifying implicit bias. Here we provide gradient matching methods that can be used to empirically estimate the implicit regularization. Our method works on networks with known regularization, recovering popular explicit penalties like ℓ_1 and ℓ_2. It also replicates known implicit effects, like the quadratic weight penalty induced by early stopping in gradient descent, demonstrating that it can be used to test theories of implicit regularization. Crucially, because our method is empirical, it can handle implicit regularization in arbitrary networks. We demonstrate this use by characterizing implicit ℓ_2 effects of dropout in deep networks. Our work shows that practitioners can use gradient matching to understand regularization in networks with implicit biases that are too complicated to derive analytically.

1 Introduction

Traditional statistical learning theory teaches us that adding more parameters to our predictive models increases the chances of overfitting to training data at the expense of their ability to generalize to novel examples. One of the great challenges of modern machine learning is to understand how massively over-parametrized neural networks which can effectively memorize their training sets still manage to generalize [31, 46]. Recent progress has shown that the solution to the overfitting problem is for learning systems to favor simplicity: successful algorithms should prioritize parsimonious or highly-compressible solutions [22, 28, 32, 37, 41, 45]. Neural networks therefore require constraints on their solution space – whether explicit or implicit – in order to achieve the impressive practical performance we observe.

We often *explicitly* regularize or constrain models, adding *inductive biases* which can improve generalization. Examples of explicit inductive biases include Bayesian priors [23, 29, 42], weight decay [14], and the translation invariance introduced in convolutional neural networks [20]. Explicit regularization is one way of enforcing simplicity.

We also generally *implicitly* regularize. A growing literature studies the implicit regularization effects of gradient-based learning [3, 4, 11, 27, 30, 35] and various algorithmic or architectural choices like early-stopping [2, 33, 44] or dropout [9, 15, 24, 25, 39, 40]. Most theory casts implicit regularization in the form of a bias towards simpler functions (often via norm or rank minimization).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

It is believed that implicit regularization is essential to the success of deep learning in the over-parameterized regime [1, 6, 8, 41, 43].

Rigorous analysis of implicit regularization is generally limited to simple settings. Much existing work focuses on tasks (deep matrix factorization [4, 13]), models (shallow and/or linear networks [9, 11, 24, 25]), and algorithms (gradient flow or full-batch gradient descent [11, 26, 38]) that differ significantly from those employed by deep learning practitioners. The claim of these theoretical studies is often that the conclusions drawn in simplified settings might extend more generally, raising the need for empirical validation in larger networks.

In this paper we develop tools to empirically estimate implicit bias, with implications both for the theoretical study of learning algorithms and for training state of the art deep learning systems. Estimating implicit bias should allow theoreticians to empirically validate their analytical results and extend the study of implicit bias to complex networks that evade mathematical analysis, thereby providing a stronger bridge between theory and practice. For empirical researchers training deep neural networks, even heuristic estimates of implicit regularization can be a useful diagnostic. While there is some literature on quantifying inductive bias, broadly construed, to our knowledge we are the first to directly estimate parametrized models of implicit bias as explicit regularizers. We summarize and contrast other approaches with our own in Appendix E. Explicitly modeling the bias permits directly testing theoretical models, and generally produces structured and interpretable results.

Our task is to find a mathematically expressible regularizer $\mathcal{R}$, such that explicitly optimizing the empirical loss $\mathcal{L}$ augmented by $\mathcal{R}$ is functionally the same as an implicitly biased procedure optimizing $\mathcal{L}$ alone. Given an a priori parametric family of regularizers, $\mathcal{R}(\cdot, \Lambda)$, we seek to estimate the values of the regularization parameters Λ. As a running example, ℓ_2 regularization has been connected to common techniques like early stopping and dropout. This regularizer can be written as $\mathcal{R}(\theta, \lambda) = \lambda\|\theta\|_2^2$ where θ are the weights of a predictive model. Estimating λ determines the strength of the technique's effective ℓ_2 regularization.

Information about the implicit regularization is clearly embedded in the dynamics of learning. Specifically, if the gradient of the training loss does not converge to zero, this may reflect additional constraints imposed by the optimization procedure which we seek to recover. Alternatively, the same information can be gleaned at each timestep from discrepancies between the gradient of the loss and the update direction. These deviations from simply going down the loss gradient are indicative of the underlying regularization.

We perform estimation by considering the learned weights to be a minimizer of a hypothetical objective combining the empirical loss *and* the implicit regularizer. Alternatively, one can consider deviations of weight updates from the strict empirical loss gradient to be guided by the implicit regularizer. Each of these perspectives implies a system of equations involving computable gradients of the loss and the candidate regularizer, in which the only unknowns are the regularization parameters Λ. This framework is general, encompassing arbitrary features (weights, activations, data, etc.) and flexible parameterizations (any differentiable function class).

Our contributions are as follows:

- **How to estimate implicit bias.** A novel empirical framework for estimating parametrized models of implicit regularization. We do so by matching the gradients of the loss to the gradients of a proposed regularizer, based on a decomposition of the empirical loss.
- **Characterizing successful estimation.** We distinguish between two modes of success: 1) recovering ground truth regularization and 2) recovering a regularizer that mimics the effect of the implicit regularization. We show that indeed we achieve (1) in simple settings where results are predicted by theory. We can evaluate (2) by estimating a regularizer and using it to retrain the model, then comparing the retrained model to the original. We then provide a theoretical characterization of the conditions needed to reproduce the original model.
- **From theory to practice.** Our proposed method bridges the gap between theory and practice. It establishes a method for theoreticians to empirically validate their theoretical results, demonstrated by our recovery of the implicit regularization of gradient descent in OLS.
- **An interpretability tool for training complex networks.** Our method also extends the study of implicit bias to complex networks that evade mathematical analysis. We demon-

strate this by estimating the implicit regularization induced by dropout in deep nonlinear networks, showing how regularization varies with dropout rate and architecture.

2 Estimating implicit regularization

2.1 Framework for estimation

Setup. We study standard supervised learning, in which we receive input data $X \in \mathcal{X}$, and target $Y \in \mathcal{Y}$. We consider a hypothesis space $\mathcal{H}$ such that every $h \in \mathcal{H}$ is a function $h : \mathcal{X} \mapsto \mathcal{Y}$. The hypotheses have some parametrization $h = h_\theta$ via the weights $\theta \in \Theta$ of the model we fit. As such the hypothesis space is defined by the chosen model's architecture. We train on a finite dataset $D = \{(X_i, Y_i), i = 1, \ldots, n\} \subseteq \mathcal{X} \times \mathcal{Y}$. We implement some learning algorithm $\mathcal{A} : \bigcup_{h=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \mapsto \mathcal{H}$ that returns a hypothesis $\hat{h}(D) \in \mathcal{H}$ with corresponding $\hat{\theta}(D) \in \Theta$.

A hypothetical minimum. The standard empirical risk minimization (ERM) framework considers algorithms for minimizing the empirical risk (also called the loss or objective function) on the training data. However, while the ERM solution finds the model weights that satisfy $\arg\min_{\theta \in \Theta} \mathcal{L}(\theta, D)$, we often obtain some $\mathcal{A}(D) = \hat{\theta}(D)$ that deviates from the strict loss minimum. For example, we often stop training before the loss has fully converged.

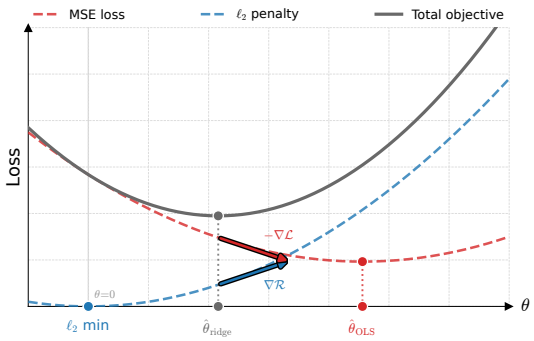


Figure 1: **Deviations from loss minima contain information about regularization.** Regularization modifies the objective function, leading to different solutions than strictly minimizing empirical loss. This gap must be explained by a regularizer ($\nabla\mathcal{R}$) counteracting any nonzero loss gradient ($-\nabla\mathcal{L}$) – this is the key logic of the paper.

Our key assumption is to view the residual loss as a quantity associated with an additional objective, namely the regularization objective. We understand implicit effects by assuming the learned $\hat{\theta}$ is an exact solution to a *regularized* empirical risk minimization (RERM) problem:

$$\hat{\theta}(D) = \arg\min_{\theta \in \Theta} \underbrace{\mathcal{L}(\theta, D)}_{\text{empirical loss}} + \underbrace{\mathcal{R}(\theta, D)}_{\text{regularization term}} ,$$

for some regularizer $\mathcal{R}$. This is a *hypothetical* condition, in that the network is explicitly trained to minimize the empirical loss. We simply hypothesize that due to some architectural or algorithmic choice, at a chosen point during training the network's weights can also be characterized as a solution to the above problem.

Computing and matching gradients. If the loss and regularizer are differentiable, then $\hat{\theta}(D)$ must satisfy

$$\nabla_\theta \mathcal{L}(\theta, D)|_{\hat{\theta}(D)} + \nabla_\theta \mathcal{R}(\theta, D)|_{\hat{\theta}(D)} = 0. \tag{1}$$

Equivalently, we can see this as a *gradient-matching condition* by rearranging:

$$-\nabla_\theta \mathcal{L}(\theta, D)|_{\hat{\theta}(D)} = \nabla_\theta \mathcal{R}(\theta, D)|_{\hat{\theta}(D)}. \tag{2}$$

The above states that the gradient of the regularizer must be approximately equal to the negative gradient of the loss. Intuitively, if we are at a minimum of an RERM objective, the net gradient in parameter space must be zero. If it were otherwise, we would move in the direction of that net gradient towards a better solution. So, if we assume the observed parameters $\hat{\theta}$ are a minimizer, then the regularizer gradients *must* negate the loss gradient!

Note that our stationarity condition implies a system of p equations where p is the dimensionality of θ. This implies that the approach in the following section can only fit regularizers with up to p parameters – the regularizer cannot have more parameters than the predictive model. We discuss this and other challenges, along with suggested remedies, in Section 2.2. Appendix D discusses computing the gradients in more detail: the loss gradient decomposes into a vector-Jacobian product that can be computed efficiently without materializing the full Jacobian.

Learning the regularizer. We can learn the regularizer by parametrizing as $\mathcal{R}(\theta, D, \Lambda)$, and then solving for Λ in Equation 2. In particular, we fit a **regularizer model** $\mathcal{R}(\theta, D, \Lambda)$ where the predictive model's weights θ are fixed and the hyperparameters Λ are learnable. Since our goal is that the above expression is small, we minimize the mean-squared-error of this gradient matching task:

$$\hat{\Lambda} = \arg\min_{\Lambda} \{ \frac{1}{p} \| \nabla_\theta \mathcal{R}(\theta, D, \Lambda)|_{\hat{\theta}(D)} + \nabla_\theta \mathcal{L}(\theta, D)|_{\hat{\theta}(D)} \|_2^2 \}. \tag{3}$$

This problem is general and allows us to use modern optimization tools like backpropagation to estimate arbitrary parameterization of Λ, but in many cases it reduces to a simple problem like linear regression and may permit closed-form solutions or more efficient implementations.

Example: weight penalty in linear models. The simplest case we can consider is that of recovering a weight norm penalty in linear models, i.e. ℓ_1 (lasso) or ℓ_2 (ridge) regression. This case eases our analysis because we have closed-form solutions for the optimal weights.

Given input data $X \in \mathbb{R}^{N \times p}$ and target vector $y \in \mathbb{R}^N$, we want to learn $\theta \in \mathbb{R}^p$ such that we generate predictions $\hat{y}(\theta) = X\theta$ which closely approximate the target vector.

We fit ridge regression with some λ:

$$\nabla_\theta \left\{ \frac{1}{2} \| y - \hat{y}(\theta) \|^2 + \frac{\lambda}{2} \| \theta \|^2 \right\} = 0,$$

resulting in: $\hat{\theta}_\lambda = (X^\top X + \lambda I)^{-1} X^\top y$.

Now we consider the task of recovering λ. Formally, given data X, y, and weights $\hat{\theta}_\lambda$ we want to find the solution $\hat{\lambda}$ that satisfies: $-X^\top (y - X\hat{\theta}_\lambda) + \hat{\lambda}\hat{\theta}_\lambda = 0_p$.

This gives us a p-dimensional system of linear equations for $\hat{\lambda}$, which holds exactly when $\hat{\lambda} = \lambda$.

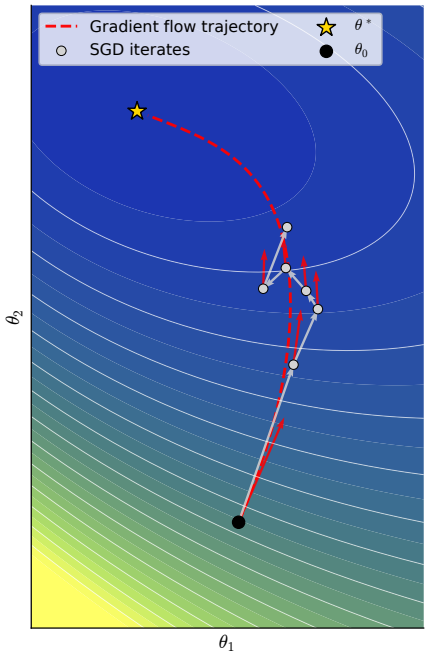


Figure 2: **Gradient deviations along the trajectory.** In a 2-parameter toy model, SGD mini-batching causes weight updates along the trajectory to deviate from the full-batch gradient. We can model the implicit regularization underlying these deviations.

Deviations along the trajectory. An alternative perspective examines implicit regularization at each optimizer step (as in [5, 34]). At a given point during the learning process, weights do not strictly follow the loss gradient; instead they follow a modified trajectory induced by the many components of the optimization process.

$$\Delta\theta = \theta_{t+1} - \theta_t = -\eta \nabla_\theta (\mathcal{L} + \mathcal{R}). \tag{4}$$

Rearranging this expression implies that regressing $-\eta^{-1}\Delta\theta - \nabla_\theta \mathcal{L}$ on $\nabla_\theta \mathcal{R}$, where each point in the trajectory yields one datapoint, offers an alternative method for learning a regularizer. We further describe this procedure and provide theoretical guarantees for the above in the form of a martingale generalization bound within Appendix C. See Section 3 for an application to estimating the implicit gradient regularization proposed by [5].

Figure 2 illustrates the trajectory deviations (in a 2-parameter model) induced by SGD mini-batching. Note that this framing contains training endpoints as a special case ($\Delta\theta_t \approx 0$), so forcing $\Delta\theta = 0$ by assumption recovers the "hypothetical minimum" perspective we described above.

Evaluating the estimated regularizer Once we fit $\hat{\Lambda}$, how can we decide if it is "correct" or even a good approximation? We posit that there are two main modes of success:

1. **Recovery of hypothesized Λ.** In settings where we know the ground truth regularization or have strong theoretical predictions about its exact parametrization, we should recover estimates $\hat{\Lambda}$ from (3) that exactly match theory.
2. **Recovery of original model weights $\hat{\theta}$: validation via explicit regularization.** Say we are interested in the implicit bias of an algorithmic or architectural feature that can be added to, or removed from, the training procedure. If we do not have any "ground truth" Λ

to work with, another way of validating our empirical estimates $\hat{\Lambda}$ is to remove (or ablate) the training feature of interest and retrain the original predictive model f with an explicitly regularized objective, namely $\mathcal{L}(\theta) + \mathcal{R}(\theta, \hat{\Lambda})$. If $\mathcal{R}(\theta, \Lambda)$ accurately captures the effective regularization, then this should recover the same solution for the optimal weights $\hat{\theta}$.

That is, the weights learned with implicit regularization are equivalent to the weights learned when early stopping is replaced with the estimated regularizer. While this doesn't necessarily mean that the estimated regularizer was used during training, it does mean that the two approaches are functionally equivalent on the training set. The second approach has been employed in prior literature, e.g. [40]. Appendix A provides a detailed discussion of theoretical guarantees for recovering the original model weights. Intuitively, if the loss converges to a relatively flat minimum, local convexity guarantees that the solution is unique and we will recover the model.

For assessing the fit itself, we can use a diagnostic analogous to regression R^2, measuring the proportion of the squared loss-gradient magnitude explained by the estimated regularizer:

$$R^2_{\text{grad}} = 1 - \frac{\|\nabla_\theta \mathcal{L}(\hat{\theta}, D) + \nabla_\theta \mathcal{R}(\hat{\theta}, D, \hat{\Lambda})\|_2^2}{\|\nabla_\theta \mathcal{L}(\hat{\theta}, D)\|_2^2}. \tag{5}$$

This is an uncentered R^2: the baseline is a zero gradient rather than the mean across gradient components. It equals one for exact gradient matching and zero when the fitted regularizer offers no improvement over a zero regularizer. Poor fit can reflect an inadequate candidate family, noisy gradients, or optimization error, while high fit alone does not establish recovery of the original model; see Appendix G.2 for practical guidance.

2.2 Regularizer identification

Which regularizers can be modeled? In order to characterize the implicit bias of some additive module $\mathcal{M}$ like an architecture (dropout layers) or algorithm (gradient descent/early stopping), you need some notion of training *without* $\mathcal{M}$ – we might say that it is *ablatable* or amenable to ablation study, so that it may be replaced by an equivalent explicitly regularized problem. Appendix J discusses attribution to individual components and their interactions, and extension to larger systems.

Collinearity of gradients. A natural application of our method is to simultaneously fit a number of candidate regularizers with scale coefficients – essentially turning the problem into linear regression over a set of regularizer gradients. That is:

$$-\nabla_\theta \mathcal{L} = \vec{\lambda}^\top \nabla_\theta \vec{\mathcal{R}} = \lambda_1 \nabla_\theta \mathcal{R}_1 + \ldots + \lambda_r \nabla_\theta \mathcal{R}_r.$$

The problem of accurately estimating the coefficient vector $\vec{\lambda}$ is more challenging when the gradients (i.e. the regression features) exhibit collinearity. In particular, if two regularizer gradients are perfectly collinear then the covariance matrix $(\nabla_\theta \mathcal{R}^T \nabla_\theta \mathcal{R})$ is singular and the OLS estimator is undefined. When the regularizer gradients are nearly collinear, the estimator is well-defined and unbiased, but has inflated variance, making estimation unstable. We elaborate and provide theoretical guarantees for both this parametric and the associated nonparametric case in Appendix B. There we show that estimation error scales in $O(\sqrt{r/p_{\text{eff}}})$, where in the linear parametric case r is the number of regularizers and p_{eff} is the effective parameter count (this is analogous to effective sample size, since parameters are samples in our regression of $\nabla_\theta \mathcal{L}$ onto $\nabla_\theta \mathcal{R}$).

Improving estimation with more diverse sampling. The approach described thus far only allows estimation of regularizers with dimension at most that of the predictive model, $p = \dim(\theta)$. A natural goal for improving estimation and enabling more complex parameterizations is to augment with additional estimating equations. Intuitively, adding more equations helps if they introduce additional regularizer gradients that do not overlap with existing ones (analogous to adding more i.i.d. data samples in regression). As mentioned above, estimation error scales in $O(\sqrt{r/p_{\text{eff}}})$ – adding more equations increases p_{eff}, and correlation among gradients decreases p_{eff}.

In an ideal setting, we get more equations by retraining the same model on a new dataset. The new training endpoint will have new weights $\hat{\theta}$ and therefore a new set of features $\nabla_\theta \mathcal{R}|_{\hat{\theta}(D)}$ and targets $-\nabla_\theta \mathcal{L}(\theta, D)|_{\hat{\theta}(D)}$ to concatenate to the regression. In synthetic data settings, we can sample new datasets that are independent, maximizing the amount of additional information obtained from each

new sample. We will later demonstrate this works for estimating the regularization due to early stopping in Figure 4B. In realistic data settings, many datasets (especially those from the same domain, like images) are correlated and therefore will lead to more similar $\hat{\theta}$ – but even sampling variability in $\hat{\theta}$ will improve our coverage.

Another natural way to obtain additional estimating equations, when constrained to a single training dataset, is through bootstrap resampling: each bootstrap replicate yields a new fitted parameter vector and thus a new approximate stationarity condition, expanding the gradient equations available for estimating Λ. Moreover, one need not resample uniformly – it is also possible to weight data samples in a way that stabilizes estimation, like upweighting samples for which the gradients are less collinear. See Appendix F for more detail and the bootstrapping algorithm for this case, as well as an example of bootstrapping applied to early stopping in OLS, as studied in Section 3.

Estimating over trajectories. When training models, we can compute the loss and regularizer gradients at any point during training. While Equation 1 will hold most closely as the model training converges, there is nothing to stop us from running our procedure earlier on in training. This opens two possible targets for estimation:

1. **Constant** Λ. Assuming that the regularizer function does not change over time, the trajectory can be used to augment the estimation of Equation 4 by contributing more gradient-matching terms – much like bootstrap replicates. See Appendix C for a detailed analysis.
2. **Time-varying** $\Lambda^{(t)}$. It is also possible that the implicit regularization changes as a function of time (or optimization steps). Then, each point along the trajectory represents a separate estimation problem. This allows for tracking how the implicit regularization evolves over the course of training, which is much more general than simply studying a model at convergence. This framing is potentially more interesting to researchers studying training dynamics rather than generalization. One can employ either of Equations 2 or 4 at each timepoint, but the hypothetical stationarity assumed for 2 is probably a poor approximation early on in training. Section 3 studies a time-varying quadratic weight penalty induced by gradient descent for OLS.

Appendix G.3 examines sensitivity to training duration in the dropout experiment.

3 Empirical Studies

The following experiments test recovery of known explicit regularization (elastic net) and theorized implicit regularization (from early stopping and discrete GD steps), and assess the adequacy of a scalar ridge candidate for dropout-trained networks.

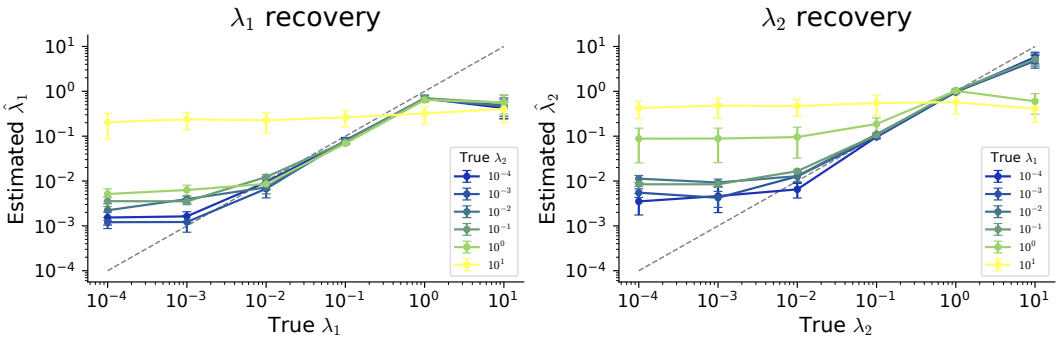


Figure 3: **Recovering explicit regularization.** Elastic-net recovery at $\beta = 10^{-3}$ on a 6×6 grid of true (λ_1, λ_2) with 10 independent dataset draws per cell. Each panel plots the estimated $\hat{\lambda}_i$ against the true λ_i on log–log axes; lines show the mean across seeds and error bars give ± 1 standard error. **Left:** $\hat{\lambda}_1$ vs. true λ_1, one line per value of true λ_2. **Right:** $\hat{\lambda}_2$ vs. true λ_2, one line per value of true λ_1. The dashed diagonal marks perfect recovery.

Explicit elastic net regularization. In order to sanity-check our approach, we first tasked our framework with recovering the parameters of a neural network explicitly trained with regularization.

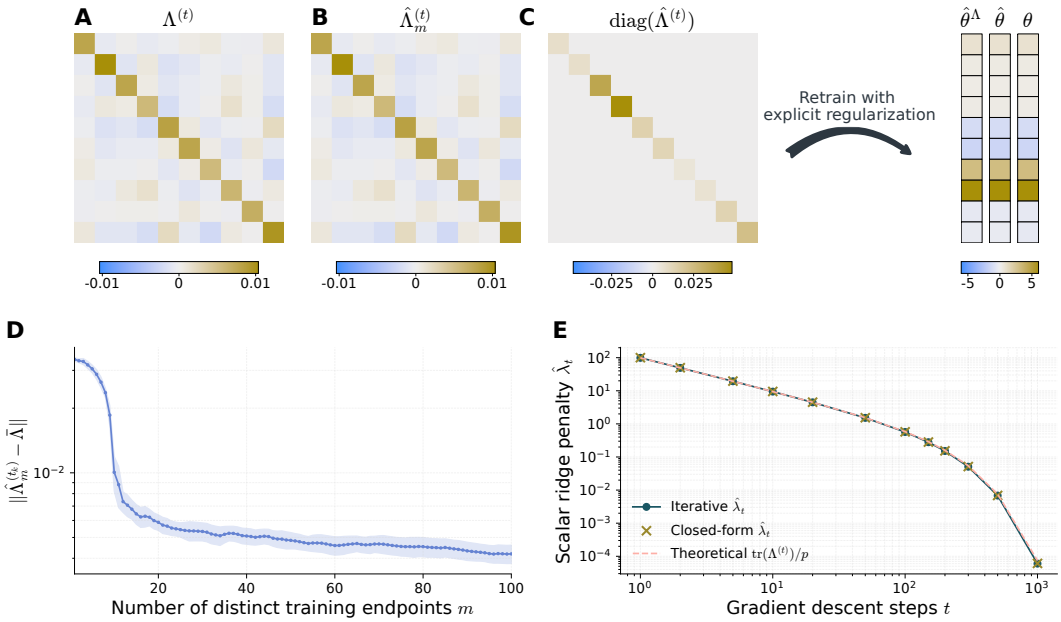


Figure 4: **Implicit regularization due to early stopping.** **(A)** The theoretical regularization matrix $\Lambda^{(t)}$ predicted by Ali et al. at the early-stopped iterate $t = 500$. **(B)** The full symmetric estimator $\hat{\Lambda}_m^{(t)}$ using $m = 10$ endpoints, each trained to the same fixed iterate $t = 500$. **(C)** The under-parametrized diagonal estimator $\mathrm{diag}(\hat{\Lambda}^{(t)})$ fit from a *single* endpoint. The regularized solution $\hat{\theta}^{\Lambda}$ under $\mathrm{diag}(\hat{\Lambda}^{(t)})$ matches the original early-stopped trained weights $\hat{\theta}$, and the ground-truth coefficients θ. **(D)** Convergence of the symmetric-matrix estimator under per-endpoint early stopping: $\|\hat{\Lambda}_m^{(t_k)} - \bar{\Lambda}\|$ as the number of endpoints m grows from 1 to 100, where $\bar{\Lambda}$ is the median of the endpoint-specific matrices $\Lambda^{(t_k)}$. Each endpoint is stopped individually when its training loss plateaus, so stop steps t_k vary. Band shows 95% CI. **(E)** The single-parameter heuristic estimate $\hat{\lambda}_t$ tracked over gradient descent steps t.

We trained deep ReLU networks with various combinations of ℓ_1 and ℓ_2 regularization (also known as elastic net).

Figure 3 shows the recovered penalties as a function of the true penalties. Note this is a difficult task because smoothing ℓ_1 near zero makes the two norms indistinguishable for very small weights ($\leq 10^{-3}$). Consequently, large penalties (which drive many weights towards zero) reduce identifiability. For very small penalties, the regularization has little effect on the optimization procedure and there is not much signal for estimation. For intermediate values of the penalty parameters, however, recovery is quite accurate.

Early stopping in OLS. It is nice to recover known explicit regularization, but the real goal is to estimate *implicit* regularization that is not in our objective function during training. To that end we study a simple example with well-understood theory: the effect of early stopping in full-batch gradient descent for ordinary least-squares linear regression [2, 33, 44]. We analyze synthetic data generated from a noisy linear model $y = X\theta + \varepsilon$, where $X \in \mathbb{R}^{1000\times 10}$ has entries $X_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, coefficients $\theta_i \overset{\text{iid}}{\sim} \mathcal{N}(0, 3^2)$, and noise $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$.

Ali et al. [2] predicts that the implicit effect of early stopping is a form of data-dependent quadratic regularization that decays over training. Consider GD with step size η on the OLS objective, starting at $\hat{\theta}^{(0)} = 0$, and assume $\eta < 1/\|X^\top X\|_{\text{op}}$. Let $\frac{1}{n}X^\top X = VSV^\top$ be the singular value decomposition. For each $t = 1, 2, 3, \ldots$, the iterate $\hat{\theta}^{(t)}$ from step t in gradient descent uniquely solves

$$\min_{\theta\in\mathbb{R}^p} \frac{1}{n}\|y - X\theta\|_2^2 + \theta^\top \Lambda^{(t)}\theta, \qquad \Lambda^{(t)} = VS\big((I - \eta S)^{-t} - I\big)^{-1}V^\top. \tag{6}$$

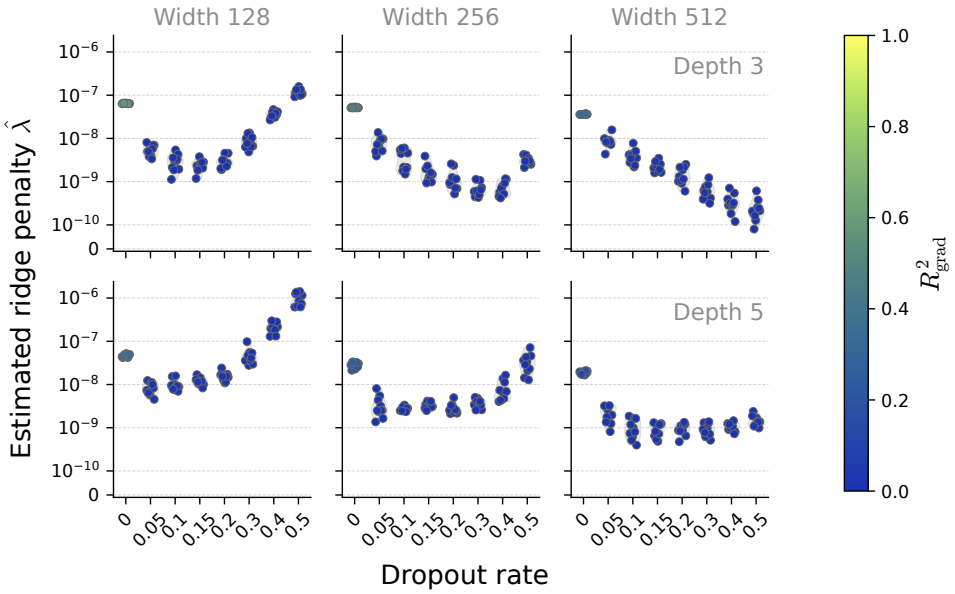


Figure 5: **Endpoint ridge projections for dropout-trained classifiers.** Exact nonnegative gradient-matching estimates $\hat{\lambda}$ for the candidate regularizer $\mathcal{R}(\theta, \lambda) = \lambda\|\theta\|_2^2$ on MNIST. Each point is one of ten seeds; columns vary width, rows vary depth, and gray violins summarize the seed distributions in displayed coordinates. Color indicates the fraction of squared clean-training loss-gradient norm explained, R^2_{grad} in (5), on a shared zero-to-one scale (lighter = better fit). The vertical axis is logarithmic above 10^{-10} and linear near zero. Zero-dropout controls are retained.

To recover this implicit regularization exactly, we need to learn the matrix $\Lambda^{(t)}$, which has $p(p+1)/2$ free parameters (it is symmetric and PSD). We only have p equations in the system of equations from a single training endpoint, posing a challenge for recovering the full matrix.

We present two approaches in Figure 4 and show how they lead to the two modes of success described in Section 2.1: recovery of hypothesized Λ (4B) and recovery of the original model weights $\hat{\theta}$ (4C). Figure 4A shows the quadratic weight penalty theoretically induced by early stopping according to Eq. 6. In 4B, we show that concatenating training endpoints from identical models trained on m distinct datasets (as discussed in Section 2.2) provides enough coverage to recover the full matrix exactly. Estimation error falls off steeply as a function of m (see 4D). In 4C, we use just a single training endpoint and force $\hat{\Lambda}^{(t)}$ to be diagonal so that it has only p parameters. We show that, though the underparametrized version diverges from the matrix predicted by theory, using it as an explicit penalty in the regression *without* early stopping yields exactly the same weights ($\hat{\theta}^{\Lambda}$) as the original unregularized model *with* early stopping ($\hat{\theta}$).

We next asked if an underparametrized model of implicit regularization could still be interpretable. After all, the true implicit regularization in a complex network might be nearly impossible to write in closed form. We may only be able to fit a heuristic model that captures some aspect or summary of its myriad effects. To test this, we fit a model with a scalar ℓ_2 regularization parameter, i.e. $\Lambda_t = \lambda_t I$. Our theory of early-stopping in OLS tells us that $\lim_{t\to\infty} \alpha^\top \Lambda_t \alpha \to 0$ because the matrix $\left((I - \eta S)^{-t} - I\right)^{-1}$ vanishes for large t (given that eigenvalues s_i are positive and $\eta s_i < 1$). We should be able to capture this behavior with our heuristic model; that is, if the theory is correct we should see $\hat{\lambda}_t \to 0$. Figure 4E shows that our heuristic estimate decays over time, with an extremely tight match to theory. We take this as an encouraging sign that we can use underparameterized models of implicit regularization to test our theory and intuitions.

Dropout. Dropout [36] randomly zeroes out hidden units during training and is broadly understood to act as an adaptive regularizer that penalizes over-reliance on specific pathways. Several theoretical analyses have shown that, for linear models, dropout is akin to adaptive ℓ_2 regularization whose strength depends on the dropout rate and the second moments of the activations [39, 40]. Here we trained a series of neural network classifiers on MNIST [21] with varying levels of dropout.

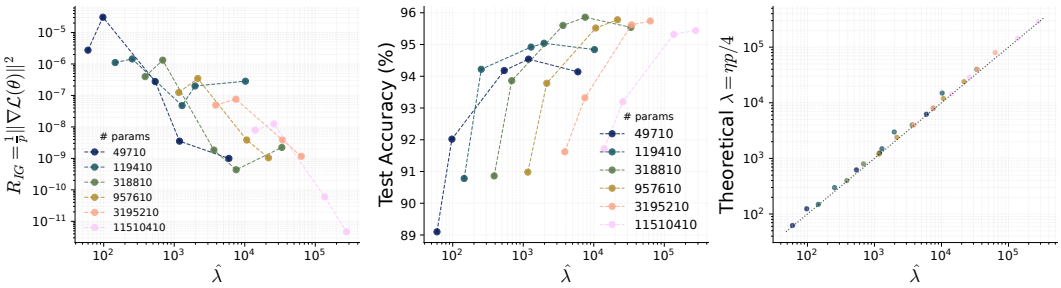


Figure 6: **Recovering implicit gradient regularization from discrete gradient steps.** Barrett and Dherin [5] derived $\lambda = \eta p/4$ for the gradient-penalty regularizer induced by discrete GD. **Left:** Value of R_{IG}, the averaged squared gradient, against the estimated scaling $\hat{\lambda}$. **Middle:** Test accuracy against $\hat{\lambda}$. **Right:** Direct recovery check comparing $\hat{\lambda}$ to the theoretical value $\eta p/4$.

Figure 5 shows the estimated ridge coefficient $\hat{\lambda}$ as a function of dropout rate, faceted by depth and width. Among positive dropout rates, narrower networks exhibit a pronounced upturn in the fitted coefficient at high dropout. Wider networks generally have smaller coefficients, with flatter or decreasing trends depending on depth. The increase in narrower networks is consistent with the intuition that stronger dropout can exert greater regularization pressure, although this relationship is not uniform across architectures.

However, the low gradient-fit R^2_{grad} values indicate that the estimated projection onto the proposed ridge regularizer explains only a small proportion of the squared residual loss-gradient magnitude. This suggests that scalar ridge provides an incomplete description of dropout's complex, data-dependent effects. Indeed, replacing dropout with each model's fitted ridge penalty and retraining from the same initialization under the same optimization and stopping procedure did not recover the original dropout model's test performance (Appendix H.4.1). Though estimated regularizers can still provide insight into the training dynamics, they may not lead to retrained models that match the original behavior. Here, the projection reveals interpretable variation in the ridge-aligned component across training configurations, while the goodness-of-fit and retraining results show the limitations of using that component to fully explain dropout.

Gradient step size. Barrett and Dherin [5] describe how the step size (η) of discrete gradient descent implicitly regularizes models by scaling a penalty on trajectories with large loss gradients. Specifically, they propose that gradient descent in $\mathbb{R}^p$ minimizing a loss $\mathcal{L}$ deviates from the exact continuous path of gradient flow. It is actually closer to the exact continuous path of a modified loss $\mathcal{L}(\theta) + \mathcal{R}(\theta, \lambda)$ where $\lambda \equiv \frac{\eta \cdot p}{4}$ and $\mathcal{R}(\theta, \lambda) \equiv \frac{\lambda}{p}\sum_{i=1}^{p}\left(\nabla_{\theta_i}\mathcal{L}(\theta)\right)^2$.

Figure 6 uses this setting as a known-bias recovery check. Rather than treating $\lambda = \eta p/4$ as given, we estimate λ from trajectory deviations and compare $\hat{\lambda}$ to the analytic value derived by Barrett and Dherin [5]. In this experiment, Eq. 4 is instantiated with the continuous gradient-flow path of L as the unregularized reference, rather than the Euler step $-\eta\nabla\mathcal{L}$; Appendix H.5 details how we compute this reference so that the residual captures the finite-step IGR correction. We also reproduce the qualitative Figure 2 relationships between stronger implicit gradient regularization, smaller averaged squared gradients, and higher test accuracy.

This application shows that our method can be used to estimate regularization in the *dynamics* of learning along the training trajectory, including early on in training. This supplements our earlier applications estimating regularization in models near convergence.

4 Conclusions

We have shown that it is possible to estimate the regularization implicitly embedded in a learning system. We accomplish this by assuming that a system jointly optimizes the empirical loss *and* a regularizer. This assumption implies a relationship among the empirical loss gradients, regularizer gradients, and observed weight updates that permits solving for unknown regularization parameters. We demonstrate that this approach can recover explicit regularization as well as theoretically predicted implicit regularization.

Our approach, as it ultimately is an instance of parametric statistical modeling, has familiar limitations. In particular, our capacity to fit models with many parameters is bound to our effective sample size. In this setup, sample size corresponds to the number of equations given by gradient matching (typically the number of parameters p in the original predictive model). In Section 2.2 we discussed ways to increase the sample size, but ultimately they may be limited by the data at hand. We also face familiar identifiability problems – regularization parameters associated with collinear gradients yield estimates with high variance. It is difficult to distinguish reliably between classes of regularizers with similar gradients. In Appendix B we give PAC bounds for both the parametric and nonparametric cases. Additionally, we frame implicit regularization as a phenomenon which can be replaced by an equivalent explicit regularizer, when estimated correctly – but in some cases it might not be feasible or sensible to replace the training technique being studied. However, many deep learning techniques can be entirely ablated, making them amenable to study.

This paradigm opens exciting avenues for future research into the implicit regularization effects of the many complex innovations and engineering tricks used in state of the art deep learning systems. Practitioners often develop training techniques in a mostly ad-hoc manner, greedily testing various training configurations and advocating for techniques that empirically improve generalization on available validation datasets. This clearly works in the long run, as the impressive evolution of deep learning proves. However, as model size and compute budgets scale, training runs become massively expensive. Estimating implicit regularization could provide a relatively cheap and practical way for practitioners to study the effects of various training configurations early on in training, by studying how the dynamics deviate from strict loss minimization. This is another axis along which to evaluate models, which would complement existing heuristics like validation loss, weight and gradient norms, or perplexity. We consider our work here to be just a starting point for empirically characterizing the implicit regularization in deep learning systems, which we believe to be essential to their capacity to generalize beyond their training data.

Acknowledgments and Disclosure of Funding

This research was supported in part by a Susquehanna PhD Fellowship awarded to Joseph H. Rudoler by Susquehanna International Group. The authors report no other direct funding or third-party support for this work and no other competing financial interests during the 36 months preceding submission.

References

[1] Yotam Alexander, Yonatan Slutzky, Yuval Ran-Milo, and Nadav Cohen. Do Neural Networks Need Gradient Descent to Generalize? A Theoretical Study, June 2025. URL http://arxiv.org/abs/2506.03931. arXiv:2506.03931 [cs].

[2] Alnur Ali, J Zico Kolter, and Ryan J Tibshirani. A continuous-time view of early stopping for least squares regression. In *The 22nd international conference on artificial intelligence and statistics*, pages 1370–1378. PMLR, 2019.

[3] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization, June 2018. URL http://arxiv.org/abs/1802.06509. arXiv:1802.06509 [cs].

[4] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit Regularization in Deep Matrix Factorization, October 2019. URL http://arxiv.org/abs/1905.13655. arXiv:1905.13655 [cs].

[5] David G. T. Barrett and Benoit Dherin. Implicit Gradient Regularization, July 2022. URL http://arxiv.org/abs/2009.11162. arXiv:2009.11162 [cs].

[6] Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *Acta Numerica*, 30:87–201, May 2021. ISSN 0962-4929, 1474-0508. doi: 10.1017/S0962492921000027. URL https://www.cambridge.org/core/product/identifier/S0962492921000027/type/journal_article.

[7] Akhilan Boopathy, William Yue, Jaedong Hwang, Abhiram Iyer, and Ila Fiete. Towards Exact Computation of Inductive Bias, June 2024. URL http://arxiv.org/abs/2406.15941. arXiv:2406.15941 [cs].

[8] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 12(1):2914, May 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-23103-1. URL https://www.nature.com/articles/s41467-021-23103-1.

[9] Jacopo Cavazza, Pietro Morerio, Benjamin Haeffele, Connor Lane, Vittorio Murino, and Rene Vidal. Dropout as a Low-Rank Regularizer for Matrix Factorization. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 435–444. PMLR, March 2018. URL https://proceedings.mlr.press/v84/cavazza18a.html.

[10] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys, April 2017. URL http://arxiv.org/abs/1611.01838. arXiv:1611.01838 [cs].

[11] Lénaïc Chizat and Francis Bach. Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. In *Proceedings of Thirty Third Conference on Learning Theory*, pages 1305–1338. PMLR, July 2020. URL https://proceedings.mlr.press/v125/chizat20a.html.

[12] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-Aware Minimization for Efficiently Improving Generalization, April 2021. URL http://arxiv.org/abs/2010.01412. arXiv:2010.01412 [cs].

[13] Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit Regularization in Matrix Factorization, May 2017. URL http://arxiv.org/abs/1705.09280. arXiv:1705.09280 [stat].

[14] Stephen Hanson and Lorien Pratt. Comparing Biases for Minimal Network Construction with Back-Propagation. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL https://proceedings.neurips.cc/paper/1988/hash/1c9ac0159c94d8d0cbedc973445af2da-Abstract.html.

[15] David P. Helmbold and Philip M. Long. On the inductive bias of dropout. *J. Mach. Learn. Res.*, 16(1):3403–3454, January 2015. ISSN 1532-4435.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Flat Minima. *Neural Computation*, 9(1):1–42, January 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.1.1. URL https://doi.org/10.1162/neco.1997.9.1.1.

[17] Alexander Immer, Lucas Torroba Hennigen, Vincent Fortuin, and Ryan Cotterell. Probing as Quantifying Inductive Bias. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1839–1851, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.129. URL https://aclanthology.org/2022.acl-long.129/.

[18] Simran Kaur, Jeremy Cohen, and Zachary C. Lipton. On the Maximum Hessian Eigenvalue and Generalization, June 2022. URL https://arxiv.org/abs/2206.10654v3.

[19] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, February 2017. URL http://arxiv.org/abs/1609.04836. arXiv:1609.04836 [cs].

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791. URL https://ieeexplore.ieee.org/document/726791.

[21] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[22] Yi Ma, Doris Tsao, and Heung-Yeung Shum. On the Principles of Parsimony and Self-Consistency for the Emergence of Intelligence, July 2022. URL http://arxiv.org/abs/2207.04630. arXiv:2207.04630 [cs, math].

[23] D. J.C. MacKay. Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469, August 1995. ISSN 0954-898X. doi: 10.1088/0954-898X/6/3/011. URL https://doi.org/10.1088/0954-898X/6/3/011.

[24] Poorya Mianjy and Raman Arora. On Dropout and Nuclear Norm Regularization. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4575–4584. PMLR, May 2019. URL https://proceedings.mlr.press/v97/mianjy19a.html.

[25] Poorya Mianjy, Raman Arora, and Rene Vidal. On the Implicit Bias of Dropout. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3540–3548. PMLR, July 2018. URL https://proceedings.mlr.press/v80/mianjy18b.html.

[26] Hancheng Min, Salma Tarmoun, Rene Vidal, and Enrique Mallada. Convergence and Implicit Bias of Gradient Flow on Overparametrized Linear Networks, May 2022. URL http://arxiv.org/abs/2105.06351. arXiv:2105.06351 [cs].

[27] Mor Shpigel Nacson, Kavya Ravichandran, Nathan Srebro, and Daniel Soudry. Implicit Bias of the Step Size in Linear Diagonal Neural Networks. In *Proceedings of the 39th International Conference on Machine Learning*, pages 16270–16295. PMLR, June 2022. URL https://proceedings.mlr.press/v162/nacson22a.html.

[28] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. SGD on Neural Networks Learns Functions of Increasing Complexity, May 2019. URL http://arxiv.org/abs/1905.11604. arXiv:1905.11604 [cs].

[29] Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, New York, NY, 1996. ISBN 978-0-387-94724-2 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0. URL http://link.springer.com/10.1007/978-1-4612-0745-0.

[30] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning, April 2015. URL http://arxiv.org/abs/1412.6614. arXiv:1412.6614 [cs].

[31] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, January 2022. URL http://arxiv.org/abs/2201.02177. arXiv:2201.02177 [cs].

[32] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5301–5310. PMLR, May 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

[33] Reginaldo J. Santos. Equivalence of regularization and truncated iteration for general ill-posed problems. *Linear Algebra and its Applications*, 236:25–33, March 1996. ISSN 0024-3795. doi: 10.1016/0024-3795(94)00114-6. URL https://www.sciencedirect.com/science/article/pii/0024379594001146.

[34] Samuel L. Smith, Benoit Dherin, David G. T. Barrett, and Soham De. On the Origin of Implicit Regularization in Stochastic Gradient Descent, January 2021. URL http://arxiv.org/abs/2101.12176. arXiv:2101.12176 [cs].

[35] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The Implicit Bias of Gradient Descent on Separable Data, November 2018. URL http://arxiv.org/abs/1710.10345. arXiv:1710.10345 [stat].

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. ISSN 1533-7928. URL http://jmlr.org/papers/v15/srivastava14a.html.

[37] Guillermo Valle-Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions, April 2019. URL http://arxiv.org/abs/1805.08522. arXiv:1805.08522 [stat].

[38] Gal Vardi, Ohad Shamir, and Nathan Srebro. On Margin Maximization in Linear and ReLU Networks, October 2022. URL http://arxiv.org/abs/2110.02732. arXiv:2110.02732 [cs].

[39] Stefan Wager, Sida Wang, and Percy Liang. Dropout Training as Adaptive Regularization, November 2013. URL http://arxiv.org/abs/1307.1493. arXiv:1307.1493 [stat].

[40] Colin Wei, Sham Kakade, and Tengyu Ma. The Implicit and Explicit Regularization Effects of Dropout. In *Proceedings of the 37th International Conference on Machine Learning*, pages 10181–10192. PMLR, November 2020. URL https://proceedings.mlr.press/v119/wei20d.html.

[41] Andrew Gordon Wilson. Deep Learning is Not So Mysterious or Different, March 2025. URL http://arxiv.org/abs/2503.02113. arXiv:2503.02113 [cs].

[42] Andrew Gordon Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization, March 2022. URL http://arxiv.org/abs/2002.08791. arXiv:2002.08791 [cs].

[43] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and Rich Regimes in Overparametrized Models. In *Proceedings of Thirty Third Conference on Learning Theory*, pages 3635–3673. PMLR, July 2020. URL https://proceedings.mlr.press/v125/woodworth20a.html.

[44] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On Early Stopping in Gradient Descent Learning. *Constructive Approximation*, 26(2):289–315, August 2007. ISSN 1432-0940. doi: 10.1007/s00365-006-0663-2. URL https://doi.org/10.1007/s00365-006-0663-2.

[45] Can Yaras, Peng Wang, Wei Hu, Zhihui Zhu, Laura Balzano, and Qing Qu. The Law of Parsimony in Gradient Descent for Learning Deep Linear Networks, June 2023. URL http://arxiv.org/abs/2306.01154. arXiv:2306.01154 [cs].

[46] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, February 2021. ISSN 0001-0782. doi: 10.1145/3446776. URL https://doi.org/10.1145/3446776.

[47] Xingxuan Zhang, Renzhe Xu, Han Yu, Hao Zou, and Peng Cui. Gradient Norm Aware Minimization Seeks First-Order Flatness and Improves Generalization. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20247–20257, Vancouver, BC, Canada, June 2023. IEEE. ISBN 979-8-3503-0129-8. doi: 10.1109/CVPR52729.2023.01939. URL https://ieeexplore.ieee.org/document/10205166/.

A Uniqueness and recovery with an estimated explicit regularizer

In this appendix we characterize how retraining a model with an explicit regularizer $\mathcal{R}$ with estimated parameters $\hat{\Lambda}$ can often recover the original predictive model parameters $\hat{\theta}$ induced by implicit regularization. This was motivated by a surprising observation in OLS: fitting explicitly regularized least-squares with a diagonal version of the implicit regularization matrix reproduced the same weights as unregularized early stopping. So not only did an explicit regularizer reproduce the predicted implicit effects, but even an *misspecified* (underparameterized) explicit regularizer reproduced the same effects!

Here we show that under various convexity and curvature assumptions, fitting $\hat{\Lambda}$ such that the resulting regularized loss is minimized at $\hat{\theta}$ actually produces a unique (or at least locally unique) solution. Therefore, retraining a model on that regularized loss recovers exactly $\hat{\theta}$. The most lenient of these assumptions – that the regularized objective has positive definite Hessian at $\hat{\theta}$ – is quite plausible in our estimation. A rich literature suggests that flatter minima have desirable properties which lead to improved generalization [16, 18], and consequently many modern training techniques (like choice of batch size [19]) optimize for flatter minima either intentionally [10, 12, 47] or by happy coincidence.

Fix an estimated hyperparameter $\hat{\Lambda}$. Let $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$ be C^2, and suppose that for the fixed $\hat{\Lambda}$, the map $\mathcal{R} : \mathbb{R}^p \to \mathbb{R}$ is C^2.

Define

$$f(\theta) \coloneqq \mathcal{L}(\theta) + \mathcal{R}(\theta, \hat{\Lambda}), \qquad \mathcal{F}(\theta) \coloneqq \nabla f(\theta).$$

Assume unconstrained optimization over $\Theta = \mathbb{R}^p$.

Local uniqueness via local convexity.

Proposition A (PD Hessian $\Rightarrow$ isolated stationary point)**.** *If $\mathcal{F}(\hat{\theta}) = 0$ and $\nabla^2 f(\hat{\theta}) \succ 0$, then: (i) $\hat{\theta}$ is a strict local minimizer of f; (ii) there exists $r > 0$ such that $\hat{\theta}$ is the unique zero of $\mathcal{F}$ in $B_r(\hat{\theta})$.*

Proof. Let $H(\theta) = \nabla^2 f(\theta)$. Since H is continuous and $H(\hat{\theta}) \succ 0$, there exist $\mu > 0$ and $r > 0$ such that $H(\theta) \succeq \mu I$ for all $\theta \in B_r(\hat{\theta})$. Hence f is μ-strongly convex on the convex ball $B_r(\hat{\theta})$. Because $\nabla f(\hat{\theta}) = 0$, $\hat{\theta}$ is the minimizer of f on $B_r(\hat{\theta})$, and strong convexity makes this minimizer unique. Therefore $\hat{\theta}$ is a strict local minimizer and the unique zero of ∇f in $B_r(\hat{\theta})$. □

Global uniqueness: two sufficient routes.

Proposition B (Strong convexity)**.** *Assume f is μ-strongly convex on $\mathbb{R}^p$ (i.e., $\nabla^2 f(\theta) \succeq \mu I$ for all θ), then f has a unique global minimizer θ^* and $\mathcal{F}(\theta) = 0$ has the unique solution $\theta = \theta^*$.*

Proof. Strong convexity implies the quadratic lower bound $f(\theta) \geq f(0) + \langle \nabla f(0), \theta \rangle + \frac{\mu}{2}\|\theta\|^2$, hence $f(\theta) \to \infty$ as $\|\theta\| \to \infty$ (coercivity) and thus existence of a minimizer. Strong convexity also yields strict convexity, so the minimizer is unique and must satisfy $\nabla f(\theta^*) = 0$. □

Proposition C (Convexity + PD Hessian at the candidate minimizer)**.** *Let f be convex on $\mathbb{R}^p$ and suppose $\mathcal{F}(\hat{\theta}) = 0$ with $\nabla^2 f(\hat{\theta}) \succ 0$. Then $\hat{\theta}$ is the unique global minimizer of f.*

Proof. For convex f, every stationary point is a global minimizer. Assume, for contradiction, there is $\theta' \neq \hat{\theta}$ with $f(\theta') = f(\hat{\theta})$. By convexity, f is constant along the segment $[\hat{\theta}, \theta']$, which implies zero directional second derivative at $\hat{\theta}$ in the direction $d = \theta' - \hat{\theta}$. But $\nabla^2 f(\hat{\theta}) \succ 0$ gives $d^\top \nabla^2 f(\hat{\theta}) d > 0$ for any $d \neq 0$, a contradiction. □

So what criteria are sufficient to exactly recover the original weights? Suppose a training procedure (with implicit regularization) produces weights $\hat{\theta}$. If we fit $\hat{\Lambda}$ so that *stationarity at $\hat{\theta}$ holds*:

$$\nabla \mathcal{L}(\hat{\theta}) + \nabla \mathcal{R}(\hat{\theta}, \hat{\Lambda}) = 0,$$

then:

- By Proposition B, if f is strongly convex on $\mathbb{R}^p$, retraining on f recovers $\hat{\theta}$ uniquely.
- By Proposition C, the same holds if f is convex and $\nabla^2 f(\hat{\theta}) \succ 0$.
- By Proposition A, even without convexity, there exists $r > 0$ such that $\hat{\theta}$ is the unique stationary point in $B_r(\hat{\theta})$. Hence any retraining method that converges to a stationary point lying in $B_r(\hat{\theta})$ must recover $\hat{\theta}$.

Stability under approximate gradient matching.

Proposition D (Perturbation bound under strong convexity)**.** *Assume f is μ-strongly convex and let $\theta^* = \arg\min f$. If $\|\mathcal{F}(\hat{\theta})\| \le \varepsilon$, then*

$$\|\hat{\theta} - \theta^*\| \le \frac{\varepsilon}{\mu} \quad \textit{and} \quad f(\hat{\theta}) - f(\theta^*) \le \frac{\varepsilon^2}{2\mu}.$$

Proof. Strong convexity implies *strong monotonicity* of the gradient: $\langle \mathcal{F}(\hat{\theta}) - \mathcal{F}(\theta^*), \hat{\theta} - \theta^* \rangle \ge \mu \|\hat{\theta} - \theta^*\|^2$, with $\mathcal{F}(\theta^*) = 0$. By Cauchy–Schwarz, $\mu\|\hat{\theta} - \theta^*\|^2 \le \|\mathcal{F}(\hat{\theta})\| \, \|\hat{\theta} - \theta^*\|$, giving the distance bound. For the function gap, strong convexity gives $f(\theta^*) \ge f(\hat{\theta}) + \langle \mathcal{F}(\hat{\theta}), \theta^* - \hat{\theta} \rangle + \frac{\mu}{2}\|\theta^* - \hat{\theta}\|^2$. Rearrange and apply Young's inequality $\langle a, b \rangle \le \frac{1}{2\mu}\|a\|^2 + \frac{\mu}{2}\|b\|^2$ with $a = \mathcal{F}(\hat{\theta})$, $b = \hat{\theta} - \theta^*$, to obtain $f(\hat{\theta}) - f(\theta^*) \le \|\mathcal{F}(\hat{\theta})\|^2/(2\mu)$. □

Linear regression with quadratic penalty (specialization)

Let

$$f(\theta) = \tfrac{1}{2}\|y - X\theta\|_2^2 + \tfrac{1}{2}\,\theta^\top \hat{\Lambda}\theta, \quad \mathcal{F}(\theta) = (X^\top X + \hat{\Lambda})\theta - X^\top y,$$

with $X \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$, and assume $\hat{\Lambda}$ is symmetric (only its symmetric part matters in ∇f).

Proposition E (Uniqueness and equality of weights)**.** *If $A := X^\top X + \hat{\Lambda} \succ 0$ and $A\hat{\theta} = X^\top y$, then: (i) f is strongly convex with unique minimizer $\theta^* = A^{-1}X^\top y$; (ii) $\theta^* = \hat{\theta}$ (explicit retraining exactly recovers the original weights).*

Proof. $A \succ 0$ implies $\nabla^2 f(\theta) = A \succ 0$, hence strong convexity and uniqueness. The normal equations for the minimizer are $A\theta^* = X^\top y$, so $\theta^* = \hat{\theta}$ by the assumed identity. □

Remark (singular case). If $A \succeq 0$ but singular, the problem is convex but not strongly convex. A minimizer exists iff $X^\top y \in \mathrm{range}(A)$, in which case the set of minimizers is

$$A^\dagger X^\top y + \ker(A).$$

If $X^\top y \notin \mathrm{range}(A)$, the objective is unbounded below. Thus unique recovery is impossible without further structure (e.g., an extra constraint or a selection rule such as minimum norm).

Summary. Suppose the trained weights $\hat{\theta}$ were originally obtained as the solution to

$$\hat{\theta} \in \arg\min_{\theta} \left\{ \mathcal{L}(\theta, D) + \mathcal{R}(\theta, \Lambda^\star) \right\},$$

for some true, possibly implicit, regularizer $\Lambda^\star$. Now consider an estimated regularizer $\hat{\Lambda}$. If $\hat{\Lambda}$ satisfies the first-order matching condition

$$\nabla_\theta \mathcal{R}(\hat{\theta}, \hat{\Lambda}) = -\nabla_\theta \mathcal{L}(\hat{\theta}, D),$$

then $\hat{\theta}$ is stationary for the regularized objective

$$\mathcal{L}(\theta, D) + \mathcal{R}(\theta, \hat{\Lambda}), \qquad \nabla_\theta \left[\mathcal{L}(\hat{\theta}, D) + \mathcal{R}(\hat{\theta}, \hat{\Lambda}) \right] = 0.$$

However, this first-order condition alone does not imply that retraining with $\hat{\Lambda}$ recovers $\hat{\theta}$. One also needs additional curvature or convexity assumptions. Under such assumptions, $\hat{\theta}$ is a local or global minimizer of

$$\theta \mapsto \mathcal{L}(\theta, D) + \mathcal{R}(\theta, \hat{\Lambda}).$$

If this minimizer is unique, then

$$\hat{\theta} = \arg\min_{\theta} \left\{ \mathcal{L}(\theta, D) + \mathcal{R}(\theta, \hat{\Lambda}) \right\}.$$

This establishes recovery of $\hat{\theta}$ conditional on the estimated regularizer $\hat{\Lambda}$. It does not necessarily establish recovery or identification of the original regularizer $\Lambda^\star$. In particular, it is possible that $\hat{\Lambda} \neq \Lambda^\star$ while both regularizers induce the same recovered parameter $\hat{\theta}$, provided they generate the same regularizer gradient at $\hat{\theta}$:

$$\nabla_\theta \mathcal{R}(\hat{\theta}, \hat{\Lambda}) = \nabla_\theta \mathcal{R}(\hat{\theta}, \Lambda^\star).$$

More generally, $\hat{\Lambda}$ and $\Lambda^\star$ need only be equivalent in the directions relevant to the stationarity and optimality conditions at $\hat{\theta}$. Therefore, $\hat{\Lambda}$ may rationalize or reproduce the trained weights without being identifiable as the true regularizer.

B PAC Guarantees

B.1 Linear parameterized case

Setup. We address the candidate regularizer case within Section 2.2, where we simultaneously fit a number of candidate regularizers with scale coefficients – essentially turning the problem into linear regression over a set of regularizer gradients.

Say we have a regularizer $\mathcal{R}(\theta, D, \Lambda)$ that is linear in r unknown coefficients:

$$\mathcal{R}(\theta, D, \Lambda) := \sum_{k=1}^{r} \lambda_k \mathcal{R}_k(\theta, D), \qquad \Lambda = (\lambda_1, ..., \lambda_r)^\top.$$

Let $b := -\nabla_\theta \mathcal{L}(\hat{\theta}, D) \in \mathbb{R}^p$. We define the endpoint regularizer-gradient design matrix

$$\Phi \in \mathbb{R}^{p \times r}, \qquad \Phi_{i,j} := \partial_{\theta_i} R_j(\hat{\theta}, D), \qquad \text{so } \nabla_\theta \mathcal{R}(\theta, D, \Lambda) = \Phi\Lambda.$$

As this is a linear regression problem, the endpoint estimator is

$$\Lambda \in \arg\min_{\Lambda \in \mathbb{R}^r} \|\Phi\Lambda - b\|_2^2.$$

The following set of assumptions suffice for a pseudo-parametric rate from using the least-squares estimator in the case where the model parameter errors are correlated.

Assumption 1 (Endpoint well-specification)**.** *There exists $\Lambda^\star \in \mathbb{R}^r$ so that $b = \Phi\Lambda^\star + \xi + a$, where $a \in \mathbb{R}^p$ is a deterministic approximation bias term.*

- *(Sub-Gaussian correlated errors) $\mathbb{E}[\xi|\Phi] = 0$, and there exists $\sigma \geq 0$ so that $\mathbb{E}[\exp(tv^\top \xi)|\Phi] \leq \exp(\sigma^2 t^2 v^\top \Sigma v/2)$ for all $v \in \mathbb{R}^p$.*
- *(Realizability) $\alpha = \frac{1}{\sqrt{p}} \|(\Phi^\top \Phi)^{-1/2} \Phi^\top a\| < \infty$.*
- *(Identifiability and conditioning) There exist κ, ρ so that $\lambda_{\min}(p^{-1}\Phi^\top \Phi) \geq \kappa > 0$, and $\rho = \lambda_{\max}((\Phi^\top \Phi)^{-1/2} \Phi^\top \Sigma \Phi (\Phi^\top \Phi)^{-1/2}) < \infty$.*

κ is a minimum eigenvalue assumption, and ρ describes an estimation error inflation factor induced by correlation across endpoint parameter errors. This is the largest eigenvalue of Σ when restricted to the column space of the regularizer gradient design matrix. Observe that this is 1 when $\Sigma = I_p$.

More explicitly, the error covariance matrix Σ is the covariance of the endpoint gradient error ξ. The final inflation factor describes how the endpoint gradient errors are correlated after projection onto the span of candidate regularizer gradients. For example, if Σ has large eigenvalues along directions that lie in the column space of Φ, then the noise is aligned with the same directions that the candidate regularizers are trying to explain, and learning then becomes difficult.

On the other hand, if two regularizer gradients are highly collinear, then $\Phi^\top \Phi$ is ill-conditioned. This is encapsulated in the minimum eigenvalue condition.

Theorem 1 (PAC Bound for the Linear Parameterized Case)**.** *Conditional on* Φ*, it holds with probability at least* $1-\delta$ *that*

$$\|\Lambda - \Lambda^\star\|_2 \leq \frac{2\sigma\sqrt{2\rho}}{\sqrt{\kappa}}\sqrt{\frac{r\log 5 + \log(2/\delta)}{p}} + \frac{\alpha}{\sqrt{\kappa}}.$$

Defining $p_{\text{eff}} = p/\rho$ for the effective parameter count, this shows that the error scales in $O(\sqrt{r/p_{\text{eff}}})$. This result illuminates a phenomenon we've seen within the experiments – that the error increases in the number of regularizer parameters we estimate, while decreasing drastically when multiple independent endpoints are used as doubling the number of independent endpoints doubles the effective parameter count.

Proof. The normal equations state that $\Phi^\top(\Phi\Lambda - b) = 0$. Substituting and rearranging, we obtain $\Phi^\top(\Phi\Lambda - \Phi\Lambda^\star - \xi - a) = 0$. The conditioning assumption allows us to invert the covariance matrix, and doing so yields

$$\Lambda - \Lambda^\star = (\Phi^\top\Phi)^{-1}\Phi^\top\xi + (\Phi^\top\Phi)^{-1}\Phi^\top a.$$

Now, because $\lambda_{\min}(p^{-1}\Phi^\top\Phi) \geq \kappa > 0$,

$$(\Phi^\top\Phi)^{-1}\Phi^\top\xi = (\Phi^\top\Phi)^{-1/2}(\Phi^\top\Phi)^{-1/2}\Phi^\top\xi \leq \frac{1}{\sqrt{p\kappa}}(\Phi^\top\Phi)^{-1/2}\Phi^\top\xi.$$

Take any vector $u \in \mathbb{R}^r$ where $\|u\| = 1$. $u^\top(\Phi^\top\Phi)^{-1/2}\Phi^\top\xi$ is now a mean-zero sub-Gaussian random variable with variance proxy $\sigma^2 u^\top(\Phi^\top\Phi)^{-1/2}\Phi^\top\Sigma\Phi(\Phi^\top\Phi)^{-1/2}u \leq \sigma^2\rho$. By the standard sub-Gaussian tail bound as well as a union bound over a $1/2$-net of the unit ball in r dimensions (which has cardinality $(1 + 2/(1/2))^r = 5^r$), we can conclude that it holds w.p. at least $1-\delta$ that

$$\|(\Phi^\top\Phi)^{-1/2}\Phi^\top\xi\|_2 \leq 2\sqrt{2\sigma^2\rho(r\log 5 + \log(2/\delta))}.$$

This implies a bound on the error term

$$\|(\Phi^\top\Phi)^{-1}\Phi^\top\xi\|_2 \leq \frac{2\sigma\sqrt{2\rho}}{\sqrt{\kappa}}\sqrt{\frac{r\log 5 + \log(2/\delta)}{p}}.$$

It remains to handle the deterministic bias term. We assumed that $\alpha = \frac{1}{\sqrt{p}}\|(\Phi^\top\Phi)^{-1/2}\Phi^\top a\| < \infty$, so

$$\|(\Phi^\top\Phi)^{-1}\Phi^\top a\|_2 \leq \|(\Phi^\top\Phi)^{-1/2}\|_{\text{op}}\|(\Phi^\top\Phi)^{-1/2}\Phi^\top a\|_2 \leq \frac{1}{\sqrt{p\kappa}}\cdot\alpha\sqrt{p},$$

yielding the result. □

B.2 Nonparametric case.

Setup. Let $b = -\nabla_\theta\mathcal{L}(\hat{\theta}, D) \in \mathbb{R}^p$. Consider a regularizer class $\mathcal{R}$. Defining $g_R = \nabla_\theta R(\hat{\theta}, D)$, this induces a gradient class $\mathcal{G} = \{g_R : R \in \mathcal{R}\} \subseteq \mathbb{R}^p$. Here, we minimize the squared error and choose $g \in \arg\min_{g\in\mathcal{G}}\|g - b\|_2^2$. To obtain a bound on the error, we make the following assumptions:

Assumption 2 (Nonparametric endpoint well-specification)**.** $b = b^\star + \xi$*, where* $b^\star$ *is the predictable endpoint bias and* ξ *is mean-zero correlated error. Additionally,*

- *(Sub-Gaussian correlated errors)* $\mathbb{E}[\xi|\Phi] = 0$*, and there exists* $\sigma \geq 0$ *so that* $\mathbb{E}[\exp(tv^\top\xi)|\Phi] \leq \exp(\sigma^2t^2v^\top\Sigma v/2)$ *for all* $v \in \mathbb{R}^p$*.*
- *(Bounded log-covering number)* $\log\mathcal{N}_\mathcal{G}(\varepsilon)$ *is finite for some* $\varepsilon > 0$*.*
- *(Correlation inflation factor) The restricted correlation inflation factor*

$$\rho(\mathcal{G}) = \sup_{g\neq g'\in\mathcal{G}}\frac{(g-g')^\top\Sigma(g-g')}{(g-g')^\top(g-g')} < \infty.$$

Theorem 2 (PAC Bound for the Nonparametric Case)**.** *Let $\hat{R}$ be the empirical risk minimizer, $\mathcal{G}$ the induced regularizer gradient class by the regularizer class $\mathcal{R}$, and $\mathcal{G}_\varepsilon$ an ε-net of $\mathcal{G}$.*

$$\frac{1}{p}\|\nabla_\theta \hat{R}(\hat{\theta}, D) + \mathbb{E}\nabla_\theta \mathcal{L}(\hat{\theta}, D)\|_2^2 - \inf_{R\in\mathcal{R}} \frac{1}{p}\|\nabla_\theta R(\hat{\theta}, D) + \mathbb{E}\nabla_\theta \mathcal{L}(\hat{\theta}, D)\|_2^2$$

is bounded w.p. at least $1-\delta$ by

$$2\varepsilon \inf_{R\in\mathcal{R}} \|\nabla_\theta R(\hat{\theta}, D) + \mathbb{E}\nabla_\theta \mathcal{L}(\hat{\theta}, D)\|_2 + \varepsilon^2 + 2\sigma \left(\sup_{g,g'\in\mathcal{G}} \frac{1}{\sqrt{p}}\|g-g'\|_2\right)\sqrt{\frac{2\rho(\mathcal{G})(\log \mathcal{N}_\mathcal{G}(\varepsilon) + \log(2/\delta))}{p}}.$$

As before, the effective parameter count is $p/\rho(\mathcal{G})$. This blows up if the error covariance matrix Σ has large eigenvalues in directions representable by differences in the gradient class. Note that any blowup is not caused by the complexity of $\mathcal{G}$ alone, but is instead caused by the interaction between differences in $\mathcal{G}$ and the error covariance matrix. The error covariance matrix describes how noise in the loss gradient target is correlated across model parameters.

Proof. Consider an ε-net of $\mathcal{G}$ that we will call $\mathcal{G}_\varepsilon$. Let

$$g_\varepsilon \in \arg\min_{g\in\mathcal{G}_\varepsilon} \|g-b\|_2^2, \qquad g_\varepsilon^\star \in \arg\min_{g\in\mathcal{G}_\varepsilon} \|g-b^\star\|_2^2.$$

Now because $\|g_\varepsilon - b\|_2^2 \le \|g_\varepsilon^\star - b\|_2^2$ and $b = b^\star + \xi$,

$$\frac{1}{p}\|g_\varepsilon - b^\star\|_2^2 - \frac{1}{p}\|g_\varepsilon^\star - b^\star\|_2^2 \le \frac{2}{p}\langle g_\varepsilon - g_\varepsilon^\star, \xi\rangle \le 2\sup_{g\in\mathcal{G}_\varepsilon}\left|\frac{1}{p}\langle g_\varepsilon - g_\varepsilon^\star, \xi\rangle\right|.$$

Now observe that $\frac{1}{p}\langle g_\varepsilon - g_\varepsilon^\star, \xi\rangle$ is sub-Gaussian with variance proxy bounded by

$$\frac{\sigma^2}{p^2}(g_\varepsilon - g_\varepsilon^\star)^\top \Sigma (g_\varepsilon - g_\varepsilon^\star) \le \frac{\sigma^2\rho(\mathcal{G}_\varepsilon)}{p^2}\sup_{g,g'\in\mathcal{G}_\varepsilon}\|g-g'\|_2.$$

It then remains to employ the sub-Gaussian tail bound and a union bound over $\mathcal{G}_\varepsilon$ to find that it holds w.p. at least $1-\delta$ that

$$\sup_{g\in\mathcal{G}_\varepsilon}\left|\frac{1}{p}\langle g_\varepsilon - g_\varepsilon^\star, \xi\rangle\right| \le \sigma \sup_{g,g'\in\mathcal{G}_\varepsilon}\frac{1}{\sqrt{p}}\|g-g'\|_2\sqrt{\frac{2\rho(\mathcal{G}_\varepsilon)(\log\mathcal{N}_\mathcal{G}(\varepsilon)+\log(2/\delta))}{p}}.$$

It then follows that $\frac{1}{p}\|g_\varepsilon - b^\star\|_2^2 - \frac{1}{p}\|g_\varepsilon^\star - b\|_2^2$ is bounded w.p. at least $1-\delta$ by

$$2\varepsilon\inf_{g\in\mathcal{G}}\|g-b^\star\| + \varepsilon^2 + 2\sigma\sup_{g,g'\in\mathcal{G}_\varepsilon}\frac{1}{\sqrt{p}}\|g-g'\|_2\sqrt{\frac{2\rho(\mathcal{G}_\varepsilon)(\log\mathcal{N}_\mathcal{G}(\varepsilon)+\log(2/\delta))}{p}}.$$

□

C Learning a regularizer on a single trajectory

This generalizes the setup within the manuscript to multiple points in a single trajectory. It turns out that you can get concentration via a martingale bound, but to an object that amounts to the regularizer that best approximates the predictable bias in T iterations.

Setup. Let $\hat{\theta}^{(1)}, ...\hat{\theta}^{(T)}$ be a single trajectory produced by the implicitly biased learning algorithm, and $\theta^{(t)}, ..., \theta^{(T)}$ be a coupled trajectory produced by a learning algorithm without said bias. Write $D_t \subseteq D$ for the minibatch at time t. Define targets

$$b_t := \frac{\hat{\theta}^{(t)} - \hat{\theta}^{(t+1)}}{\eta_t} - \frac{\theta^{(t)} - \theta^{(t+1)}}{\eta_t},$$

which we obtain by rearranging for example $\theta^{(t+1)} = \theta^{(t)} - \eta_t g_t$, where g_t is the update direction.

Examples. This encompasses a host of setups. For example, if one is comparing SGD to batch GD, we obtain

$$b_t = \nabla_\theta \mathcal{L}(\hat{\theta}^{(t)}, D_t) - \nabla_\theta \mathcal{L}(\theta^{(t)}, D).$$

If one is comparing SGD with dropout to plain SGD, we obtain

$$b_t = \frac{\hat{\theta}^{(t)} - \hat{\theta}^{(t+1)}}{\eta_t} - \nabla_\theta \mathcal{L}(\hat{\theta}^{(t)}, D_t).$$

And finally, if we only consider the endpoint $t = T$, we recover the formulation from earlier in the paper:

$$b_T = -\nabla_\theta \mathcal{L}(\theta^{(t)}, D).$$

Regression. As before, we regress $b_1, ..., b_T$ on $\nabla_\theta R(\hat{\theta}^{(t)}, D, \Lambda)$, optimizing over either $R \in \mathcal{R}$ or Λ depending on whether one deals with a nonparametric regularizer class or a parametric regularizer class.

Consider the filtration $\mathcal{F}_t$. Write

$$b_t^\star := \mathbb{E}[b_t \mid \mathcal{F}_t], \qquad M_{t+1} := b_t - b_t^\star.$$

Then it follows that

$$b_t = b_t^\star + M_{t+1}, \qquad \mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = 0.$$

Nonparametric case. We consider the nonparametric case. Define a regularizer class $\mathcal{R}$, with ρ-log-covering number $\log \mathcal{N}_\mathcal{R}(\rho)$. To combat heteroskedasticity, we accommodate p.s.d. weight matrices W_t that are $\mathcal{F}_t$-measurable. Ideally we have $W_t \approx \mathbb{E}[M_{t+1} M_{t+1}^\top \mid \mathcal{F}_t]^{-1} =: \Sigma_t^{-1}$. This induces the loss

$$\hat{\mathcal{E}}_T(R) := \sum_{t=0}^{T-1} \|W_t^{1/2}(b_t - \nabla_\theta R(\hat{\theta}^{(t)}, D))\|_2^2, \text{ and we naturally select } \hat{R}_T \in \operatorname*{arg\,min}_{R \in \mathcal{R}} \hat{\mathcal{E}}_T(R).$$

Their population equivalents are

$$\mathcal{E}_T(R) := \sum_{t=0}^{T-1} \|W_t^{1/2}(b_t^\star - \nabla_\theta R(\hat{\theta}^{(t)}, D))\|_2^2, \text{ and } R_T^\star \in \operatorname*{arg\,min}_{R \in \mathcal{R}} \mathcal{E}_T(R),$$

yielding the best approximation within $\mathcal{R}$ to the predictable bias $b_t^\star := \mathbb{E}[b_t \mid \mathcal{F}_t]$.

We also require

1. **Identifiability.** $\forall R, R' \in \mathcal{R}\ \ \exists t \in [T]$ s.t. $\nabla_\theta R(\hat{\theta}^{(t)}, D) \neq \nabla_\theta R'(\hat{\theta}^{(t)}, D)$.
2. **Measurability.** The maps $R \mapsto \nabla_\theta R(\hat{\theta}^{(t)}, D)$ and weight matrices W_t are measurable for each t, and $b_t \in L^2$.
3. **Bounded increments.** Define $\Delta_t(R, R') := W_t^{1/2}(\nabla_\theta R(\hat{\theta}^{(t)}, D) - \nabla_\theta R'(\hat{\theta}^{(t)}, D))$, and the martingale increments $X_{t+1}(R, R') = \langle \Delta_t(R, R'), W_t^{1/2} M_{t+1} \rangle$. Assume these increments are almost surely bounded: $|X_{t+1}(R, R')| \leq c(R, R')$ for all $t = 1, ..., T$.

It is also helpful to define the quadratic variation $V_T(R, R') := \sum_{t=0}^{T-1} \mathbb{E}[X_{t+1}(R, R')^2 \mid \mathcal{F}_t]$. Then, we can obtain the following generalization bound:

Theorem 3 (Within-trajectory generalization bound)**.** *With probability at least* $1 - \delta$*,*

$$\mathcal{E}_T(\hat{R}_T) - \inf_{R \in \mathcal{R}} \mathcal{E}_T(R) \leq \sup_{R, R' \in \mathcal{R}} O\Bigg(\sqrt{V_T(R, R') \log\left(\frac{\mathcal{N}_\mathcal{R}(1/T)}{\delta}\right)} + c(R, R') \log\left(\frac{\mathcal{N}_\mathcal{R}(1/T)}{\delta}\right)\Bigg).$$

Proof. We first write

$$\hat{\mathcal{E}}_T(R) = \sum_{t=0}^{T-1} \|W^{1/2} b_t^\star + W^{1/2} M_{t+1} - W^{1/2} \nabla_\theta R(\hat{\theta}^{(t)}, D))\|_2^2.$$

Then, we can expand the square to find that

$$\hat{\mathcal{E}}_T(R) = \mathcal{E}_T(R) + 2 \sum_{t=0}^{T-1} \langle W^{1/2} b_t^\star - W^{1/2} \nabla_\theta R(\hat{\theta}^{(t)}, D)), W^{1/2} M_{t+1} \rangle + \sum_{t=0}^{T-1} \|W^{1/2} M_{t+1}\|_2^2.$$

Continuing the argument, we note that $\hat{\mathcal{E}}_T(\hat{R}_T) = \inf_{R \in \mathcal{R}} \hat{\mathcal{E}}_T(R) \leq \hat{\mathcal{E}}_T(R)$, so it must hold that

$$\begin{aligned}
&\mathcal{E}_T(\hat{R}_T) + 2 \sum_{t=0}^{T-1} \langle W^{1/2} b_t^\star - W^{1/2} \nabla_\theta \hat{R}_T(\hat{\theta}^{(t)}, D)), W^{1/2} M_{t+1} \rangle + \sum_{t=0}^{T-1} \|W^{1/2} M_{t+1}\|_2^2 \\
&\leq \mathcal{E}_T(R) + 2 \sum_{t=0}^{T-1} \langle W^{1/2} b_t^\star - W^{1/2} \nabla_\theta R(\hat{\theta}^{(t)}, D)), W^{1/2} M_{t+1} \rangle + \sum_{t=0}^{T-1} \|W^{1/2} M_{t+1}\|_2^2.
\end{aligned}$$

Cancelling and rearranging terms, we arrive at an expression for $\mathcal{E}_T(\hat{R}_T) - \mathcal{E}_T(R)$ that amounts to

$$\begin{aligned}
\mathcal{E}_T(\hat{R}_T) - \mathcal{E}_T(R) &= 2 \sum_{t=0}^{T-1} \langle W^{1/2} \nabla_\theta \hat{R}_T(\hat{\theta}^{(t)}, D)) - W^{1/2} \nabla_\theta R(\hat{\theta}^{(t)}, D)), W^{1/2} M_{t+1} \rangle \\
&\leq 2 \sup_{R, R' \in \mathcal{R}} \left| \sum_{t=0}^{T-1} X_{t+1}(R, R') \right|.
\end{aligned}$$

Now, note that $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = 0$ implies that $\mathbb{E}[X_{t+1}(R, R') \mid \mathcal{F}_t] = 0$ is a martingale increment, with quadratic variation $V_T(R, R')$ and almost surely bounded increments $|X_{t+1}(R, R')| \leq c(R, R')$. We can then apply Freedman's inequality to find

$$\mathbb{P}\left(\left| \sum_{t=0}^{T-1} X_{t+1}(R, R') \right| \geq \sqrt{2 V_T(R, R') \varepsilon} + c(R, R') \varepsilon / 3 \right) \leq 2 \exp(-\varepsilon).$$

Applying a union bound over a $1/T$-net of $\mathcal{R}$ then yields our desired result:

$$\begin{aligned}
\mathcal{E}_T(\hat{R}_T) - \inf_{R \in \mathcal{R}} \mathcal{E}_T(R) \leq \sup_{R, R' \in \mathcal{R}} O\Bigg(&\sqrt{V_T(R, R') \log\left(\frac{\mathcal{N}_{\mathcal{R}}(t/T)}{\delta} \right)} \\
&+ c(R, R') \log\left(\frac{\mathcal{N}_{\mathcal{R}}(1/T)}{\delta} \right) \Bigg).
\end{aligned}$$

□

D Computing loss and regularizer gradients

Computational complexity. Often, the loss is of the form $\mathcal{L}(\theta, D) = \mathcal{L}(\hat{Y}(\theta), D)$, where $\hat{Y}(\theta) = \hat{Y}(\theta, D)$ is the output of the model with weights θ. In that case, Equation 2 becomes

$$\nabla_\theta \mathcal{R}(\theta, D)|_{\hat{\theta}(D)}^\top = - \left(\frac{\partial \mathcal{L}(\hat{Y}, D)}{\partial \hat{Y}} \right)^\top \nabla_\theta \hat{Y}(\theta)|_{\hat{\theta}(D)},$$

where $\frac{\partial \mathcal{L}}{\partial \hat{Y}}$ is the partial derivative of the empirical loss with respect to the model output/predictions, and $\nabla_\theta \hat{Y}(\theta)$ is the Jacobian of the model output with respect to the model's weights.

For example taking the empirical loss to be mean-squared error: $\mathcal{L}(\theta, D) = \frac{1}{2} \|Y - \hat{Y}(\theta)\|^2$, we obtain

$$\nabla_\theta \mathcal{R}(\theta, D)|_{\hat{\theta}(D)} = (Y - \hat{Y}(\hat{\theta}))^\top \nabla_\theta \hat{Y}(\theta)|_{\hat{\theta}(D)}.$$

This is just the product of the model's residuals $(Y - \hat{Y})$ and its Jacobian evaluated at the learned weights $(\nabla_\theta \hat{Y}(\theta)|_{\hat{\theta}(D)})$, both of which can be readily computed.

It is typically easy to derive and compute $\frac{\partial \mathcal{L}}{\partial \hat{y}}$ for standard loss functions, which means we can compute the vector-Jacobian product $(\frac{\partial \mathcal{L}}{\partial \hat{y}})^\top \nabla_\theta \hat{Y}(\theta)$ instead of calculating and storing the Jacobian itself. This vector-Jacobian product has dimensionality $1 \times p$, or equivalently p if gradients are treated as column vectors. Computing it avoids explicitly forming the full Jacobian $\nabla_\theta \hat{Y}(\theta) \in \mathbb{R}^{k \times p}$, where k is output dimension of the model. Moreover, this computation doesn't depend on the choice of $\mathcal{R}$ at all – this will be helpful when we iterate over candidate regularizers, as this quantity need not be recomputed.

As mentioned in the main text, our stationarity condition implies a system of p equations where p is the dimensionality of θ. To see this, note that $\nabla_\theta \mathcal{R}(\theta, D) \in \mathbb{R}^p$, $\left(\frac{\partial \mathcal{L}(\hat{y}, D)}{\partial \hat{y}}\right)^\top \in \mathbb{R}^k$ where k is the output size of the model, and $\nabla_\theta \hat{y}(\theta) \in \mathbb{R}^{k \times p}$.

E Related work on estimating inductive bias

[7] propose a Bayesian framework for quantifying the *amount* of inductive bias needed to achieve generalization on some prediction task. They define inductive bias of a task as the negative log probability that a hypothesis h achieves some test error rate ε – intuitively, if a randomly sampled hypothesis $h \sim p_h$ is unlikely (small probability) then a large inductive bias is required to reliably find a generalizing hypothesis (large negative log-probability).

[17] propose a method of quantifying inductive bias based on probing intermediate representations. They consider Bayesian evidence as a proxy for inductive bias, formalizing it as the maximum evidence (how likely it is that a particular dataset could have been generated by a given model) for some intermediate representation, over all possible probes in a function class (e.g. linear probes). The intuition is that overly-simple models are unlikely because they don't explain the data well and complex models are unlikely because they are so high-dimensional that many candidate models can also explain the data equally well.

Neither of these approaches involves actually estimating the parameters of the kinds of implicit regularizers suggested in the literature, as we do in the present work. Rather they quantify inductive bias as a scalar quantity and try to judge the amount of inductive bias present in various models or representations. In these settings, any predictive model has a fixed amount of inductive bias on a prediction task – there is no way to evaluate competing hypotheses for how that bias is produced. By directly estimating the parameters of hypothesized regularizers, our framework enables empirically analyzing and contrasting theoretical predictions about implicit bias in realistic models.

F Bootstrapping

Bootstrap resampling is used only in the experiment in this appendix, not in Figures 3–6.

In Figure 7 we replicate the OLS early-stopping recovery experiment (Figure 4) using bootstrap resampling of a single training set rather than independent draws of ground-truth weights. Each bootstrap endpoint is trained from scratch on a row-resampled $(X^{*(b)}, y^{*(b)})$ for $b = 1, \ldots, m-1$, with the canonical training endpoint included as $b = 0$. Compared to the multi-dataset experiment, bootstrap recovery is qualitatively weaker: the multi-dataset fixed-t system is exactly consistent (all endpoints share the same X and hence the same theoretical $\Lambda^{(t)}$), so it reaches numerical recovery at $m = p$, whereas the bootstrap system is only approximately consistent ($\Lambda^{*(b),(t)}$ varies with the resampled $X^{*(b)}$) and exhibits a residual model-misspecification floor. Recovery is also gated on observation noise: the spread in $\hat{\theta}^{*(b)}$ scales as $\sigma/\sqrt{n}$, so for very clean data (small σ) bootstrap thetas cluster too tightly around $\hat{\theta}(D)$ for the LS system to be identifiable. The σ sweep in Panel F shows the practical regime: bootstrap recovery beats reporting zero only when the OLS sampling variance is non-negligible relative to the coefficient magnitude.

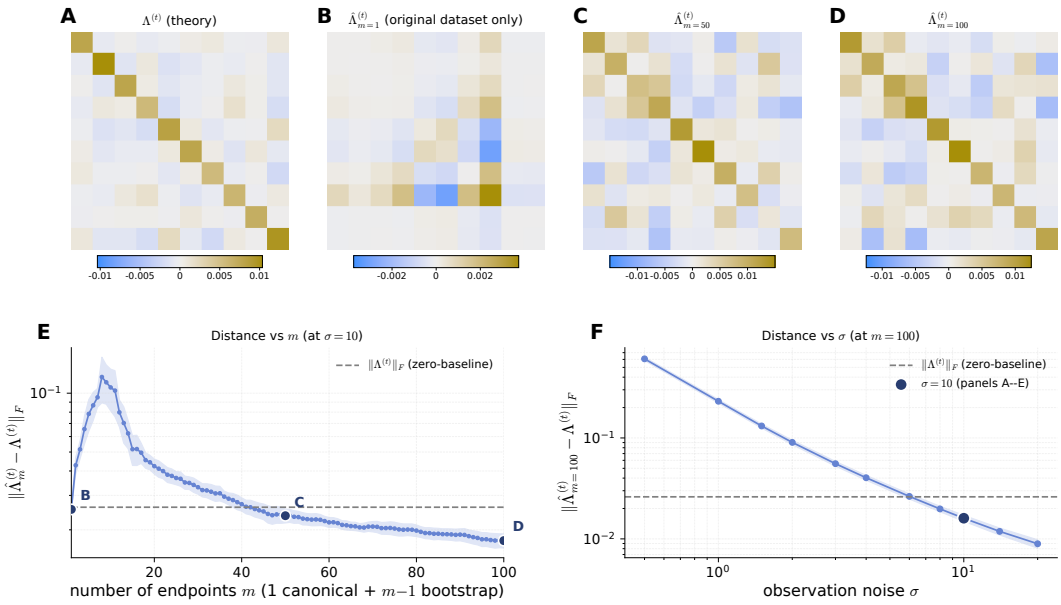


Figure 7: **Bootstrap recovery of the early-stopped GD regularizer.** Parallel to Figure 4 but with bootstrap resamples of a single training set in place of independent ground-truth weight redraws. We use a noisier variant of the OLS DGP ($\sigma = 10$ rather than $\sigma = 1$ in the main composite figure): bootstrap recovery requires sufficient OLS sampling variance ($\propto \sigma/\sqrt{n}$) for the bootstrap-induced spread in $\hat{\theta}$ to span the $p(p+1)/2 = 55$-dim space of symmetric matrices. **(A)** Theoretical $\Lambda^{(t)}$ at the canonical fixed iterate $t = 500$. **(B)** The minimum-norm fit from the single canonical training endpoint. With only p stationarity equations in $p(p+1)/2$ unknowns, the LS system is underdetermined and `lstsq` returns the smallest-Frobenius-norm $\hat{\Lambda}$ satisfying the constraints; most of the matrix is left at zero. **(C)** Recovery from $m = 50$ endpoints (1 canonical + 49 bootstrap). **(D)** Recovery from $m = 100$ endpoints, comfortably past the rank threshold $m \approx p$ where the LS design matrix transitions from underdetermined to overdetermined. **(E)** Distance to theory $\|\hat{\Lambda}^{(t)}_m - \Lambda^{(t)}\|_F$ as m grows from 1 to 100 at $\sigma = 10$, with markers at the m values used in panels B–D. The dashed gray line is $\|\Lambda^{(t)}\|_F$, equivalent to the trivial baseline $\hat{\Lambda} = 0$. Band shows mean and 95% CI across 10 pools of bootstrap resamples. **(F)** Distance at $m = 100$ swept over σ (mean and 95% CI across 10 pools of bootstrap resamples). Below $\sigma \approx 6$ bootstrap recovery is worse than the zero-baseline; above it the recovered matrix is increasingly informative.

Algorithm 1 Bootstrap-augmented estimation

Input: Dataset D, learning algorithm $\mathcal{A}$, regularizer family $\mathcal{R}(\theta, D, \Lambda)$, number of bootstrap replicates B

Output: Estimate $\hat{\Lambda}$

1: Train the predictive model on D to obtain $\hat{\theta}(D)$
2: **for** $b = 1, \ldots, B$ **do**
3: Sample a bootstrap sample $D^{*(b)}$ from D with replacement
4: Retrain the predictive model on $D^{*(b)}$ to obtain $\hat{\theta}^{*(b)}$
5: **end for**
6: Estimate $\hat{\Lambda}$ by minimizing

$$\left\| \nabla_\theta \mathcal{R}(\theta, D, \Lambda)\big|_{\hat{\theta}(D)} + \nabla_\theta \mathcal{L}(\theta, D)\big|_{\hat{\theta}(D)} \right\|_2^2 + \sum_{b=1}^{B} \left\| \nabla_\theta \mathcal{R}(\theta, D^{*(b)}, \Lambda)\big|_{\hat{\theta}^{*(b)}} + \nabla_\theta \mathcal{L}(\theta, D^{*(b)})\big|_{\hat{\theta}^{*(b)}} \right\|_2^2$$

7: **return** $\hat{\Lambda}$

G Additional considerations for stable estimation of implicit regularization

G.1 Normalizing gradients

When candidate regularizers induce gradients with very different norms, we use a normalized gradient-matching variant that first rescales each candidate gradient to unit norm, fits coefficients in that normalized basis, and then maps back to the original regularization parameters.

G.2 Evaluating goodness of fit

Evaluating quality of fit is central to the utility of the framework. Equation 5 compares the squared residual after adding the fitted regularizer gradient with the squared loss gradient before adding it. It is analogous to the uncentered R^2 in linear regression: a value near one indicates that the candidate family closely matches the target gradient, zero indicates no improvement over the zero regularizer, and a negative value indicates a larger residual. If the family includes the zero regularizer and the gradient-matching objective is minimized exactly on the same target, the score cannot be negative.

Both gradients should be evaluated at the same weights and on the same dataset, using the fitted regularizer in its original units. To assess fit to the empirical loss gradient, use the full dataset or a sufficiently accurate average of minibatch gradients before computing the squared norms. Averaging scores computed separately on minibatches is a different diagnostic and can be dominated by minibatch variability. When the loss gradient is nearly zero, the ratio becomes sensitive to small numerical or sampling errors; report the loss-gradient and residual norms alongside the score, and treat the score as undefined when its denominator is zero.

A low score may indicate misspecification, but it can also arise from noisy gradient estimates or failure to optimize the regularizer coefficients. These possibilities should be checked before rejecting the candidate family. Moreover, an endpoint that is still moving under its training procedure may reflect incomplete optimization as well as implicit regularization. A high score establishes a good gradient match at the evaluated weights, but does not identify a unique regularizer or show that it captures the effect of a particular training mechanism. Controlled ablations and retraining with the fitted explicit penalty provide stronger checks, as discussed in Section 2.1.

G.3 Sensitivity to training duration and convergence

The estimated regularizer can depend on when we evaluate the predictive model during training. Fitting at successive checkpoints helps assess this sensitivity: changes in the estimate may reflect evolving implicit effects as well as incomplete optimization of the predictive model. This is distinct from error in fitting the regularizer coefficients at fixed weights.

Figure 8 illustrates this for the dropout experiment in Section 3 and Figure 5. We use the ten depth-3, width-256 runs with dropout rate 0.3 from the final grid, following the training and stopping protocol in Appendix H.4. At each saved checkpoint we compute the exact scalar ridge fit to the full clean-training loss gradient. The fitted coefficients decrease substantially early in training and change less in absolute magnitude later on, reaching small positive values at the stopping endpoints. Because these coefficients approach zero, small absolute changes can still correspond to appreciable relative changes. This comparison characterizes sensitivity to training duration; coefficient stability alone does not establish stationarity or adequacy of the candidate family.

H Further experiment details

Code and data availability. The implementation and reproduction workflow are publicly available at `https://github.com/jrudoler/estimating-implicit-regularization`. The repository documents the software environment, dependency versions, and commands for training, analysis, and figure generation. Synthetic data are generated by the experiment scripts; MNIST is downloaded through TorchVision.

Existing assets and licenses. We use MNIST [21], created by Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. Its dataset card lists the MIT license (`https://huggingface.co/datasets/ylecun/mnist`). PyTorch and TorchVision are distributed under BSD 3-Clause licenses

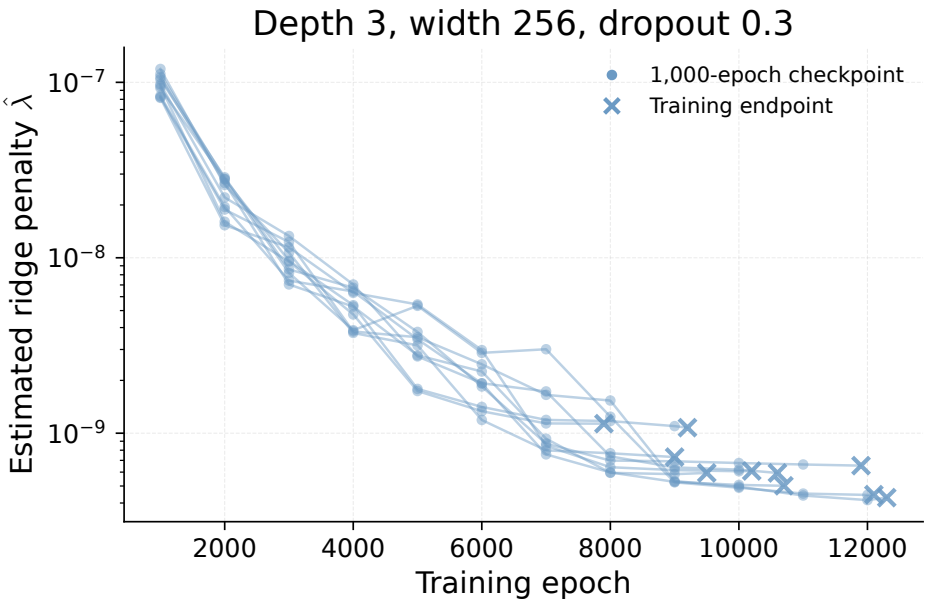


Figure 8: **Sensitivity of the estimated dropout ridge coefficient to training duration.** Each translucent curve represents one of ten seeds. Circles mark saved checkpoints every 1,000 epochs; crosses mark each run's training-loss stopping endpoint (7,900–12,300 epochs). The vertical axis is logarithmic. Estimates are computed independently at each checkpoint, without normalizing by the final epoch or coefficient. All signed coefficients shown are positive and therefore equal their nonnegative projections. Pilot extensions beyond the stopping endpoint are excluded.

(`https://github.com/pytorch/pytorch/blob/main/LICENSE` and `https://github.com/pytorch/vision/blob/main/LICENSE`, respectively). The reproduction workflow downloads MNIST from the upstream source rather than redistributing it as a new dataset.

H.1 Illustrative figures (Figs. 1 and 2)

Figures 1 and 2 are programmatic illustrations rather than empirical experiments. Figure 1 illustrates ridge regression on a scalar linear model $y = \theta x + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, $\sigma = 1.8$, $n = 20$ observations, $x \sim \mathcal{N}(0, 1)$, true $\theta = 2.0$, and ℓ_2 penalty strength $\lambda = 0.8$. The MSE, ℓ_2 penalty, and total objective are plotted as functions of the scalar parameter θ; gradient arrows at $\hat{\theta}_{\text{ridge}}$ confirm the equal-and-opposite stationarity condition $\nabla_\theta \mathcal{L} + \nabla_\theta \ell_2 = 0$. Figure 2 simulates a two-parameter $\tanh(wx + b)$ network on 96 training points $x \in [-2.5, 2.5]$ with target $y = 0.75 \tanh(1.4x - 0.35) + 0.12 \sin(2.7x)$; gradient flow is approximated by full-batch GD at learning rate 4×10^{-3} for 3000 steps, and SGD at learning rate 0.40 with batch size 8 for 6 steps.

H.2 Recovering elastic net

We trained a non-linear neural network with one hidden layer (200 units) and ReLU activation to predict a scalar output from a 10-dimensional input. The model was trained using stochastic gradient descent with a learning rate of 10^{-3} for up to 200 epochs, with early stopping (patience = 15) based on training loss.

Synthetic data were generated by sampling inputs $X \in \mathbb{R}^{5000 \times 10}$ from a standard normal distribution, and computing targets as

$$y = X\theta + \varepsilon, \quad \theta_i \overset{\text{iid}}{\sim} \mathcal{N}(0, 5^2), \quad \varepsilon_j \overset{\text{iid}}{\sim} \mathcal{N}(0,\, 0.5^2).$$

Data were split 80/20 into training and test sets, and the network was trained in mini-batches of size 32. The bias estimator $\hat{\lambda}$ was fit by Adam at learning rate 10^{-2} for up to 1000 epochs with early-stopping patience 50 on the gradient-matching loss. For each $(\lambda_1, \lambda_2, \beta)$ in a grid search, we

repeated the pipeline on independent synthetic datasets, resampling X, β, ε, and the train/test split with distinct random seeds (ten seeds per configuration). In Figure 3 we plot the mean estimated $\hat{\lambda}_i$ against the true λ_i with ± 1 standard error bars across replicates, so that variance across runs is visible alongside any systematic bias.

We first attempted elastic net regularization (combining ℓ_1 and ℓ_2 penalties), but it failed to recover the true parameters. This is likely due to the non-smoothness of the ℓ_1 norm, which hinders gradient-based optimization and biases solutions toward zero.

Instead, we applied a smoothed ℓ_1 penalty, related to the Huber loss:

$$\mathcal{R}(\theta, \beta) = \begin{cases} \frac{1}{2\beta}\theta^2 & \text{if } |\theta| < \beta \\ |\theta| - \frac{1}{2}\beta & \text{otherwise} \end{cases},$$

with smoothing parameter $\beta = 0.001$ and ℓ_1 penalty weight set to 1.0.

The gradient is

$$\frac{d\mathcal{R}}{d\theta} = \begin{cases} \dfrac{\theta}{\beta}, & |\theta| < \beta, \\ \operatorname{sign}(\theta), & |\theta| \geq \beta. \end{cases}$$

For $|\theta| < \beta$, the behavior is purely quadratic—identical to an ℓ_2 penalty of strength $1/\beta$—whereas for $|\theta| \geq \beta$ it is linear (ℓ_1). When the coefficient λ_1 on this penalty is very large (and β small), nearly all θ lie in the quadratic region, so $\lambda_1 \mathcal{R}(\theta) \approx \frac{\lambda_1}{2\beta}\theta^2$, i.e. an effective ℓ_2 penalty of strength $\frac{\lambda_1}{\beta}$, which produces the same severe under-fitting characteristic of heavy ℓ_2 regularization. Therefore when the penalty is large and many θ are close to zero the λ_1, λ_2 are not identifiable.

Figure 3 summarizes elastic-net recovery at $\beta = 10^{-3}$ on a 6×6 grid of true (λ_1, λ_2) with 10 independent dataset draws per cell. Each panel plots the estimated $\hat{\lambda}_i$ against the true λ_i on log–log axes, with one line per value of the other true penalty and error bars giving ± 1 standard error across seeds; the dashed diagonal marks perfect recovery. For large penalty values, smoothing near zero makes estimation inaccurate because the true parameters are not identifiable when many θ are small.

H.3 OLS early-stopping experiments

All OLS panels in Figure 4 use a shared noisy data-generating process. We draw $X \in \mathbb{R}^{1000 \times 10}$ with iid standard normal entries, draw coefficients $\theta_i \overset{\text{iid}}{\sim} \mathcal{N}(0, 3^2)$, and observe

$$y = X\theta + \varepsilon, \qquad \varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, 1).$$

The observation noise means that the empirical OLS minimizer is an unbiased but sample-specific estimate of the coefficient vector, so the trained weights need not coincide exactly with the ground truth even when the effective regularizer reproduces the observed-data solution.

For the single-endpoint diagonal estimate and the scalar λ_t trace, we use seed 56 for the shared draw of (X, θ, ε). The diagonal estimate (Panel C) trains full-batch GD from zero for up to 500 epochs with early-stopping patience 5 and no minimum improvement threshold (i.e. any decrease counts as improvement). The scalar trace (Panel E) evaluates a single deterministic GD trajectory at checkpoints $t \in \{1, 2, 5, 10, 20, 50, 100, 150, 200, 300, 500, 1000\}$ and fits one ridge coefficient at each checkpoint by gradient matching with Adam (learning rate 0.05, maximum $30{,}000$ epochs, patience 2000). The GD trajectory for Panel E is distinct from the early-stopped model in Panels A–C; it runs for exactly the number of steps needed to reach $t = 1000$ without early stopping, and the bias estimator at each checkpoint is independently early-stopped.

For the full symmetric-matrix estimator, we hold the design matrix X fixed and form 10 independent endpoint pools of 100 endpoints each, with a newly drawn coefficient vector and independent observation noise per endpoint.

Panel B (fixed iterate). To ensure the stacked system $\Lambda^{(t)}\hat{\theta}_k = -\nabla L_k(\hat{\theta}_k)$ has a single well-defined theoretical target $\Lambda^{(t)}$, all endpoints in the Panel B pools are stopped at the same fixed iterate t equal to the early-stopping epoch of the single-endpoint run from Panels A and C. Each endpoint runs full-batch GD from zero for up to 2000 epochs; the iterate at the shared t is extracted without early stopping.

Panel D (variable early stopping). To demonstrate recovery in a more realistic and challenging setting, Panel D uses per-endpoint early stopping: each endpoint runs full-batch GD from zero for up to 2000 epochs with patience 5 and no minimum improvement threshold, and its individual iterate count t_k is determined by when its own training loss plateaus. Because t_k varies across endpoints, the theoretical target used to evaluate the distance $\|\hat{\Lambda}_m^{(t_k)} - \Lambda^{(t_k)}\|$ is the per-pool median of the endpoint-specific matrices $\Lambda^{(t_k)}$; the variance in t_k across endpoints within a pool is small in practice, making this a mild approximation. The smooth decay in Panel D (with no identifiability cliff at $m = p$) reflects the richer signal in variable-stop endpoints compared to a fixed-t setup.

Figure 4E uses the same noisy OLS data-generating process as above, and we trained the regression model using full-batch gradient descent with step size $\eta = 10^{-2}$ for the normalized squared loss, evaluating $\hat{\theta}^{(t)}$ at $t \in \{1, 2, 5, 10, 20, 50, 100, 150, 200, 300, 500, 1000\}$. At each t we fit a single-parameter ridge bias by gradient matching using Adam (learning rate 0.05, up to 30,000 epochs, early-stopping patience 2000). The figure shows a single trajectory (not averaged).

H.4 Implicit bias of dropout

For Figure 5, we train fresh ReLU classifiers on MNIST with dropout rates $p \in \{0, 0.05, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50\}$, depths $d \in \{3, 5\}$ (number of hidden layers), and widths $w \in \{128, 256, 512\}$, with ten seeds per configuration (480 models). Dropout follows each hidden ReLU except the last; batch normalization and explicit weight decay are disabled. We retain the same fixed 54,000-example subset of the MNIST training set across runs; the remaining 6,000 examples and the test set are not used for training, stopping, or coefficient estimation.

Training and stopping. We use shuffled mini-batch SGD with momentum 0.9, batch size 2048, and initial learning rate 0.05. The stopping statistic is the nonoverlapping 100-epoch mean of sample-weighted training cross-entropy, with dropout active. An improvement must exceed 10^{-7} in absolute loss units; the relative threshold is zero. After 500 epochs without such improvement, we reduce the learning rate tenfold, down to 0.0005, resetting the best-loss reference and patience after each reduction. At the minimum learning rate, we stop after 1,000 epochs without improvement. The cap is 100,000 epochs; all 480 models stopped by the loss rule, at 4,800–24,200 epochs (median 10,100). No gradient diagnostic or coefficient sign enters stopping.

Closed-form ridge fit. At each stopping endpoint $\hat{\theta}$, let $g = \nabla_\theta \mathcal{L}_{\text{clean}}(\hat{\theta})$ be the mean cross-entropy gradient over all 54,000 training examples with dropout disabled. For $\mathcal{R}(\theta, \lambda) = \lambda\|\theta\|_2^2$, including weights and biases, we compute

$$\hat{\lambda}_{\text{signed}} = -\frac{\langle g, 2\hat{\theta}\rangle}{\|2\hat{\theta}\|_2^2}, \qquad \hat{\lambda} = \max(0, \hat{\lambda}_{\text{signed}}).$$

Model arithmetic is float32; sample-weighted gradient accumulation and scalar products are float64. No iterative coefficient optimizer is used. Signed estimates are retained; all happened to be positive. Figure colors show R^2_{grad} from (5), not raw fitting loss or prediction accuracy.

Pilot robustness and retained trajectories. Before expanding the grid, we extended seven seed-123 pilots: depth 3/width 256 at dropout 0, 0.3, and 0.5, plus the four remaining architectures at dropout 0.5. Each continued at the terminal learning rate for at least as many additional epochs as its stopping duration. The predeclared practical check requires either both signed coefficients within $[-10^{-7}, 10^{-7}]$ or relative change at most 5%. Six pairs met the near-zero criterion, with relative changes up to 17.66%; the remaining pair changed by 4.30%. This is not a uniform relative-accuracy or stationarity guarantee. The figure uses the original stopping endpoints, not the extended ones. Initial, every-1,000-epoch, stopping, and pilot-extension checkpoints are retained. Zero-dropout controls are neither removed nor baseline-subtracted; the endpoint fit can reflect residual optimization gradients even when the direct dropout contribution is zero.

H.4.1 Validation by explicit replacement

We examined whether the estimated regularizer could reproduce the behavior of a dropout-trained model when applied explicitly during training without dropout. We used the ten depth-3, width-256

MNIST classifiers trained with dropout rate 0.3 in Figure 5. For each seed, we restarted from that model's saved initial weights and trained without dropout, either with no explicit penalty or with the fitted ridge penalty $\hat{\lambda}_s\|\theta\|_2^2$, including weights and biases. Each coefficient was taken directly from its corresponding dropout endpoint, without pooling, baseline subtraction, or tuning; these coefficients ranged from 4.29×10^{-10} to 1.13×10^{-9}.

The replacements used the same training split, initialization, mini-batch SGD settings, precision, learning-rate schedule, and training-loss stopping rule as the final dropout grid described above. The explicit penalty contributed to the parameter updates, while training cross-entropy alone controlled learning-rate reductions and stopping. Thus, the stopping criterion was shared, although the stopping epochs could differ. The no-penalty controls reproduced the original zero-dropout endpoints exactly, including their parameter values. Test data were used only for evaluation after training, not to select coefficients or stopping points.

Median test accuracy was 98.345% for the dropout targets, compared with 97.86% without dropout and 97.85% with the fitted ridge penalty. The ridge replacements had lower test accuracy than their paired dropout targets for all ten seeds, with a mean shortfall of 0.508 percentage points. They disagreed with the target on a median of 1.655% of test predictions, although all models correctly classified every training example. Thus, the fitted scalar ridge penalty did not recover the original model's test performance or predictions in this configuration. This is a concrete example in which an estimated regularizer can summarize a component of the endpoint gradient without providing a sufficient replacement for the training procedure. Together with the low gradient-fit R^2_{grad}, this result illustrates the limits of a scalar ridge description of dropout's complex, data-dependent effects, while leaving the fitted projection useful for interpreting variation across training configurations.

H.5 Recovering Barrett implicit gradient regularization

This appendix details the known-coefficient recovery experiment for the implicit gradient regularizer derived by Barrett and Dherin [5]. The goal is not to revalidate all of Barrett and Dherin's empirical claims about generalization. Instead, we use their derivation as a calibration target: if our trajectory-gradient-matching estimator is working, it should recover the coefficient $\lambda = \eta p/4$ from observed deviations between discrete gradient descent and gradient flow.

Relation to Equation 4. The trajectory-deviation estimator in Eq. 4 compares an observed update to an unregularized reference trajectory. In the Barrett–Dherin setting, the observed update is one full-batch GD/Euler step, while the relevant unregularized reference is the continuous gradient flow of L over the same elapsed time η, not the linearized displacement $-\eta\nabla L(\theta)$. Comparing directly to $-\eta\nabla L(\theta)$ would give zero residual for full-batch GD; using the gradient-flow displacement instead isolates the finite-step backward-error term that corresponds to implicit gradient regularization.

Background and notation. For a loss $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$ with gradient $g(\theta) := \nabla\mathcal{L}(\theta)$ and Hessian $H(\theta) := \nabla^2\mathcal{L}(\theta)$, discrete gradient descent with step size η iterates $\theta_{k+1} = \theta_k - \eta\, g(\theta_k)$. Barrett and Dherin [5] apply backward error analysis to this Euler integration of the continuous gradient flow $\dot{\theta} = -g(\theta)$ and show that, for sufficiently smooth $\mathcal{L}$ and sufficiently small η, the discrete iterates lie close to the *modified* gradient flow

$$\dot{\theta} \;=\; -g(\theta) \;-\; \tfrac{\eta}{2}\, H(\theta)\, g(\theta) \;+\; O(\eta^2) \;=\; -\nabla\widetilde{\mathcal{L}}(\theta) \;+\; O(\eta^2), \qquad \widetilde{\mathcal{L}}(\theta) \;=\; \mathcal{L}(\theta) + \lambda\, R_{IG}(\theta),$$

where $\lambda = \eta p/4$ and $R_{IG}(\theta) = \|g(\theta)\|_2^2/p$. The implicit regularizer's gradient is $\nabla\mathcal{R}(\theta,\lambda) = (2\lambda/p)\, H(\theta)g(\theta) = (\eta/2)\, H(\theta)g(\theta)$ at Barrett's identification of λ.

A trajectory-based estimator for λ. Trained neural networks are not stationary points of the augmented loss $\widetilde{\mathcal{L}}$, so the static stationarity condition $g + \nabla\mathcal{R} = 0$ used elsewhere in this paper does not hold meaningfully along the trajectory. Instead, we estimate λ from the trajectory-deviation form. Fix a parameter vector θ and compare a single full-batch GD step

$$\Delta\theta_{\mathrm{GD}} \;=\; -\eta\, g(\theta)$$

to the displacement produced by the *true* continuous gradient flow of $\mathcal{L}$ over elapsed time η,

$$\Delta\theta_{\mathrm{flow}} \;=\; \varphi_\eta(\theta) - \theta, \qquad \varphi_t(\theta) \;:=\; \text{solution at time } t \text{ of } \dot{\theta}(s) = -g(\theta(s)),\ \theta(0) = \theta.$$

A Taylor expansion of φ_η gives $\Delta\theta_{\text{flow}} = -\eta g + (\eta^2/2)\, Hg + O(\eta^3)$, so the per-unit-time discrepancy

$$\mathcal{T}(\theta, \eta) \;:=\; \frac{\Delta\theta_{\text{flow}} - \Delta\theta_{\text{GD}}}{\eta} \;=\; \frac{\eta}{2}\, H(\theta)\, g(\theta) \;+\; O(\eta^2)$$

matches $\nabla\mathcal{R}(\theta, \lambda)$ exactly when $\lambda = \eta p/4$. Given observations $\{(H(\theta_t)g(\theta_t),\, \mathcal{T}(\theta_t, \eta))\}_{t=1}^{T}$ at parameter values θ_t visited along a probe trajectory, the scalar least-squares fit admits a closed form:

$$\widehat{\frac{\lambda}{p}} \;=\; \arg\min_{c} \sum_{t=1}^{T} \big\| \mathcal{T}(\theta_t, \eta) - 2c\, H(\theta_t) g(\theta_t) \big\|_2^2 \;=\; \tfrac{1}{2}\, \frac{\sum_t \big\langle H(\theta_t) g(\theta_t),\, \mathcal{T}(\theta_t, \eta) \big\rangle}{\sum_t \big\| H(\theta_t) g(\theta_t) \big\|_2^2}. \tag{7}$$

Computing the reference flow. We compute $\Delta\theta_{\text{flow}}$ with classical fourth-order Runge–Kutta integration of $\dot\theta = -g(\theta)$, rather than by taking smaller Euler steps. Sub-stepped Euler would introduce the same first-order backward-error term we are trying to estimate, so it would partially build the target into the numerical reference. RK4 gives a high-accuracy approximation to the continuous gradient flow while preserving the interpretation of $\mathcal{T}$ as the discrepancy between one discrete GD step and the corresponding continuous-time flow segment.

Experimental protocol for Figure 6. We train five-hidden-layer MLPs on MNIST with hidden width $w \in \{50, 100, 200, 400, 800, 1600\}$ and step size $\eta \in \{5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}, 5 \times 10^{-1}\}$. We use $\tanh$ activations rather than ReLU activations because the estimator for this regularizer requires $H(\theta)g(\theta)$; equivalently, it differentiates the gradient of the gradient-penalty regularizer, so the loss must be twice differentiable in the parameters. Training uses mini-batch stochastic gradient descent with batch size 128 on a 10,000-sample MNIST training subset, with a 1,000-sample held-out validation subset (drawn disjointly from the MNIST training split) and a 5,000-sample test subset, with no weight decay, dropout, momentum, or learning-rate schedule.

We evaluate each model at the checkpoint with highest *validation* accuracy among checkpoints that fit the training data. We use validation rather than test accuracy for checkpoint selection so that the test accuracy reported in Figure 6 (Middle) is independent of the selection criterion. Because the smooth $\tanh$ networks and reduced training subset do not always reach exact interpolation within the computational budget, we use a relaxed train-accuracy threshold of 90% and exclude runs that fail to reach it. At the selected snapshot $\theta^\star$, we compute $R_{IG}(\theta^\star) = \|g(\theta^\star)\|_2^2/p$ and estimate λ by running $T = 5$ additional full-batch probe steps from $\theta^\star$, collecting $(H(\theta_t)g(\theta_t),\, \mathcal{T}(\theta_t, \eta))$ at each step. The Hessian-vector product is computed by double backpropagation, and each RK4 reference uses $k = 10$ substeps. For computational tractability, the probe loss for R_{IG} and $\hat\lambda$ is evaluated on a fixed 2,048-example training subset.

Caveats on the per-step estimator. The identity $\mathcal{T} = (\eta/2)\, Hg + O(\eta^2)$ is a local Taylor/backward-error prediction under smoothness of $\mathcal{L}$. Thus the recovery panel in Figure 6 should be read as a calibration check for our trajectory estimator against a known analytic coefficient, not as an independent proof that IGR explains test accuracy. The left and middle panels retain the qualitative Barrett Figure 2 comparisons between $\hat\lambda$, R_{IG}, and test accuracy; the right panel is the direct methodological check, plotting $\hat\lambda$ against $\eta p/4$.

I Compute resources

We summarize the computational resources used for the experiments below. Experimental protocols, including data generation, model architectures, optimization, and stopping criteria, are described in Appendix H.3 and the other experiment-specific sections. The times reported here are allocated compute budgets, not measured run times or minimum hardware requirements. Code and instructions for running the experiments are available in the accompanying GitHub repository.

Hardware. Experiments used CPU servers with 64 x86_64 cores and 1 TB of RAM, with NVIDIA L40S GPUs for neural-network training. Each GPU training job used a single GPU; the per-job CPU and host-memory allocations are given below. The full server resources were shared across jobs.

Linear regression. The OLS early-stopping and bootstrap experiments used 10-dimensional linear regression and ran on CPUs. Matrix-recovery and bootstrap-recovery jobs were allocated 2 CPUs, 8 GB of RAM, and one hour each. The λ-versus-epochs experiment was allocated 2 CPUs, 16 GB of RAM, and four hours. These experiments include single-endpoint recovery at $\sigma = 1$ and $\sigma = 10$, fixed-iterate and variable-stop recovery with 100 endpoints in each of five pools, and bootstrap recovery with 100 resamples in each of ten pools, together with the noise-level sweep in Appendix H.3.

Implicit gradient regularization. The experiments in Appendix H.5 used one GPU, 4 CPUs, and 32 GB of RAM per job. The width and step-size sweep comprised six widths and seven step sizes, with 50 epochs of mini-batch SGD on the MNIST subset. Each setting also used five full-batch probe steps, with ten fourth-order Runge–Kutta (RK4) substeps per probe; the sweep was allocated four hours. The four longer trajectory experiments used synthetic data at $\eta \in \{0.01, 0.03\}$ for 50 gradient-descent steps, and MNIST with $\tanh$ and ReLU activations at $\eta = 0.01$ for 30 steps. Each was allocated six hours to accommodate Hessian–gradient products and the RK4 reference flow used by the trajectory-deviation estimator.

Elastic-net recovery. The elastic-net experiment comprised a 6×6 grid of (λ_1, λ_2) values with ten seeds, for 360 runs in total. Runs were distributed across eight parallel workers, each allocated one GPU, 4 CPUs, 32 GB of RAM, and four hours to process its assigned runs.

Dropout and ridge replacement. Figure 5 uses 480 models: two depths, three widths, eight dropout rates, and ten seeds. Each model was trained with one GPU, 2 CPUs, and 24 GB of RAM, with a four-hour allocation. Seven pilot runs were also extended to assess the robustness of the estimated coefficient. The training-duration analysis in Figure 8 uses estimates at checkpoints from the same training runs and requires no additional training. The matched replacement experiment (Appendix H.4.1) added 20 runs: for each of ten seeds, one model trained with the fitted ridge penalty and one without that penalty. Each replacement run was allocated one GPU, 2 CPUs, 24 GB of RAM, and one hour.

J Scaling to larger and more complex systems

Our experiments are a first step toward studying implicit regularization in contemporary deep learning systems. Applying the framework to CNNs or large language models requires care in choosing the candidate regularizer family and isolating the effects of different training mechanisms. Each new architecture or training technique could be a substantial study in its own right. Here we describe the challenges and considerations for extending this work.

Model size and estimation. The size of a model, strictly speaking, is not the only limiting factor. Our approach regresses the loss gradient, a p-dimensional vector, on the gradients of r candidate regularizers. For a fixed family, increasing p can provide more estimating equations; the relevant statistical quantities are the dimension r, the conditioning of the candidate gradients, and the effective parameter count p_{eff} described in Appendix B. Correlated errors can limit the information gained from additional parameters. Thus parameter count alone does not determine how much training data or how many endpoints are needed. Larger models also increase the cost of computing gradients, and obtaining additional training endpoints can be expensive, even though loss gradients can be computed without storing the full Jacobian (Appendix D).

Explicit, implicit, and interacting effects. Explicit regularization enters the training objective directly; implicit regularization arises through architectural or algorithmic choices that affect the updates or selected solution. Our estimates describe an effective regularizer conditional on the training configuration. Attributing this regularizer to a particular component requires controlled ablations, since its effect may depend on other components. Such interactions can themselves be modeled by including candidate regularizers representing joint effects and fitting them across configurations in which the relevant components are varied.

Disentangling modern training procedures. Large language models may involve multiple datasets and varying objectives during pre-training and post-training, in addition to explicit regu-

larization and many architecture and optimizer modifications. Studying these effects requires accounting for known explicit penalties, isolating a data regime and objective, and conducting ablation studies on mechanisms that may induce implicit regularization. Similar considerations apply to CNNs and vision transformers: the challenge is to distinguish the effects of the components used together during training. Access to model weights alone does not necessarily provide the training data, objective, or optimizer state needed for these comparisons. The degree of access to the training environment therefore affects which questions can be studied with an open-weight model.

Choosing and transferring regularizers. Theoretical work remains important for proposing mathematical models that capture the effects of training procedures. There is nuance and skill in finding simple descriptions of complex phenomena, and demonstrating a novel implicit bias requires both an informative candidate family and experiments that isolate the effect of interest. Transferring an estimated bias between architectures poses a further challenge: a regularizer expressed in terms of weights requires a clear map between their parameterizations. Architectural choices such as parameter-free skip connections offer a more tractable setting because models with and without the mechanism can share the same weight coordinates. This could allow controlled experiments asking whether an explicit regularizer reproduces the effects of the architectural choice. Developing and validating such extensions remains future work.

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: The abstract and introduction claim an empirical gradient-matching framework for estimating implicit regularization, with validation on explicit penalties, early stopping, dropout, and step-size-induced regularization. These claims are matched by the framework in Section 2, the experiments in Section 3, and the conclusions in Section 4.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Section 4 discusses limitations including effective-sample-size constraints, collinearity and identifiability of regularizer gradients, difficulty distinguishing regularizers with similar gradients, and cases where an implicit effect may not be sensibly replaced by an explicit regularizer.

   Guidelines:

   - The answer [N/A] 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: The assumptions behind the gradient-matching framework are stated in Section 2, and the formal results are accompanied by assumptions and proofs in Appendices A–C.

   Guidelines:

   - The answer [N/A] 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: The paper gives the estimator equations and Appendix H provides the main experimental protocols, including data-generating processes, MNIST splits and subsets, architectures, optimizers, learning rates, batch sizes, epoch limits, early-stopping rules, checkpoint choices, and seed counts. The implementation and reproduction workflow are publicly available at `https://github.com/jrudoler/estimating-implicit-regularization`; see the code and data availability paragraph in Appendix H.

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: The public repository at `https://github.com/jrudoler/estimating-implicit-regularization` includes the implementation, dependency/workflow information, and scripts for reproducing the main experimental results. The experiments use synthetic data generated by the scripts and MNIST, with data access and preprocessing described in Appendix H and the repository documentation.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix H specifies the training and test details for the main figures, including synthetic-data distributions, train/test splits, MNIST subsets, model architectures, dropout/depth/width grids, optimizers, learning rates, batch sizes, epoch limits, early-stopping criteria, and estimator-fitting details.

   Guidelines:

   - The answer [N/A] 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: The experiments report uncertainty or replicate-level variability: Fig. 3 uses means with ± 1 standard-error bars across seeds, Fig. 4D reports a 95% confidence band, and Fig. 5 plots individual-seed estimates with 10 seeds per configuration. The paper does not use formal hypothesis tests, since several experiments are calibration or recovery checks against known explicit or analytic targets.

   Guidelines:

   - The answer [N/A] 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: The paper reports model sizes, batch sizes, epoch limits, and dataset/subset sizes. Appendix I discusses the compute resources used for the experiments, which were run on an internal cluster with GPU.

   Guidelines:

   - The answer [N/A] 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: The research described is methodological and uses synthetic data and MNIST experiments; it does not describe human-subject data, private or sensitive data, or high-risk model or dataset releases.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The paper discusses positive scientific and practical impacts, such as giving practitioners a cheaper diagnostic for understanding training configurations as compute budgets scale. It does not however discuss negative societal impacts or misuse scenarios – the method introduced by the paper is itself a tool for furthering understanding of neural networks and their inner workings. As such it has potential to accelerate the development of neural networks and thereby shares the many risks of advancing AI, but none of those are in any way specific to our work.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models or datasets; the experiments use synthetic data and MNIST, and the contribution is a methodological estimator rather than a deployable model or scraped dataset with high misuse risk.

Guidelines:

- The answer [N/A] 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: Appendix H credits MNIST and gives its dataset-card license (MIT), together with the BSD 3-Clause licenses and upstream license URLs for PyTorch and TorchVision. MNIST is downloaded from upstream rather than redistributed as a new dataset.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: The released research code is documented at `https://github.com/jrudoler/estimating-implicit-regularization`, including environment setup, workflow commands, and experiment descriptions. Appendix H links to the release. No new dataset or model checkpoint is introduced as a separate asset.

    Guidelines:

    - The answer [N/A] 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: [N/A]

Justification: The paper does not include crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The paper does not include human-subject research, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development and experiments do not involve LLMs as an important, original, or non-standard component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.