A DUAL REPRESENTATION OF INFLUENCE FUNCTIONS FOR LINEARIZABLE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we present a dual representation of influence functions, whose computational complexity scales with dataset size rather than model size. Both analytically and experimentally, we show that this representation can be an efficient alternative to the original influence functions for estimating changes in parameters, model outputs and loss due to data point removal, when model size is large relative to dataset size, or when evaluating the original influence functions in parameter space is infeasible. The dual representation, however, is limited to linearizable models, which are models whose behavior can be approximated by their linearizations throughout training, and requires materializing a matrix, whose size grows with the product of model output dimension and dataset size.

1 INTRODUCTION

Interpretability and privacy are critical considerations in many machine learning applications. However, many modern machine learning models, particularly deep neural networks, lack transparency and may inadvertently memorize undesirable information or expose sensitive information to users. This creates challenges for understanding model behavior, and raises concerns about the reliability of model predictions and the risk of exposing private information. As a result, there is growing interest in developing methods that improve interpretability and provide mechanisms that remove the influence of sensitive or undesirable information from trained models.

Influence functions, originally developed in robust statistics (Hampel, 1974) and later introduced to machine learning by Koh & Liang (2017), aim to address these issues by providing a way of estimating how model parameters and parameter-dependent functions change when training data points are perturbed. This ability has made influence functions useful for understanding model behavior, where they can be used to identify the training data points whose presence is estimated to have the largest effect on a particular prediction or loss, as well as for machine unlearning (Nguyen et al., 2025), which is an emerging field that aims to efficiently remove the influence of specific data points from a trained model so that the resulting model behaves as if it had been retrained from scratch without those data points. Despite being successfully used to unlearn and understand the behavior of small models (Koh & Liang, 2017; Guo et al., 2020), influence functions cannot be readily scaled to large models due to the need to compute an inverse Hessian vector product (IHVP), whose computational complexity grows with the number of model parameters and the cost of evaluating model.

Due to this increasing computational complexity, exact IHVP evaluation can become computationlly infeasible for large models, or, even when it remains feasible, its cost can eventually exceed the cost of retraining the model, eliminating the computational advantage of using influence functions. The inability to evaluate the IHVP both quickly and exactly for large models has led to sustanined interest in scaling influence functions by approximating the IHVP, either directly or through an approximation of the Hessian matrix (Koh & Liang, 2017; Schioppa et al., 2022; Grosse et al., 2023). However, these approaches can reduce the reliability of influence functions because they introduce additional approximation error into a quantity that is already an approximation. The resulting error therefore consists of both the intrinsic approximation error of the influence function and the additional error introduced by approximating the IHVP.

Given the current state of influence function research, the use of influence functions thus faces a dilemma. On the one hand, obtaining reliable influence function estimates requires evaluating

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the IHVP exactly, but exact evaluation is only practical for relatively small models. On the other hand, scaling influence functions to larger models generally requires approximating the IHVP, which can introduce additional approximation error and reduce the reliability of the resulting influence function estimates. Motivated by this dilemma, we investigate whether influence functions can be re-represented in a way that makes them inherently more scalable, in other words, whether a new representation can be found that allows influence functions to be evaluated both quickly and exactly for large models without approximating the IHVP. Inspired by the dual formulation of linear regression, in this work, we introduce a dual representation of the influence functions for linearizable models, which shifts the computational complexity of IHVP from scaling with model size to scaling with dataset size. We show that the dual representation can be a more efficient alternative to the original representation when model size is large relative to dataset size, and that it provides a means of estimating changes in model outputs and loss due to data point removal for models with infinitely many parameters.

2 PRELIMINARIES

This section introduces our notations and provides conceptual background needed for the development of the dual representation of influence functions. The derivations of the results in this section are provided in Appendix B.

Let $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ be a training dataset of input-output pairs $(\mathbf{x}, \mathbf{y})$, where $\mathcal{X} \subseteq \mathbb{R}^{d_{\text{in}}}$ and $\mathcal{Y} \subseteq \mathbb{R}^{d_{\text{out}}}$ denote input and output spaces. Let $f : \mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\boldsymbol{\theta}}} \to \mathbb{R}^{d_{\text{out}}}$ be a parameterized machine learning model, and consider the regularized empirical risk over $\mathcal{D}$

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) \coloneqq \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \left(\ell(f(\mathbf{x}, \boldsymbol{\theta}), \mathbf{y}) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 \right),$$

where $\lambda > 0$ denotes a regularization constant and $\ell : \mathbb{R}^{d_{\text{out}}} \times \mathbb{R}^{d_{\text{out}}} \to \mathbb{R}_{\geq 0}$ denotes a loss function. When ℓ is convex in $\boldsymbol{\theta}$, $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$ is strongly convex and has a unique minimizer $\boldsymbol{\theta}^\star \coloneqq \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$ for any dataset $\mathcal{D}$. Let $\mathbf{X} \in \mathbb{R}^{|\mathcal{D}|\times d_{\text{in}}}$ denotes the feature matrix of $\mathcal{D}$, the gradient and Hessian of such empirical risk with respect to vectorized model outputs $f(\mathbf{X}, \boldsymbol{\theta}) \coloneqq \text{vec}(\{f(\mathbf{x}, \boldsymbol{\theta})\}_{\mathbf{x}\in\mathcal{D}}) \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|}$ are denoted as $\nabla_{f(\mathbf{X},\boldsymbol{\theta})}\mathcal{L}_{\mathcal{D}}$ and $\nabla^2_{f(\mathbf{X},\boldsymbol{\theta})}\mathcal{L}_{\mathcal{D}}$ respectively.

2.1 INFLUENCE FUNCTIONS

Let $\mathcal{D}_f$ denote a forget dataset, which contains data points that need to be deleted from $\mathcal{D}$ and whose influence must be removed from $\boldsymbol{\theta}^\star$. The remaining data points form a retain dataset, which is denoted as $\mathcal{D}_r$. Assume, without loss of generality, that data points in $\mathcal{D}_f$ are the first $|\mathcal{D}_f|$ data points in $\mathcal{D}$. Consider an upweighted version of $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$, defined as $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, -\frac{1}{|\mathcal{D}|}) \coloneqq \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) - \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta})$, where the contributions of the data points in $\mathcal{D}_f$ are upweighted by $-\frac{1}{|\mathcal{D}|}$.

The idea of influence functions is that, instead of solving $\boldsymbol{\theta}_r^\star \coloneqq \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \mathcal{L}_{\mathcal{D}_r}(\boldsymbol{\theta})$ from scratch, one can use the first-order Taylor approximation to approximate $\boldsymbol{\theta}_r^\star$ by

$$\boldsymbol{\theta}_r^\star \approx \boldsymbol{\theta}^\star + \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\mathbf{H}_{\boldsymbol{\theta}^\star}^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta}^\star), \tag{1}$$

where $\mathbf{H}_{\boldsymbol{\theta}^\star} \coloneqq \nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star, -\frac{1}{|\mathcal{D}|}) \in \mathbb{R}^{d_{\boldsymbol{\theta}}\times d_{\boldsymbol{\theta}}}$ is the Hessian of the upweighted empirical risk, and $\mathbf{H}_{\boldsymbol{\theta}^\star}^{-1}$ may be further approximated by $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star)^{-1}$ when $\frac{|\mathcal{D}_f|}{|\mathcal{D}|}$ is small. In addition, for any differentiable function $g : \mathbb{R}^{d_{\boldsymbol{\theta}}} \to \mathbb{R}^k$, $g(\boldsymbol{\theta}_r^\star)$ can be approximated by evaluating the first-order Taylor expansion of $g(\boldsymbol{\theta})$ around $\boldsymbol{\theta}^\star$ at $\boldsymbol{\theta}_r^\star$. For example, let $\mathbf{z}_t \coloneqq (\mathbf{x}_t, \mathbf{y}_t)$ denote a test data point, the change in model output and loss at $\mathbf{z}_t$ caused by removing $\mathcal{D}_f$ can be approximated by

$$f(\mathbf{x}_t, \boldsymbol{\theta}_r^\star) - f(\mathbf{x}_t, \boldsymbol{\theta}^\star) \approx \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_t, \boldsymbol{\theta}^\star)(\boldsymbol{\theta}_r^\star - \boldsymbol{\theta}^\star) \tag{2}$$

$$\mathcal{L}_{\{\mathbf{z}_t\}}(\boldsymbol{\theta}_r^\star) - \mathcal{L}_{\{\mathbf{z}_t\}}(\boldsymbol{\theta}^\star) \approx \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\{\mathbf{z}_t\}}(\boldsymbol{\theta}^\star)^\top(\boldsymbol{\theta}_r^\star - \boldsymbol{\theta}^\star), \tag{3}$$

where $\boldsymbol{\theta}_r^\star - \boldsymbol{\theta}^\star$ is given by (1).

The computational bottleneck of (1) is the IHVP: $\mathbf{H}_{\boldsymbol{\theta}^\star}^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta}^\star)$. Since $\mathbf{H}_{\boldsymbol{\theta}^\star}$ scales with the number of model parameters, materializing it is prohibitively expensive for large models. This rules out analytical inversion or the use of standard linear system solvers that require access to the full matrix. Consequently, exact computation of the IHVP typically relies on the conjugate gradient (CG) method, which solves the linear system

$$\mathbf{H}_{\boldsymbol{\theta}^\star}\mathbf{s} = \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta}^\star) \tag{4}$$

iteratively using only a linear operator $\mathbf{v} \mapsto \mathbf{H}_{\boldsymbol{\theta}^\star}\mathbf{v}$ that evaluates the Hessian vector products (HVPs), without explicitly forming the Hessian matrix.

3 METHODOLOGY

This section derives the dual representation of influence functions for linearized models, and discusses the advantages of the dual representation. The proofs and derivations of the results in this section are provided in Appendices A and B.

3.1 A DUAL FORMULATION

Consider the linearization of $f(\cdot, \boldsymbol{\theta})$ around $\boldsymbol{\theta}'$, defined as $f^{\text{lin}}(\cdot, \boldsymbol{\theta}) \coloneqq f(\cdot, \boldsymbol{\theta}') + \nabla_{\boldsymbol{\theta}} f(\cdot, \boldsymbol{\theta}')(\boldsymbol{\theta} - \boldsymbol{\theta}')$, and suppose it is trained by minimizing the regularized empirical risk

$$\hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta}) \coloneqq \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\left(\ell(f^{\text{lin}}(\mathbf{x}, \boldsymbol{\theta}), \mathbf{y}) + \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2\right).$$

Define $\hat{\boldsymbol{\theta}}^\star \coloneqq \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$ and consider the reparameterization

$$\boldsymbol{\theta} \coloneqq \phi(\Delta\boldsymbol{\alpha}) \coloneqq \hat{\boldsymbol{\theta}}^\star + \nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)^\top \Delta\boldsymbol{\alpha},$$

where $\Delta\boldsymbol{\alpha} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|}$ and $\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}') \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|\times d_{\boldsymbol{\theta}}}$. Under the reparameterization, $f^{\text{lin}}(\cdot, \boldsymbol{\theta})$ becomes $g^{\text{lin}}(\cdot, \Delta\boldsymbol{\alpha}) \coloneqq f^{\text{lin}}(\cdot, \hat{\boldsymbol{\theta}}^\star) + \mathbf{K}(\cdot, \mathbf{X})\Delta\boldsymbol{\alpha}$ and $\hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$ becomes

$$\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}) \coloneqq \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\left(\ell(g^{\text{lin}}(\mathbf{x}, \Delta\boldsymbol{\alpha}), \mathbf{y}) + \frac{\lambda}{2}\|\phi(\Delta\boldsymbol{\alpha}) - \boldsymbol{\theta}'\|_2^2\right),$$

where $\mathbf{K}(\cdot, \cdot) \coloneqq \nabla_{\boldsymbol{\theta}} f(\cdot, \boldsymbol{\theta}')\nabla_{\boldsymbol{\theta}} f(\cdot, \boldsymbol{\theta}')^\top$ denotes the empirical Neural Tangent Kernel (NTK) (Jacot et al., 2018). The optimization problem $\min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^{d_{\text{out}}|\mathcal{D}|}} \tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha})$ is an equivalent dual formulation of $\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$. The following theorem characterizes their relationship.

Theorem 3.1. *If ℓ is convex with respect to model output, $\lambda > 0$, and $\nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)$ has full row rank, then for any nonempty $\mathcal{D}_r \subseteq \mathcal{D}$, the optimization problems: $\min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^{d_{out}|\mathcal{D}|}} \tilde{\mathcal{L}}_{\mathcal{D}_r}(\Delta\boldsymbol{\alpha})$ and $\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$, have unique minimizers $\Delta\boldsymbol{\alpha}_r^\star$ and $\hat{\boldsymbol{\theta}}_r^\star$ that are related by $\Delta\boldsymbol{\alpha}_r^\star \underset{\phi^{-1}(\cdot)}{\overset{\phi(\cdot)}{\rightleftharpoons}} \hat{\boldsymbol{\theta}}_r^\star$. Moreover, $\Delta\boldsymbol{\alpha}_r^\star$ admits the form*

$$\Delta\boldsymbol{\alpha}_r^\star \coloneqq \begin{bmatrix} \frac{1}{\lambda}\nabla_{f^{lin}(\mathbf{X}_f, \hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}} \\ -\frac{1}{\lambda}\left(\nabla_{f^{lin}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}_r^\star)}\hat{\mathcal{L}}_{\mathcal{D}_r} - \nabla_{f^{lin}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}}\right) \end{bmatrix}.$$

3.2 INFLUENCE ON $\Delta\boldsymbol{\alpha}$

The dual formulation provides an alternative way to obtain $\hat{\boldsymbol{\theta}}^\star$. Specifically, $\hat{\boldsymbol{\theta}}^\star$ can be obtained by first obtaining $\Delta\boldsymbol{\alpha}^\star = \arg\min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^{d_{\text{out}}|\mathcal{D}|}} \tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha})$, then mapping $\Delta\boldsymbol{\alpha}^\star$ back to $\hat{\boldsymbol{\theta}}^\star$ via $\phi(\cdot)$. Therefore, to study the influence of removing a subset of data points from $\mathcal{D}$ on $\hat{\boldsymbol{\theta}}^\star$, it suffices to study the influence on $\Delta\boldsymbol{\alpha}^\star$. For any nonempty $\mathcal{D}_r \subseteq \mathcal{D}$, define $\Delta\boldsymbol{\alpha}_r^\star \coloneqq$

$\arg\min_{\boldsymbol{\Delta\alpha}\in\mathbb{R}^{d_{\text{out}}|\mathcal{D}|}} \tilde{\mathcal{L}}_{\mathcal{D}_r}(\Delta\boldsymbol{\alpha})$, by following the same derivation as in Section 2.1, $\Delta\boldsymbol{\alpha}_r^\star$ can be approximated by

$$\Delta\boldsymbol{\alpha}_r^\star \approx \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\mathbf{H}_{\Delta\boldsymbol{\alpha}^\star}^{-1}\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0}), \tag{5}$$

where $\mathbf{H}_{\Delta\boldsymbol{\alpha}^\star} \coloneqq \nabla^2_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}}(\mathbf{0}, -\frac{1}{|\mathcal{D}|}) \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|\times d_{\text{out}}|\mathcal{D}|}$ and $\mathbf{H}_{\Delta\boldsymbol{\alpha}^\star}^{-1}$ may be further approximated by $\nabla^2_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}}(\mathbf{0})^{-1}$ when $\frac{|\mathcal{D}_f|}{|\mathcal{D}|}$ is small. Furthermore, let $\mathbf{K}$, $\mathbf{K}_r$ and $\mathbf{K}_f$ denote $\mathbf{K}(\mathbf{X},\mathbf{X})$, $\mathbf{K}(\mathbf{X}_r,\mathbf{X})$ and $\mathbf{K}(\mathbf{X}_f,\mathbf{X})$, respectively, the $\mathbf{H}_{\Delta\boldsymbol{\alpha}^\star}$ and $\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0})$ in (5) can be expressed as

$$\mathbf{H}_{\Delta\boldsymbol{\alpha}^\star} = \frac{|\mathcal{D}_r|}{|\mathcal{D}|}\left(\mathbf{K}_r^\top\nabla^2_{f^{\text{lin}}(\mathbf{X}_r,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}_r}\,\mathbf{K}_r + \lambda\mathbf{K}\right) \tag{6}$$

$$\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0}) = \mathbf{K}_f^\top\nabla_{f^{\text{lin}}(\mathbf{X}_f,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}_f} + \lambda\mathbf{K}\boldsymbol{\alpha}^\star, \tag{7}$$

where $\boldsymbol{\alpha}^\star \coloneqq -\frac{1}{\lambda}\nabla_{f^{\text{lin}}(\mathbf{X},\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|}$ and $\nabla^2_{f^{\text{lin}}(\mathbf{X}_r,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}_r} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}_r|\times d_{\text{out}}|\mathcal{D}_r|}$. After obtaining $\Delta\boldsymbol{\alpha}_r^\star$, the change in model output and loss at a test data point $\mathbf{z}_t$ can be approximated by

$$f^{\text{lin}}(\mathbf{x}_t,\hat{\boldsymbol{\theta}}_r^\star) - f^{\text{lin}}(\mathbf{x}_t,\hat{\boldsymbol{\theta}}^\star) \approx \mathbf{K}(\mathbf{x}_t,\mathbf{X})\Delta\boldsymbol{\alpha}_r^\star \tag{8}$$

$$\hat{\mathcal{L}}_{\{\mathbf{z}_t\}}(\hat{\boldsymbol{\theta}}_r^\star) - \hat{\mathcal{L}}_{\{\mathbf{z}_t\}}(\hat{\boldsymbol{\theta}}^\star) \approx \nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\mathbf{0})^\top\Delta\boldsymbol{\alpha}_r^\star, \tag{9}$$

where $\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\mathbf{0})$ is given by the same expression as (7), after replacing $\mathcal{D}_f$ with $\{\mathbf{z}_t\}$ and $\mathbf{X}_f$ with $\mathbf{x}_t$. Moreover, (8) and (9) can be evaluated at multiple test data points simultaneously via vectorization. Let $\mathcal{D}_t$ denote a test dataset, the change in model output at all data points in $\mathcal{D}_t$ can be computed by $\mathbf{K}_t\Delta\boldsymbol{\alpha}_r^\star$, and the change in loss at all data points in $\mathcal{D}_t$ can be computed by

$$|\mathcal{D}_t|\nabla_{f^{\text{lin}}(\mathbf{X}_t,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}_t} \odot (\mathbf{K}_t\,\Delta\boldsymbol{\alpha}_r^\star) + \lambda\boldsymbol{\alpha}^{\star\top}\mathbf{K}\,\Delta\boldsymbol{\alpha}_r^\star\mathbf{1}, \tag{10}$$

where $\mathbf{K}_t$ denotes $\mathbf{K}(\mathbf{X}_t,\mathbf{X})$ and $\mathbf{1} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}_t|}$ denotes the all-ones vector [1].

Equation (5), (8), and (9) are the $\Delta\boldsymbol{\alpha}$-space counterparts of (1), (2) and (3). Although working in the $\Delta\boldsymbol{\alpha}$-space may appear less convenient due to the additional step of mapping back to the $\boldsymbol{\theta}$-space, it nonetheless offers two advantages over working directly in $\boldsymbol{\theta}$-space.

3.3 A Reduced Linear System

It can be seen from Theorem 3.1 that, entries of $\Delta\boldsymbol{\alpha}_r^\star$ corresponding to data points in $\mathcal{D}_f$, i.e., $\Delta\boldsymbol{\alpha}_{r,f}^\star \coloneqq \frac{1}{\lambda}\nabla_{f^{\text{lin}}(\mathbf{X}_f,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}}$, are known once $\hat{\boldsymbol{\theta}}^\star$ is known. This gives rise to a computational shortcut for evaluating (5). Let $\mathbf{K}_{ij}$ denotes $\mathbf{K}(\mathbf{X}_i,\mathbf{X}_j)$, (5) can be expressed in block matrix form as

$$\begin{bmatrix}\mathbf{H}^{ff}_{\Delta\boldsymbol{\alpha}^\star} & \mathbf{H}^{fr}_{\Delta\boldsymbol{\alpha}^\star}\\ \mathbf{H}^{rf}_{\Delta\boldsymbol{\alpha}^\star} & \mathbf{H}^{rr}_{\Delta\boldsymbol{\alpha}^\star}\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{\alpha}^\star_{r,f}\\ \Delta\boldsymbol{\alpha}^\star_{r,r}\end{bmatrix} = \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\begin{bmatrix}\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0})_f\\ \nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0})_r\end{bmatrix},$$

where

$$\mathbf{H}^{ij}_{\Delta\boldsymbol{\alpha}^\star} = \frac{|\mathcal{D}_r|}{|\mathcal{D}|}\left(\mathbf{K}_{ir}\nabla^2_{f^{\text{lin}}(\mathbf{X}_r,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}_r}\,\mathbf{K}_{rj} + \lambda\mathbf{K}_{ij}\right) \tag{11}$$

$$\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0})_i = \mathbf{K}_{if}\nabla_{f^{\text{lin}}(\mathbf{X}_f,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}_f} + \lambda\mathbf{K}_i\boldsymbol{\alpha}^\star. \tag{12}$$

Since $\Delta\boldsymbol{\alpha}^\star_{r,f}$ is known and $\mathbf{H}^{rr}_{\Delta\boldsymbol{\alpha}^\star} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}_r|\times d_{\text{out}}|\mathcal{D}_r|}$ is positive definite when $\mathbf{H}_{\Delta\boldsymbol{\alpha}^\star}$ is positive definite, to find $\Delta\boldsymbol{\alpha}_r^\star$, it suffices to find $\Delta\boldsymbol{\alpha}^\star_{r,r}$ by solving

$$\mathbf{H}^{rr}_{\Delta\boldsymbol{\alpha}^\star}\Delta\boldsymbol{\alpha}^\star_{r,r} = \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0})_r - \mathbf{H}^{rf}_{\Delta\boldsymbol{\alpha}^\star}\Delta\boldsymbol{\alpha}^\star_{r,f}. \tag{13}$$

The linear system in (13) has two key properties: (i) the dimension of $\mathbf{H}^{rr}_{\Delta\boldsymbol{\alpha}^\star}$ grows with $d_{\text{out}}|\mathcal{D}_r|$ instead of $d_{\boldsymbol{\theta}}$, and (ii) the size of the system decreases as $|\mathcal{D}_f|$ increases. Therefore, depending on the values of d_{out} and $|\mathcal{D}_f|$, the system can be solved either by forming $\mathbf{H}^{rr}_{\Delta\boldsymbol{\alpha}^\star}$ explicitly and using a standard linear system solver, or by applying CG with an operator that evaluates $\mathbf{H}^{rr}_{\Delta\boldsymbol{\alpha}^\star}\mathbf{v}$. In either case, the computational cost of solving the system is independent of model size, and a larger $|\mathcal{D}_f|$ yields a smaller system, reducing the cost even further.

[1] Equation (10) can be used directly for scalar-valued model, i.e., $d_{\text{out}} = 1$. For vector-valued model, one needs to sum every d_{out} consecutive rows of the first term and change the dimension of $\mathbf{1}$ to $\mathbb{R}^{|\mathcal{D}_t|}$.

3.4 KERNEL TRICK

The $\Delta\boldsymbol{\alpha}$-space representation of influence functions allows influence functions to be evaluated using the kernel trick. This follows from observing that $\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha})$ is a special case of the more general empirical risk

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha}) \coloneqq \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \Big(\ell(h(\mathbf{x},\boldsymbol{\alpha}),\mathbf{y}) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top \boldsymbol{\Theta}(\mathbf{X},\mathbf{X})\boldsymbol{\alpha}\Big),$$

where $h(\cdot,\boldsymbol{\alpha}) \coloneqq f'(\cdot) + \boldsymbol{\Theta}(\cdot,\mathbf{X})\boldsymbol{\alpha}$ and $\boldsymbol{\Theta}(\mathbf{X},\mathbf{X}) \succ 0$ is the Gram matrix induced by kernel function $\boldsymbol{\Theta}(\cdot,\cdot)$ on $\mathcal{D}$. Specifically, if $f'(\cdot) = f(\cdot,\boldsymbol{\theta}')$ and $\boldsymbol{\Theta}(\cdot,\cdot) = \mathbf{K}(\cdot,\cdot)$, then $\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}) = \mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha}^\star + \Delta\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}^\star \coloneqq \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^{d_{\text{out}}|\mathcal{D}|}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha})$. Therefore, by replacing $f'(\cdot)$ and $\boldsymbol{\Theta}(\cdot,\cdot)$ with suitable alternatives and following the same derivation as the original influence functions, the influence of removing data points can be computed for a broad class of models of the form $h(\cdot,\boldsymbol{\alpha})$ trained by minimizing $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha}^\star + \Delta\boldsymbol{\alpha})$, or equivalently, $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha})$. This also includes models whose $\boldsymbol{\theta}$-space representation contains infinitely many parameters. For example, by replacing $\boldsymbol{\Theta}(\cdot,\cdot)$ with the analytical NTK $\mathbf{K}^\infty(\cdot,\cdot)$ (Jacot et al., 2018; Lee et al., 2019), we can compute the influence of removing data points on the loss and model outputs of infinitely wide neural networks whose NTKs admit analytical forms.

3.5 DISCUSSION

The estimates provide by the original representation of influence functions are most reliable when the empirical risk is strongly convex with respect to the model parameters, and the model is trained to convergence, i.e., $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star) = \mathbf{0}$. In addition to the conditions stated in Theorem 3.1, the reliable use of the dual representation requires $\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}^\star) = \mathbf{0}$, which can be achieved either by training $g^{\text{lin}}(\cdot,\Delta\boldsymbol{\alpha})$ to convergence, or by training $f^{\text{lin}}(\cdot,\boldsymbol{\theta})$ to convergence.

Although evaluating (5) is more computationally efficient than evaluating (1) when $d_{\text{out}}|\mathcal{D}_r| \ll d_{\boldsymbol{\theta}}$, the evaluations of (8) and (9) do not offer an obvious computational advantage over those of (2) and (3) when the NTK does not admit an analytical form. This may appear to limit the computational advantage of the dual representation for finite models, where the empirical NTK matrix must be constructed from per-example gradients. However, this concern is mitigated by the fact that, once $\Delta\boldsymbol{\alpha}_r^\star$ is obtained, it can be mapped back to $\hat{\boldsymbol{\theta}}_r^\star$ through $\phi(\cdot)$, and (2) and (3) can then be used to study the influence on model output and loss. Compared with the approach that first compute $\hat{\boldsymbol{\theta}}_r^\star$ using (1) and then applies (2) and (3), our approach is more efficient as long as the combined cost of evaluting (5) and applying the map $\phi(\cdot)$ is lower than the cost of evaluting (1). The experiment results shown in Figure 1, where we compare the time it takes to obtain an estimate of $\hat{\boldsymbol{\theta}}_r^\star$, using the two approaches, confirm that this combined cost is indeed lower in the settings considered.

Apart from linear and linearized models, the dual representation of influence functions can also be applied to nonlinear models, provided that they are linearizable. Informally, we say a parameterized model is linearizable at some reference point $\boldsymbol{\theta}'$ if the model and its linearization around $\boldsymbol{\theta}'$ produce similar training dynamics. Nonlinear models can be made approximately linearizable by training with $\hat{\mathcal{L}}_{\mathcal{D}}$ using large values of λ. This is because a nonlinear model is well approximated by its linearization around $\boldsymbol{\theta}'$ when its parameters change little from $\boldsymbol{\theta}'$ during training, and the regularization term $\lambda\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2$ discourages the parameters from deviating too far from $\boldsymbol{\theta}'$ when λ is large. Therefore, when trained with large λ, the above results may still hold even if the linearized model $f^{\text{lin}}(\cdot,\boldsymbol{\theta})$ in $\hat{\mathcal{L}}_{\mathcal{D}}$ is replaced by its nonlinear counterpart $f(\cdot,\boldsymbol{\theta})$.

4 EXPERIMENTS

This section presents empirical verification of the advantages claimed in Sections 3.3 and 3.4 for working in $\Delta\boldsymbol{\alpha}$-space and investigates how the choice of λ affects the training dynamics of nonlinear models and the effectiveness of the two representations of influence functions. All experiments are implemented in Python using JAX (Bradbury et al., 2018) and Neural Tangents (Novak et al., 2020; 2022; Hron et al., 2020; Sohl-Dickstein et al., 2020; Han et al., 2022) libraries, and all results are obtained on two NVIDIA RTX 3080 Ti GPUs. The code to reproduce the experiments in this section can be found in `https://anonymous.4open.science/r/ntk_influence`.

4.1 Implementation Details

The original and dual representations of influence functions are evaluated by solving (4) and (13), respectively, using CG. The operator $\mathbf{v} \mapsto \mathbf{H}_{\boldsymbol{\theta}^\star}\mathbf{v}$ is implemented via automatic differentiation, while the operator $\mathbf{v} \mapsto \mathbf{H}^{rr}_{\Delta\boldsymbol{\alpha}^\star}\mathbf{v}$ is implemented based on (11), where the HVP $\nabla^2_{f^{\text{lin}}(\mathbf{X}_r,\hat{\boldsymbol{\theta}}^\star)}\hat{\mathcal{L}}_{\mathcal{D}_r}\mathbf{v}$ is computed using automatic differentiation and $\mathbf{K}_{rr}\mathbf{v}$ is computed as a matrix vector product.

We assume that $\mathbf{K}$ is precomputed and stored, so that $\mathbf{K}_{rr}$ can be obtained by slicing $\mathbf{K}$. In practice, constructing $\mathbf{K}$ in advance may be advantageous, since, once it is constructed, it can be reused across different influence function calls to study the influence of removing different subsets of data points. This is because, no matter which data points we remove, the components of (13) can always be expressed as submatrices of $\mathbf{K}$, and these submatrices can be obtained by simply slicing $\mathbf{K}$. In practice, $\mathbf{K}_{rr}$ may be stored either on a single GPU or sharded across multiple GPUs to reduce per-device memory usage. In the latter case, we implement a sharded version of the matrix vector product $\mathbf{K}_{rr}\mathbf{v}$ that allows $\mathbf{K}_{rr}\mathbf{v}$ to be evaluated while keeping the shards of $\mathbf{K}_{rr}$ distributed across multiple devices.

4.2 Linearized Models

We verify the computational advantage of the dual representation of influence functions, hereafter referred to as the $\Delta\boldsymbol{\alpha}$-space method, by comparing its unlearning performance with the the original influence functions, hereafter referred to as the $\boldsymbol{\theta}$-space method, on two linearized neural networks and two datasets. For each model-dataset pair, we train the model on full dataset, remove $p \in \{10, 30, 50, 70, 90\}$ percent of the data points randomly from either all classes or a specific class of the dataset, and apply both methods for unlearning. The unlearning performance of both methods is evaluated using the following metrics:

1. **Cold-start runtime**, defined as the time required to run each method on its first execution. This includes not only the time to prepare and solve the linear systems, but also one-time overhead such as compilation, memory allocation, and other initialization costs. This metric compares unlearning speed in scenarios where unlearning is required only once. To accurately measure this runtime, for each value of p, the experiment is run in a separate process to prevent reuse of compiled functions and cached resources;
2. **Warm-start runtime**, defined as the mean and standard deviation of the runtime over five subsequent executions after initialization overhead has been incurred. This metric compares unlearning speed in scenarios where unlearning needs to be performed repeatedly on different $\mathcal{D}_f$ of the same size;
3. **Relative ℓ_2 distance** between the unlearned model parameters $\boldsymbol{\theta}_u$ and the retrained model parameters $\hat{\boldsymbol{\theta}}^\star_r$, which measures how closely the unlearned model aligns with the retrained model in parameter space. A smaller value indicates better alignment between them;
4. **Accuracy** of the unlearned models and the retrained model on $\mathcal{D}_f$, which is used to assess forgetting quality. The closer the accuracy of the unlearned model is to that of the retrained model, the stronger the indication that the unlearned model has forgotten the designated data points to a degree similar to the retrained model.

We consider the linearizations of two models: a fully connected neural network (FCNN) with three hidden layers of width 1024 and ReLU activations, and a convolutional neural network (CNN) composed of three blocks, each consisting of a convolutional layer with 128 channels and a 3×3 kernel, followed by a ReLU activation and average pooling. The FCNN and CNN are both linearized around their initialization and trained, respectively, on a subset of MNIST (LeCun et al., 2002) consisting of 200 data points from each of the 10 classes and on a subset of CIFAR-10 (Krizhevsky & Hinton, 2009) consisting of 1000 data points from each of classes 0 (airplane) and 1 (automobile). For FCNN-MNIST pair, training is performed by minimizing $\hat{\mathcal{L}}_{\mathcal{D}}$ with cross entropy loss and $\lambda = 10^{-2}$ using full-batch gradient descent, and data points are randomly removed from all classes of the constructed dataset. For the CNN-CIFAR-10 pair, training is performed by minimizing $\hat{\mathcal{L}}_{\mathcal{D}}$ with squared error loss and $\lambda = 10^{-1}$ using momentum, and data points are removed from class 0 of the constructed dataset. To highlight the effect of unlearning, we include a random perturbation baseline that perturbs the trained model parameter $\hat{\boldsymbol{\theta}}^\star$ with random noise $\boldsymbol{\epsilon} \in \mathbb{R}^{d_{\boldsymbol{\theta}}}$, where $\boldsymbol{\epsilon}$ satisfies

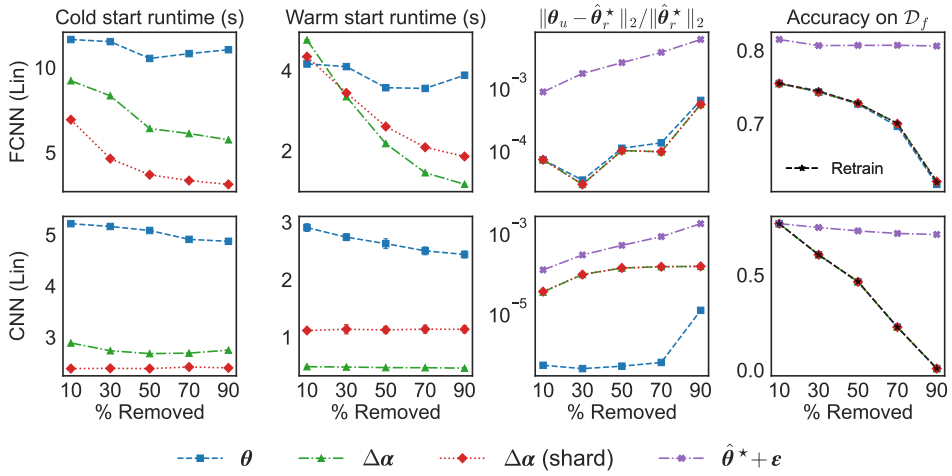


Figure 1: **Comparison of the unlearning performance of θ-space and $\Delta\alpha$-space methods on linearized FCNN and CNN trained on subsets of MNIST and CIFAR10.** The top and bottom rows correspond to the linearized FCNN and CNN, respectively. We compare the θ-space method with two implementations of the $\Delta\alpha$-space method, and with a random perturbation baseline. From left to right, we report cold-start runtime, warm-start runtime, relative parameter distance, and test accuracy as a function of the percentage of removed data. The dashed black curves in the rightmost panes indicate the accuracy of the retrained model. We observe that, when model size is large relative to dataset size, the $\Delta\alpha$-space method requires less runtime while maintaining unlearning performance comparable to that of the θ-space method.

$\|\epsilon\|_2 = \|\hat{\theta}^\star - \hat{\theta}_r^\star\|_2$. The unlearning performance of the θ-space and $\Delta\alpha$-space methods on these models is shown in Figure 1.

In both experiments, the two implementations of the $\Delta\alpha$-space method generally achieve lower runtimes than the θ-space method. These results are consistent with our expectations, since the dataset size in both experiments are small relative to the model size, i.e., $d_{\text{out}}|\mathcal{D}| \ll d_\theta$. Consequently, the cost of solving (13) is smaller than that of solving (4). In addition, the unlearning performance of the $\Delta\alpha$-space method is comparable to that of the θ-space method, which can be seen by the small relative ℓ_2 distance between $\hat{\theta}_r^\star$ and θ_u produced by both methods and by their close agreement with the retrained model in accuracy on $\mathcal{D}_f$.

4.3 Effect of λ

To investigate how λ affects the training dynamics of nonlinear models and how these dynamics compare with those of their linearizations, we simultaneously train a nonlinear model and its linearization, using the same loss and optimizer, and record their training dynamics for different values of λ.

We adopt a scalar-valued FCNN with two hidden layers of width 2048 and ReLU activations. Both the model and its linearization around initialization are trained by minimizing $\hat{\mathcal{L}}_\mathcal{D}(\theta)$ with squared error loss on a binary MNIST dataset consisting of 5000 data points from each of the classes 3 and 8, using full-batch gradient descent with a learning rate of 0.1. During training, we record the relative ℓ_2 distance between the parameters of the FCNN and its linearization, denoted by θ and θ^{lin}, respectively; the root mean squared error (RMSE) between their predictions on $\mathcal{D}_t$; the accuracy performance of both models on $\mathcal{D}_t$; and the gradient norms of the parameters of the FCNN and its linearization. These quantities, recorded over 4000 epochs of training, are summarized in Figure 2.

We observe that larger values of λ make the training dynamics of the FCNN more closely resemble that of its linearization, as evidenced by the small relative ℓ_2 distance between their parameters and small difference between their outputs throughout training. In addition, $\|\nabla_\theta \mathcal{L}_\mathcal{D}(\theta)\|_2$ approaches 0 more rapidly for larger values of λ, indicating that, when trained with $\lambda\|\theta - \theta'\|_2^2$, both the model and its linearization converge faster under stronger regularization. However, this comes at the cost of worse test accuracy, as larger values of λ lead to progressively poorer test accuracy.

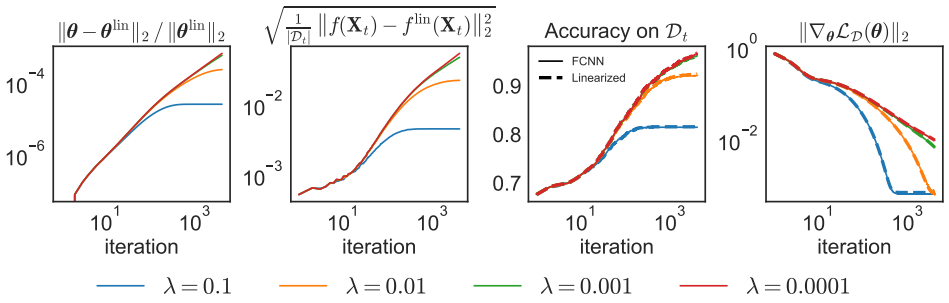


Figure 2: **Comparison of the training dynamics of a FCNN and its linearization under different values of λ.** The left two panes show the relative ℓ_2 distance between the parameters of FCNN and its linearization, and the RMSE between their predictions on $\mathcal{D}_t$. The right two panes show the accuracies of the FCNN and its linearization on $\mathcal{D}_t$, and the gradient norms of their parameters. We observe that larger values of λ make the FCNN track its linearization more closely and converge faster, but at the cost of worse test accuracy.

Since the dual representation requires the model to be linearizable, and the original representation also benefits from working with linear models, because they make the empirical risk strongly convex in $\boldsymbol{\theta}$, the choice of λ plays an important role in the effectiveness of $\boldsymbol{\theta}$-space and $\Delta\boldsymbol{\alpha}$-space methods when they are applied to nonlinear models. This sensitivity to λ is illustrated in Figure 3a, where data points from both classes are unlearned from the aforementioned FCNN, trained with different values of λ, using both methods. It can be seen that both methods produce unlearned model that more faithfully match the retrained model, when applied to the FCNN trained with larger value of λ. This behvior is expected because, as λ decreases, the model parameters leave the linearizable regime and behave increasinlgy nonlinear, violating the assumptions underlying both representations of influence functions and causing both to produce unreliable estimates.

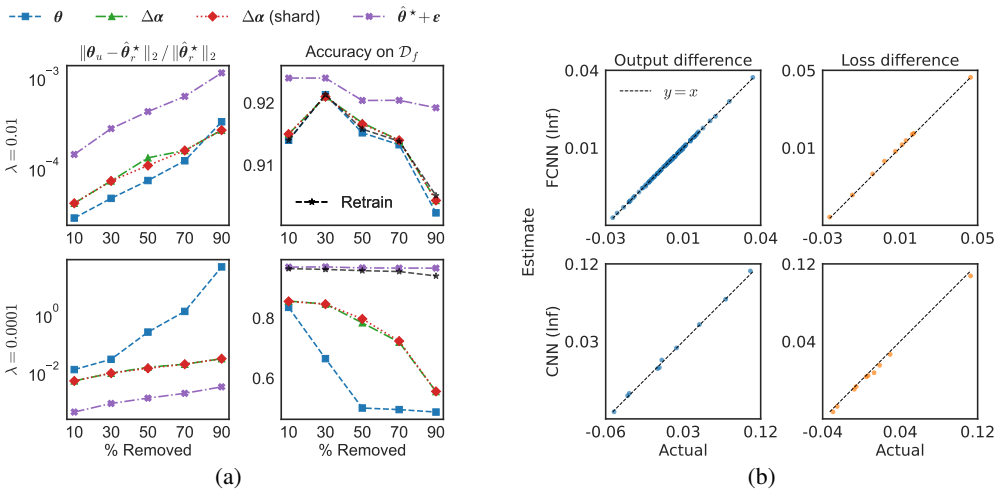
4.4 Infinitely Wide Neural Networks

We showcase the applicability of $\Delta\boldsymbol{\alpha}$-space method to infinitely wide neural networks for estimating changes in model outputs and loss induced by data point removal. We consider two infinitely wide neural networks derived from a FCNN with three hidden layers and ReLU activations, and from a CNN with three convolutional layers and ReLU activations, respectively. The infinitely wide FCNN is trained by minimizing $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha})$ with cross entropy loss and $\lambda = 10^{-1}$ on a subset of MNIST consisting of 1000 data points from each of the 10 classes, whereas the infinitely wide CNN is trained by minimizing $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\alpha})$ with squared error loss and $\lambda = 10^{-1}$ on a subset of CIFAR-10 consisting of 500 data points from each of classes 0 and 1. In both cases, optimization is performed using gradient descent with momentum.

For each model-dataset pair, the model is trained on the constructed dataset and on a retain dataset obtained by randomly removing 50% of data points from each class of the dataset, after which the actual changes in model outputs and loss due to the removal of data points are evaluated at 10 randomly selected test data points. The estimated changes in model outputs and loss are obtained by first solving (13), and then evaluating $\mathbf{K}_t \Delta\boldsymbol{\alpha}_r^\star$ and (10) at the test data points, with $\mathbf{K}(\cdot, \cdot)$ replaced by $\mathbf{K}^\infty(\cdot, \cdot)$ throughout the process. A comparison of the actual and estimated differences in model outputs and loss is summarized in Figure 3b, where, for the outputs of infinitely wide FCNN, all 10 output dimensions are shown. It can be seen that the estimates provided by $\Delta\boldsymbol{\alpha}$-space method are accurate for both infinitely wide neural networks.

5 Related Work

Our work builds on the work of Koh & Liang (2017), who applied influence functions from robust statistics to machine learning to study the effect of perturbing a training data point on model parameters, loss, and outputs. In their work, they identified the scalability issue of influence functions, namely that evaluating the IHVP using CG becomes computationally expensive as dataset size and model size increase, and consequently proposed to use a stochastic estimation method to approxi-



(a) (b)

Figure 3: **(a) Comparison of the unlearning performance of $\boldsymbol{\theta}$-space and $\Delta\boldsymbol{\alpha}$-space methods on a nonlinear model under different values of λ.** The dashed black curves indicate the accuracy of the retrained model. We observe that, when trained with larger value of λ, both $\boldsymbol{\theta}$-space and $\Delta\boldsymbol{\alpha}$-space methods produce unlearned model that more faithfully match the retrained model. **(b) Comparison of estimated and actual changes in the outputs and loss of two infinitely wide neural networks at** 10 **test data points after removing** 50% **of data points from** $\mathcal{D}$**.**

mate these quantities. However, Basu et al. (2021) later observed that the stochastic estimation of the IHVP can make influence estimates unreliable in deep learning settings. Several later works focus on improving the scalability of influence functions in $\boldsymbol{\theta}$-space by developing more efficient methods for approximating the IHVP. For example, Schioppa et al. (2022) scale up influence functions by using Arnoldi iteration to approximate the IHVP, and demonstrate applicability to vision transformers with up to 300 million parameters, while Grosse et al. (2023) employ the Eigenvalue-corrected Kronecker-Factored Approximate Curvature (EK-FAC) to approximate the IHVP, extending influence functions to large language models with up to 52 billion parameters.

6 Conclusion

We presented a dual representation of influence functions for linearizable models, which shifts the computational complexity of IHVP from scaling with model size to scaling with dataset size. We showed analytically that, when the number of model parameters is large relative to the product of model output dimension and retain dataset size, the dual representation allows the influence of removing data points on model parameters to be computed more efficiently than the original representation, without introducing any additional approximation. Furthermore, the dual representation makes it possible to analyze the influence on model outputs and loss in the limit where the number of model parameters goes to infinity. Experimental results obtained by comparing the dual and original representations of influence functions on different machine unlearning tasks further confirmed the computational advantages of the dual representation in settings where working directly in parameter space is difficult or infeasible. Nonetheless, the dual representation is limited in that it is applicable only to linearizable models and requires the computation and storage of the NTK matrix, which can make its use expensive for datasets with many data points and classes.

We believe that, when complemented by the dual representation, influence functions can be used to efficiently unlearn or understand the behavior of linear and linearized models regardless of model size. They may also be applicable to nonlinear models whose parameters remain close to their initialization, such as models obtained by fine tuning a pretrained model. More broadly, the dual representation may offer a different path to scalability in a literature where scalability has largely been pursued by developing increasingly sophisticated approximations to the IHVP, while also opening new research directions. One particularly interesting question to answer in the future is whether existing IHVP approximation techniques can be adapted to the dual representation to further reduce the computational cost of influence functions.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks:

- Assisting with writing proofs and derivations;
- Assisting with implementing experiments and plotting figures;
- Understanding unfamiliar technical concepts;
- Rephrasing sentences and checking grammar.

We did not use generative AI tools for any other tasks that require disclosure. All AI-assisted work was reviewed and verified for correctness. We take full responsibility for the final content of this work.

ETHICS STATEMENT

To the best of our knowledge, this work does not raise any questions regarding the Code of Ethics.

REPRODUCIBILITY STATEMENT

Section 4 describes the setup for each experiment, including the GPU used to obtain the results. Additional implementation details, together with instructions and code for reproducing all experiments, are available in the GitHub repository linked in the paper. Running all experiments takes approximately 1.5–2 hours. All theoretical results are accompanied by proofs or derivations, which are provided in Appendices A and B, respectively.

REFERENCES

S Basu, P Pope, and S Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*, 2021.

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

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint*, 2023.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning (ICML)*, 2020.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 1974.

Insu Han, Amir Zandieh, Jaehoon Lee, Roman Novak, Lechao Xiao, and Amin Karbasi. Fast neural kernel embeddings for general activations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning (ICML)*, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 2002.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *ACM Transactions on Intelligent Systems and Technology*, 2025.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations (ICLR)*, 2020. URL `https://github.com/google/neural-tangents`.

Roman Novak, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Fast finite width neural tangent kernel. In *International Conference on Machine Learning (ICML)*, 2022.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *AAAI Conference on Artificial Intelligence*, 2022.

Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint*, 2020.

A PROOFS

Theorem 3.1 can be proved by proving the following three propositions. When trained under the risk $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$, the difference between the parameters obtained by training on any nonempty subset of $\mathcal{D}$ and those obtained by training on $\mathcal{D}$ lies in a linear subspace of $\mathbb{R}^{d_{\boldsymbol{\theta}}}$. This observation is formalized in the following proposition.

Proposition A.1. *For any nonempty $\mathcal{D}_r \subseteq \mathcal{D}$, let $\hat{\boldsymbol{\theta}}^\star$ and $\hat{\boldsymbol{\theta}}_r^\star$ denote the model parameters obtained by solving $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$ and $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$, respectively. Then:*

$$\hat{\boldsymbol{\theta}}_r^\star - \hat{\boldsymbol{\theta}}^\star = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top \Delta\boldsymbol{\alpha}_r^\star,$$

where $\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}') \in \mathbb{R}^{d_{out}|\mathcal{D}|\times d_{\boldsymbol{\theta}}}$ and $\Delta\boldsymbol{\alpha}_r^\star \in \mathbb{R}^{d_{out}|\mathcal{D}|}$ are defined as

$$\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}') \coloneqq \begin{bmatrix} \nabla_{\boldsymbol{\theta}} f(\mathbf{x}^{(1)}, \boldsymbol{\theta}') \\ \vdots \\ \nabla_{\boldsymbol{\theta}} f(\mathbf{x}^{(|\mathcal{D}|)}, \boldsymbol{\theta}') \end{bmatrix} \quad \Delta\boldsymbol{\alpha}_r^\star \coloneqq \begin{bmatrix} \frac{1}{\lambda} \nabla_{f^{lin}(\mathbf{X}_f, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} \\ -\frac{1}{\lambda} \left(\nabla_{f^{lin}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}_r^\star)} \hat{\mathcal{L}}_{\mathcal{D}_r} - \nabla_{f^{lin}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} \right) \end{bmatrix}.$$

Proof of Proposition A.1. Since $\hat{\boldsymbol{\theta}}^\star$ and $\hat{\boldsymbol{\theta}}_r^\star$ are the minimizers of $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$ and $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$, respectively, by first-order necessary condition for optimality, they must satisfy

$$0 = \nabla_{\boldsymbol{\theta}} \hat{\mathcal{L}}_{\mathcal{D}}(\hat{\boldsymbol{\theta}}^\star) = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top \nabla_{f^{\text{lin}}(\mathbf{X}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} + \lambda(\hat{\boldsymbol{\theta}}^\star - \boldsymbol{\theta}')$$
$$0 = \nabla_{\boldsymbol{\theta}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\hat{\boldsymbol{\theta}}_r^\star) = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}_r, \boldsymbol{\theta}')^\top \nabla_{f^{\text{lin}}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}_r^\star)} \hat{\mathcal{L}}_{\mathcal{D}_r} + \lambda(\hat{\boldsymbol{\theta}}_r^\star - \boldsymbol{\theta}'),$$

where $\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}') \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|\times d_{\boldsymbol{\theta}}}$ and $\nabla_{f^{\text{lin}}(\mathbf{X}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|}$ are defined as

$$\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}') \coloneqq \begin{bmatrix} \nabla_{\boldsymbol{\theta}} f(\mathbf{x}^{(1)}, \boldsymbol{\theta}') \\ \vdots \\ \nabla_{\boldsymbol{\theta}} f(\mathbf{x}^{(|\mathcal{D}|)}, \boldsymbol{\theta}') \end{bmatrix} \quad \nabla_{f^{\text{lin}}(\mathbf{X}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} \coloneqq \frac{1}{|\mathcal{D}|} \begin{bmatrix} \nabla_f \ell(f^{\text{lin}}(\mathbf{x}^{(1)}, \hat{\boldsymbol{\theta}}^\star), \mathbf{y}^{(1)}) \\ \vdots \\ \nabla_f \ell(f^{\text{lin}}(\mathbf{x}^{(|\mathcal{D}|)}, \hat{\boldsymbol{\theta}}^\star), \mathbf{y}^{(|\mathcal{D}|)}) \end{bmatrix},$$

and $\nabla_{\boldsymbol{\theta}} f(\mathbf{X}_r, \boldsymbol{\theta}')$ and $\nabla_{f^{\text{lin}}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}_r^\star)} \hat{\mathcal{L}}_{\mathcal{D}_r}$ are defined in the same way. Rearranging terms yields

$$\hat{\boldsymbol{\theta}}^\star = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top \boldsymbol{\alpha}^\star + \boldsymbol{\theta}'$$
$$\hat{\boldsymbol{\theta}}_r^\star = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}_r, \boldsymbol{\theta}')^\top \boldsymbol{\alpha}_r^\star + \boldsymbol{\theta}',$$

where $\boldsymbol{\alpha}^\star \coloneqq -\frac{1}{\lambda} \nabla_{f^{\text{lin}}(\mathbf{X}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|}$ and $\boldsymbol{\alpha}_r^\star \coloneqq -\frac{1}{\lambda} \nabla_{f^{\text{lin}}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}_r^\star)} \hat{\mathcal{L}}_{\mathcal{D}_r} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}_r|}$. Assume, without loss of generality, that data points in $\mathcal{D}_f$ are the first $|\mathcal{D}_f|$ data points in $\mathcal{D}$, then by defining $\tilde{\boldsymbol{\alpha}}_r^\star \coloneqq \left[\mathbf{0}^\top, \boldsymbol{\alpha}_r^{\star\top}\right]^\top \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|}$, $\hat{\boldsymbol{\theta}}_r^\star$ can be rewritten as $\hat{\boldsymbol{\theta}}_r^\star = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top \tilde{\boldsymbol{\alpha}}_r^\star + \boldsymbol{\theta}'$. Subtracting $\hat{\boldsymbol{\theta}}^\star$ from $\hat{\boldsymbol{\theta}}_r^\star$ yields

$$\hat{\boldsymbol{\theta}}_r^\star - \hat{\boldsymbol{\theta}}^\star = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top \Delta\boldsymbol{\alpha}_r^\star,$$

where

$$\Delta\boldsymbol{\alpha}_r^\star = \tilde{\boldsymbol{\alpha}}_r^\star - \boldsymbol{\alpha}^\star = \begin{bmatrix} \frac{1}{\lambda} \nabla_{f^{\text{lin}}(\mathbf{X}_f, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} \\ -\frac{1}{\lambda} \left(\nabla_{f^{\text{lin}}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}_r^\star)} \hat{\mathcal{L}}_{\mathcal{D}_r} - \nabla_{f^{\text{lin}}(\mathbf{X}_r, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}} \right) \end{bmatrix}.$$

□

Proposition A.1 implies that, for every nonempty $\mathcal{D}_r \subseteq \mathcal{D}$, the solution to $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ always belongs to the convex subspace $\mathcal{C} \coloneqq \hat{\boldsymbol{\theta}}^\star + \text{col}\left(\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top\right)$ of $\mathbb{R}^{d_{\boldsymbol{\theta}}}$, where $\text{col}(A)$ denotes the column space of A. Therefore, the original unconstrained problem can be reformulated as a constrained one, and the following proposition shows that the two problems are equivalent in the sense that they yield the same unique minimizer.

Proposition A.2. *Assume ℓ is convex with respect to model output, then for any nonempty $\mathcal{D}_r \subseteq \mathcal{D}$, $\arg\min_{\boldsymbol{\theta}\in\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ and $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ have the same unique minimizer.*

Proof of Proposition A.2. By definition of minimizer and the fact that $\mathcal{C} \subset \mathbb{R}^{d_{\boldsymbol{\theta}}}$, we know $\hat{\boldsymbol{\theta}}_r^\star \coloneqq \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ and $\hat{\boldsymbol{\theta}}_{r,c}^\star \coloneqq \arg\min_{\boldsymbol{\theta}\in\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ satisfy

$$\hat{\mathcal{L}}_{\mathcal{D}_r}(\hat{\boldsymbol{\theta}}_r^\star) \le \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta}),\ \forall \boldsymbol{\theta} \in \mathcal{C}$$
$$\hat{\mathcal{L}}_{\mathcal{D}_r}(\hat{\boldsymbol{\theta}}_{r,c}^\star) \le \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta}),\ \forall \boldsymbol{\theta} \in \mathcal{C}.$$

Thus, $\hat{\boldsymbol{\theta}}_r^\star$ and $\hat{\boldsymbol{\theta}}_{r,c}^\star$ are both minimizers of $\arg\min_{\boldsymbol{\theta}\in\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$. When ℓ is convex with respect to the model output, $\arg\min_{\boldsymbol{\theta}\in\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ is a convex optimization problem with a strictly convex objective function. As a result, the global minimizer is unique, and hence $\hat{\boldsymbol{\theta}}_r^\star = \hat{\boldsymbol{\theta}}_{r,c}^\star$. □

To solve the constrained problem, one can introduce a reparameterization that enforces the constraint, and reformulates the problem as an unconstrained one. We enforce the constraint $\boldsymbol{\theta} \in \mathcal{C}$ via the reparameterization

$$\boldsymbol{\theta} \coloneqq \phi(\Delta\boldsymbol{\alpha}) \coloneqq \hat{\boldsymbol{\theta}}^\star + \nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)^\top \Delta\boldsymbol{\alpha},$$

where $\Delta\boldsymbol{\alpha} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}|}$. Under the reparameterization, $f^{\text{lin}}(\cdot, \boldsymbol{\theta})$ becomes $g^{\text{lin}}(\cdot, \Delta\boldsymbol{\alpha}) \coloneqq f^{\text{lin}}(\cdot, \hat{\boldsymbol{\theta}}^\star) + \mathbf{K}(\cdot, \mathbf{X})\Delta\boldsymbol{\alpha}$ and $\hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$ becomes

$$\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}) \coloneqq \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \Big(\ell(g^{\text{lin}}(\mathbf{x}, \Delta\boldsymbol{\alpha}), \mathbf{y}) + \frac{\lambda}{2}\|\phi(\Delta\boldsymbol{\alpha}) - \boldsymbol{\theta}'\|_2^2\Big),$$

where $\mathbf{K}(\cdot,\cdot) \coloneqq \nabla_{\boldsymbol{\theta}} f(\cdot, \boldsymbol{\theta}')\nabla_{\boldsymbol{\theta}} f(\cdot, \boldsymbol{\theta}')^\top$ denotes the empirical Neural Tangent Kernel (NTK) (Jacot et al., 2018), and $\mathbf{K}(\mathbf{X}_1, \mathbf{X}_2) \in \mathbb{R}^{d_{\text{out}}N_1 \times d_{\text{out}}N_2}$ for feature matrices $\mathbf{X}_1 \in \mathbb{R}^{N_1 \times d_{\text{in}}}$ and $\mathbf{X}_2 \in \mathbb{R}^{N_2 \times d_{\text{in}}}$. $\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha})$ is a dual representation of $\hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$ and an unconstrained formulation of the constrained problem. The relationship between the solutions to the unconstrained problem in $\Delta\boldsymbol{\alpha}$ and the constrained problem in $\boldsymbol{\theta}$ is stated in the following proposition.

Proposition A.3. *If ℓ is convex with respect to model output, $\lambda > 0$, and $\nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)$ has full row rank, then for any nonempty $\mathcal{D}_r \subseteq \mathcal{D}$, $\arg\min_{\boldsymbol{\Delta\alpha}\in\mathbb{R}^{d_{out}|\mathcal{D}|}} \tilde{\mathcal{L}}_{\mathcal{D}_r}(\Delta\boldsymbol{\alpha})$ and $\arg\min_{\boldsymbol{\theta}\in\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ have unique minimizers $\boldsymbol{\Delta\alpha}_r^\star$ and $\hat{\boldsymbol{\theta}}_{r,c}^\star$ that are related by*

$$\boldsymbol{\Delta\alpha}_r^\star \underset{\phi^{-1}(\cdot)}{\overset{\phi(\cdot)}{\rightleftharpoons}} \hat{\boldsymbol{\theta}}_{r,c}^\star.$$

Proof of Proposition A.3. When ℓ is convex with respect to the model output and $\lambda > 0$, both $\arg\min_{\boldsymbol{\Delta\alpha}\in\mathbb{R}^{d_{\text{out}}|\mathcal{D}|}} \tilde{\mathcal{L}}_{\mathcal{D}_r}(\Delta\boldsymbol{\alpha})$ and $\arg\min_{\boldsymbol{\theta}\in\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ are convex optimization problems with strictly convex objective functions. As a result, their minimizers $\boldsymbol{\Delta\alpha}_r^\star$ and $\hat{\boldsymbol{\theta}}_{r,c}^\star$ are unique. If $\nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)$ has full row rank, then $\phi(\cdot)$ is injective and for every $\boldsymbol{\theta} \in \mathrm{Im}(\phi)$ there exists exactly one $\Delta\boldsymbol{\alpha}$ such that $\phi(\Delta\boldsymbol{\alpha}) = \boldsymbol{\theta}$. Since, by Propositions A.1 and A.2, we know $\hat{\boldsymbol{\theta}}_{r,c}^\star = \phi(\Delta\boldsymbol{\alpha}_r^\star)$ and $\hat{\boldsymbol{\theta}}_{r,c}^\star \in \mathrm{Im}(\phi)$, we can conclude that $\boldsymbol{\Delta\alpha}_r^\star \underset{\phi^{-1}(\cdot)}{\overset{\phi(\cdot)}{\rightleftharpoons}} \hat{\boldsymbol{\theta}}_{r,c}^\star$. □

Proposition A.2 and A.3 together imply that, for linearized models and any nonempty $\mathcal{D}_r \subseteq \mathcal{D}$, solving $\arg\min_{\boldsymbol{\Delta\alpha}\in\mathbb{R}^{d_{\text{out}}|\mathcal{D}|}} \tilde{\mathcal{L}}_{\mathcal{D}_r}(\Delta\boldsymbol{\alpha})$ is equivalent to solving $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$, in the sense that both lead to the same parameter $\hat{\boldsymbol{\theta}}_{r,c}^\star \coloneqq \arg\min_{\boldsymbol{\theta}\in\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{D}_r}(\boldsymbol{\theta})$ and the same empirical risk $\hat{\mathcal{L}}_{\mathcal{D}_r}(\hat{\boldsymbol{\theta}}_{r,c}^\star)$.

B DERIVATIONS

We present detailed derivations for the equations shown in Section 2.1 and 3.2.

B.1 DERIVATION FOR (1)

The following derivation, adapted from (Koh & Liang, 2017), serves to introduce our notation and to generalize their single data point removal setting to a multiple data points removal setting.

In the multiple data points removal setting, we define the upweighted empirical risk as $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \epsilon) \coloneqq \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) + |\mathcal{D}_f|\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta})\epsilon$. Substituting $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$ and $\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta})$ with their respective definitions and setting $\epsilon = -\frac{1}{|\mathcal{D}|}$ yields $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, -\frac{1}{|\mathcal{D}|}) = \frac{|\mathcal{D}_r|}{|\mathcal{D}|}\mathcal{L}_{\mathcal{D}_r}(\boldsymbol{\theta})$. Since $\mathcal{L}_{\mathcal{D}_r}(\boldsymbol{\theta})$ is strongly convex and $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, -\frac{1}{|\mathcal{D}|})$ is a scalar multiple of $\mathcal{L}_{\mathcal{D}_r}(\boldsymbol{\theta})$, the two optimization problems $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \mathcal{L}_{\mathcal{D}_r}(\boldsymbol{\theta})$ and $\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{d_{\boldsymbol{\theta}}}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, -\frac{1}{|\mathcal{D}|})$ share the same unique minimizer, i.e., $\boldsymbol{\theta}_r^\star = \boldsymbol{\theta}^\star(-\frac{1}{|\mathcal{D}|})$.

By first order necessary condition for optimality, $\boldsymbol{\theta}^\star$ and $\boldsymbol{\theta}^\star(\epsilon)$ satisfy $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star) = \mathbf{0}$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star(\epsilon), \epsilon) = \mathbf{0}$. When ϵ is small, $\boldsymbol{\theta}^\star(\epsilon)$ is close to $\boldsymbol{\theta}^\star$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star(\epsilon), \epsilon)$ can be accurately approximated by its first-order Taylor expansion around $\boldsymbol{\theta}^\star$, so the optimality condition for $\boldsymbol{\theta}^\star(\epsilon)$ becomes $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star, \epsilon) + \nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star, \epsilon)\left(\boldsymbol{\theta}^\star(\epsilon) - \boldsymbol{\theta}^\star\right) \approx \mathbf{0}$. Rearranging terms and taking first- and second-order derivatives of $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \epsilon)$ with respect to $\boldsymbol{\theta}$ yields

$$\begin{aligned}\boldsymbol{\theta}^\star(\epsilon) &\approx \boldsymbol{\theta}^\star - \nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star, \epsilon)^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star, \epsilon) \\ &= \boldsymbol{\theta}^\star - \left(\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star) + |\mathcal{D}_f|\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta}^\star)\epsilon\right)^{-1}|\mathcal{D}_f|\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta}^\star)\epsilon.\end{aligned}$$

When $\epsilon = -\frac{1}{|\mathcal{D}|}$, using the relation $\boldsymbol{\theta}_r^\star = \boldsymbol{\theta}^\star(-\frac{1}{|\mathcal{D}|})$, the above equation becomes $\boldsymbol{\theta}_r^\star \approx \boldsymbol{\theta}^\star + \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\mathbf{H}^{-1}_{\boldsymbol{\theta}^\star}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta}^\star)$, where $\mathbf{H}^{-1}_{\boldsymbol{\theta}^\star} \coloneqq (\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star) - \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}_f}(\boldsymbol{\theta}^\star))^{-1}$ can be further approximated by $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}^\star)^{-1}$ when $\frac{|\mathcal{D}_f|}{|\mathcal{D}|}$ is small, due to the identity $(\mathbf{A}+\epsilon\mathbf{X})^{-1} = \mathbf{A}^{-1} - \epsilon\mathbf{A}^{-1}\mathbf{X}\mathbf{A}^{-1} + \mathcal{O}(\epsilon^2)$.

B.2 DERIVATION FOR (5)

The derivation of (5) is identical to that of (1), with only a few minor tweaks. Specifically, by replacing $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \epsilon)$ with $\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}, \epsilon) \coloneqq \tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}) + |\mathcal{D}_f|\tilde{\mathcal{L}}_{\mathcal{D}_f}(\Delta\boldsymbol{\alpha})\epsilon$, but otherwise following the same derivation as in Section B.1, one can show that

$$\Delta\boldsymbol{\alpha}_r^\star \approx \Delta\boldsymbol{\alpha}^\star + \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\mathbf{H}^{-1}_{\Delta\boldsymbol{\alpha}^\star}\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\Delta\boldsymbol{\alpha}^\star),$$

where $\mathbf{H}^{-1}_{\Delta\boldsymbol{\alpha}^\star} \coloneqq \left(\nabla^2_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}^\star) - \frac{|\mathcal{D}_f|}{|\mathcal{D}|}\nabla^2_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_f}(\Delta\boldsymbol{\alpha}^\star)\right)^{-1}$ can be further approximated by $\nabla^2_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}^\star)^{-1}$ when $\frac{|\mathcal{D}_f|}{|\mathcal{D}|}$ is small. To obtain (5), it remains to show that $\Delta\boldsymbol{\alpha}^\star = \mathbf{0}$, which follows from Propositions A.2 and A.3. Specifically, these two propositions imply that, $\Delta\boldsymbol{\alpha}^\star$, $\hat{\boldsymbol{\theta}}^\star$ and $\hat{\boldsymbol{\theta}}_c^\star \coloneqq \arg\min_{\boldsymbol{\theta}\in\mathcal{C}}\hat{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\theta})$ are related by

$$\Delta\boldsymbol{\alpha}^\star \underset{\phi^{-1}(\cdot)}{\overset{\phi(\cdot)}{\rightleftarrows}} \hat{\boldsymbol{\theta}}_c^\star = \hat{\boldsymbol{\theta}}^\star.$$

Therefore, $\hat{\boldsymbol{\theta}}^\star = \hat{\boldsymbol{\theta}}^\star + \nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)^\top \Delta\boldsymbol{\alpha}^\star$ and rearranging terms yields $\nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)^\top \Delta\boldsymbol{\alpha}^\star = \mathbf{0}$. Since $\nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)$ is assumed to have full row rank, $\nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)^\top$ has full column rank and a trivial null space, i.e., $\ker(\nabla_{\boldsymbol{\theta}} f\left(\mathbf{X}, \boldsymbol{\theta}'\right)^\top) = \{\mathbf{0}\}$, which implies that $\Delta\boldsymbol{\alpha}^\star = \mathbf{0}$.

B.3 DERIVATION FOR EQUATIONS (6) AND (7)

For any dataset $\mathcal{D}_i$, by definition of $\tilde{\mathcal{L}}_{\mathcal{D}_i}(\Delta\boldsymbol{\alpha})$, $\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_i}(\Delta\boldsymbol{\alpha})$ and $\nabla^2_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_i}(\Delta\boldsymbol{\alpha})$ are given by

$$\nabla_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_i}(\Delta\boldsymbol{\alpha}) = \mathbf{K}(\mathbf{X}, \mathbf{X}_i)\nabla_{g^{\text{lin}}(\mathbf{X}_i, \Delta\boldsymbol{\alpha})}\tilde{\mathcal{L}}_{\mathcal{D}_i} + \lambda\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')(\hat{\boldsymbol{\theta}}^\star - \boldsymbol{\theta}') + \lambda\mathbf{K}\Delta\boldsymbol{\alpha} \tag{14}$$

$$\nabla^2_{\Delta\boldsymbol{\alpha}}\tilde{\mathcal{L}}_{\mathcal{D}_i}(\Delta\boldsymbol{\alpha}) = \mathbf{K}(\mathbf{X}, \mathbf{X}_i)\nabla^2_{g^{\text{lin}}(\mathbf{X}_i, \Delta\boldsymbol{\alpha})}\tilde{\mathcal{L}}_{\mathcal{D}_i}\mathbf{K}(\mathbf{X}_i, \mathbf{X}) + \lambda\mathbf{K}, \tag{15}$$

where $\mathbf{K}(\mathbf{X}, \mathbf{X}_i) \coloneqq \left[\mathbf{K}(\mathbf{X}, \mathbf{x}^{(1)}), \cdots, \mathbf{K}(\mathbf{X}, \mathbf{x}^{(|\mathcal{D}_i|)})\right] \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}| \times d_{\text{out}}|\mathcal{D}_i|}$,

$$\nabla_{g^{\text{lin}}(\mathbf{X}_i, \Delta\boldsymbol{\alpha})} \tilde{\mathcal{L}}_{\mathcal{D}_i} \coloneqq \frac{1}{|\mathcal{D}_i|} \begin{bmatrix} \nabla_f \ell(g^{\text{lin}}(\mathbf{x}^{(1)}, \Delta\boldsymbol{\alpha}), \mathbf{y}^{(1)}) \\ \vdots \\ \nabla_f \ell(g^{\text{lin}}(\mathbf{x}^{(|\mathcal{D}_i|)}, \Delta\boldsymbol{\alpha}), \mathbf{y}^{(|\mathcal{D}_i|)}) \end{bmatrix} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}_i|}.$$

and $\nabla^2_{g^{\text{lin}}(\mathbf{X}_i, \Delta\boldsymbol{\alpha})} \tilde{\mathcal{L}}_{\mathcal{D}_i} \in \mathbb{R}^{d_{\text{out}}|\mathcal{D}_i| \times d_{\text{out}}|\mathcal{D}_i|}$ is a block diagonal matrix whose blocks are

$$\frac{1}{|\mathcal{D}_i|} \nabla^2_f \ell(g^{\text{lin}}(\mathbf{x}, \Delta\boldsymbol{\alpha}), \mathbf{y}) \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}, \ \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{D}_i.$$

$\nabla_{\Delta\boldsymbol{\alpha}} \tilde{\mathcal{L}}_{\mathcal{D}_f}(\Delta\boldsymbol{\alpha})$, $\nabla^2_{\Delta\boldsymbol{\alpha}} \tilde{\mathcal{L}}_{\mathcal{D}_f}(\Delta\boldsymbol{\alpha})$ and $\nabla^2_{\Delta\boldsymbol{\alpha}} \tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha})$ are obtained by replacing $\mathcal{D}_i$ in (14) and (15) with $\mathcal{D}_f$ and $\mathcal{D}$.

Equation (7) is obtained by evaluating $\nabla_{\Delta\boldsymbol{\alpha}} \tilde{\mathcal{L}}_{\mathcal{D}_f}(\Delta\boldsymbol{\alpha})$ at $\Delta\boldsymbol{\alpha}^\star = \mathbf{0}$,

$$\begin{aligned} \nabla_{\Delta\boldsymbol{\alpha}} \tilde{\mathcal{L}}_{\mathcal{D}_f}(\mathbf{0}) &= \mathbf{K}(\mathbf{X}, \mathbf{X}_f) \nabla_{g^{\text{lin}}(\mathbf{X}_f, \mathbf{0})} \tilde{\mathcal{L}}_{\mathcal{D}_f} + \lambda \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')(\hat{\boldsymbol{\theta}}^\star - \boldsymbol{\theta}') \\ &= \mathbf{K}(\mathbf{X}_f, \mathbf{X})^\top \nabla_{f^{\text{lin}}(\mathbf{X}_f, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\mathcal{D}_f} + \lambda \mathbf{K} \boldsymbol{\alpha}^\star, \end{aligned}$$

where we have used the fact that $\hat{\boldsymbol{\theta}}^\star - \boldsymbol{\theta}' = \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top \boldsymbol{\alpha}^\star$ and $g^{\text{lin}}(\cdot, \mathbf{0}) = f^{\text{lin}}(\cdot, \hat{\boldsymbol{\theta}}^\star)$. By definition of $\mathbf{H}_{\Delta\boldsymbol{\alpha}}$, we have

$$\begin{aligned} \mathbf{H}_{\Delta\boldsymbol{\alpha}} &= \nabla^2_{\mathbf{\Delta\boldsymbol{\alpha}}} \tilde{\mathcal{L}}_{\mathcal{D}}(\Delta\boldsymbol{\alpha}) - \frac{|\mathcal{D}_f|}{|\mathcal{D}|} \nabla^2_{\Delta\boldsymbol{\alpha}} \tilde{\mathcal{L}}_{\mathcal{D}_f}(\Delta\boldsymbol{\alpha}) \\ &= \frac{|\mathcal{D}_r|}{|\mathcal{D}|} \left(\mathbf{K}(\mathbf{X}_r, \mathbf{X})^\top \nabla^2_{g^{\text{lin}}(\mathbf{X}_r, \Delta\boldsymbol{\alpha})} \tilde{\mathcal{L}}_{\mathcal{D}_r} \mathbf{K}(\mathbf{X}_r, \mathbf{X}) + \lambda \mathbf{K} \right), \end{aligned}$$

evaluating $\mathbf{H}_{\Delta\boldsymbol{\alpha}}$ at $\Delta\boldsymbol{\alpha}^\star = \mathbf{0}$ and using the equality $g^{\text{lin}}(\cdot, \mathbf{0}) = f^{\text{lin}}(\cdot, \hat{\boldsymbol{\theta}}^\star)$ gives (6).

B.4 Derivation for (8) and (9)

The first-order Taylor expansion of $g^{\text{lin}}(\mathbf{x}_t, \Delta\boldsymbol{\alpha}_r^\star)$ and $\tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\Delta\boldsymbol{\alpha}_r^\star)$ around $\Delta\boldsymbol{\alpha}^\star = 0$ gives

$$\begin{aligned} g^{\text{lin}}(\mathbf{x}_t, \Delta\boldsymbol{\alpha}_r^\star) - g^{\text{lin}}(\mathbf{x}_t, \mathbf{0}) &= \mathbf{K}(\mathbf{x}_t, \mathbf{X}) \Delta\boldsymbol{\alpha}_r^\star \\ \tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\Delta\boldsymbol{\alpha}_r^\star) - \tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\mathbf{0}) &\approx \nabla_{\Delta\boldsymbol{\alpha}} \tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\mathbf{0})^\top \Delta\boldsymbol{\alpha}_r^\star. \end{aligned}$$

When $g^{\text{lin}}(\mathbf{x}_t, \Delta\boldsymbol{\alpha})$ and $\tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\Delta\boldsymbol{\alpha})$ are evaluated at $\mathbf{0}$, by their definitions, we have $g^{\text{lin}}(\mathbf{x}_t, \mathbf{0}) = f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}^\star)$ and

$$\tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\mathbf{0}) = \ell(f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}^\star), \mathbf{y}_t) + \frac{\lambda}{2} \|\hat{\boldsymbol{\theta}}^\star - \boldsymbol{\theta}'\|_2^2 = \hat{\mathcal{L}}_{\{\mathbf{z}_t\}}(\hat{\boldsymbol{\theta}}^\star).$$

When they are evaluated at $\Delta\boldsymbol{\alpha}_r^\star$, by Proposition A.1 and definition of $f^{\text{lin}}(\cdot, \boldsymbol{\theta})$, we have

$$\begin{aligned} g^{\text{lin}}(\mathbf{x}_t, \Delta\boldsymbol{\alpha}_r^\star) &= f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}^\star) + \mathbf{K}(\mathbf{x}_t, \mathbf{X})(\tilde{\boldsymbol{\alpha}}_r^\star - \boldsymbol{\alpha}^\star) \\ &= f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}^\star) + \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_t, \boldsymbol{\theta}')(\hat{\boldsymbol{\theta}}_r^\star - \boldsymbol{\theta}') - \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_t, \boldsymbol{\theta}')(\hat{\boldsymbol{\theta}}^\star - \boldsymbol{\theta}') \\ &= f(\mathbf{x}_t, \boldsymbol{\theta}') + \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_t, \boldsymbol{\theta}')(\hat{\boldsymbol{\theta}}_r^\star - \boldsymbol{\theta}') \\ &= f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}_r^\star), \end{aligned}$$

and

$$\begin{aligned} \tilde{\mathcal{L}}_{\{\mathbf{z}_t\}}(\Delta\boldsymbol{\alpha}_r^\star) &= \ell(f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}_r^\star), \mathbf{y}_t) + \frac{\lambda}{2} \|\hat{\boldsymbol{\theta}}^\star + \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}')^\top \Delta\boldsymbol{\alpha}_r^\star - \boldsymbol{\theta}'\|_2^2 \\ &= \ell(f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}_r^\star), \mathbf{y}_t) + \frac{\lambda}{2} \|\hat{\boldsymbol{\theta}}_r^\star - \boldsymbol{\theta}'\|_2^2 \\ &= \hat{\mathcal{L}}_{\{\mathbf{z}_t\}}(\hat{\boldsymbol{\theta}}_r^\star). \end{aligned}$$

Combining these results gives (8) and (9).

B.5 DERIVATION FOR (10)

Evaluating (9) at all data points in $\mathcal{D}_t$ is equivalent to performing the matrix vector product:

$$\begin{bmatrix} \left(\mathbf{K}(\mathbf{x}_t^{(1)}, \mathbf{X})^\top \nabla_{f^{\text{lin}}(\mathbf{x}_t^{(1)}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\{\mathbf{z}_t^{(1)}\}} + \lambda \mathbf{K} \boldsymbol{\alpha}^\star\right)^\top \\ \vdots \\ \left(\mathbf{K}(\mathbf{x}_t^{(|\mathcal{D}_t|)}, \mathbf{X})^\top \nabla_{f^{\text{lin}}(\mathbf{x}_t^{(|\mathcal{D}_t|)}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\{\mathbf{z}_t^{(|\mathcal{D}_t|)}\}} + \lambda \mathbf{K} \boldsymbol{\alpha}^\star\right)^\top \end{bmatrix} \Delta \boldsymbol{\alpha}_r^\star.$$

Assume f^{lin} has scalar-valued output, the above product can be rewritten as

$$\begin{bmatrix} \nabla_{f^{\text{lin}}(\mathbf{x}_t^{(1)}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\{\mathbf{z}_t^{(1)}\}} \\ \vdots \\ \nabla_{f^{\text{lin}}(\mathbf{x}_t^{(|\mathcal{D}_t|)}, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\{\mathbf{z}_t^{(|\mathcal{D}_t|)}\}} \end{bmatrix} \odot \left(\begin{bmatrix} \mathbf{K}(\mathbf{x}_t^{(1)}, \mathbf{X}) \\ \vdots \\ \mathbf{K}(\mathbf{x}_t^{(|\mathcal{D}_t|)}, \mathbf{X}) \end{bmatrix} \Delta \boldsymbol{\alpha}_r^\star \right) + \lambda \begin{bmatrix} \boldsymbol{\alpha}^{\star\top} \mathbf{K} \Delta \boldsymbol{\alpha}_r^\star \\ \vdots \\ \boldsymbol{\alpha}^{\star\top} \mathbf{K} \Delta \boldsymbol{\alpha}_r^\star \end{bmatrix},$$

which gives (10). The generalization to vector-valued f^{lin} follows by noticing

$$\mathbf{1}^\top \left(\nabla_{f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\{\mathbf{z}_t\}} \odot (\mathbf{K}(\mathbf{x}_t, \mathbf{X}) \Delta \boldsymbol{\alpha}_r^\star) \right) = \nabla_{f^{\text{lin}}(\mathbf{x}_t, \hat{\boldsymbol{\theta}}^\star)} \hat{\mathcal{L}}_{\{\mathbf{z}_t\}}^\top \mathbf{K}(\mathbf{x}_t, \mathbf{X}) \Delta \boldsymbol{\alpha}_r^\star,$$

where $\mathbf{1} \in \mathbb{R}^{d_{\text{out}}}$ denotes the all-ones vector.