DIVA: DATASET DERIVATIVE OF A LEARNING TASK

Anonymous authors
Paper under double-blind review

ABSTRACT

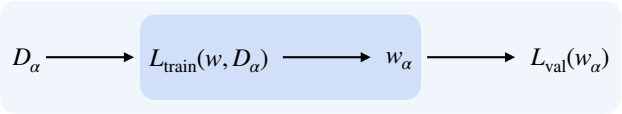
We present a method to compute the derivative of a learning task with respect to a dataset. A learning task is a function from a training set to the validation error, which can be represented by a trained deep neural network (DNN). The "dataset derivative" is a linear operator, computed around the trained model, that informs how perturbations of the weight of each training sample affect the validation error, usually computed on a separate validation dataset. Our method, DIVA (Differentiable Validation) hinges on a closed-form differentiable expression of the leave-one-out cross-validation error around a pre-trained DNN. Such expression constitutes the dataset derivative. DIVA could be used for dataset auto-curation, for example removing samples with faulty annotations, augmenting a dataset with additional relevant samples, or rebalancing. More generally, DIVA can be used to optimize the dataset, along with the parameters of the model, as part of the training process without the need for a separate validation dataset, unlike bi-level optimization methods customary in AutoML. To illustrate the flexibility of DIVA, we report experiments on sample auto-curation tasks such as outlier rejection, dataset extension, and automatic aggregation of multi-modal data.

1 INTRODUCTION

Consider the following seemingly disparate questions. *(i) Dataset Extension:* Given a relatively small training set, but access to a large pool of additional data, how to select from from the latter samples to augment the former? *(ii) Dataset Curation:* Given a potentially large dataset riddled with annotation errors, how to automatically reject such outlier samples? *(iii) Dataset Reweighting:* Given a finite training set, how to reweight the training samples to yield better generalization performance?

These three are examples of *Dataset Optimization*. In order to solve this problem with differentiable programming, one can optimize a loss of the model end-to-end, which requires *differentiating the model's loss with respect to the dataset.* Our main contribution is an efficient method to compute such a *dataset derivative*. This allows learning an importance weight α_i for each datum in a training dataset $\mathcal{D}$, extending the optimization from the weights $\mathbf{w}$ of a parametric model such as a deep neural network (DNN), to also include the weights of the dataset.

As illustrated in the following diagram, standard optimization in machine learning works by finding the weights $\mathbf{w}_\alpha$ that minimize the training loss $L_{\text{train}}(\mathbf{w}, D_\alpha) = \sum_i \alpha_i \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$ on a given (weighted) dataset D_α (dark box). We solve a more general learning problem (light box) by jointly optimizing the dataset $\mathcal{D}_\alpha$ in addition to $\mathbf{w}$. To avoid the trivial solution $\alpha = 0$, it is customary in AutoML to optimize $\mathcal{D}_\alpha$ by minimizing the validation error computed on a disjoint dataset. This makes for inefficient use of the data, which has to be split between training and validation sets. Instead, we leverage a closed-form expression of the leave-one-out cross-validation error to jointly optimize the model and data weights during training, without the need to create a separate validation set.



The intermediate block in the diagram (which finds the optimal weights $\mathbf{w}_\alpha$ for the the training loss on $\mathcal{D}_\alpha$) is usually non-differentiable with respect to the dataset, or the derivative is prohibitively

expensive to compute. DIVA leverages recent progress in deep learning linearization Achille et al. (2020), to derive a closed-form expression for the derivative of the final loss (validation error) with respect to the dataset weights. In particular, Achille et al. (2020) have shown that, by replacing cross-entropy with least-squares, replacing ReLu with leaky-ReLu, and performing suitable pre-conditioning, the linearized model performs on par with full non-linear fine-tuning. We also leverage a classical result to compute the leave-one-out loss of a linear model in closed-form (Rifkin & Lippert, 2007; Green & Silverman, 1993). This allows us to optimize the LOO loss without requiring a separate validation set, setting DIVA apart from bi-level optimization customary in AutoML.

To illustrate the many possible uses of the dataset derivative, we run experiments with a simplified version of DIVA to cleanup a dataset of noisy annotations, to extend a training set with additional data from an external pool, identify meaningful data augmentation, and to perform multi-modal expansion using a CLIP model (Radford et al., 2021).

Rather than using the full linearization of the model derived by Achille et al. (2020), we restrict the gradient to its last layer, cognizant that we are not exploiting the full power of LQF and thereby obtaining only a lower-bound of performance improvement. Despite that restriction, our results show consistent improvements from dataset optimization, at the modest computational cost of a forward pass over the dataset to optimize the importance weights.

To summarize, our main contributions are:

1. We introduce a method to compute the dataset derivative in closed form, DIVA.
2. We illustrate the use of DIVA to perform dataset optimization by minimizing directly the leave-one-out error without the need for an explicit validation dataset.
3. We perform experiments with a simplified model that, despite not using the full power of the linearization, shows consistent improvements in dataset extension, re-weighting, outlier rejection and automatic aggregation of multi-modal data.

Our method presents several limitations. The dataset derivative of a learning task is computed around a point represented by a pre-trained model. It only allows local optimization around this point. Moreover, we only compute a restriction of the linearization to the dimensions spanned by the last few layers. In general, this yields suboptimal results compared to full global optimization from scratch, if one could compute that at scale. Nonetheless, the linearized setting is consistent with the practice of fine-tuning pre-trained models in light of the results of Achille et al. (2020), see also (Radford et al., 2021; Rezende et al., 2017; Mormont et al., 2018; Hosny et al., 2018).

2 Related Work

AutoML. State of the art performance in image classification tasks often relies on large amount of human expertise in selecting models and adjusting the training settings for the task at hand (Li et al., 2020a). Automatic machine learning (AutoML) (Feurer et al., 2019; He et al., 2021) aims to automate model selection (Cawley & Talbot, 2010) and the training settings by instead using meta-algorithms for the different aspects of the learning settings. Such methods follow a bi-level optimization framework, optimizing the training settings in the outer level, and traditional model optimization in the inner level (Jenni & Favaro, 2018). AutoML has focused on achieving better results via automatic model selection (Deshpande et al., 2021; Feurer et al., 2019) including neural architecture search (NAS) (Zoph & Le, 2016; Elsken et al., 2019; Liu et al., 2019). Other important AutoML topics include hyper-parameter selection (Li et al., 2017; Akiba et al., 2019) and data augmentation (Cubuk et al., 2018; Lim et al., 2019; Chen et al., 2020; Behl et al., 2020), which are closer to our settings of optimizing the dataset weights. Since the main signal for a model's performance is the final validation loss, which requires full optimization of the model for each evaluation, AutoML approaches often incur a steep computational costs. Alternatively, other methods follow alternating optimization of the criteria, such as the work of Ren et al. (2018) that approximates full network optimization with a single SGD step to learn to reweight the training set dynamically. *Differentiable AutoML* alleviates outer-optimization costs while optimizing the final validation error via differentiable programming, by utilizing proxy losses and continuous relaxation that enable differentiation. Different approaches to differentiable AutoML include differentiable NAS (Liu et al., 2018; Wu et al., 2019), data augmentation (Liu et al., 2021; Li et al., 2020b), and hyper-parameter optimization (Andrychowicz et al., 2016). The DIVA dataset derivative follows the differentiable

AutoML framework by enabling direct optimization of the dataset with respect to the final validation error of the model.

Importance sampling. While our dataset optimization problem may seem superficially similar to importance sampling, the optimization objective is different. Importance sampling aims to reweight the training set to make it more similar to the test distribution or to speed up convergence. On the other hand, DIVA objective is to optimizes a validation loss of the model, even if this requires making the training distribution significantly different from the testing distribution. Importance sampling methods have a long history in the MCMC machine learning literature where the sampling is conditioned on the predicted importance of samples (Metropolis & Ulam, 1949; Liu, 2008). In deep learning, importance sampling methods have been studied theoretically for linearly-separable data (Byrd & Lipton, 2019) and recently in more generality (Xu et al., 2021). Furthermore, there exist many importance sampling heuristics in deep learning training including different forms of hard sample mining (Shrivastava et al., 2016; Xue et al., 2019; Chang et al., 2017), weighting based on a focal loss (Lin et al., 2017), re-weighting for imbalance, (Cui et al., 2019; Huang et al., 2019; Dong et al., 2017) and gradient based scoring (Li et al., 2019). We emphasize that DIVA's optimization of the sample weights is not based on a heuristic but is rather a differentiable AutoML method driven by optimization of a proxy of the test error. Further, DIVA allows optimization of the dataset weights with respect to an arbitrary loss and also allows for dataset extension computation.

LOO based optimization. Leave-one-out cross validation is well established in statistical learning (Stone, 1977). In ridge regression, the LOO model predictions for the validation samples have a closed-form expression that avoids explicit cross validation computation (Green & Silverman, 1993; Rifkin & Lippert, 2007) enabling efficient and scalable unbiased estimate of the test error. Efficient LOO has been widely used as a criterion for regularization (Pedregosa et al., 2011; Quan et al., 2010; Birattari et al., 1999; Thapa et al., 2020), hyper-parameter selection (Hwang & Shim, 2017) and optimization (Wen et al., 2008). Most similar to our dataset derivative are methods that: (1) optimize a restricted set of parameters, such as kernel bandwidth, in weighted least squares (Cawley, 2006; Hong et al., 2007) (2) locally weighted regression methods (*memorizing regression*) (Atkeson et al., 1997; Moore et al., 1992), or (3) methods that measure the impact of samples based on LOO predictions (Brodley & Friedl, 1999; Nikolova et al., 2021).

Dataset selection & sample impact measures. Koh & Liang (2017) measure the effect of changes of a training sample weight on a final validation loss through per-sample weight gradients, albeit without optimizing the dataset and requiring a separate validation set. Their proposed expression for the per-sample gradient, however, does not scale easily to our problem of dataset optimization. In contrast, in proposition 3 we introduce an efficient closed-form expression for the derivative of the whole datasets. Moreover, in proposition 3, we show how to optimize the weights with respect to a cross-validation loss which does not require a separate set. In Pruthi et al. (2020), the authors present a sample-impact measure for interpretability based on a validation set; for dataset extension, Yan et al. (2020) presents a coarse dataset extension method based on self-supervised learning. Dataset distillation and core set selection methods aim to decrease the size of the dataset (Wang et al., 2018) by selecting a representative dataset subset (Hwang et al., 2020; Jeong et al., 2020; Coleman et al., 2019; Joneidi et al., 2020; Trichet & Bremond, 2018; Killamsetty et al., 2021). While DIVA is capable of removing outliers, in this work we do not approach dataset selection from the perspective of making the dataset more computationally tractable by reducing the number of samples.

3 METHOD

In supervised learning, we use a parametrized model $f_{\mathbf{w}}(\mathbf{x})$ to predict a target output y given an input $\mathbf{x}$ coming from a joint distribution $(\mathbf{x}, y) \sim \mathcal{T}$. Usually, we are given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with samples (x, y) assumed to be independent and identically distributed (i.i.d.) according to $\mathcal{T}$. The training set $\mathcal{D}$ is then used to assemble the empirical risk for some per-sample loss ℓ,

$$L_{\text{train}}(\mathbf{w}; \mathcal{D}) = \sum_{i=1}^{N} \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i),$$

which is minimized to find the optimal model parameters $\mathbf{w}_{\mathcal{D}}$:

$$\mathbf{w}_{\mathcal{D}} = \operatorname*{argmin}_{\mathbf{w}} L_{\text{train}}(\mathbf{w}; \mathcal{D}).$$

The end goal of empirical risk minimization is that weights will also minimize the *test loss*, computed using a separate test set. Nonetheless $\mathcal{D}$ is often biased and differs from the distribution $\mathcal{T}$. In addition, from the perspective of optimization, different weighting of the training loss samples can enable or inhibit good learning outcomes of the task $\mathcal{T}$ (Lin et al., 2017).

Dataset Optimization. In particular, it may not be the case that sampling the training set $\mathcal{D}$ i.i.d. from $\mathcal{T}$ is the best option to guarantee generalization, nor it is realistic to assume that $\mathcal{D}$ is a fair sample. Including in-distribution samples that are too difficult may negatively impact the optimization, while including certain out-of-distribution examples may aid the generalization on $\mathcal{T}$. It is not uncommon, for example, to improve generalization by training on a larger dataset containing out-of-distribution samples coming from other sources, or generating out-of-distribution samples with data augmentation. We call Dataset Optimization the problem of finding the optimal subset of samples, real or synthetic, to include or exclude from a training set $\mathcal{D}$ in order to guarantee that the weights $\mathbf{w}_{\mathcal{D}}$ trained on $\mathcal{D}$ will generalize as much as possible.

Differentiable Dataset Optimization. Unfortunately, a naïve brute-force search over the 2^N possible subsets of $\mathcal{D}$ is unfeasible. The starting idea of DIVA is to instead solve a more general continuous optimization problem that can however be optimized end-to-end. Specifically, we parameterize the choice of samples in the augmented dataset through a set of non-negative continuous sample weights α_i which can be optimized by gradient descent along with the weights of the model. Let $\alpha = (\alpha_1, \ldots, \alpha_N)$ be the vector of the sample weights and denote the corresponding weighted dataset by $\mathcal{D}_\alpha$. The training loss on $\mathcal{D}_\alpha$ is then defined as:

$$L_{\mathcal{D}}(\mathbf{w}; \mathcal{D}_\alpha) = \sum_{i=1}^{N} \alpha_i \, \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i). \tag{1}$$

Note that if all α_i's are either 0 or 1, we are effectively selecting only a subset of $\mathcal{D}$ for training. As we will show, this continuous generalization allows us to optimize the sample selection in a differentiable way. In principle, we would like to find the sample weights $\alpha^* = \operatorname{argmin}_\alpha L_{\mathcal{D}_{\text{test}}}(\mathbf{w}_\alpha)$ that lead to the best generalization. Since we do not have access to the test data, in practice this translates to optimizing α with respect to an (unweighted) validation loss L_{val}:

$$\alpha^* = \operatorname{argmin}_\alpha L_{\text{val}}(\mathbf{w}_\alpha).$$

We can, of course, compute a validation loss using a separate validation set. However, as we will see in Section 3.3, we can also use a leave-one-out cross-validation loss directly on the training set, without any requirement of a separate validation set.

In order to efficiently optimize α by gradient-descent, we need to compute the dataset derivative $\nabla_\alpha L_{\text{val}}(\mathbf{w}_\alpha)$. By the chain rule, this can be done by computing $\nabla_\alpha \mathbf{w}_\alpha$. However, the training function $\alpha \to \mathbf{w}_\alpha$ that finds the optimal weights $\mathbf{w}_\alpha$ of the model given the sample weights α may be non-trivial to differentiate or may not be differentiable at all (for example, it may consist of thousands of steps of SGD). This would prevent us from minimizing α end-to-end.

In the next section, we show that if, instead of linearizing the $\mathbf{w}_\alpha$ end-to-end in order to compute the derivative, we linearize the model *before* the optimization step, the derivative can both be written in closed-form and computed efficiently, thus giving us a tractable way to optimize α.

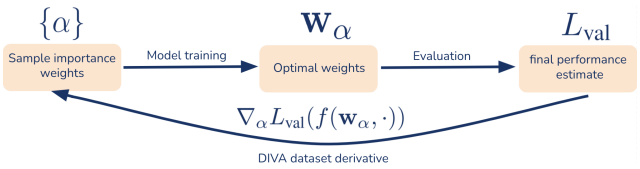


Figure 1: The DIVA dataset derivative is computed end-to-end from the final validation loss

3.1 Linearization

In real-world applications, the parametric model $f_{\mathbf{w}}(\mathbf{x})$ is usually a deep neural network. Recent work (Achille et al., 2020; Mu et al., 2020) have shown that in many cases, a deep neural network can

be transformed to an equivalent linear model that can be trained on a simple quadratic loss and still reach a performance similar to the original model. Given a model $f_{\mathbf{w}}(\mathbf{x})$, let $\mathbf{w}_0$ denote an initial set of weights, For example, $\mathbf{w}_0$ could be obtained by pre-training on a large dataset such as ImageNet (if the task is image classification). Following Achille et al. (2020); Mu et al. (2020), we consider a linearization $f^{\text{lin}}_{\mathbf{w}}(\mathbf{x})$ of the network $f_{\mathbf{w}}(\mathbf{x})$ given by the first-order Taylor expansion of $f_{\mathbf{w}}(\mathbf{x})$ around $\mathbf{w}_0$:

$$f^{\text{lin}}_{\mathbf{w}}(\mathbf{x}) = f_{\mathbf{w}_0}(\mathbf{x}) + \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}) \cdot (\mathbf{w} - \mathbf{w}_0). \tag{2}$$

Intuitively, if fine-tuning does not move the weights much from the initial pre-trained weights $\mathbf{w}_0$, then $f^{\text{lin}}_{\mathbf{w}}(\mathbf{x})$ will remain a good approximation of the network while becoming linear in $\mathbf{w}$ (but still remaining highly non-linear with respect to the input $\mathbf{x}$). Effectively, this is equivalent to training a linear classifier using the gradients $\mathbf{z}_i := \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}_i)$ as features (Mu et al., 2020).

Although $f^{\text{lin}}_{\mathbf{w}}(\mathbf{x})$ is a linear model, the optimal weights $\mathbf{w}_\alpha$ may still be a complex function of the training data, depending on the loss function used. Achille et al. (2020) showed that equivalent performance can be obtained by replacing the empirical cross-entropy with the regularized least-squares loss:

$$L_{\mathcal{D}}(\mathbf{w}) = \sum_{i=1}^{N} \|f^{\text{lin}}_{\mathbf{w}}(\mathbf{x}) - \mathbf{y}_i\|^2 + \lambda \|\mathbf{w}\|^2 \tag{3}$$

where $\mathbf{y}$ denotes the one-hot encoding vector of the label y_i. In Achille et al. (2020), it is shown that linearized models are equivalent from the standpoint of performance on most standard tasks and classification benchmarks, and better in the low-data regime, which is where the problem of "dataset augmentation" is most relevant. The advantage of using this loss is that the optimal weights $\mathbf{w}^*$ can now be written in closed-form as

$$\mathbf{w}^* = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top (\mathbf{Y} - f_{\mathbf{w}_0}(\mathbf{X})), \tag{4}$$

where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$ is the matrix of the Jacobians $\mathbf{z}_i = \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}_i)$. While our method can be applied with no changes to linearization of the full network, for simplicity in our experiments we restrict to linearizing only the last layer of the network. This is equivalent to using the network as a fixed feature extractor and training a linear classifier on top the last-layer features, that is, $\mathbf{z}_i = f^{L-1}_{\mathbf{w}_0}(\mathbf{x}_i)$ are the features at the penultimate layer.

3.2 Computation of the Dataset Derivative

We now show that for linearized models we can compute the derivative $\nabla_\alpha \mathbf{w}_\alpha$ in closed-form. For the α-weighted dataset, the objective in eq. (3) with L_2 loss for the linearized model is written as,

$$\mathbf{w}_\alpha = \operatorname*{argmin}_{\mathbf{w}} L_{\mathcal{D}}(\mathbf{w}; \mathcal{D}_\alpha) = \operatorname*{argmin}_{\mathbf{w}} \sum_{i=1}^{N} \alpha \|\mathbf{w}^\top \mathbf{z}_i - \mathbf{y}_i\|^2 + \lambda \|\mathbf{w}\|^2. \tag{5}$$

where $\mathbf{z}_i = \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}_i)$ as in the previous section. Note that $\alpha \|\mathbf{w}^\top \mathbf{z}_i - \mathbf{y}_i\|^2 = \|\mathbf{w}^\top \mathbf{z}^\alpha_i - \mathbf{y}^\alpha_i\|$, where $\mathbf{z}^\alpha_i := \sqrt{\alpha}\mathbf{z}_i$ and $\mathbf{y}^\alpha_i := \sqrt{\alpha}\mathbf{y}_i$. Using this, we can reuse eq. (4) to obtain the following closed-form solution for $\mathbf{w}_\alpha$:

$$\mathbf{w}_\alpha = (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}, \tag{6}$$

where we have taken $\mathbf{D}_\alpha = \text{diag}(\alpha)$. In particular, note that $\mathbf{w}_\alpha$ is now a differentiable function of α. The following proposition gives a closed-form expression for the derivative.

Proposition 1 (Model-Dataset Derivative $\nabla_\alpha \mathbf{w}_\alpha$)**.** *For the ridge regression problem equation 5 and $\mathbf{w}_\alpha$ defined as in equation 6, define*

$$\mathbf{C}_\alpha = (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1}. \tag{7}$$

Then the Jacobian of $\mathbf{w}_\alpha$ with respect to α is given by

$$\nabla_\alpha \mathbf{w}_\alpha = \mathbf{Z}\mathbf{C}_\alpha \circ \big((\mathbf{I} - \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha)\mathbf{Y}\big), \tag{8}$$

Where we write $\mathbf{A} \circ \mathbf{B} \in \mathbb{R}^{n \times m \times k}$ for the batch-wise outer product of $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$ along the common dimension k, i.e., $(\mathbf{A} \circ \mathbf{B})_{ijk} = a_{ij} b_{ik}$

Figure 2: **Examples of the reweighting done by DIVA. (Left)** Samples from the FGVC Aircraft classification dataset that are up-weighted by DIVA and **(Right)** samples that are down-weighted because they increase the test error. Down-weighted samples tend to have planes in non-canonical poses, multiple planes, or not enough information to classify the plane correctly.

The Jacobian $\nabla_\alpha \mathbf{w}_\alpha$ would be rather large to compute explicitly. Fortunately, the end-to-end gradient $L_{\text{val}}(\mathbf{w}_\alpha)$ if the final validation loss can still be computed efficiently, as we now show. Given a validation dataset $\mathcal{D}_{\text{val}}$, the validation loss is:

$$L_{\text{val}}(\mathbf{w}_\alpha) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{val}}} \ell(f_{\mathbf{w}_\alpha}(\mathbf{x}_i), y_i). \tag{9}$$

The following gives the expression to optimize α end-to-end with respect to the validation loss.

Proposition 2 (Validation Loss Dataset Derivative)**.** *Define* $\mathbf{L}$ *as the loss function derivative with respect to the network outputs as,*

$$\mathbf{L} = \left[\frac{\partial \ell}{\partial f}(f(\mathbf{x}_1), y_1), \cdots \frac{\partial \ell}{\partial f}(f(\mathbf{x}_N), y_N) \right]$$

Then the dataset derivative importance weights with respect to final validation is given by

$$\nabla_\alpha L_{val}(\mathbf{w}_\alpha) = \mathbf{Z}\mathbf{C}_\alpha\mathbf{Z}^\top \times \left(\mathbf{L}^\top \mathbf{Y}^\top (\mathbf{I} - \mathbf{D}_\alpha \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top)\right). \tag{10}$$

3.3 Leave-one-out optimization

It is common in AutoML to optimize the hyper-parameters with respect to a separate validation set. However, using a separate validation may not be practical in limited data settings, which are a main focus of dataset optimization. To remedy this, we now show that we can instead optimize α by minimizing a leave-one-out cross-validation loss that only requires a training set:

$$L_{\text{LOO}}(\alpha) = \sum_{i=1}^{N} \ell(f_{\mathbf{w}_\alpha^{-i}}(\mathbf{x}_i), y_i), \tag{11}$$

where $\mathbf{w}_\alpha^{-i}$ are the optimal weights obtained by training with the loss eq. (5) on the entire dataset $\mathcal{D}$ *except* for the i-th sample $(\mathbf{x}_i, y_i)$. This may seem counter-intuitive, since we are optimizing the weights of the training samples using a validation loss defined on the training set itself. It is useful to recall that $\mathbf{w}_\alpha^{-i}$ minimizes the α-weighted L_2 loss on the training set (minus the i-th example):

$$\mathbf{w}_\alpha^{-i} = \arg\min_{\mathbf{w}} L_{\mathcal{D}}^{-i}(w, \mathcal{D}_\alpha) = \arg\min_{\mathbf{w}} \sum_{j \neq i} \alpha_j \|f_w(x_j) - y_j\|^2 + \lambda \|\mathbf{w}\|^2. \tag{12}$$

Meanwhile, α minimizes the *unweighted* validation loss in eq. (11). This prevents the existence of degenerate solutions for α.

Computing L_{LOO} naively would require training n classifiers, but fortunately, in the case of a linear classifier with the L_2 loss, a more efficient closed-form solution exists (Green & Silverman, 1993; Rifkin & Lippert, 2007). Generalizing those results to the case of a weighted loss, we are able to derive the following expression.

Proposition 3. *Define*

$$\mathbf{R}_\alpha = \mathbf{Z}^\top \sqrt{\mathbf{D}_\alpha} (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \sqrt{\mathbf{D}_\alpha} \mathbf{Z}$$

Then α-weighted LOOV predictions defined in eq. (12) *admit a closed-form solution:*

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) = \left[\frac{\mathbf{R}_\alpha \sqrt{\mathbf{D}_\alpha}\mathbf{Y} - \mathit{diag}(\mathbf{R}_\alpha)\sqrt{\mathbf{D}_\alpha}\mathbf{Y}}{\mathit{diag}(\sqrt{\mathbf{D}_\alpha} - \sqrt{\mathbf{D}_\alpha}\mathbf{R}_\alpha)} \right]_i, . \tag{13}$$

where $\mathit{diag}(\mathbf{A}) = [a_{11}, \ldots, a_{nn}]$ denotes the vector containing the diagonal of $\mathbf{A}$, and the division between vectors is element-wise.

Note that the prediction $f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i)$ on the i-th sample when training on all the other samples is a differentiable function of α. Composing eq. (13) in eq. (11), we compute the derivative $\nabla_\alpha L_{\text{LOO}}(\alpha)$, which allows us to optimize the cross-validation loss with respect to the sample weights, without the need of a separate validation set. We give the closed-form expression for $\nabla_\alpha L_{\text{LOO}}(\alpha)$ in the Appendix.

3.4 Dataset Optimization with DIVA

We can now apply the closed-form expressions for $\nabla_\alpha L_{\text{val}}(\alpha)$ and $\nabla_\alpha L_{\text{LOO}}(\alpha)$ for differentiable dataset optimization. We describe the optimization using L_{val}, but the same applies to L_{LOO}.

DIVA Reweight. The basic task consists in reweighting the samples of an existing dataset in order to improve generalization. This can curate a dataset by reducing the influence of outliers or wrong labels, or by reducing possible imbalances. To optimize the dataset weights, we use gradient descent in the form:

$$\alpha \leftarrow \alpha - \eta \nabla_\alpha L_{\text{val}}. \tag{14}$$

It is important to notice that L_{val} is an unbiased estimator of the test loss only at the first step, hence optimizing using eq. (14) for multiple steps can lead to over-fitting (see Appendix). Therefore, we apply only 1-3 gradient optimization steps with a relatively large learning rate $\eta \simeq 0.1$. This early stopping both regularizes the solution and decreases the wall-clock time required by the method. We initialize α so that $\alpha_i = 1$ for all samples.

DIVA Extend. The dataset gradient also allows us to extend an existing dataset. Given a core dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and an external (potentially noisy) data pool $\mathcal{E} = \{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i=N+1}^{N+M}$, we want to find the best samples from $\mathcal{E}$ to add to $\mathcal{D}$. For this we merge $\mathcal{D}$ and $\mathcal{E}$ in a single dataset and initialize α such that $\alpha_i = 1$ for samples of $\mathcal{D}$ and $\alpha_i = 0$ for samples of $\mathcal{E}$ (so that initially the weighted dataset matches $\mathcal{D}$). We then compute $\nabla_\alpha L_{\text{val}}(\alpha)$ to find the top k samples of $\mathcal{E}$ that have the largest negative value of $\nabla_\alpha L_{\text{val}}(\alpha)_i$, i.e., the samples that would give the largest reduction in validation error if added to the training set and add them to $\mathcal{D}$. This is repeated until the remaining samples in $\mathcal{E}$ all have positive value for the derivative (adding them would not further improve the performance).

Detrimental sample detection. The i-th component of $\nabla_\alpha L_{\text{val}}$ specifies the influence of the i-th sample on the validation loss. In particular, $(\nabla_\alpha L_{\text{val}})_i > 0$ implies that the sample increases the validation loss, hence it is detrimental (e.g., it is mislabeled or overly represented in the dataset). We can select the set of detrimental examples by thresholding $\nabla_\alpha L_{\text{val}}$:

$$\text{Detrimental}(\epsilon) = \big\{i : \ (\nabla_\alpha L_{\text{val}})_i \geq \epsilon\big\}. \tag{15}$$

4 Results

For our models we use standard residual architectures (ResNet) models pre-trained on ImageNet (Deng et al., 2009) and Places365 (Zhou et al., 2017). For our experiments on dataset optimization we consider datasets that are smaller than the large scale datasets used for pre-training as we believe they reflect more realistic conditions for dataset optimization. For our experiments we use the CUB-200 (Welinder et al., 2010), FGVC-Aircraft, (Maji et al., 2013), Stanford Cars (Krause et al., 2013), Caltech-256 (Griffin et al., 2007), Oxford Flowers 102 (Nilsback & Zisserman, 2008), MIT-67 Indoor (Quattoni & Torralba, 2009), Street View House Number (Netzer et al., 2011), and the Oxford Pets (Parkhi et al., 2012) visual recognition and classification datasets. In all experiments, we use the network as a fixed feature extractor, and train a linear classifier on top of the network features using the weighted L_2 loss eq. (5) and optimize the α weights using DIVA.

Dataset AutoCuration. We use DIVA Reweight to optimize the importance weights of samples from several fine-grain classification datasets. While the datasets have already been manually curated by experts to exclude out-of-distribution or mislabeled examples, we still observe that in all cases DIVA can further improve the test error of the model (Table 1). To understand how DIVA achieves this, in Figure 2 we show the most up-weighted (left) and down-weighted (right) examples on the FGVC Aircraft classification task Maji et al. (2013). We observe that DIVA tends to give more weight to clear, canonical examples, while it detects as detrimental (and hence down-weights) examples that contain multiple planes (making the label uncertain), or that do not clearly show the plane, or show non-canonical poses. We compare DIVA Reweight with two other re-weighting approaches:

Dataset	Original	DIVA Reweight	Chang et al. (2017)†	Ren et al. (2018)*	Gain
Aircrafts	57.58	**54.64**	70.48	81.82 (80.62)	+2.94
Cub-200	39.30	**36.93**	57.85	72.55 (75.35)	+2.36
MIT Indoor-67	32.54	**31.27**	37.84	64.48 (58.06)	+1.27
Oxford Flowers	20.23	**19.16**	22.82	48.80 (55.46)	+1.07
Stanford Cars	58.91	**56.31**	75.87	83.09 (84.50)	+2.56
Caltech-256	23.98	**21.29**	37.52	58.44 (52.77)	+2.69

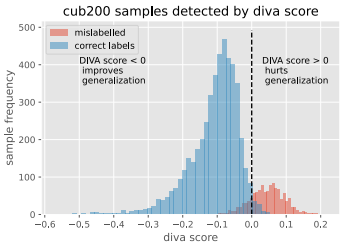
Table 1: Test error of DIVA Reweight to curate several fine-grain classification datasets. We use a ResNet-34 pretrained on ImageNet as feature extractor and train a linear classifier on top of the last layer. Note that DIVA Reweight can improve performance even on curated and noiseless datasets whereas other reweighting methods based on hard-coded rules may be detrimental in this case.

Ren et al. (2018), that applies re-weighting using information extracted from a separate validation gradient step, and Chang et al. (2017), which reweighs based on the uncertainty of each prediction (threshold-closeness weighting scheme). For Ren et al. (2018), we set aside 20% of the training samples as validation for the reweight step, but use all samples for the final training (in parentheses). We notice that both baselines, which are designed to reweight noisy datasets, underperform with respect to DIVA on noiseless datasets.

Dataset extension. We test the capabilities of DIVA Extend to extend a dataset with additional samples of the distribution. In Figure 4 and Table 2 (in the Appendix), we observe that DIVA is able to select the most useful examples and reaches an optimal performance generalization error using significantly less samples than the baseline uniform selection. Moreover, we notice that DIVA identifies a smaller subset of samples that provides better test accuracy than adding all the pool samples to the training set.

Detrimental sample detection. To test the ability of DIVA to detect training samples that are detrimental for generalization, we artificially introduce wrong labels in the dataset. In Section 3.4 we claimed that detrimental examples can be detected by looking at the samples for which the derivative $\nabla_\alpha L_{\text{LOO}}(\alpha)_i$ is positive. To verify this, in Figure 3 we plot the histogram of the derivatives for correct and mislabeled examples. We observe that most mislabeled examples have positive derivative. In particular, we can directly classify an example as mislabeled if the derivative is positive. In Figure 3 we report the F1 score and AUC obtained in a mislabeled sample detection task using the DIVA gradients.

Multi-modal learning. Recent multi-modal models such as CLIP (Radford et al., 2021) can embed both text and images in the same vector spaces. This allows to boost the performance on few-shot image classification tasks by also adding to the training set textual descriptions of the classes, such as the label name. However, training on label names may also hurt the performance, for example if the label name is not known by the CLIP model. To test this, we create a few-shot task by selecting 20 images per class from the Caltech-256. We then use DIVA Extend to select an increasing number of



Dataset	F1-score ($\epsilon = 0$)	AUC
Cub200	0.87	0.98
Aircrafts	0.68	0.90
MIT Indoor-67	0.86	0.98
Stanford Cars	0.75	0.93
Caltech-256	0.92	0.99
Oxford Flowers	0.83	0.97

Figure 3: **(Left)** Distribution of LOO DIVA gradients for correctly labelled and mislabelled samples in CUB-200 dataset (20% of the samples are mislabeled by replacing their label uniformly at random). **(Right) DIVA for outlier rejection.** We use DIVA on a ResNet-34 network linearization and detect mislabelled samples (outliers) in a dataset present with 20% label noise. Selection is based on $\nabla_\alpha(L_{\text{val}}(\mathbf{w}_\alpha))_i > \epsilon$.

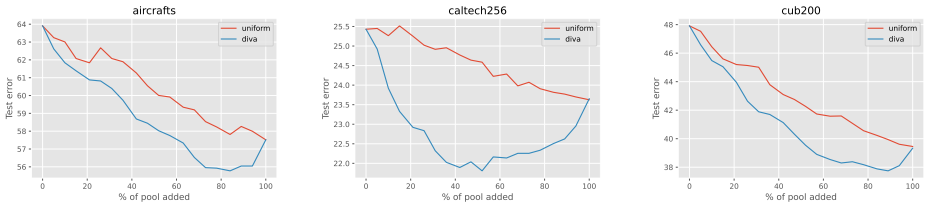


Figure 4: **DIVA Extend.** We show the test error achieved by the model as we extend a dataset with samples selected from a dataset pool using either DIVA Extend (red line) or uniform sampling (blue line). The pool set matches the same distribution as the training set. In all cases DIVA Extend outperforms uniform sampling and identifies subsets of the pool set with better performance than the whole pool. We also note that using only a subset selected by DIVA as opposed to using the whole pool, actually improves the test accuracy.

labels to add to the training set. In Figure 5 (right), we show that DIVA can select the beneficial label embeddings to add in order to improve the few-shot test performance. However, when forced to add all labels, including detrimental ones, the test error starts to increase again.

Data augmentation. To further test the versatility of DIVA, we qualitatively evaluate DIVA Reweight on the task of tuning the probabilities with which we apply a given data augmentation procedure. Let $t_1, \ldots, t_K$ be a set of data augmentation transformations. Let $\mathcal{D}^{t_k}$ be the result of applying the data augmentation t_k to $\mathcal{D}$. We can create an augmented dataset $\mathcal{D}^{\text{aug}} = \mathcal{D} \cup \mathcal{D}^{t_0} \cup \ldots \cup \mathcal{D}^{t_K}$, by merging all transformed datasets. We then apply DIVA Reweight on $\mathcal{D}^{\text{aug}}$ to optimize the weight α of the samples. Based on the updated importance weights we estimate the optimal probability with which to apply the transformation t_k as $p_k = (\sum_{i \in \mathcal{D}^{t_k}} \alpha_i)/(\sum_i \alpha_i)$. In particular we select common data augmentation procedures, horizontal flip and vertical flip, and we tune their probability on the Street View House Number, Oxford Flowers and the Oxford Pets classification tasks. We observe that DIVA assigns different probabilities to each transformation depending on the task (Figure 5): on the number classification task DIVA penalizes both vertical and horizontal flips, which may confuse different classes (such 2 and 5, 6 and 9). On an animal classification task (Oxford Pets) DIVA does not penalize horizontal flips, but penalizes vertical flips since they are out of distributions. Finally, on Flowers classification DIVA gives equal probability to all transformations (most flower pictures are frontal so all rotations and flips are valid).

5 Discussion

In this work we present a gradient-based method to optimize a dataset. In particular we focus on sample reweighting, extending datasets, and removing outliers from noisy datasets. We note that by developing the notion of a dataset derivative we are capable of improving dataset quality in multiple disparate problems in machine learning. The dataset derivative we present is given in closed-form and enables general reweighting operations on datasets based on desired differentiable validation losses. In cases where a set-aside validation loss is not available we show the use of the leave-one-out framework enables computing and optimizing a dataset "for free" and derive the first closed-form dataset derivative based on the LOO framework.

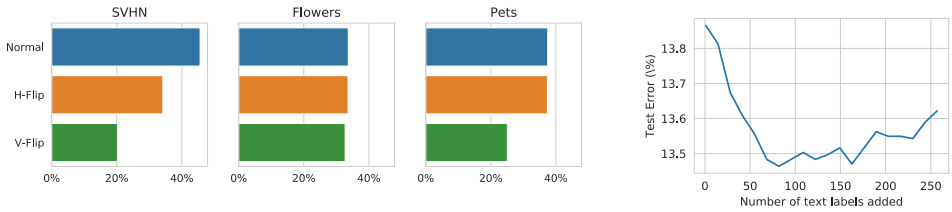


Figure 5: **(Left) DIVA can select the best data augmentation for each task.** We optimize the probability with which each data augment transformation is applied. DIVA optimizes the transformation to apply for the particular task. **(Right) Use of DIVA extend on a multi-modal task.** Selecting only the most beneficial text embeddings to use in a multi-modal classification task (as scored by DIVA) outperforms blindly using all available text embeddings.

REFERENCES

Alessandro Achille, Aditya Golatkar, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Lqf: Linear quadratic fine-tuning. *arXiv preprint arXiv:2012.11140*, 2020.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*, 2016.

Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning. *Lazy learning*, pp. 11–73, 1997.

Harkirat Singh Behl, Atilim Güneş Baydin, Ran Gal, Philip HS Torr, and Vibhav Vineet. Autosimulate:(quickly) learning synthetic data generation. In *European Conference on Computer Vision*, pp. 255–271. Springer, 2020.

Mauro Birattari, Gianluca Bontempi, and Hugues Bersini. Lazy learning meets the recursive least squares algorithm. *Advances in neural information processing systems*, pp. 375–381, 1999.

Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pp. 872–881. PMLR, 2019.

Gavin C Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *The 2006 IEEE international joint conference on neural network proceedings*, pp. 1661–1668. IEEE, 2006.

Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107, 2010.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30:1002–1012, 2017.

Chih-Yang Chen, Che-Han Chang, and Edward Y Chang. Hypernetwork-based augmentation. *arXiv preprint arXiv:2006.06320*, 2020.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2019.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.

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

Aditya Deshpande, Alessandro Achille, Avinash Ravichandran, Hao Li, Luca Zancato, Charless Fowlkes, Rahul Bhotika, Stefano Soatto, and Pietro Perona. A linearized framework and a new benchmark for model selection for fine-tuning. *arXiv preprint arXiv:2102.00084*, 2021.

Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1851–1860, 2017.

Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*, pp. 113–134. Springer, Cham, 2019.

Peter J Green and Bernard W Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Xia Hong, Sheng Chen, and Chris J Harris. A kernel-based two-class classifier for imbalanced data sets. *IEEE Transactions on neural networks*, 18(1):28–41, 2007.

Khalid M Hosny, Mohamed A Kassem, and Mohamed M Foaud. Skin cancer classification using deep learning and transfer learning. In *2018 9th Cairo International Biomedical Engineering Conference (CIBEC)*, pp. 90–93. IEEE, 2018.

Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *IEEE transactions on pattern analysis and machine intelligence*, 42(11):2781–2794, 2019.

Changha Hwang and Jooyong Shim. Geographically weighted least squares-support vector machine. *Journal of the Korean Data and Information Science Society*, 28(1):227–235, 2017.

Myunggwon Hwang, Yuna Jeong, and Wonkyung Sung. Data distribution search to select core-set for machine learning. In *Proceedings of the 9th International Conference on Smart Media & Applications (SMA 2020), Jeju, Korea*, pp. 17–19, 2020.

Simon Jenni and Paolo Favaro. Deep bilevel learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 618–633, 2018.

Yuna Jeong, Myunggwon Hwang, and Wonkyung Sung. Dataset distillation for core training set construction. 2020.

Mohsen Joneidi, Saeed Vahidian, Ashkan Esmaeili, Weijia Wang, Nazanin Rahnavard, Bill Lin, and Mubarak Shah. Select to better learn: Fast and accurate deep learning using data selection from nonlinear manifolds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7819–7829, 2020.

KrishnaTeja Killamsetty, Durga Sivasubramanian, Baharan Mirzasoleiman, Ganesh Ramakrishnan, Abir De, and Rishabh K. Iyer. GRAD-MATCH: A gradient matching based data subset selection for efficient learning. *CoRR*, abs/2103.00123, 2021. URL `https://arxiv.org/abs/2103.00123`.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

Buyu Li, Yu Liu, and Xiaogang Wang. Gradient harmonized single-stage detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8577–8584, 2019.

Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the hyperparameters for fine-tuning. *arXiv preprint arXiv:2002.11770*, 2020a.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy M. Hospedales, Neil Martin Robertson, and Yongxin Yang. DADA: differentiable automatic data augmentation. 2020b.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *arXiv preprint arXiv:1905.00397*, 2019.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Aoming Liu, Zehao Huang, Zhiwu Huang, and Naiyan Wang. Direct differentiable augmentation search. *arXiv preprint arXiv:2104.04282*, 2021.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 82–92, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Andrew W. Moore, Daniel J. Hill, and Michael P. Johnson. An empirical investigation of brute force to choose features, smoothers and function approximators. In *Computational Learning Theory and Natural Learning Systems*. MIT Press, 1992.

Romain Mormont, Pierre Geurts, and Raphaël Marée. Comparison of deep transfer learning strategies for digital pathology. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2262–2271, 2018.

Fangzhou Mu, Yingyu Liang, and Yin Li. Gradients as features for deep representation learning. *arXiv preprint arXiv:2004.05529*, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Natalia Nikolova, Rosa M Rodríguez, Mark Symes, Daniela Toneva, Krasimir Kolev, and Kiril Tenekedjiev. Outlier detection algorithms over fuzzy data with weighted least squares. *International Journal of Fuzzy Systems*, pp. 1–23, 2021.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracking gradient descent. *arXiv preprint arXiv:2002.08484*, 2020.

Tingwei Quan, Xiaomao Liu, and Qian Liu. Weighted least squares support vector machine local region method for nonlinear time series prediction. *Appl. Soft Comput.*, 10(2):562–566, March 2010. ISSN 1568-4946.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420. IEEE, 2009.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pp. 4334–4343. PMLR, 2018.

Edmar Rezende, Guilherme Ruppert, Tiago Carvalho, Fabio Ramos, and Paulo De Geus. Malicious software classification using transfer learning of resnet-50 deep neural network. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1011–1014. IEEE, 2017.

Ryan M Rifkin and Ross A Lippert. Notes on regularized least squares. 2007.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 761–769, 2016.

Mervyn Stone. Asymptotics for and against cross-validation. *Biometrika*, pp. 29–35, 1977.

Mishal Thapa, Sameer B Mulani, and Robert W Walters. Adaptive weighted least-squares polynomial chaos expansion with basis adaptivity and sequential adaptive sampling. *Computer Methods in Applied Mechanics and Engineering*, 360:112759, 2020.

Remi Trichet and Francois Bremond. Dataset optimization for real-time pedestrian detection. *IEEE access*, 6:7719–7727, 2018.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Wen Wen, Zhifeng Hao, and Xiaowei Yang. A heuristic weight-setting strategy and iteratively updating algorithm for weighted least-squares support vector regression. *Neurocomputing*, 71 (16-18):3096–3103, 2008.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

Da Xu, Yuting Ye, and Chuanwei Ruan. Understanding the role of importance weighting for deep learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=_WnwtieRHxM`.

Jiabin Xue, Jiqing Han, Tieran Zheng, Jiaxing Guo, and Boyong Wu. Hard sample mining for the improved retraining of automatic speech recognition. *arXiv preprint arXiv:1904.08031*, 2019.

Xi Yan, David Acuna, and Sanja Fidler. Neural data server: A large-scale search engine for transfer learning data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3893–3902, 2020.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

APPENDIX

We structure the appendix as follows: We present additional experiments in Section A and we describe the details of the experiments in Section B. In Section C we provide proofs for the propositions in the paper and additional discussion on the methods.

A ADDITIONAL EXPERIMENTS

Validation overfitting. When updating the dataset using the dataset derivative there is a risk of overfitting to the validation set after repeated applications of the derivative. Namely the validation loss is initially an unbiased estimate of the test loss yet after using it to update the dataset repeatedly it eventually will start overfitting. In our settings, we notice that when using a small number of gradient updates (< 5) and with step sizes $\eta \sim 0.1$ we are able to avoid overfitting and improve the *test* error when optimizing the dataset derivative based on the validation loss. In this experiment we present the final test and validation classification errors of optimized datasets. As we optimize with respect to the validation loss, it is indeed clear that the validation loss decreases dramatically yet more importantly are the effect of the test accuracy.

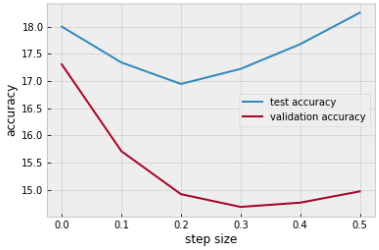


Figure 6: The test error decreases as the dataset is optimized with respect to the validation set until eventually overfitting commences. The validation set error decreases more significantly as the dataset is optimized directly on the validation set, yet for very large step-sizes the first order optimization becomes inaccurate. The plot uses the Caltech-256 dataset to illustrate the overfitting

DIVA Extend plots. In Figure 4 we report the results on all the remaining datasets following the set-up of Figure 4. In Table 2 we report the accuracy of DIVA Extend and uniform sampling on the various datasets when adding 50% of samples from the pool.

Dataset	DIVA Extend	Uniform	Improvement
Aircrafts	58.00	60.01	+2.01
Cub-200	39.42	42.29	+2.87
MIT Indoor-67	32.54	33.73	+1.19
Oxford Flowers	20.56	23.29	+2.73
Stanford Cars	60.37	62.45	+2.09
Caltech-256	21.97	24.55	+2.59

Table 2: Results of using DIVA Extend to select the best samples to extend several fine-grain classification datasets. We train a linear classifier on top of a ResNet-34 pretrained on ImageNet, and compare the test performance when extending the target training dataset with 50% of the pool samples selected either uniformly at random or via DIVA Extend.

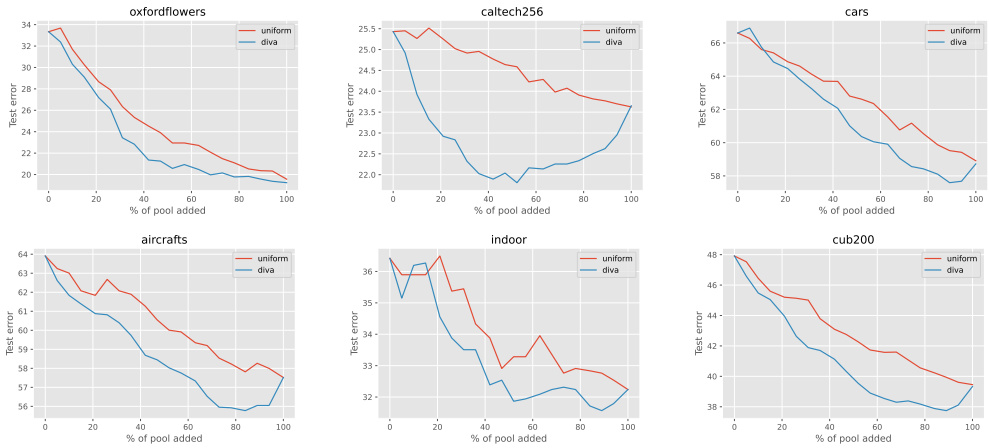


Figure 7: Same plots as Figure 4 on the other fine-grained datasets.

B EXPERIMENTAL DETAILS

Dataset details For our experiments we utilize several fine-grain classification datasets from the computer vision community that are standard for fine-tuning image classification tasks (CUB-200 Welinder et al. (2010), FGVC-Aircraft, Maji et al. (2013), Stanford Cars Krause et al. (2013), Caltech-256 Griffin et al. (2007), Oxford Flowers 102 Nilsback & Zisserman (2008), MIT-67 Indoor Quattoni & Torralba (2009), Street View House Number Netzer et al. (2011), and the Oxford Pets Parkhi et al. (2012)). Some of the datasets do not follow a default train-test split and we use the following splits commonly used in the literature for the datasets,

- Oxford Flowers 102: We use the original 1020 images in the training split without incorporating the default validation set.
- Caltech-256: We split the dataset into a training set with 60 images from each class for training, and use remaining data for testing.

Pre-training setup For the pre-trained networks we use for fine-tuning, we use the pre-trained configurations available on PyTorch's torchvision package. In particular the reference networks are pre-trained using the ImageNet Deng et al. (2009) dataset. The images embedded by the network are pre-processed via standard resizing and center cropping (256 resize, followed by a 224 cropping).

Regularization parameter λ: To get the best unweighted dataset baseline to compare with the optimized dataset, for each of the un-optimized original datasets, we first search for optimal λ values in $\lambda \in \{2^n$ for $n \in \{-20, -19, \ldots 4\}\}$ to measure the classifier's performance. After selecting optimal λ we proceed with dataset optimization with the optimal λ values. Note that DIVA does not require λ to be optimal and improvements are even more significant for un-optimized λ.

Dataset Derivative Computation In Section C we derive the closed-form dataset derivatives used for DIVA. We computed the closed-form solution analytically and we verified our results using automatic differentiation tools on large number of conditions including synthetic and real data. This has been another method to verify the correctness of the derivative formulas.

DIVA experimental details

DIVA Extend For Table 2 and Figure 4 we first split the original training set into 50% training subset and 50% pool subset that will be used to selectively extend the training subset with DIVA or other extension approaches. We run DIVA Extend LOO and uniform sampling to add pool samples to the training set. In both settings we incrementally extend the training subset from the pool (For DIVA, by selecting samples with top DIVA score) in each step we extend an equal number of samples ((# pool samples) // (# number of steps)). In the figure we present the test error as a function of training set size and compare DIVA sample selection with selecting the same number of samples at uniform from the pool set. In Table 2, we present the improvement in test accuracy at the 50% extend mark

of the pool set (e.g. extending 25% of the original training set) between DIVA and uniform sample selection of the same number of samples. For both experiments we use the ResNet-34 architecture.

DIVA Reweigh In Table 1 we use DIVA Reweight LOO with the same parameters for all of the datasets we consider. The DIVA parameters we use are $K = 4$ for the number of steps and $\eta = 0.15$ for the step-size. As with DIVA Extend, we use ResNet 34 for the architecture for the representation.

DIVA validation loss We find the cross entropy loss to work better as the loss function applied to the validation predictions (both in LOO and regular validation). Further for LOO we find it crucial to apply the validation loss only on mis-classified LOO predictions to improve the test accuracy of the model, this can be interpreted as the cross entropy loss with a "hard margin hinge" loss.

Outlier rejection For the results presented in the side-by-side Table and Figure 3 we apply 20% random label noise to each class in the dataset and use DIVA LOO to compute the normalized DIVA gradient (DIVA score) of each sample. The F1 score reports classification by thresholding with $\epsilon = 0$ and the AUC is computing by thresholds spanning detection of no samples, to detection of all samples.

C PROOFS OF PROPOSITIONS

We write the proposition statements for convenience.

Proposition (Model-Dataset Derivative $\nabla_\alpha \mathbf{w}_\alpha$)**.** *For the ridge regression problem eq.* (5) *and* $\mathbf{w}_\alpha$ *defined as in eq.* (6)*, define*

$$\mathbf{C}_\alpha = (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1}. \tag{16}$$

Then the Jacobian of $\mathbf{w}_\alpha$ *with respect to* α *is given by*

$$\nabla_\alpha \mathbf{w}_\alpha = \mathbf{Z}\mathbf{C}_\alpha \circ \big((\mathbf{I} - \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha)\mathbf{Y}\big). \tag{17}$$

Proof of Proposition 1:
We recall the settings of the problem are $\mathbf{Z} \in \mathbb{R}^{n \times m}$, $\mathbf{Y} \in \mathbb{R}^{n \times k}$, $\mathbf{w}_\alpha \in \mathbb{R}^{m \times k}$ and the importance weights $\alpha \in \mathbb{R}^n$. Here n is the number of samples, m is the number of parameters for each output of the model, and k is the number of classes (represented via the one-hot convention).

The derivation of $\nabla_\alpha \mathbf{w}_\alpha$ we present is computational in nature and without the loss of generality we consider the derivative for a single output class $k = 1$ (one-vs-all classification naturally extends). For the single class settings, the relevant dimensions are $\mathbf{w}_\alpha \in \mathbb{R}^m$ and $\nabla_\alpha \mathbf{w}_\alpha \in \mathbb{R}^{m \times n}$ (for numerator layout convention of the derivative). To further simplify we consider the derivative entrywise for single index α_r,

$$\frac{\partial \mathbf{w}_\alpha}{\partial \alpha_r} \in \mathbb{R}^{m \times 1}.$$

The closed-form solution is,

$$\mathbf{w}_\alpha = \mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}.$$

using chain rule,

$$\frac{\partial \mathbf{w}_\alpha}{\partial \alpha_r} = \frac{\partial \mathbf{C}_\alpha}{\partial \alpha_r} \times \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y} + \mathbf{C}_\alpha \times \frac{\partial \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}}{\partial \alpha_r}.$$

We compute the two parts of the derivative in turn:
Let $\mathbf{K}_\alpha = (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})$ so that $\mathbf{K}_\alpha = \mathbf{C}_\alpha^{-1}$. Then

$$\frac{\partial \mathbf{K}_\alpha}{\partial \alpha_r} = \frac{\partial}{\partial \alpha_r}(\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I}) = \mathbf{Z}_{r,:}^\top \mathbf{Z}_{r,:}$$

where $\mathbf{Z}_{r,:} \in \mathbb{R}^{1 \times m}$ is the rth row of $\mathbf{Z}$. Next note,

$$\mathbf{C}_\alpha \mathbf{K}_\alpha = \mathbf{I}$$

differentiating both sides,

$$\frac{\partial \mathbf{C}_\alpha \mathbf{K}_\alpha}{\partial \alpha_r} = \mathbf{0}.$$

Applying chain rule we get,

$$\frac{\partial \mathbf{C}_\alpha \mathbf{K}_\alpha}{\partial \alpha_r} = \frac{\partial \mathbf{C}_\alpha}{\partial \alpha_r}\mathbf{K}_\alpha + \mathbf{C}_\alpha \frac{\partial \mathbf{K}_\alpha}{\partial \alpha_r},$$

rearranging, substituting $\frac{\partial \mathbf{K}_\alpha}{\partial \alpha_r}$, and multiplying to the right by $\mathbf{C}_\alpha$

$$\frac{\partial \mathbf{C}_\alpha}{\partial \alpha_r} = -\mathbf{C}_\alpha \mathbf{Z}_{r,:}^\top \mathbf{Z}_{r,:}\mathbf{C}_\alpha.$$

Next, by direct computation $\frac{\partial \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}}{\partial \alpha_r}$ satisfies

$$\frac{\partial \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}}{\partial \alpha_r} = y_r \cdot \mathbf{Z}_{r,:}^\top.$$

Combining the original terms we have

$$\begin{aligned}\frac{\partial \mathbf{w}_\alpha}{\partial \alpha_r} &= -\big(\mathbf{C}_\alpha \mathbf{Z}_{r,:}^\top \mathbf{Z}_{r,:}\mathbf{C}_\alpha\big) \times \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y} + y_r \cdot \mathbf{C}_\alpha \times \mathbf{Z}_{r,:}^\top \\ &= \mathbf{C}_\alpha \mathbf{Z}_{r,:}^\top \big(y_r - \mathbf{Z}_{r,:}\mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}\big).\end{aligned}$$

Now the full derivative is written as

$$\boxed{\nabla_\alpha \mathbf{w}_\alpha = \mathbf{Z}\mathbf{C}_\alpha \circ \big((\mathbf{I} - \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha)\mathbf{Y}\big)}$$

Proposition (Validation Loss Dataset Derivative)**.** *Define* $\mathbf{L}$ *as the matrix of the loss function derivative with respect to network training outputs as,*

$$\mathbf{L} = \left[\frac{\partial \ell}{\partial f}(f_{\mathbf{w}_\alpha}(\mathbf{x}_1), y_1), \cdots \frac{\partial \ell}{\partial f}(f(\mathbf{x}_N), y_N)\right].$$

Then the dataset derivative of the importance weights with respect to final validation loss is given by

$$\nabla_\alpha L_{val}(\mathbf{w}_\alpha) = \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top \times \big(\mathbf{L}^\top \mathbf{Y}^\top (\mathbf{I} - \mathbf{D}_\alpha \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top)\big). \tag{18}$$

Proof of Proposition 2:
This follows from the chain rule combined with simplification of broadcasting terms. We again consider the single output settings with the single coordinate derivative $\frac{\partial L_{\text{val}}}{\partial \alpha_r}$ which is given as,

$$\frac{\partial L_{\text{val}}}{\partial \alpha_r} = \nabla_{\mathbf{w}} L_{\text{val}} \frac{\partial \mathbf{w}_\alpha}{\partial \alpha_r}.$$

With

$$\begin{aligned}&= \sum_{i=1}^{n} \mathbf{L}_{:,i} \times \mathbf{z}_i^\top \\ &= \mathbf{L}^\top \mathbf{Z}.\end{aligned}$$

Therefore

$$\big(\nabla_\alpha L_{\text{val}}(\mathbf{w}_\alpha)\big)_i = \sum_{j,k} \big(\nabla_\alpha \mathbf{w}_\alpha\big)_{i,j,k} (\mathbf{L}^\top \mathbf{Z})_{j,k}.$$

Now $\mathbf{L}, \mathbf{Z}$ can be separated into the two terms of $\nabla_\alpha \mathbf{w}_\alpha$,

$$\boxed{\nabla_\alpha L_{\text{val}}(\mathbf{w}_\alpha) = \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top \times \big(\mathbf{L}^\top \mathbf{Y}^\top (\mathbf{I} - \mathbf{D}_\alpha \mathbf{Z}\mathbf{C}_\alpha \mathbf{Z}^\top)\big)} \tag{19}$$

Next we consider the derivations for the Leave One Out (LOO) framework. In the LOO framework one applies cross-validation to a training set $\{(\mathbf{z}_1, y_1), \ldots (\mathbf{z}_n, y_n)\}$ by running n-fold cross validation, where in each fold, the ith sample $(\mathbf{z}_i, y_i)$ is taken out and is used for validation while the optimal classifier is solved for the remaining of the training task,

$$\mathbf{w}_{-i}, = \arg\min_{\mathbf{w}} \sum_{j \neq i} \ell(f(\mathbf{z}_j), y_j). \tag{20}$$

Then the LOO prediction at the ith index is defined as $(\mathbf{f}_{\text{LOO}})_i = f_{\mathbf{w}_{-i}}(\mathbf{z}_i)$. Below we prove the LOO predictions can be written in closed-form without explicit cross validation calculations.

Proposition 4 (Closed-form LOO prediction vector)**.** *Define the LOO vector predictions as,*

$$\mathbf{f}_{\text{LOO}} = [f_{\mathbf{w}_{-1}}(\mathbf{z}_1), \dots, f_{\mathbf{w}^{-n}}(\mathbf{z}_n)]^\top$$

and define

$$\mathbf{R} = \mathbf{Z}^\top(\mathbf{Z}^\top\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}$$

then for the learning task eq. (20) *LOO predictions are given as*

$$\mathbf{f}_{\text{LOO}} = \frac{\mathbf{R}\mathbf{y} - \textit{diag}(\mathbf{R})\mathbf{y}}{\mathbf{I} - \textit{diag}(\mathbf{R})}. \tag{21}$$

Proof of Proposition 4:
The proof is reproduced from Rifkin & Lippert (2007) for completeness.
Without the loss of generality we derive the LOO prediction of $\mathbf{z}_n$. Namely given, $\{(\mathbf{z}_1, y_1), \dots (\mathbf{z}_n, y_n)\}$ we use $\{(\mathbf{z}_1, y_1), \dots (\mathbf{z}_{n_1}, y_{n-1})\}$ for training and validate using $\{(\mathbf{z}_n, y_n\}$. Denote $\mathbf{w}^{-n}$ as be the optimal solution to this training task with regularization parameter λ and define the (currently unknown) LOO prediction as

$$q = f_{\mathbf{w}^{-n}}(\mathbf{z}_n).$$

We define the modified learning task consisting of $\{(\mathbf{z}_1, y_1), \dots (\mathbf{z}_{n-1}, y_{n-1}), (\mathbf{z}_n, q)\}$ where we added the data point $(\mathbf{z}_n, q)$. Note that the optimal solution with λ regularization to the modified learning task is again $\mathbf{w}^{-n}$ since q is taken to have a zero residual. therefore the solution to the modified learning problem can be written in closed-form as,

$$\mathbf{w}^{-n} = (\mathbf{Z}^\top\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^\top[\mathbf{y}_{1:n-1}, q]^\top.$$

Using $\mathbf{w}^{-n}$ we write the equation for the LOOV prediction q,

$$q = \langle \mathbf{z}_n, \mathbf{w}^{-i} \rangle = \mathbf{z}_n^\top(\mathbf{Z}^\top\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^\top[\mathbf{y}_{1:n-1}, q]^\top.$$

Let $\mathbf{R} = \mathbf{Z}(\mathbf{Z}^\top\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^\top$ then we have

$$q = \mathbf{z}_n^\top\mathbf{w}_{-i} = \mathbf{R}_{n,:}[\mathbf{Y}_{1:n-1}, q]^\top$$

and by adding and subtracting $\mathbf{R}_{:,n}$ multiplied by $[0, y_n]^\top$ we get,

$$q = \mathbf{z}_n^\top\mathbf{w}_{-i} = \mathbf{R}_{:n-1,n}[\mathbf{Y}_{1:n-1}]^\top + \mathbf{R}_{:,n}[0, q] + \mathbf{R}_{:,n}[0, y_n]^\top - \mathbf{R}_{:,n}[0, y_n]^\top.$$

Re-arranging we have

$$q - \mathbf{R}_{nn}q = \mathbf{R}_{:,n}\mathbf{y} - y_n\mathbf{R}_{n,n}$$

Solving for q we get,

$$q = \frac{\mathbf{R}_{:,n}\mathbf{y} - y_n\mathbf{R}_{n,n}}{1 - \mathbf{R}_{n,n}}$$

And without the loss of generality the full prediction vector is given as,

$$\boxed{\mathbf{f}_{\text{LOO}} = \frac{\mathbf{R}\mathbf{y} - \text{diag}(\mathbf{R})\mathbf{y}}{\text{diag}(\mathbf{I} - \mathbf{R})}} \tag{22}$$

Given the closed-form LOOV expression we may use $\mathbf{f}_{\text{LOO}}$ for the validation loss to compute the dataset derivative on L_{val} without any additional validation data. While this may seem contradictory as we are optimizing the dataset validated via the weighting duality between sample loss weighting and data scaling, we define the leave one out value predictions in the weighted dataset settings and evaluate on the original (unweighted) data points as,

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) \tag{23}$$

with,

$$\mathbf{w}_\alpha^{-i} = \arg\min_{\mathbf{w}} \sum_{j \neq i} \alpha_j \ell(f(\mathbf{z}_j), y_j). \tag{24}$$

Therefore the α-weighted LOO term is $f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i)$ is faithful to the original distribution despite being trained with the weighted loss eq. (24). We in fact are able to show that α-weighted LOO formulation also admits a closed-form solution that satisfies our definition and for DIVA LOO we utilize the derivative of the closed-form to optimize the dataset.

Proposition. *Define*

$$\mathbf{R}_\alpha = \mathbf{Z}^\top \sqrt{\mathbf{D}_\alpha}(\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \sqrt{\mathbf{D}_\alpha}\mathbf{Z}$$

Then the α-weighted LOOV predictions defined in eq. (12) *admit a closed-form solution:*

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) = \left[\frac{\mathbf{R}_\alpha \sqrt{\mathbf{D}_\alpha}\mathbf{Y} - \mathit{diag}(\mathbf{R}_\alpha)\sqrt{\mathbf{D}_\alpha}\mathbf{Y}}{\mathit{diag}(\sqrt{\mathbf{D}_\alpha} - \sqrt{\mathbf{D}_\alpha}\mathbf{R}_\alpha)} \right]_i, \tag{25}$$

Further the LOO dataset derivative is well-defined and satisfies the following gradient condition,

$$\mathit{diag}(\nabla_\alpha \mathbf{f}_{\mathit{LOO}}) = \mathbf{0}. \tag{26}$$

The gradient condition $\mathrm{diag}(\nabla_\alpha \mathbf{f}_{\mathrm{LOO}}) = \mathbf{0}$ implies that the LOO prediction at $\mathbf{z}_i$ does not depend on α_i and ensures that the closed-form solution is well defined in the α-weighted settings and is differentiable.

Proof of Proposition 3:

The proof builds on Proposition 4 for the weighted settings.

In general since LOO expression describes the weighting problem, it must be shown that the introduction of the weights do not break the argument of Proposition 4. Considering the optimization problem in eq. (12). We also use the duality between the loss weights and data scaling to note that $\mathbf{w}_\alpha^- i$ can be derived by considering the unweighted LOO eq. (20) with the modified dataset,

$$\{(\sqrt{\alpha_1}\mathbf{z}_1, \sqrt{\alpha_1}y_1), \ldots (\sqrt{\alpha_n}\mathbf{z}_n, \sqrt{\alpha_n}y_n)\}. \tag{27}$$

Indeed in this settings with the newly defined data the derivation in Proposition 4 of the final prediction vector of the LOO entries holds with the same optimal solution $\mathbf{w}_\alpha$ due to the duality between data scaling and loss weights. Nonetheless the closed-form LOO predictions $\mathbf{f}_{\mathrm{LOO}}$ are evaluated at the data-points $\sqrt{\alpha_i}\mathbf{z}_i$. Since the model is linear the final predictions at the original data-points of the weighted training settings are written as $\mathbf{f}_{\mathrm{LOO}}/\sqrt{\alpha}$.

Noting the weighted training problem can be expressed as $\tilde{\mathbf{Y}} = \sqrt{\mathbf{D}_\alpha}\mathbf{Y},\ \tilde{\mathbf{Z}} = \sqrt{\mathbf{D}_\alpha}\mathbf{Z}$ we use Proposition 4 to write analogously

$$\mathbf{R}_\alpha = \mathbf{Z}^\top \sqrt{\mathbf{D}_\alpha}(\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \sqrt{\mathbf{D}_\alpha}\mathbf{Z} \tag{28}$$

and divide by $\sqrt{\alpha}$ by multiplying the denominator by $\sqrt{\mathbf{D}_\alpha}$,

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) = \left[\frac{\mathbf{R}_\alpha \sqrt{\mathbf{D}_\alpha}\mathbf{Y} - \mathrm{diag}(\mathbf{R}_\alpha)\sqrt{\mathbf{D}_\alpha}\mathbf{Y}}{\mathrm{diag}(\sqrt{\mathbf{D}_\alpha} - \sqrt{\mathbf{D}_\alpha}\mathbf{R}_\alpha)} \right]_i, \tag{29}$$

Since the derivation at the ith index of the weighted LOO prediction, $(\mathbf{f}_{\mathrm{LOO}})_i = (\mathbf{w}_\alpha^{-i})^\top \mathbf{z}_i$ is entirely independent of α_i, we have

$$\frac{\partial (f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i))_i}{\partial \alpha_i} = 0.$$