Forgetting Outside the Box
Supplementary Material

Aditya Golatkar, Alessandro Achille, and Stefano Soatto

In the Supplementary Material we:

- Appendix A: Provide implementation details for our scrubbing procedure;
- Appendix B: Show further experiments on more datasets (CIFAR-10, Lacuna, TinyImagenet) and models (AllCNN, ResNet).
- Appendix C: Provide proofs for all propositions and equations in the paper;

A Experimental Details

We train our models with SGD (learning rate $\eta = 0.01$ and momentum $m = 0.9$) using weight decay ($\lambda = 0.1$). All models are trained to convergence (we stop the training 5 epochs after the model achieves zero training error).

Pre-training and weight-decay. In all cases, we use pre-trained models (we pre-train the models on CIFAR-100/Lacuna-100/TinyImageNet (first 150 classes) respectively), and denote by w_0 the pre-trained weight configuration. One important point is that we change the weight decay regularization term from the standard $\|w\|_2^2$ (which pulls the weights toward zero) to $\|w - w_0\|_2^2$ (which pulls the weights toward the initialization). This serves two purposes, (i) It ensures that the weights remain close to the initialization (in our case, a pre-trained network). This further helps the weight during training to remain in the neighborhood of the initialization where the the linear approximation (NTK) is good; (ii) With this change, the training dynamics of the weights/activations of a linearized network only depend on the relative change in the weights from its initial value (see [27, Section 2.2] for more details).

NTK matrix, weight decay and cross-entropy. For clarity, in Section 4 and Proposition 2 we only considered a unregularized MSE regression problem. To apply the theory to the more practical case of a classification cross-entropy loss with weight-decay regularization, we need the following changes.

First, using weight decay the NTK matrix becomes $\Theta = \nabla f_0(\mathcal{D}) \nabla f_0(\mathcal{D})^T + \lambda I$, where λ is the weight decay coefficient. Second, when using the cross-entropy loss, the gradients $\nabla_{f_t^{\text{lin}}(\mathcal{D})} L$ of the loss function can be approximated as $\nabla_{f_w(x)} L \approx \nabla_{f_{w_0(x)}} L + H_{f_{w_0}(x)}(f_w(x) - f_{w_0}(x))$, where $H_{f_{w_0}(x)}$ is the Hessian of the loss with respect to the output activations. With these two together, we obtain $\Theta = H_{f_{w_0}(x)} \nabla f_0(\mathcal{D}) \nabla f_0(\mathcal{D})^T + \lambda I$ as the NTK matrix to use in our setting.

We did however notice that replacing $H_{f_{w_0}(x)}$ with the identity matrix I works better in practice. This may be due to $H_{f_{w_0}(x)}$ estimating the wrong curvature when the softmax saturates. Also we found that — while in principle identical as long as the network remains in the linear regime — linearizing

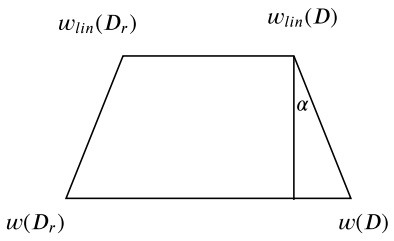

Fig. 5: Isosceles Trapezium Trick: $\|w(\mathcal{D}_r) - w(\mathcal{D})\| = \|w_{\text{lin}}(\mathcal{D}_r) - w_{\text{lin}}(\mathcal{D})\| + 2\sin\alpha\|w_{lin}(\mathcal{D}) - w(\mathcal{D})\|$. This allows us to match outputs of the linear dynamic model with the real output, without having to match the effective learning rate of the two, and while being more robust to wrong estimation of the curvature by the linearized model.

around $w(\mathcal{D})$ provides a better estimate of the scrubbing direction compared to linearizing around w_0.

Trapezium trick. We observe that the linearized dynamics of in eq. (8) and eq. (9) correctly approximate the training direction but they usually undershoot and give a smaller norm solution than SGD. This may be due to difficulty in matching the learning rate of continuous gradient descent and discrete SGD. To overstep these issues in a robust way, we use the following simple "trapezium trick" (Figure 5) to renormalize the scrubbing vector obtained with the linear dynamics: Instead of trying to predicting the unknown $w(\mathcal{D}_r)$ directly using the scrubbing vector suggested by eq. (10), we compute the two final points of the linearized dynamics $w_{\text{lin}}(\mathcal{D})$ and $w_{\text{lin}}(\mathcal{D}_r)$, and approximate $w(\mathcal{D}_r)$ by constructing the isosceles trapezium in Figure 5. Effectively, this rescales the ideal linearized forgetting direction $w_{\text{lin}}(\mathcal{D}_r) - w_{\text{lin}}(\mathcal{D})$ to correct for the undershooting.

B Additional Experiments

In Figure 6 we use a PCA projection to show the geometry of the loss landscape after convergence and the training paths obtained by training the weights on the full dataset $\mathcal{D}$ or only on $\mathcal{D}_r$. We observe that the loss landscape around the pretraining point is smooth and almost convex. Moreover, the two training paths remain close to each other. This supports our choice of using a simple linearized approximation to compute the shift that jumps from one path to the other.

In Figure 7 and Figure 8 we show additional experiments on several architectures (ResNet-18, All-CNN) and datasets (Lacuna, CIFAR-10, TinyImageNet). In all cases we observe a similar qualitative behavior to the one discussed in the paper.

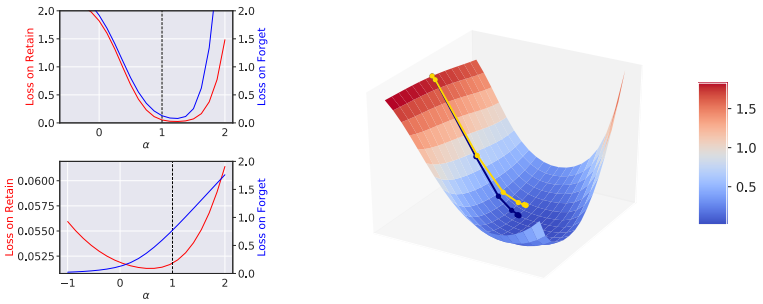

Fig. 6: **(Right) The loss landscape and training dynamics after pre-training are smooth and regular.** This justifies our linearization approach to study the dynamics. The black and yellow lines are the training paths on $\mathcal{D}$ and $\mathcal{D}_r$ respectively. Notice that they remain close. **(Upper left) Loss along the line** joining the model at initialization ($\alpha = 0$) and the model after training on $\mathcal{D}$ ($\alpha = 1$) (the black path). **(Lower left)** Loss along the line joining the end point of the two paths ($\alpha = 0$ and 1 respectively), which is the ideal scrubbing direction.

C Proofs

Markov chain in Section 3.1. We consider the retain set $\mathcal{D}_r$ (not shown) as an observed random variable, while the cohort to forget $\mathcal{D}_f$ is an hidden variable sampled randomly from the data distribution. The directed edge $\mathcal{D}_f \to w$ in the Markov chain derives from the fact that we first sample $\mathcal{D}_f$, to obtain the full complete training set $\mathcal{D} = \mathcal{D}_r \sqcup \mathcal{D}_f$, and then train the network on $\mathcal{D}$ to obtain the weights w.

Proof of Lemma 1. We have the following upper-bound for $I(\mathcal{D}_f; f_{S(w)}(\mathbf{x}))$:

$$
\begin{aligned}
I(\mathcal{D}_f; f_{S(w)}(\mathbf{x})) &= I(\mathcal{D}_f; f_{S(w)}(\mathbf{x})) \\
&= \mathbb{E}_{\mathcal{D}_f}\left[\mathrm{KL}\left(p(f_{S(w)}(\mathbf{x})|\mathcal{D}_f \cup \mathcal{D}_r) \,\|\, p(f_{S(w)}(\mathbf{x})|\mathcal{D}_r)\right)\right] \\
&= \mathbb{E}_{\mathcal{D}_f}\mathbb{E}_{p(f_{S(w)}(\mathbf{x})|\mathcal{D}_f \cup \mathcal{D}_r)}\left[\log \frac{p(f_{S(w)}(\mathbf{x})|\mathcal{D}_f \cup \mathcal{D}_r)}{p(f_{S(w)}(\mathbf{x})|\mathcal{D}_r)}\right] \\
&= \mathbb{E}_{\mathcal{D}_f}\mathbb{E}_{p(f_{S(w)}(\mathbf{x})|\mathcal{D}_f \cup \mathcal{D}_r)}\left[\log \frac{p(f_{S(w)}(\mathbf{x})|\mathcal{D}_f \cup \mathcal{D}_r)}{p(f_{S(w)}(\mathbf{x})|\mathcal{D}_r)}\right. \\
&\qquad\qquad \left. + \log \frac{p(f_{S_0(w)}(\mathbf{x})|\mathcal{D}_r)}{p(f_{S_0(w)}(\mathbf{x})|\mathcal{D}_r)}\right] \\
&= \mathbb{E}_{\mathcal{D}_f}\left[\mathrm{KL}\left(p(f_{S(w)}(\mathbf{x})|\mathcal{D}_f \cup \mathcal{D}_r) \,\|\, p(f_{S_0(w)}(\mathbf{x})|\mathcal{D}_r)\right)\right] \\
&\qquad\qquad - \mathrm{KL}\left(p(f_{S(w)}(\mathbf{x})|\mathcal{D}_r) \,\|\, p(f_{S_0(w)}(\mathbf{x})|\mathcal{D}_r)\right) \\
&\leq \mathbb{E}_{\mathcal{D}_f}\left[\mathrm{KL}\left(p(f_{S(w)}(\mathbf{x})|\mathcal{D}_f \cup \mathcal{D}_r) \,\|\, p(f_{S_0(w)}(\mathbf{x})|\mathcal{D}_r)\right)\right]
\end{aligned}
$$

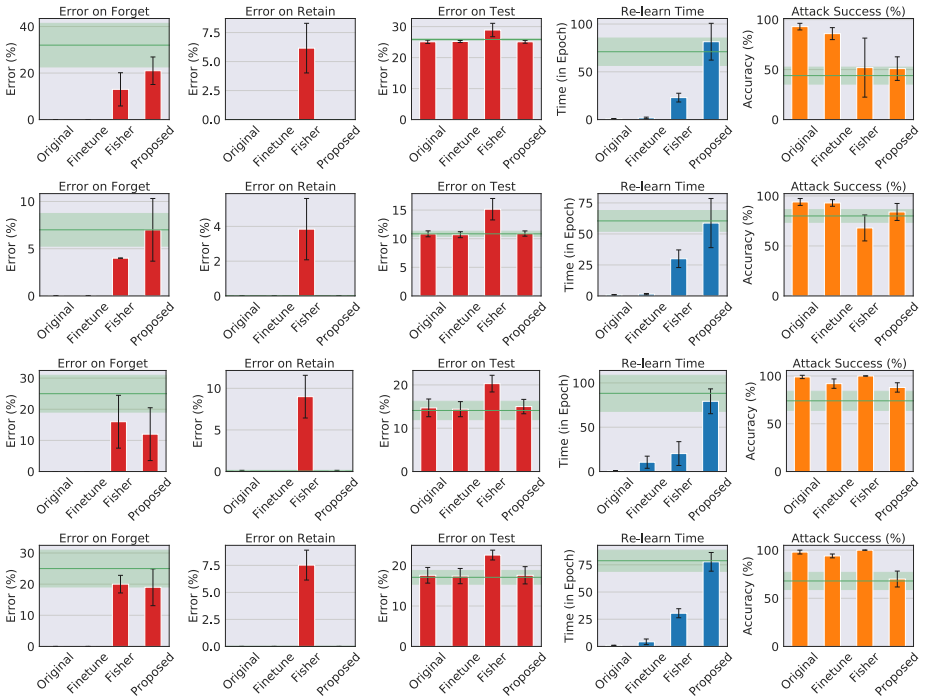

Fig. 7: Same experiment as Figure 2 for different architectures and datasets. **(Row 1)**: ResNet-18 on CIFAR, **(Row 2)**: ResNet-18 on Lacuna, **(Row 3)**: All-CNN on TinyImageNet and **(Row 4)**: ResNet-18 on TinyImageNet. In all the experiments we observe that for different readout functions the proposed method lies in the green (target) region.

where the last inequality follows from the fact that KL-divergence is always non-negative.

Proof of Lemma 2. We have the inequalities:

$$\begin{aligned} I(\mathcal{D}_f; f_{S(w)}(\mathbf{x})) &\leq \mathbb{E}_{\mathcal{D}_f}\left[\mathrm{KL}\left(p(f_{S(w_{\mathcal{D}})}(\mathbf{x}))\,\|\,p(f_{S_0(w_{\mathcal{D}_r})}(\mathbf{x}))\right)\right] \\ &\leq \mathbb{E}_{\mathcal{D}_f,\epsilon}\Big[\mathrm{KL}\left(p(f_{S(w_{\mathcal{D}})}(\mathbf{x}))\,\|\,p(f_{S_0(w_{\mathcal{D}_r})}(\mathbf{x}))\right)\Big] \end{aligned}$$

where the first inequality comes from Lemma 1, and the second inequality is from [16, Proposition 2].

Proof of eq. (5). The activations of a scrubbed network (using the Gaussian scrubbing procedure in eq. (4)) for a given sample $\mathbf{x}$ are given by $f_{h(w)+n}(\mathbf{x})$, where $n \sim N(0, \Sigma)$. By linearizing the activations (using NTK formalism) around $h(w)$, we obtain the distribution of the scrubbed activations:

$$f_{h(w)+n}(\mathbf{x}) \sim N(f_{h(w)}(\mathbf{x}), \nabla_w f_{h(w)}(\mathbf{x}) \Sigma \nabla_w f_{h(w)}(\mathbf{x})^T)$$

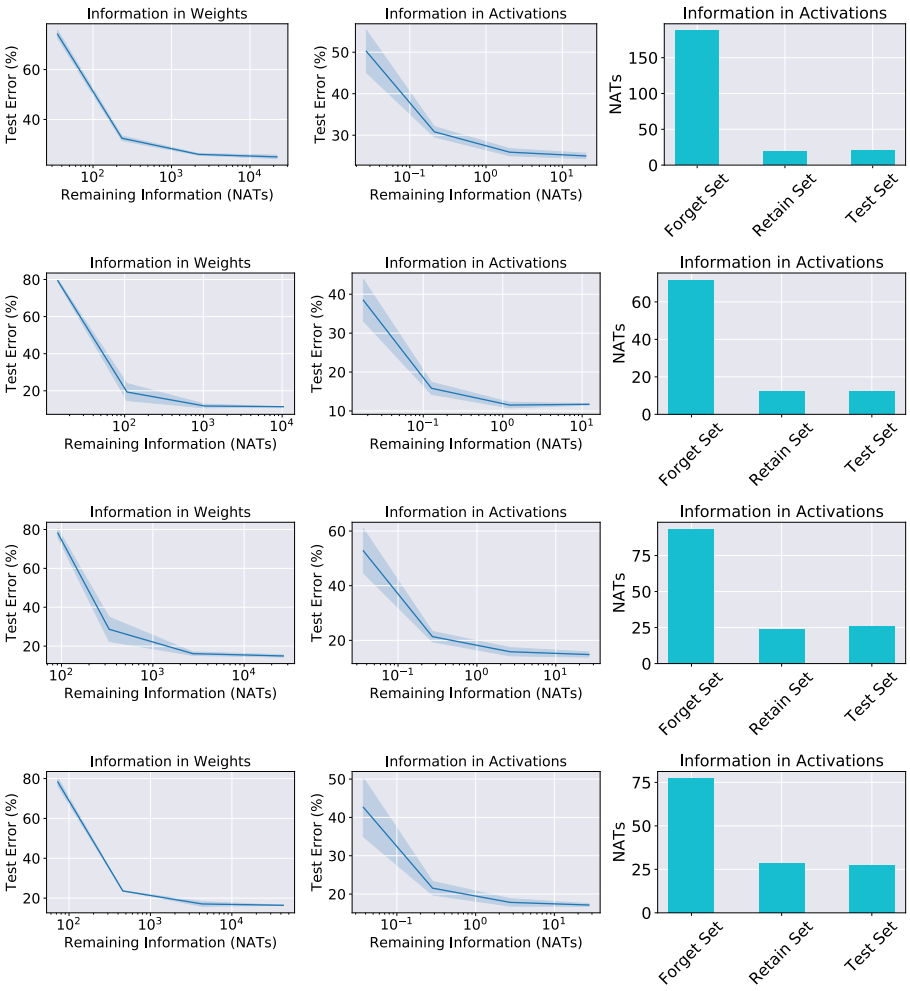

Fig. 8: Same experiment as Figure 3 for different architectures and datasets. **(Row-1)**: ResNet-18 on CIFAR, **(Row-2)**: ResNet-18 on Lacuna, **(Row-3)**: All-CNN on TinyImageNet, **(Row-4)**: ResNet-18 on TinyImageNet. We observe consistent behaviour across different architectures and datasets.

For the original model we compute this at $w = w_{\mathcal{D}}$ and take h to be the the NTK scrubbing shift in eq. (10). The baseline model does not use any shift, so $h(w) = w$, and is computed at $w = w_{\mathcal{D}_r}$.

Proof of Proposition 1, eq. (6). As in [16, Example 2].

Proof of Proposition 1, eq. (7). Using Equation (5) we write the distribution of the activations of a scrubbed network:

$$f_{S(w_{\mathcal{D}})}(\mathbf{x}) \sim N(f_{h(w_{\mathcal{D}})}(\mathbf{x}), J\Sigma J^T)$$

where $J = \nabla_w f_{h(w_{\mathcal{D}})}(\mathbf{x})$. We can similarly write the activations for baseline as:

$$f_{S_0(w_{\mathcal{D}_r})}(\mathbf{x}) \sim N(f_{w_{\mathcal{D}_r}}(\mathbf{x}), J'\Sigma_0 J'^T)$$

where $J' = \nabla_w f_{w_{\mathcal{D}_r}}(\mathbf{x})$ Using the two distributions, we rewrite the bound in Lemma 2 as:

$$\begin{aligned} I(\mathcal{D}_f; f_{S(w)}(\mathbf{x})) &\leq \mathbb{E}_{\mathcal{D}_f,\epsilon}\big[\,\mathrm{KL}\left(N(f_{h(w_{\mathcal{D}})}(\mathbf{x}), J\Sigma J^T) \,\|\, N(f_{w_{\mathcal{D}_r}}(\mathbf{x}), J'\Sigma_0 J'^T\right)\big] \\ &= \mathbb{E}_{\mathcal{D}_f,\epsilon}\big[\Delta f^T {\Sigma'_{\mathbf{x}}}^{-1} \Delta f + \mathrm{tr}(\Sigma_{\mathbf{x}} {\Sigma'_{\mathbf{x}}}^{-1}) - \log|\Sigma_{\mathbf{x}} {\Sigma'_{\mathbf{x}}}^{-1}| - n\big] \end{aligned}$$

where $\Delta f = f_{h(w_{\mathcal{D}})}(\mathbf{x}) - f_{w_{\mathcal{D}_r}}(\mathbf{x})$, $\Sigma_{\mathbf{x}} = J\Sigma J^T$, $\Sigma'_{\mathbf{x}} = J'\Sigma_0 J'^T$, and we used the closed form expression for the KL divergence of two normal distributions.

Proof of Proposition 2. Let $\mathcal{D} = \mathcal{D}_r \cup \mathcal{D}_f$ be the complete training set. To keep the notation simpler, we assume that the loss is an mean-square-error regression loss (we discuss classification using cross-entropy in Appendix B).

Let w_0 be the weights obtained after pre-training on $\mathcal{D}_{\text{pre-train}}$. Taking inspiration from the NTK analysis [23, 27] we approximate the activations $f_w(x)$ for a test datum x after fine-tuning on $\mathcal{D}$ using the liner approximation $f_w^{\text{lin}}(x) = f_{w_0}(x) + \nabla_w f_{w_0}(x)(w - w_0)$. To keep the notation uncluttered, we write $f_0(x)$ instead of $f_{w_0}(x)$.

The training dynamics under the linear approximation (assuming continuous gradient descent) are then given by eq. (8) and (9), and will converge at the final solution (see [27] for more details):

$$w_{\text{lin}}(\mathcal{D}) = \nabla f_0(\mathcal{D})^T \Theta^{-1}(f_0(\mathcal{D}) - Y) + w_0.$$

Here $\nabla f_0(\mathcal{D}) \in \mathbb{R}^{Nc \times p}$ is the gradient of the output with respect to the parameters (at initialization) for all the samples in $\mathcal{D}$ stacked along the rows to form a matrix (p is the number of parameters in the model, $N = |\mathcal{D}|$ and c is the number of classes), $\Theta = \nabla f_0(\mathcal{D}) \nabla f_0(\mathcal{D})^T$ ($\in \mathbb{R}^{Nc \times Nc}$) is the NTK matrix. Similarly, Y is the matrix formed by stacking all ground-truth labels one below the other and $f_0(\mathcal{D}) \in \mathbb{R}^{(Nc \times 1)}$ are the stacked outputs of the DNN at initialization on $\mathcal{D}$.

Similarly the baseline solution (training only on the data to retain) is:

$$w_{\text{lin}}(\mathcal{D}_r) = \nabla f_0(\mathcal{D}_r)^T \Theta_{rr}^{-1}(f_0(\mathcal{D}_r) - Y_r) + w_0,$$

where $\Theta_{rr} = \nabla f_0(\mathcal{D}_r)^T \nabla f_0(\mathcal{D}_r)$. The optimal scrubbing vector (at least for the linearized model) would then be $\delta w := w_{\text{lin}}(\mathcal{D}_r) - w_{\text{lin}}(\mathcal{D})$: adding δw to the weights obtained by training on $\mathcal{D}$ makes us forget the extra examples ($\mathcal{D}_f$), so that we obtain weights equivalent to training on $\mathcal{D}_r$ alone. We now derive the simplified expression in eq. (10) for the optimal scrubbing vector δw. We start by rewriting $w_{\text{lin}}(\mathcal{D})$ using block matrixes:

$$\begin{aligned} w_{\text{lin}}(\mathcal{D}) &= \nabla f_0(\mathcal{D})^T \Theta^{-1}(f_0(\mathcal{D}) - Y) + w_0 \\ &= \left[\nabla f_0(\mathcal{D}_r)^T \nabla f_0(\mathcal{D}_f)^T\right] \begin{bmatrix} \Theta_{rr} & \Theta_{rf} \\ \Theta_{rf}^T & \Theta_{ff} \end{bmatrix}^{-1} \begin{bmatrix} f_0(\mathcal{D}_r) - Y_r \\ f_0(\mathcal{D}_f) - Y_f \end{bmatrix} + w_0 \end{aligned}$$

We can expand the inverse of the NTK matrix using the following equations:

$$\begin{bmatrix} \Theta_{rr} & \Theta_{rf} \\ \Theta_{rf}^T & \Theta_{ff} \end{bmatrix}^{-1} = \begin{bmatrix} \left[\Theta_{rr} - \Theta_{rf}\Theta_{ff}^{-1}\Theta_{rf}^T\right]^{-1} & -\Theta_{rr}^{-1}\Theta_{rf}M \\ -M\Theta_{rf}^T\Theta_{rr}^{-1} & M \end{bmatrix}$$

Where $M = \left[\Theta_{ff} - \Theta_{rf}^T\Theta_{rr}^{-1}\Theta_{rf}\right]^{-1}$. Using Woodbury Matrix Identity:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

We obtain:

$$\left[\Theta_{rr} - \Theta_{rf}\Theta_{ff}^{-1}\Theta_{rf}^T\right]^{-1} = \Theta_{rr}^{-1} + \Theta_{rr}^{-1}\Theta_{rf}M\Theta_{rf}^T\Theta_{rr}^{-1}$$

Thus,

$$\begin{bmatrix} \Theta_{rr} & \Theta_{rf} \\ \Theta_{rf}^T & \Theta_{ff} \end{bmatrix}^{-1} = \begin{bmatrix} \Theta_{rr}^{-1} + \Theta_{rr}^{-1}\Theta_{rf}M\Theta_{rf}^T\Theta_{rr}^{-1} & -\Theta_{rr}^{-1}\Theta_{rf}M \\ -M\Theta_{rf}^T\Theta_{rr}^{-1} & M \end{bmatrix}$$

Using the above relation we get

$$\begin{aligned} w_{\text{lin}}(\mathcal{D}) = &\nabla f_0(\mathcal{D}_r)^T \left(\Theta_{rr}^{-1} + \Theta_{rr}^{-1}\Theta_{rf}M\Theta_{rf}^T\Theta_{rr}^{-1}\right)(f_0(\mathcal{D}_r) - Y_r) \\ &- \nabla f_0(\mathcal{D}_f)^T M\Theta_{rf}^T\Theta_{rr}^{-1}(f_0(\mathcal{D}_r) - Y_r) \\ &- \nabla f_0(\mathcal{D}_r)^T\Theta_{rr}^{-1}\Theta_{rf}M(f_0(\mathcal{D}_f) - Y_f) \\ &+ \nabla f_0(\mathcal{D}_f)^T M(f_0(\mathcal{D}_f) - Y_f) + w_0. \end{aligned}$$

Finally, using this the optimal shift δw for scrubbing the weights is

$$\begin{aligned}
\Delta w &= w_{\text{lin}}(\mathcal{D}_r) - w_{\text{lin}}(D) \\
&= -\nabla f_0(\mathcal{D}_r)^T \Theta_{rr}^{-1} \Theta_{rf} M \Theta_{rf}^T \Theta_{rr}^{-1} (f_0(\mathcal{D}_r) - Y_r) \\
&+ \nabla f_0(\mathcal{D}_f)^T M \Theta_{rf}^T \Theta_{rr}^{-1} (f_0(\mathcal{D}_r) - Y_r) \\
&+ \nabla f_0(\mathcal{D}_r)^T \Theta_{rr}^{-1} \Theta_{rf} M (f_0(\mathcal{D}_f) - Y_f) \\
&- \nabla f_0(\mathcal{D}_f)^T M (f_0(\mathcal{D}_f) - Y_f) \\
&= \Big[I - \nabla f_0(\mathcal{D}_r)^T \Theta_{rr}^{-1} \nabla f_0(\mathcal{D}_r) \Big] \nabla f_0(\mathcal{D}_f)^T M \Big[\Theta_{rf}^T \Theta_{rr}^{-1} (f_0(\mathcal{D}_r) - Y_r) \Big] \\
&+ \Big[I - \nabla f_0(\mathcal{D}_r)^T \Theta_{rr}^{-1} \nabla f_0(\mathcal{D}_r) \Big] \nabla f_0(\mathcal{D}_f)^T M \Big[(Y_f - f_0(\mathcal{D}_f)) \Big] \\
&= P \nabla f_0(\mathcal{D}_f)^T M V,
\end{aligned}$$

where $P = I - \nabla f_0(\mathcal{D}_r)^T \Theta_{rr}^{-1} \nabla f_0(\mathcal{D}_r)$, $M = \left[\Theta_{ff} - \Theta_{rf}^T \Theta_{rr}^{-1} \Theta_{rf} \right]^{-1}$ and $V = [(Y_f - f_0(\mathcal{D}_f)) + \Theta_{rf}^T \Theta_{rr}^{-1} (Y_r - f_0(\mathcal{D}_r))]$.