Supplemental Material for
HyDRA: Hypergradient Data Relevance Analysis for Interpreting Deep Neural Networks

Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao
1School of Computer Science and Engineering, Nanyang Technological University
2Alibaba-NTU Singapore Joint Research Institute
{yuanyuan.chen, boyang.li, han.yu, pengcheng.wu, ascymiao}@ntu.edu.sg
*Corresponding authors

Contents

Experiment Settings 1
Additional Examples 1
Additional Results for Debugging Training Data 1
Growth of Training Time 2
Approximation Error Analysis 2
Bounds with Lipschitz Continuity 2
Bounds with Relaxed Conditions 4
Mini-batch Hypergradient 5

Experiment Settings

MNIST and Fashion-MNIST are divided into training sets of 50,000, validation sets of 10,000, and test sets of 10,000 samples, respectively. CIFAR-10 is divided into a training set of 40,000, a validation set of 10,000, and a test set of 10,000 samples, respectively.

LeNet-5 consists of 61,706 trainable parameters. We replace all of the tanh activation function with ReLU in our experiments. We also use the DenseNet-40 model with a growth rate of 12, which contains 176,122 trainable parameters.

Table 3 gives the detailed hyperparameters and other settings used in our experiments. For the reduce-on-plateau schedule, we recognize a plateau when the sum of the training loss and the validation loss does not decrease more than 0.01% of the best value found in two epochs.

All networks are optimized using SGD with momentum of 0.9. On MNIST and Fashion-MNIST, we train LeNet-5 using SGD with momentum for 20 epochs with a batch size of 64. This procedure obtains test accuracies of 99.09% and 89.99% respectively. On CIFAR-10, we train DenseNet-40 using SGD with momentum for 150 epochs with a batch size of 64 and obtains test accuracy of 90.5%.

In the first training data debugging experiment, we perform early stopping after 20 epochs of training and report the accuracy in the last epoch. In the second training data debugging experiment, we train the networks for 50 epochs.

Additional Examples

Two data points from MNIST with extreme contributions are shown in Table 1. The first is labeled as 5 but closely resembles 6. The second is labeled as 8 and has an unusual upper half. Their average contribution to all test data points is 4-5 orders of magnitudes higher than the average (≈ 10^-9).

Due to their unusual appearances, these data points stand out from the rest of the training data and hence exert large influence on the model.

The two training samples heavily influence the two test data points in Table 2. The first test sample in Table 2 is misclassified as 5. HyDRA indicates this error may be caused by the training sample that conflates 5 and 6. The second test sample is wrongly classified as 3. Regardless, the outlandish looking 8 probably helped in bringing the training loss on this digit down.

Additional Results for Debugging Training Data

We report the results for debugging training data with the label noise rate r set to 50% in Table 4. For all datasets, training the network from scratch using samples chosen by
Table 2: Test samples strongly influenced by training samples in Table 1.

| Test Sample | Influencer | Contrib. | True / Predicted Label | Model Conf. |
|-------------|------------|----------|------------------------|-------------|
| ![Image](58x625 to 90x657) | ![Image](110x625 to 142x657) | -0.31 | 6 / 5 | 0.78 |
| ![Image](58x589 to 90x621) | ![Image](110x589 to 142x589) | 0.091 | 8 / 3 | 0.80 |

HyDRA leads to better accuracy than influence functions. In addition, HyDRA produces significantly better performance than training directly on noisy data on MNIST and CIFAR10. An exception happens on Fashion-MNIST, where filtering noisy data points using either method performs worse than not filtering the data at all.

**Growth of Training Time**

Our theoretical analysis indicates that the training time of HyDRA scales linearly with the number of model parameters and the number of data points whose contributions are tracked through the training trajectory. In this section, we empirically study how training time of HyDRA grows with those two factors.

We perform the experiments on a server with an AMD Ryzen 7 3800X 8-Core processor, 32 GB of main memory, two 2 GeForce RTX 2080 Ti GPU, each with 12GB memory, and 1 TB XPG GAMMIX S50 solid state drive. We used three different networks, LeNet5 (Lecun et al. 1998), DenseNet-40 (Huang et al. 2017), and MobileNet V2 (Sandler et al. 2019), which have 61,706, 176,122, and 2,236,682 trainable parameters respectively. For each network, we record the training time when tracking the contribution of 400, 2,000 and 10,000 training data points.

Figure 1 shows the average training time per tracked data point and per network parameter. We note that the average training time does not increase, which indicates the training time scales sublinearly initially and grows linearly afterwards.

**Approximation Error Analysis**

In this section, we will analyze the approximation error introduced by dropping the \( H^\text{tr} \) term in the case of vanilla GD. After dropping the term, the recurrent update equation becomes

\[
\tilde{\nabla}_{t,i} = \nabla_{t-1,i} - \eta_t \lambda \nabla_{t-1,i} - \eta_t g_{t-1,i}.
\]  

We are interested in bounding the norm of the approximation error, which is defined below.

**Definition 1.** The approximation error at the \( t \)th iteration is defined as

\[
e_t := \nabla_{t,i} - \tilde{\nabla}_{t,i},
\]  

where \( \tilde{\nabla}_{t,i} \) is the approximation of \( \nabla_{t,i} \).

Before the analysis, we also need some moderate conditions about the optimization process.

**Condition 1.** The training loss \( \mathcal{L}_{\text{train}}(x, y, w) \) is twice differentiable.

**Condition 2.** The optimization process converges. That is,

\[
\lim_{t \to \infty} w_t = \hat{w}.
\]  

**Bounds with Lipschitz Continuity**

First, we bound the error under several moderate conditions regarding the optimization process. In the next subsection, we show how these conditions can be further relaxed.

**Condition 3.** The empirical risk function \( \mathcal{L}_{\text{train}}^\epsilon \) has Lipschitz-continuous gradients with Lipschitz constant \( L \). Formally,

\[
\left\| \frac{\partial \mathcal{L}_{\text{train}}^\epsilon (w_1)}{\partial w_1} - \frac{\partial \mathcal{L}_{\text{train}}^\epsilon (w_2)}{\partial w_2} \right\| \leq L \| w_1 - w_2 \|, \forall w_1, w_2.
\]  

Figure 1: The training time scales up with respect to #Parameters and #Tracked Samples.
tion constrains the eigenvalues of $H$ tends to be close to $w$ takes value from a compact set. The latter is likely true since applied by, for example, that the Lipschitz continuity is a regular condition, which is implemented by, for example, that $C_{\min}$ is twice differentiable and $w$ takes value from a compact set. The latter is likely true since $w$ tends to be close to $u_0$. It is worth noting that this condition constrains the eigenvalues of $H_t$ to the range $[-L, L]$.

**Condition 4.** The learning rate sequence $\eta_t$ is non-increasing and lower-bounded by 0. That is,

$$\eta_t \geq \eta_{t+1} > 0, \forall t. \quad (5)$$

Since $\eta_t$ and $\lambda$ are both typically quite small, we assume their product is also small.

**Condition 5.** The product $0 < \eta_t \lambda < 1, \forall t$.

Finally, the contribution $\nabla_{t,i}$ should be bounded, or efforts to estimate it would end in vain.

**Condition 6.** The contribution measure sequence $\nabla_{t,i}$ does not diverge as $t \to \infty$ and is bounded by a constant $M_w$. More formally,

$$\lim_{t \to \infty} \| \nabla_{t,i} \| < M_w, \forall i. \quad (6)$$

**Theorem 1.** With conditions 1-6, the norm of the approximation error is bounded by

$$\| e_t \| < LM_w \eta_{t \lambda}. \quad (7)$$

Proof. First, we have the recursive formula:

$$e_0 = 0, \quad (8)$$

$$e_t = (1 - \eta_t \lambda) e_{t-1} - \eta_t H^*_{t-1} \nabla_{t-1,i}. \quad (9)$$

After solving it, we get

$$e_t = \sum_{j=1}^{t} (-\eta_j)(1 - \eta_j \lambda)^{t-j} H^*_{j-1} \nabla_{j-1,i}. \quad (10)$$

By the triangle inequality,

$$\| e_t \| \leq \sum_{j=1}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} \| H^*_{j-1} \nabla_{j-1,i} \|. \quad (11)$$

We then have

$$\sum_{j=1}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} \| H^*_{j-1} \nabla_{j-1,i} \| \leq \sum_{j=1}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} \| L \nabla_{j-1,i} \|$$

$$= \sum_{j=1}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} LM_w$$

$$\leq LM_w \eta_1 \sum_{j=0}^{t-1} (1 - \eta_j \lambda)^{j} \quad (12)$$

$$= LM_w \eta_1 \frac{1 - (1 - \eta_t \lambda)^{t}}{\eta_t \lambda} \quad (13)$$

$$< LM_w \eta_1 \frac{1 - c}{\eta_t \lambda} \quad (14)$$

As such, we obtain the desired inequality. $\square$

Furthermore, if we know that the learning rate decays sufficiently exponentially, the approximation error diminishes when $t$ tends to infinity.

**Condition 7.** The learning rate sequence $\eta_t$ decays exponentially at rate $c$, which is less than $1 - \eta_t \lambda$. That is,

$$\eta_{t+1} = c \eta_t, \forall t, \quad (13)$$

$$0 < c < 1 - \eta_t \lambda. \quad (14)$$
Theorem 2. With conditions 1-7 and the learning rate schedule in condition 7, the approximation error diminishes when $t$ tends to infinity.

$$
\lim_{t \to \infty} \|e_t\| = 0.
$$

Proof. Under the specific learning rate schedule, we have

$$
\|e_t\| \leq LM_w \sum_{j=1}^{t} \eta_j (1 - \eta_j \lambda)^{t-j}
= LM_w \sum_{j=1}^{t} \eta_j c_j (1 - \eta_j \lambda)^{t-j}
\leq LM_w \frac{\eta_1}{c} (1 - \eta_1 \lambda)^t \sum_{j=1}^{t} \left( \frac{c}{1 - \eta_1 \lambda} \right)^j
= LM_w \frac{\eta_1}{c} (1 - \eta_1 \lambda)^t \frac{c}{1 - \frac{c}{1 - \eta_1 \lambda}}.
$$

(16)

When $t \to \infty$, $(1 - \eta_1 \lambda)^t$ and $(c/(1 - \eta_1 \lambda))^{t+1}$ go to zero, and the claim follows.

Bounds with Relaxed Conditions

The condition 3 in the previous section may appear to be too restrictive. In this subsection, we replace it with a more relaxed condition and show that the error is still uniformly bounded in the limit.

Condition 8. The Hessian sequence $H^{er}$ converges as $t \to \infty$

$$
\lim_{t \to \infty} H^{er}_t = \tilde{H}^{er},
$$

(17)

where $\tilde{H}^{er}$ is the Hessian of the empirical risk $\mathcal{L}^{er}_{\text{train}}$ at $\tilde{w}$.

Since $w$ converges, it makes sense to assume that Hessian converges, too.

Corollary 1. The eigenvalues of $H^{er}$ converge to the eigenvalues of $\tilde{H}^{er}$.

By this corollary, we can find an index $N$ such that $|\kappa_{\text{max}}(H^{er}_t) - \kappa_{\text{max}}(\tilde{H}^{er})| < \delta, \forall t \geq N$, given an arbitrarily small $\delta$, where $\kappa_{\text{max}}$ is the eigenvalue with maximal absolute value. Now, we fix such an infinitesimal $\delta_{\text{eigen}}$ and the corresponding index $N_{\text{eigen}}$.

Finally, since $\eta_t$ and $\lambda$ are both typically quite small, we assume their product is eventually small.

Theorem 3. With conditions 1-6, and 8, and let $t_0 = N_{\text{eigen}} + 1$ be the start of the tail portion of the optimization, we can upper bound the error’s norm as

$$
\lim_{t \to \infty} \|e_{t,t_0}\| < (\kappa_{\text{max}}(\tilde{H}^{er}) + \delta_{\text{eigen}})M_w \eta_{t_0} \frac{1}{\eta_1 \lambda}.
$$

(18)

where $e_{t,t_0}$ is a shorthand for a two-part sum that constitute $e_t$,

$$
e_t = e_{t,t_0} = \sum_{j=1}^{t_0-1} (-\eta_j)(1 - \eta_j \lambda)^{t-j} H^{er}_{j-1} \nabla_{j-1,i}
+ \sum_{j=t_0}^{t} (-\eta_j)(1 - \eta_j \lambda)^{t-j} H^{er}_{j-1} \nabla_{j-1,i}.
$$

(19)

Proof. Again, we have the recursive formula:

$$
e_0 = 0,
$$

$$
e_t = (1 - \eta_t \lambda)e_{t-1} - \eta_t H^{er}_{t-1} \nabla_{t-1,i}.
$$

(20)

(21)

After solving it, we get

$$
e_t = \sum_{j=1}^{t} (-\eta_j)(1 - \eta_j \lambda)^{t-j} H^{er}_{j-1} \nabla_{j-1,i}.
$$

(22)

Now if $t \geq t_0$, then by the triangle inequality,

$$
\|e_{t,t_0}\| \leq \sum_{j=1}^{t_0-1} \eta_j (1 - \eta_j \lambda)^{t-j} \|H^{er}_{j-1} \nabla_{j-1,i}\|
+ \sum_{j=t_0}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} \|H^{er}_{j-1} \nabla_{j-1,i}\|.
$$

(23)

Consider the second part first, we have

$$
\sum_{j=t_0}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} \|H^{er}_{j-1} \nabla_{j-1,i}\|
< \sum_{j=t_0}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} (\kappa_{\text{max}}(\tilde{H}^{er}) + \delta_{\text{eigen}}) \|\nabla_{j-1,i}\|
\leq (\kappa_{\text{max}}(\tilde{H}^{er}) + \delta_{\text{eigen}})M_w \eta_{t_0} \sum_{j=t_0}^{t} (1 - \eta_j \lambda)^{t-j}
\leq (\kappa_{\text{max}}(\tilde{H}^{er}) + \delta_{\text{eigen}})M_w \eta_{t_0} \frac{1 - (1 - \eta_j \lambda)^{t-t_0}}{\eta_1 \lambda}
< (\kappa_{\text{max}}(\tilde{H}^{er}) + \delta_{\text{eigen}})M_w \eta_{t_0} \frac{1}{\eta_1 \lambda}.
$$

(24)

Note that the first part of the right-hand side $\to 0$ as $t \to \infty$. In other words, for any small $\delta_1 > 0$, there is a number $N_1 > t_0$, such that for all $t \geq N_1$,

$$
\sum_{j=1}^{t_0-1} \eta_j (1 - \eta_j \lambda)^{t-j} \|H^{er}_{j-1} \nabla_{j-1,i}\| < \delta_1.
$$

(25)

Taken together, for any infinitesimal $\delta_1 > 0$, there exist an index $N_1$ such that

$$
\|e_{t,t_0}\| < (\kappa_{\text{max}}(\tilde{H}^{er}) + \delta_{\text{eigen}})M_w \eta_{t_0} \frac{1}{\eta_1 \lambda} + \delta_1, \forall t \geq N_1.
$$

(26)

This is the definition of the limit that we seek to prove.

If we add condition 7, we would have the same conclusion as before.

Theorem 4. With the extra condition (7),

$$
\lim_{t \to \infty} \|e_{t,t_0}\| = 0.
$$

(27)
Proof.
Reconsidering the second part, which can be rewritten as
\[
\sum_{j=t_0}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} \left\| H_{j-1} \nabla_{j-1,i} \right\|
\]
\[
\leq (\kappa_{\max}(H) + \delta_{\text{eigen}}) M_w \sum_{j=t_0}^{t} \eta_j (1 - \eta_j \lambda)^{t-j}
\]
\[
\leq (\kappa_{\max}(H) + \delta_{\text{eigen}}) M_w \sum_{j=t_0}^{t} c j^{1-\eta_0} (1 - \eta_j \lambda)^{t-j}
\]
\[
= (\kappa_{\max}(H) + \delta_{\text{eigen}}) M_w \frac{\eta_0}{c \eta_0} \sum_{j=t_0}^{t} c j^{1-\eta_0} (1 - \eta_j \lambda)^{t-j}
\]
\[
\leq (\kappa_{\max}(H) + \delta_{\text{eigen}}) M_w \frac{\eta_0}{c \eta_0} (1 - \eta_0 \lambda)^{t} \sum_{j=t_0}^{t} \left( \frac{c}{1 - \eta_j \lambda} \right)^j
\]
\[
\leq (\kappa_{\max}(H) + \delta_{\text{eigen}}) M_w \frac{\eta_0}{c \eta_0} (1 - \eta_0 \lambda)^{t} \sum_{j=t_0}^{t} \left( \frac{c}{1 - \eta_j \lambda} \right)^j.
\]  
(28)

Introducing \( q = \frac{c}{1 - \eta_1 \lambda} \) for simplification, we have
\[
\sum_{j=t_0}^{t} \eta_j (1 - \eta_j \lambda)^{t-j} \left\| H_{j-1} \nabla_{j-1,i} \right\|
\]
\[
\leq (\kappa_{\max}(H) + \delta_{\text{eigen}}) M_w \frac{\eta_0}{c \eta_0} (1 - \eta_0 \lambda)^{t} \frac{q^{t_0} - q^{t+1}}{1 - q}
\]
\[
\leq (\kappa_{\max}(H) + \delta_{\text{eigen}}) M_w \frac{\eta_0}{c \eta_0} (1 - \eta_0 \lambda)^{t} \frac{q^{t_0}}{1 - q}.
\]  
(29)

Finally, since \( \lim_{t \to \infty} (1 - \eta_t \lambda)^{t} = 0 \), the desired conclusion follows. \( \Box \)

**Mini-batch Hypergradient**

Here we consider mini-batch training with a batch size of \( B \), and other symbols are of the same meanings as above. Formally, the loss function at \( t^{th} \) batch is
\[
\mathcal{L}_{\text{batch}}(w_{t-1}) = \frac{1}{B} \sum_{(x,y) \in \text{batch}_t} \ell(x, w_{t-1}, y) + \mathbb{1}_{\text{batch}_t}(i) \frac{N_e}{B} \ell(x, w_{t-1}, y_i) + \frac{1}{2} w_{t-1}^\top w_{t-1},
\]  
(30)

where the indicator function \( \mathbb{1} \) is introduced to determine whether the \( i^{th} \) training point is in the batch.

As before, we have the initial conditions
\[
\nabla_{0,i} = 0, \quad \frac{d w_0}{d e_i} = 0.
\]  
(33)

Then the recurrence formula
\[
\frac{d w_t}{d e_i} = p \frac{d w_{t-1}}{d e_i} + H_{t-1} \nabla_{t-1,i} + \lambda \nabla_{t-1,i} + \mathbb{1}_{\text{batch}_t}(i) \frac{N_e}{B} \ell(x, w_{t-1}, y_i) + \frac{1}{2} w_{t-1}^\top w_{t-1},
\]  
(35)

where \( H_t \) denote the Hessian of the regularizer-free batch loss. Also, we could omit \( H_t \) here.

**References**

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2019. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*. 