A Overview

The appendix is organized as follows: First, in Section B, we discuss the potential social impacts of our work. Next, in Section C, we provide experimental details on the qualitative measurements on image classification datasets presented in Section 6. Then, in Section D, we illustrate the extension of generalized representers to explain vector-output target functions. Finally, we provide proofs of our theoretical results in Section E.

B Potential Negative Social Impact of Our Work

Our approach has the potential to bring about various social consequences by allowing individuals to modify model predictions through adjustments to training samples. These effects can involve the risks of attacking established models or exacerbating biases and ethical concerns.
We set the regularization parameter

We then follow Yeh et al.

We use the three methods mentioned in Section 4, which are toxicity/Agnews datasets respectively.

The total number of parameters in the model is

They are trained with SGD optimizer with learning rates

The CNNs we use for vision datasets comprise

respect to the target function predictions.

Target derivative:

and the parameters of the target function

tangent kernel, we have an extra bias term:

search gradient descent algorithm to solve Eqn. (7)

Due to the large number of parameters in the target model (over 1.6 million), we do not implement the generalized representer with surrogate derivative global importance and neural tangent kernel since it is impractical to store all feature maps of training samples. Also, we apply the influence function only on the last layer [6] since the computation of the inverse Hessian matrix is too costly.

C.2 Computation of Global Importances

We use the three methods mentioned in Section 4 which are surrogate derivative, target derivative, and tracking representers.

Surrogate derivative: We first transform Eqn. (4) to the primal form: let \( f_K(x) = \langle \theta_{sug}, \Phi(x) \rangle \) be a kernel machine that has finite-dimensional feature maps and is parameterized by \( \theta_{sug} \). Eqn. (4) can then be transformed to:

\[
\theta_{sug}^{\star} = \arg\min_{\theta_{sug} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\langle \theta_{sug}, \Phi(x_i) \rangle, f(x_i)) + \frac{\lambda}{2} \| \theta_{sug} \|^2. \tag{17}
\]

We then follow Yeh et al. [7] to take the target model parameters as initialization and use the line-search gradient descent algorithm to solve Eqn. (17). When the kernel function is chosen as a neural tangent kernel, we have an extra bias term: \( f_K(x) = \sum_{i=1}^{n} \langle \partial_{\theta} f_{\theta}(x_i), \theta_{sug} - \theta \rangle + f_{\theta}(x) \), where \( \theta \) is the parameters of the target function \( f_{\theta} \).

The optimization problem then becomes

\[
\theta_{sug}^{\star} = \arg\min_{\theta_{sug} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\langle \theta_{sug} - \theta, \partial_{\theta} f_{\theta}(x_i) \rangle + f_{\theta}(x_i), f_{\theta}(x_i)) + \frac{\lambda}{2} \| \theta_{sug} \|^2. \tag{18}
\]

We set the regularization parameter \( \lambda = 2 \times 10^{-2} \) in our experiments. Then we compute the global importance with Eqn. (6), i.e.

\[
\alpha_i = \frac{\partial \mathcal{L}(\langle \theta_{sug}^{\star} - \theta, \partial_{\theta} f_{\theta}(x_i) \rangle + f_{\theta}(x_i), f_{\theta}(x_i))}{\partial (\theta_{sug} - \theta, \partial_{\theta} f_{\theta}(x_i))}.
\]

Target derivative: We directly adopt Eqn. (7) that computes derivatives of the loss function with respect to the target function predictions.

Tracking representers: In our experiment, we use the following relaxation for further speed up:

\[
\alpha_i^{\star} = - \sum_{t \in [T]} \eta^{(t)} \frac{1}{|B^{(t)}|} \frac{\partial \mathcal{L}(f^{(t-1)}(x_i), y_i)}{\partial f^{(t-1)}(x_i)}, \tag{19}
\]
where we assume the target model is trained with (stochastic) gradient descent, $f(t)(x_i)$ denotes the target model at $t$th iteration during training, and $B(t)$ and $\eta(t)$ are the corresponding mini-batch and learning rate. Similar to the intuition of replacing the surrogate derivative with target derivative, we track the target model’s training trajectory directly instead of solving Eqn. with kernel gradient descent.

C.3 Computation of Kernel functions

For last layer embeddings and neural tangent kernels, we directly use Eqn. to compute feature maps. For influence function kernels, we use public implementation for Pytorch. It uses stochastic estimation to speed up the computation of the inversed hessian vector product and sets the damping factor to 0.01.

C.4 Computation of TracInCP

For TracInCP, we compute over 7 evenly spaced checkpoints, including the initial parameter $\theta(0)$ and the final parameter $\theta(T)$.

C.5 Hardware usage

For all of our experiments, we use 4 NVIDIA A6000 GPUs each with 40GB of memory and 96 CPU cores. To run the experiment on the vision datasets in Section 6, we retrain the model over 11 (number of methods) × 2 (for AUC-DEL+ and AUC-DEL−) × 50 (number of random seeds) × 10 (number of test samples per model) = 11,000 times.

We train 5 models simultaneously on a single GPU for speed up. It takes around 4 days to run the experiment in Section 6.

C.6 Supplementary Experimental Results

Apart from the numerical results in terms of AUC-DEL−, we also provide deletion curves showing changes in model predictions for different numbers of training samples being removed in Figure 1, 2, 3 and 4.

Figure 1: Deletion curves for the MNIST dataset. Larger DEL− is better since it indicates a method finds more negative impact training samples.

C.7 Licence of Datasets

For the image classification datasets, MNIST has GNU general public license v3.0. The CIFAR-10 dataset is under MIT License. The Toxicity dataset has license cc0-1.0 The AGnews dataset has licensed non-commercial use.

https://github.com/nimarb/pytorch_influence_functions
(a) Exp I: comparing global importances.  
(b) Exp II: comparing kernels  
(c) Exp III: Comparing generalized representers.

Figure 2: Deletion curves for the **CIFAR-10 dataset**. Larger DEL is better since it indicates a method finds more negative impact training samples.

(a) Exp I: comparing global importances.  
(b) Exp II: comparing kernels  
(c) Exp III: Comparing generalized representers.

Figure 3: Deletion curves for the **Toxicity dataset**. Larger DEL is better since it indicates a method finds more negative impact training samples.

(a) Exp I: comparing global importances.  
(b) Exp II: comparing kernels  
(c) Exp III: Comparing generalized representers.

Figure 4: Deletion curves for the **AGnews dataset**. Larger DEL is better since it indicates a method finds more negative impact training samples.
D Extension to Explain Vector Output Target Functions

In the main paper, we assume that the target function has a scalar output for notational convenience. However, most modern machine models have multi-dimensional outputs as is common for multi-class classification problems. In this section, we show that generalized representers can be extended to vector output by utilizing kernels for vector-valued functions.

**Definitions:** Given a target function \( f : \mathbb{R}^d \rightarrow \mathbb{R}^c \), a dataset \( D := \{ (x_i, y_i) \}_{i=1}^n \) with each training sample \( (x_i, y_i) \in \mathbb{R}^d \times \mathcal{Y} \), and a Mercer kernel \( K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^{c \times c} \), the generalized representer has the following form:

\[
\phi(f, (x_i, y_i) \rightarrow \mathbf{x}) = \mathbf{\alpha}^T K(x_i, \mathbf{x}) \in \mathbb{R}^c , \ \forall i \in [n],
\]

where the global importance \( \mathbf{\alpha} \in \mathbb{R}^{n \times c} \) is a matrix with \( \alpha_{ij} \) the importance of the \( i \)th training sample to the \( j \)th prediction, and \( \mathbf{\alpha}_i \in \mathbb{R}^{c \times 1} \) is the \( i \)th row of the matrix.

**D.1 Vector-valued Global Importance**

We discuss the extension of the three methods in Section 4, which are surrogate derivative, target derivative, and tracking representers, for vector-valued target functions.

**Surrogate derivative:** The surrogate derivative in Eqn. (6) for vector-valued target functions is:

\[
\mathbf{\alpha}_{ij}^{\text{sur}} = -\frac{1}{n\lambda} \frac{\partial \mathcal{L}(\hat{f}_K(x_i), f(x_i))}{\partial f_K(x_i)_j}, \ \forall i \in [n] \text{ and } \forall j \in [c],
\]

where \( \hat{f}_K : \mathbb{R}^p \rightarrow \mathbb{R}^c \) is the minimizer of Eqn. (4), and \( \hat{f}_K(x_i)_j \) is its \( j \)th output. Eqn. (6) can be obtained by taking derivative of Eqn. (4) with respect to \( \alpha_{ij} \).

**Target derivative:** The target derivative can be represented as

\[
\mathbf{\alpha}_{ij}^{\text{tar}} = -\frac{1}{n\lambda} \frac{\partial \mathcal{L}(f(x_i), y_i)}{\partial f(x_i)_j}, \ \forall i \in [n] \text{ and } \forall j \in [c],
\]

where \( f(x_i)_j \) denotes \( j \)th output of the target function evaluated on \( x_i \).

**Tracking representers:** The formulation of the tracking representers for vector-valued target functions is as below:

\[
\mathbf{\alpha}_{ij}^{\text{tra}} = -\sum_{t \in [T]: t \in B(t)} \eta(t) \frac{\partial \mathcal{L}(\hat{f}_K^{(t-1)}(x_i), f(x_i))}{\partial f_K^{(t-1)}(x_i)_j} \forall i \in [n] \text{ and } \forall j \in [c],
\]

where \( B(t) \) and \( \eta(t) \) denote the minibatch and the learning rate used to solve Eqn. (4), and \( \hat{f}_K^{(t-1)}(x_i)_j \) is the function at the \( (t-1) \)th time step and \( \hat{f}_K^{(t-1)}(x_i) \) denotes its \( j \)th output of \( \hat{f}_K^{(t-1)}(x_i) \).

**D.2 Kernels for Vector-valued Target Functions**

In this section, we provide formulas for kernels mentioned in Section 5 for vector-valued target functions.

**Penultimate-layer embeddings:** Let the target model \( f \) be \( f(x) = \Phi_{\Theta_1}(x)^T \Theta_2 \), where \( \Phi_{\Theta_1} : \mathbb{R}^d \rightarrow \mathbb{R}^t \) maps inputs to the penultimate layer, and \( \Theta_2 \in \mathbb{R}^{t \times c} \) is the last layer weight matrix. The kernel function is defined as

\[
K_{\text{LL}}(x, z) = \langle \Phi_{\Theta_1}(x), \Phi_{\Theta_1}(z) \rangle \cdot I_c \in \mathbb{R}^{c \times c}, \ \forall x, z \in \mathbb{R}^d,
\]

where \( I_c \) is the \( c \times c \) identity matrix. Since the kernel outputs the same value for all entries on the diagonal, we can consider the target function \( f(x) = [f_1(x), \ldots, f_c(x)] \) as \( c \) distinct scalar output functions and compute generalized representers for each of these functions.
Neural tangent kernel: When the target function has multi-dimensional outputs, the neural tangent kernel, \( K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}^{c \times c} \), is defined by
\[
K_{\text{NTK}, \theta}(x, z) = \left( \frac{\partial f_\theta(x)}{\partial \theta} \right) \left( \frac{\partial f_\theta(z)}{\partial \theta} \right)^\top,
\]
where \( \frac{\partial f_\theta}{\partial \theta} \in \mathbb{R}^{c \times p} \) is the Jacobian matrix consisting of derivative of the function \( f_\theta \) with respect to its parameter \( \theta \).

Influence function kernel: let \( \frac{\partial f_\theta}{\partial \theta} \in \mathbb{R}^{c \times p} \) is the Jacobian matrix defined above, the influence function kernel can be defined as
\[
K_{\text{Inf}, \theta}(x, z) = \frac{\partial f_\theta(x)^\top}{\partial \theta} H_\theta^{-1} \frac{\partial f_\theta(z)}{\partial \theta}.
\]
The influence function then can be written as
\[
\phi_{\text{Inf}}(f_\theta, (x_i, y_i) \rightarrow x) = -\left( \frac{\partial \mathcal{L}(f_\theta(x_i), y_i)}{\partial \theta} \right) H_\theta^{-1} \frac{\partial f_\theta(x)}{\partial \theta} \left( \frac{\partial \mathcal{L}(f_\theta(y_i), y_i)}{\partial \theta} \right)^\top K_{\text{Inf}, \theta}(x_i, x).
\]
E Proof of Theoretical Results

In this section, we prove our theoretical results.

E.1 Proof of Theorem

Proof. We first prove that if explanation \( \phi(\cdot) \) satisfies the continuity, self-explanation, symmetric zero, symmetric cycle, and irreducibility axioms for any training samples \( D \) containing \( n \) training samples \( \{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R} \) for all \( i \in [n] \), we have

\[
\phi(f, x_i \rightarrow x_j) = \alpha_i K(x_i, x_j), \quad \forall i, j \in [n],
\]

for some \( \alpha \in \mathbb{R}^n \) and some continuous positive-definite kernel \( K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \).

Definitions: We first split the training data \( \{x_i, y_i\} \) into two parts: (1) \( \{x_i, y_i\} \) such that \( \phi(f, x_i \rightarrow x_j) \neq 0 \), and (2) \( \{x_i, y_i\} \) such that \( \phi(f, x_i \rightarrow x_j) = 0 \).

We then construct an undirected graph where the points are all training data points \( \{x_i\} \) and all \( \phi(f, x_i \rightarrow x_j) \neq 0 \) and all \( \phi(f, x_i \rightarrow x_j) = 0 \). If \( \phi(f, x_i \rightarrow x_j) \neq 0 \), then there are paths connecting \( x_i \) and \( x_j \).

If \( \phi(f, x_i \rightarrow x_j) = 0 \), then the reference point \( f, x_{p_i} \rightarrow x_{p_{j+1}} \), for all \( j \in [0, k-1] \). Define

\[
\alpha' = \begin{cases} 
1, & \text{if } x_i \text{ is the reference point.} \\
\frac{\phi(f, x_{p_i} \rightarrow x_{p_{i+1}})}{\phi(f, x_{p_i} \rightarrow x_{p_{i+1}})} \cdots \frac{\phi(f, x_{p_i} \rightarrow x_{p_{j}})}{\phi(f, x_{p_i} \rightarrow x_{p_{j}})}, & \text{if } x_i \text{ is in a connected component with reference point } x_r, \\
0, & \text{if } i \in \{m+1, \ldots, n\}.
\end{cases}
\]

By the symmetric cycle axiom, \( \alpha' \) is well defined and independent of how \( p \) is chosen since if two distinct paths connect \( x_i \) to \( x_r \), \( \alpha' \) by calculating through both paths will be equal by the symmetric cycle axiom. Formally, define \( \alpha'(p) = \frac{\phi(f, x_{p_i} \rightarrow x_{p_{i+1}})}{\phi(f, x_{p_i} \rightarrow x_{p_{i+1}})} \cdots \frac{\phi(f, x_{p_i} \rightarrow x_{p_{j}})}{\phi(f, x_{p_i} \rightarrow x_{p_{j}})} \), we have

\[
\alpha'(p) = \frac{\phi(f, x_{p_0} \rightarrow x_{p_0})}{\phi(f, x_{p_0} \rightarrow x_{p_0})} \cdots \frac{\phi(f, x_{p_0} \rightarrow x_{q_0})}{\phi(f, x_{p_0} \rightarrow x_{q_0})} \cdots \frac{\phi(f, x_{p_{k-1}} \rightarrow x_{p_{k-1}})}{\phi(f, x_{p_{k-1}} \rightarrow x_{p_{k-1}})} = 1,
\]

by the symmetric cycle axiom since \( p_k = q_k \) and \( p_0 = q_0 \). Furthermore, we know that \( \alpha' \neq 0 \) for all \( i \), since all terms are non-zero by definition of connected component.

We then define

\[
K'(x_i, x_j) = \begin{cases} 
\phi(f, x_i \rightarrow x_j) / \alpha', & \text{for } i \in [m] \text{ and } j \in [n]/\{i\}, \\
\phi(f, x_i \rightarrow x_j) / \alpha', & \text{for } i = j \in [m], \\
0, & \text{for } i \in \{m+1, \ldots, n\} \text{ and } j \in [m], \\
\alpha' / \alpha_j, & \text{for } i, j \in \{m+1, \ldots, n\}.
\end{cases}
\]

Prove that \( K' \) is a symmetric function. We first consider the case where both \( i \) and \( j \) belong to \([m]\).

If \( x_i \) and \( x_j \) are in the same connected component, by definition the reference point \( x_r \) will be the same for \( x_i \) and \( x_j \). If \( \phi(f, x_i \rightarrow x_j) = 0 \), by symmetric-zero axiom \( \phi(f, x_j \rightarrow x_i) = 0 \), and thus \( K'(x_i, x_j) = K'(x_j, x_i) = 0 \). If \( \phi(f, x_i \rightarrow x_j) \neq 0 \) and \( \phi(f, x_j \rightarrow x_i) \neq 0 \), without loss of generality, assume the path from \( x_i \) to \( x_r \) is \( p \) and the path from \( x_j \) to \( x_r \) is \( q \), where path \( p \) has length \( k_1 \) and path \( q \) has length \( k_1 \). If \( x_i \) and \( x_j \) are in the same connected component, by definition the reference point \( x_r \) will be the same for \( x_i \) and \( x_j \). If \( \phi(f, x_i \rightarrow x_j) = 0 \), by symmetric-zero axiom \( \phi(f, x_j \rightarrow x_i) = 0 \), and thus \( K'(x_i, x_j) = K'(x_j, x_i) = 0 \). If \( \phi(f, x_i \rightarrow x_j) \neq 0 \) and \( \phi(f, x_j \rightarrow x_i) \neq 0 \), without loss of generality, assume the path from \( x_i \) to \( x_r \) is \( p \) and the path from \( x_j \) to \( x_r \) is \( q \), where path \( p \) has length \( k_1 \) and path \( q \) has length \( k_2 \). Then

\[
K'(x_i, x_j) = \frac{\phi(f, x_i \rightarrow x_j)}{\phi(f, x_j \rightarrow x_i)}. \frac{\phi(f, x_i \rightarrow x_j) \cdots \phi(f, x_{p_{k-1}} \rightarrow x_{p_{k-1}}) \cdots \phi(f, x_{q_{k-1}} \rightarrow x_{q_{k-1}})}{\phi(f, x_j \rightarrow x_i) \cdots \phi(f, x_{p_{k-1}} \rightarrow x_{p_{k-1}}) \cdots \phi(f, x_{q_{k-1}} \rightarrow x_{q_{k-1}})} = 1.
\]
where the last equality is implied by the symmetric cycle axiom. Next, if \( x_i \) and \( x_j \) are not in the same connected component, there is no edge between \( x_i \) and \( x_j \). It implies that either \( \phi(f, x_i \rightarrow x_j) = 0 \) or \( \phi(f, x_j \rightarrow x_i) = 0 \) by the definition of the graph. However, due to the symmetric zero axiom, we should have \( \phi(f, x_i \rightarrow x_j) = \phi(f, x_j \rightarrow x_i) = 0 \). Thus we have \( K'(x_i, x_j) = K'(x_j, x_i) = 0 \).

If \( i \) and \( j \) are not in the same connected component, it implies that at least one \( K'(x_i, x_j) \) and \( K'(x_j, x_i) \) should be zero by definition of the connected component. However, it implies \( K'(x_i, x_j) = K'(x_j, x_i) = 0 \) due to the symmetric zero axiom. Finally, if at least one of \( i, j \) is not in \( [m] \), we have \( K'(x_i, x_j) = K'(x_j, x_i) \) by the definition of \( K' \).

Lastly, we consider the case either \( i \) or \( j \) belongs to \( \{m + 1, \ldots, n\} \). If \( i \in \{m + 1, \ldots, n\} \) and \( j \in [m] \), we have
\[
K'(x_i, x_j) = \phi(f, x_j \rightarrow x_i) / \alpha' = K'(x_j, x_i)
\]
Similarly, the results hold for the case when \( j \in \{m + 1, \ldots, n\} \) and \( i \in [m] \). If we have both \( i, j \) belong to \( \{m + 1, \ldots, n\} \), we have \( K'(x_i, x_j) = K'(x_j, x_i) = 0 \).

Therefore, \( K'(x_i, x_j) = K'(x_j, x_i) \) for any two training points \( x_i, x_j \) (if \( i = j \) this also trivially holds).

**Prove \( K' \) is a continuous positive-definite kernel:** Now, we have \( \phi(f, x_i \rightarrow x_j) = \alpha_i K'(x_i, x_j) \) for any two arbitrary training points \( x_i, x_j \) for some symmetric kernel function \( K' \) satisfying \( K'(x_i, x_i) \geq 0 \) for all training samples \( x_i \).

Next, since the explanation further satisfies the continuity and irreducibility axioms for all kinds of training points, the kernel should be a continuous positive definite kernel by Mercer’s theorem \[63\].

**Prove the other direction:** Assume that some \( K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \) is some continuous positive-definite kernel function and the explanation can be expressed as
\[
\phi(f, x_i \rightarrow x_j) = \alpha_i K'(x_i, x_j), \forall i, j \in [n].
\]
Below we prove that it satisfies the continuity, self-explanation, symmetric zero, symmetric cycle, and irreducibility axioms.

First, it satisfies the continuity axiom since the kernel is continuous. Second, if \( \phi(f, x_i \rightarrow x_i) = 0 \) for some training point \( x_i \), we have \( \alpha_i K'(x_i, x_i) = 0 \). This implies that either \( \alpha_i = 0 \) or \( K'(x_i, x_i) = 0 \) for all \( x \in \mathbb{R}^d \) since it is a positive-definite kernel. Therefore, it satisfies the self-explanation axiom.

Third, we prove that it satisfies the symmetric zero axiom. Since we have \( \phi(f, x_i \rightarrow x_j) \neq 0 \), it implies \( \alpha_i \neq 0 \). Therefore if we have \( \phi(f, x_j \rightarrow x_i) = 0 \) for some \( x_i, x_j \), it implies \( K'(x_i, x_j) = 0 \).

Since the kernel is symmetric, it also implies \( \phi(f, x_j \rightarrow x_i) = \alpha_j K'(x_j, x_i) = 0 \).

Next, we prove that it satisfies the symmetric zero axiom.
\[
\prod_{i=1}^{k} \phi(f, x_{t_i} \rightarrow x_{t_{i+1}}) = \prod_{i=1}^{k} \alpha_i K'(x_{t_i}, x_{t_{i+1}}) = \prod_{i=1}^{k} \alpha_{t_{i+1}} K'(x_{t_{i+1}}, x_{t_i}) = \prod_{i=1}^{k} \phi(f, x_{t_{i+1}} \rightarrow x_{t_i}),
\]
where we use the fact that \( K' \) is a symmetric kernel in the third equation.

Finally, by Mercer theorem, it satisfies the irreducibility axiom.

\[\square\]

**E.2 Proof of Proposition** \[6\]

**Proof:** We solve the following reparameterized objective as in Eqn. \[5\],
\[
\hat{\alpha} = \arg\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{n} \sum_{i=1}^{n} L \left( \sum_{j=1}^{n} \alpha_j K(x_i, x_j), f(x_i) \right) + \frac{\lambda}{2} \alpha^\top \mathbf{K} \alpha \right\}.
\]
By the first-order optimality condition, we have

\[
\frac{\partial}{\partial \alpha} \left( \frac{1}{n} \sum_{i=1}^{n} \mathcal{L} \left( \sum_{j=1}^{n} \alpha_j K(x_i, x_j), f(x_i) \right) + \frac{\lambda}{2} \alpha^T K \alpha \right) \bigg|_{\alpha = \hat{\alpha}} = 0
\]

By the chain rule, we have

\[
\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}' \left( \sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j), f(x_i) \right) K e_i + \lambda \hat{\alpha} = 0,
\]

where \(e_i \in \mathbb{R}^{n \times 1}\) denotes the \(i^{th}\) unit vector and \(\mathcal{L}' \left( \sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j), f(x_i) \right) \in \mathbb{R}\) is the partial derivative of \(\mathcal{L}\) with respect to its first input. Then we arrange the equation:

\[
K \left( \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}' \left( \sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j), f(x_i) \right) e_i + \lambda \hat{\alpha} \right) = 0
\]

Therefore, we must have \(\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}' \left( \sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j), f(x_i) \right) e_i + \lambda \hat{\alpha} \in \text{null}(K)\). It implies that

\[
\hat{\alpha} \in \left\{ -\frac{1}{n \lambda} \sum_{i=1}^{n} \mathcal{L}' \left( \sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j), f(x_i) \right) e_i + \text{null}(K) \right\}.
\]

We note that \(\sum_{j=1}^{n} \hat{\alpha}_j K(x_i, x_j) = (K e_i)^\top \hat{\alpha} = e_i^\top (K \hat{\alpha})\) is a constant for all alphas in the above solution set. Therefore, we have

\[
\hat{\alpha} \in \{ \alpha^* + \text{null}(K) \},
\]

where \(\alpha^*_i = -\frac{1}{n \lambda} \frac{\partial \mathcal{L}(f_K(x_i), f(x_i))}{\partial f_K(x_i)}\) for all \(i \in [n]\).

\[\square\]

### E.3 Proof of Proposition 9

**Proof.** Let \(v \in \text{null}(K)\). We consider the norm of \(f_v = \sum_{i=1}^{n} v_i K(x_i, \cdot)\):

\[
\|f_v\|_H^2 = \langle f_v, f_v \rangle = \left\langle \sum_{i=1}^{n} v_i K(x_i, \cdot), \sum_{i=1}^{n} v_i K(\cdot, x_i) \right\rangle \quad (\text{kernels are symmetric to its inputs})
\]

\[
= \sum_{i=1}^{n} v_i \langle K(x_i, \cdot), v \rangle \sum_{i=1}^{n} v_i \langle v, K(\cdot, x_i) \rangle
\]

\[
= v^\top K v
\]

\[
= 0 \quad (K v = 0),
\]

which implies that \(f_v(x) = 0\) for all \(x \in \mathbb{R}^d\).

\[\square\]

### E.4 Proof of Corollary 12

By plugging the kernel function \(K_{1L}(x, z) = \langle \Phi_{\theta_1}(x), \Phi_{\theta_1}(z) \rangle, \forall x, z \in \mathbb{R}^d\) to Proposition 8, we have

\[
\hat{f}_K(\cdot) = \sum_{i=1}^{n} \alpha^*_i K_{LL}(x_i, \cdot), \quad \text{with } \alpha^*_i = -\frac{1}{n \lambda} \frac{\partial \mathcal{L}(\hat{f}_K(x_i), f(x_i))}{\partial \hat{f}_K(x_i)}.
\]