A Connection between the Fisher and Hessian

The Fisher information matrix of a conditional distribution parameterized by θ, $p_\theta(y|x_i)$, where we do not have an explicit representation for $p(x)$, is

$$\begin{aligned} F_\theta &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{p_\theta} \nabla_\theta \log p_\theta(y|x_i) \nabla_\theta \log p_\theta(y|x_i)^T \\ &= \frac{1}{n}\sum_{i=1}^{n} \Big[\underbrace{\mathbb{E}_{p_\theta} \frac{\nabla^2_\theta p_\theta(y|x_i)}{p_\theta(y|x_i)}}_{=0} - \mathbb{E}_{p_\theta} \nabla^2_\theta \log p_\theta(y|x_i) \Big] \\ &= -\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{p_\theta} \nabla^2_\theta \log p_\theta(y|x_i). \end{aligned}$$

The second equality can be obtained by carrying out the differentiation $\nabla^2_\theta \log p_\theta(y|x)$. The third equality critically relies on the expectation being taken with respect to the model's distribution, as well as being able to switch the order of the expectation and differentiation.

When our model has learned a distribution $p_\theta(y|x)$ close to the "true" distribution $p(y|x)$, we may approximate F_θ by replacing $\mathbb{E}_{p_\theta}$ with a Monte Carlo estimate based on the target values, y_i, in the training set. This gives us

$$F_\theta \approx -\frac{1}{n}\sum_{i=1}^{n} \nabla^2_\theta \log p_\theta(y_i|x_i) = H_\theta.$$

The final equality with H_θ (as defined in Eq. (2)) holds when $\ell(z,\theta) = -\log p_\theta(y_i|x_i)$. Therefore, our two expressions for influence, Eq. (3) and Eq. (6), approximately agree when the loss is the negative log likelihood, and y given x is actually distributed as $p_{\theta^*}(y|x)$. In practice, we have found that the Fisher and Hessian return similar examples when used to compute $\mathcal{I}_{\text{test},i}$.

B Proofs of RelatIF

Recall **Proposition 3.1**: *Assume that Eq. (4) and Eq. (5) hold with equality. Then Eq. (10) is equivalent to*

$$\underset{i\in\{1,\ldots,n\}}{\arg\max} \frac{|\mathcal{I}_{\text{test},i}|}{\|H^{-1}_{\theta^*} g_i\|}. \quad (14)$$

Proof. The $\arg\max$ in Eq. (10) is just a search over the n examples in our training set. Therefore the solution to Eq. (10) amounts to solving $\max|\ell(z_{\text{test}}, \theta^*_{i,\epsilon_i}) - \ell(z_{\text{test}}, \theta^*)|$ subject to the constraints for each i, then choosing the z_i for which this quantity is largest. We assume Eq. (5) holds with equality, and so it follows

$$|\ell(z_{\text{test}}, \theta^*_{i,\epsilon_i}) - \ell(z_{\text{test}}, \theta^*)| = |\mathcal{I}_{\text{test},i}\epsilon_i|.$$

Note, that $\mathcal{I}_{\text{test},i}\epsilon_i$ is a linear function in ϵ_i. The maximum of its absolute value will therefore be on one of the endpoints imposed by the constraint on ϵ_i.

We assume that approximation in Eq. (4) is exact, yielding $\theta^*_{i,\epsilon_i} - \theta^* = -H^{-1}_{\theta^*} g_i \epsilon_i$. Using this equality to describe the change in parameters, the constraint in Eq. (10) can be written as

$$\|H^{-1}_{\theta^*} g_i \epsilon_i\|^2 \le \delta^2 \implies |\epsilon_i| \le \frac{|\delta|}{\|H^{-1}_{\theta^*} g_i\|}.$$

We now have an explicit constraint on ϵ_i. Plugging either endpoint of this interval into $|\mathcal{I}_{\text{test},i}\epsilon_i|$ yields the same value. Substituting that value into the outer $\arg\max$ we get,

$$\underset{i\in\{1,\ldots,n\}}{\arg\max} \frac{|\mathcal{I}_{\text{test},i}||\delta|}{\|H^{-1}_{\theta^*} g_i\|}.$$

Since δ does not depend on i, it can be dropped from the $\arg\max$, and the result follows. □

Recall **Proposition 3.3**: Assume Eq. (8) and Eq. (9) hold with equality. Then Eq. (12) is equivalent to

$$\underset{i\in\{1,\ldots,n\}}{\arg\max} \frac{|\mathcal{I}_{\text{test},i}|}{\sqrt{\mathcal{I}_{i,i}}}. \quad (15)$$

Proof. The proof follows the proof of Proposition 3.1. We need only consider the new constraint. Assuming Eq. (9) holds with equality, we have

$$\ell(z_j, \theta^*_{i,\epsilon_i}) - \ell(z_j, \theta^*) = \mathcal{I}_{j,i}\epsilon_i = -g_j^T F^{-1}_{\theta^*} g_i \epsilon_i.$$

We introduce the notation $g(z) = \nabla_\theta \ell(z, \theta^*)$ to signify the gradient of the loss of a point $z = (x, y)$ sampled from the model's distribution, p^*_θ. We substitute into the constraint, which yields

$$\mathbb{E}_{p_{\theta^*}}(-g(z)^T F^{-1}_{\theta^*} g_i \epsilon_i)^2 \le \delta^2.$$

Expanding and rearranging the left hand side, yields

$$\begin{aligned} &\mathbb{E}_{p_{\theta^*}}(g(z)^T F^{-1}_{\theta^*} g_i \epsilon_i)^T (g(z)^T F^{-1}_{\theta^*} g_i \epsilon_i) \\ &\qquad = g_i^T F^{-1}_{\theta^*} \mathbb{E}_{p_{\theta^*}}\left[g(z)g(z)^T\right] F^{-1}_{\theta^*} g_i \epsilon_i^2, \end{aligned}$$

where we have moved the constant terms outside of the expectation. Because our loss functions here is negative log-likelihood, $\mathbb{E}_{p_{\theta^*}}[g(z)g(z)^T]$ is the is the definition of the Fisher information matrix, F^*_θ. It agrees with the presentation in Section 2.2 when we replace the expectation over $z = (x, y)$ with $\sum_j E_{p_{\theta^*}(y|x_j)}$. Thus the constraint can be reduced to

$$g_i^T F^{-1}_{\theta^*} g_i \epsilon_i^2 = \mathcal{I}_{i,i}\epsilon_i^2 \le \delta^2.$$

The Fisher, F_{θ^*}, is positive definite, therefore $\mathcal{I}_{i,i}$ is positive, and the constraint is equivalent to $|\epsilon_i| \le \frac{|\delta|}{\sqrt{\mathcal{I}_{i,i}}}$. The rest of the proof follows Proposition 3.1. □

C Computational Considerations

Inverting the Hessian The Hessian of the total loss, H_{θ^*}, is positive definite if θ^* is the local minimum of the objective function. If the requirements for positive definiteness of H_θ are not met, for example because the optimization procedure has not fully converged, we can add a damping term to the Hessian to make it invertible (i.e., $\tilde{H}_\theta = H_\theta + \lambda I$ where I is the identity matrix). Adding the damping term is equivalent to L_2 regularization, and allows us to form a convex quadratic approximation to the loss function. Koh & Liang (2017) demonstrate that influence functions computed with a damping term still give meaningful results in practice. Empirically we have also found this to be the case (see Fig. 6).

IF and RelatIF in large models The Hessian of the total loss, H_{θ^*}, is a P by P matrix, where P is the number of model parameters. In even just mid-sized models, it becomes near impossible to explicitly form H_{θ^*} in memory. Koh & Liang (2017) propose using LiSSA (Agarwal et al., 2017) to estimate the inverse Hessian vector products needed to compute influence. We have found that this works well in practice after some tuning. However, because the denominator in RelatIF is a function of the training example z_i we cannot use the same s_{test} trick suggested by Koh & Liang (2017). This has led us to explore using a number of block diagonal approximations to the Hessian, principally motivated by K-FAC (Martens & Grosse, 2015). Specifically, we use these approximations to pre-compute the denominator in RelatIF ($\sqrt{\mathcal{I}_{i,i}}$ or $\|H_{\theta^*}^{-1} g_i\|$) for every training example z_i. We are then able to use s_{test}.

D Qualitative comparison of θ-RelatIF and ℓ-RelatIF

Fig. 7 offers a comparison between the top positively influential training examples recovered by ℓ-RelatIF and θ-RelatIF. As this figure shows, the examples returned by both method are visually similar.

E More explanation examples

Fig. 8 and Fig. 9 offer more examples of using RelatIF and IF for explaining the prediction of a ConvNet trained on CIFAR10 data set.

F Comparison to support vectors

Consider a soft-SVM optimization problem, with a dimensionality of the feature space lower than the number of training points. Via working with a dual formulation of the objective, one can uncover that the optimal weights $w^\star$ lie in the linear span of some training samples, called support vectors (see, e.g., (Shalev-Shwartz & Ben-David, 2014), Ch.15).

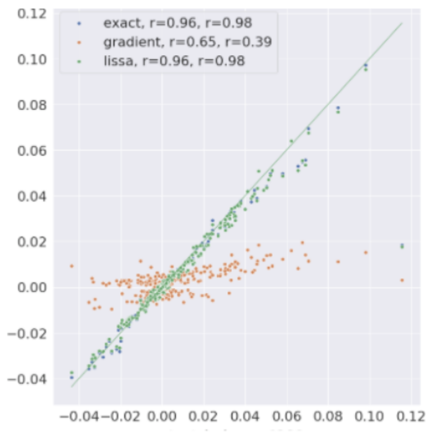
Let $\psi(\cdot)$ map inputs x to a feature space. Define a kernel function $K(x_i, x_j) = \langle\psi(x_i), \psi(x_j)\rangle$ for all inputs x_i, x_j. Let w^* be an output of a solution to a soft-margin SVM.

The support vectors are the training samples indexed by i, such that $y_i\langle w^*, \psi(x_i)\rangle \leq 1$, i.e., support vectors are the training samples that are inside the margin or misclassified. By computing the gradients of a soft-SVM objective, one can see that IF induces a ranking of training samples according to $y_i y_{test} K(x_i, x_{test})$, where $\{x_i\}_i$ are support vectors (IF equals to zero for training samples that are not support vectors). In other words, the highest influence samples are support vectors that are most similar to the test point in the feature space. Similarly, repeating the same calculation for the loss-relative influence, one would rank the support vectors based on $y_i y_{test} K(x_i, x_{test})/\sqrt{K(x_i, x_i)}$.

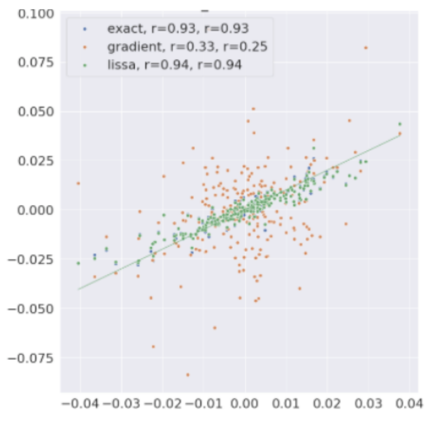
In summary, IF gives a way to distinguish which of the training samples *among all support vectors* are the ones that are most similar to the test sample in the feature space. RelatIF normalizes this similarity in the feature space by the norm of the training sample in the feature space, which is equivalent to IF evaluated on training samples that are unit vectors in the feature space.

G Application of example-based explanations

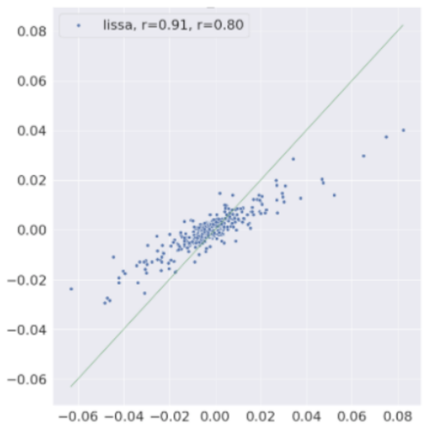
Consider the example shown in Fig. 10. The input image in the top left corner is classified as a bird, however, the predicted probability for class plane is also very high. Using RelatIF we can identify the top relevant training examples to each label (i.e., bird and plane) and assess the model's decision based on them. Comparing the similarity between the test image and relevant training examples to each class, suggests that this image should be classified as a plane. Also, these explanations could be used as a guide to collect more training examples for improving the model performance (e.g., what kind of training examples are useful for correcting a specific misclassification).



(a) Logistic Regression on MNIST



(b) ConvNet on MNIST



(c) AlexeNet on small ImageNet

Figure 6: Evaluating the accuracy of the estimated error for leave-one-out retraining by influence functions with different hessian approximations. in the figure, "exact" refers to exact Hessian, "gradient" referes to using identity matrix for Hessian and "lissa" refers to using the LiSSA method for approximating inverse Hessian vector products. In the case of AlexNet, the results are only reported for LiSSA due computational/memory complexity. For all of these models a small damping coefficient has been added to the diagonal of the Hessian to make it invertible. As this figure shows, in the case of LiSSA and exact Hessian, the correlation between influence function estimation for change of loss and the actual leave-one-out retraining loss is high.

H Qualitative comparison with k-nearest neighbors

Fig. 11 offers a qualitative comparison between using RelatIF and k-nearest neighbors for explaining the prediction of a ConvNet trained on CIFAR10 data set.

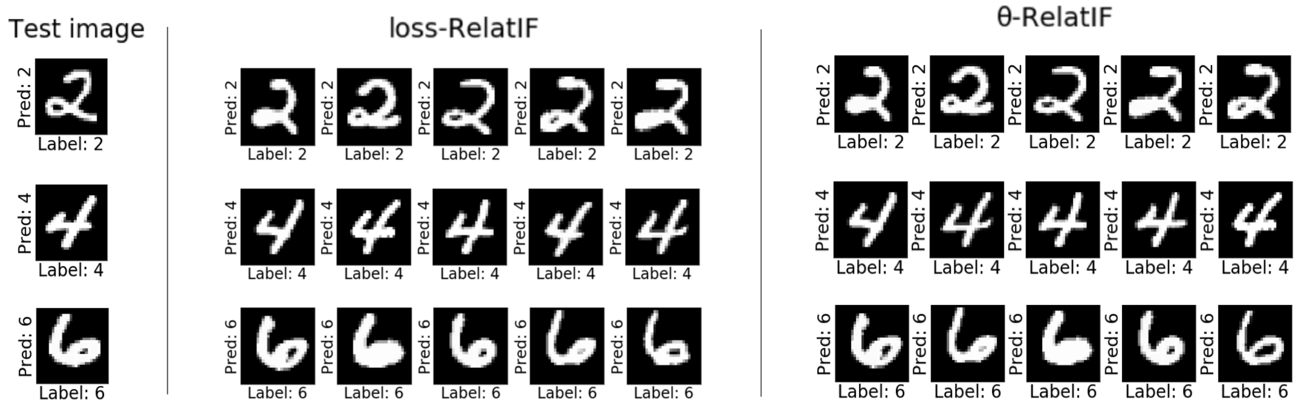


Figure 7: Comparison of the top positively influential training examples recovered by ℓ-RelatIF and θ-RelatIF. The classifier is a logistic regression trained on MNIST. Each row shows a test sample and the top five positively influential training examples for the predicted label selected by each method. The true class and the predicted label for each example is marked. The example returned by both method are visually similar.

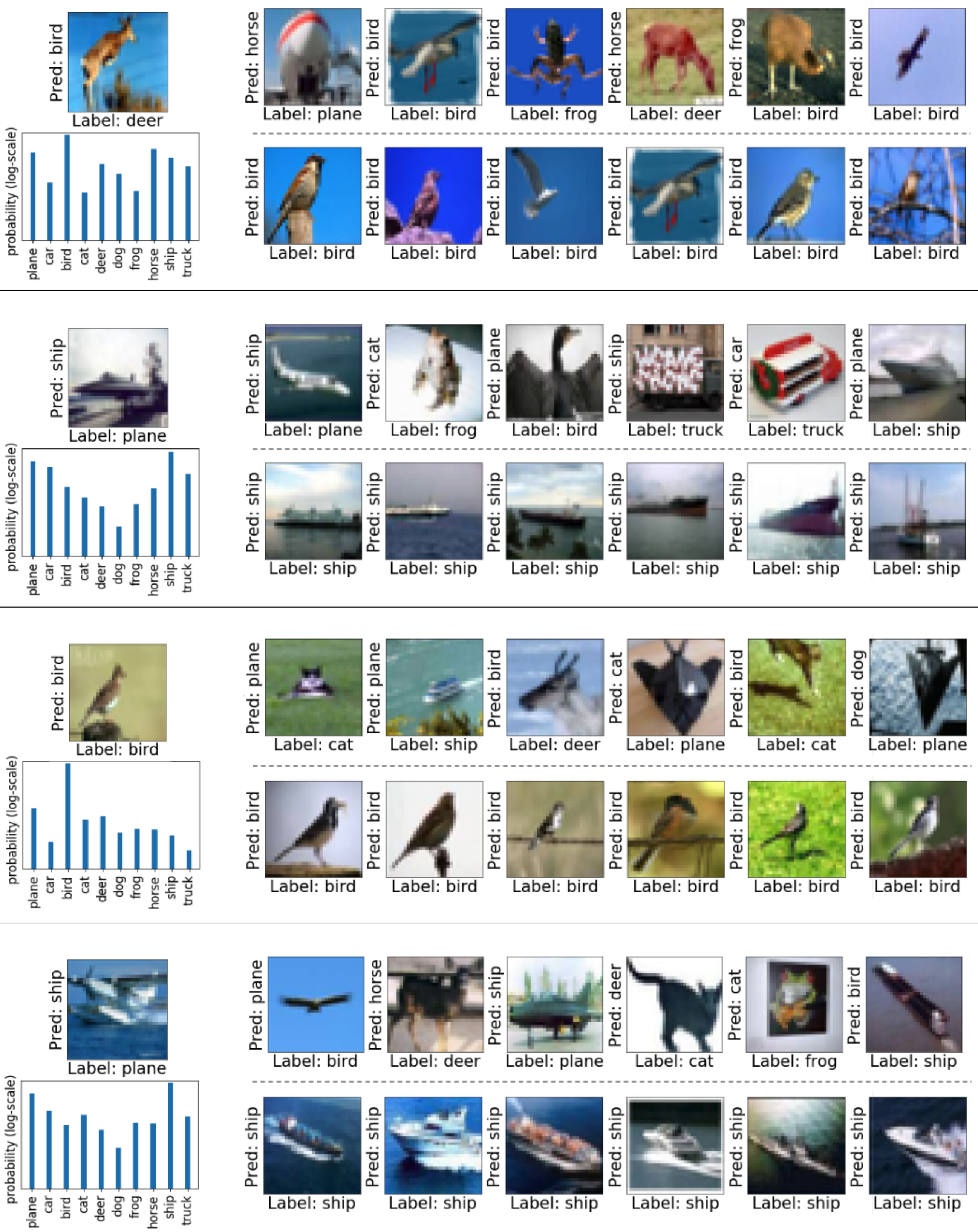


Figure 8: Generating example-based explanation using IF and RelatIF for the model prediction. The model is a ConvNet trained on CIFAR10 data set. For each test sample in the left column, the recovered training examples for explaining the model prediction using IF and RelatIF are depicted in the first and second row, respectively.

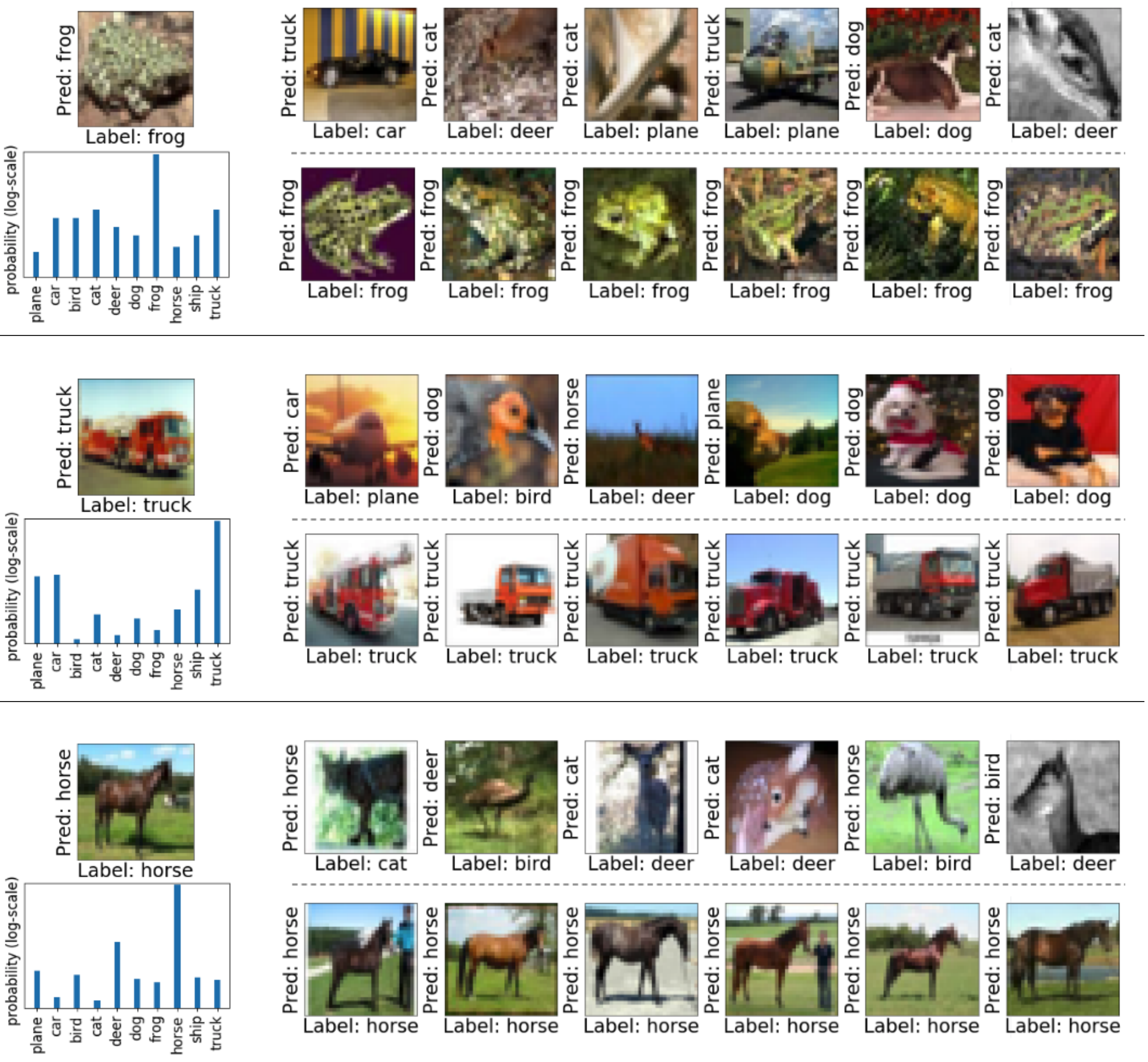


Figure 9: Generating example-based explanation using IF and RelatIF for the model prediction. The model is a ConvNet trained on CIFAR10 data set. For each test sample in the left column, the recovered training examples for explaining the model prediction using IF and RelatIF are depicted in the first and second row, respectively.

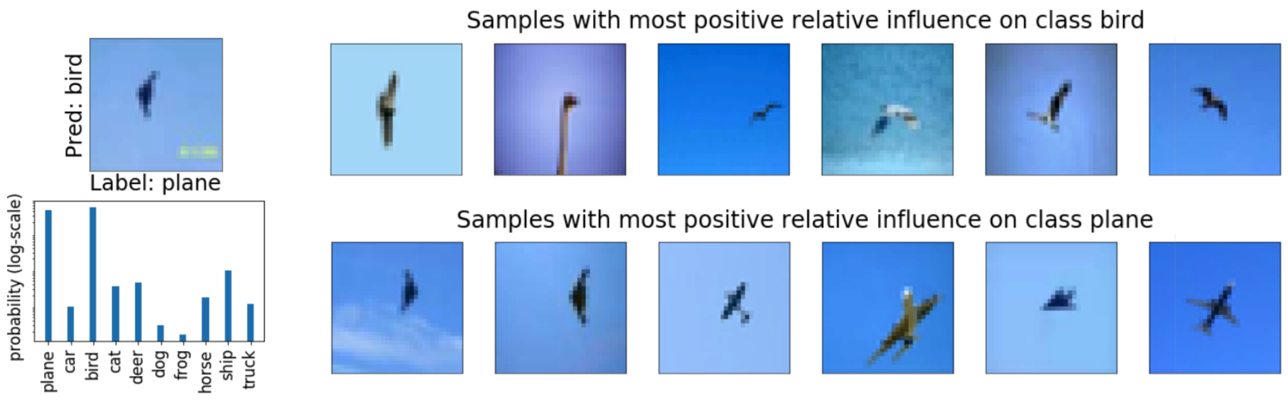


Figure 10: Using RelatIF to identify training examples with the top positive relevant influence on the the predicted (bird) and true label (plane).

I The connection between RelatIF and leave-one-out retraining

RelatIF is an approximation for the ratio of change in the test sample loss to global changes of the model (i.e., norm of change in the model parameters or root sum of square change in loss over the training set) resulted from leave-one-out retraining. Table 4 reports this ratio for different methods.

Method	$\Delta\ell_{\text{test}}/\|\Delta\theta\|$	$\Delta\ell_{\text{test}}/\sqrt{\sum(\Delta\ell_i)^2}$
IF	0.311 ± 0.167	0.016 ± 0.008
θ-RelatIF	0.459 ± 0.226	0.026 ± 0.010
ℓ-RelatIF	0.475 ± 0.229	0.027 ± 0.010
Nearest-N	0.304 ± 0.182	0.018 ± 0.009

Table 4: The ratio of change in the test loss ($\Delta\ell_{\text{test}}$) to the global changes, i.e., the norm of the change in the model parameters ($\|\Delta\theta\|$), or root sum of square change in loss over the training set ($\sqrt{\sum(\Delta\ell_i)^2}$). The results come from removing the mostly positively influential training sample as determined by different methods, then retraining the model (i.e., leave-one-out retraining). The model is a logistic regression trained on MNIST. The experiment is repeated for 100 randomly selected test samples. The mean $\pm$ standard error are reported. One can approximate the leave-one-out change of these ratios using θ-RelatIF and ℓ-RelatIF, respectively. Note that these ratios are higher for the samples identified by RelatIF than for those identified by IF or NN. This is due to the fact that RelatIF identifies the points that maximize these ratios.

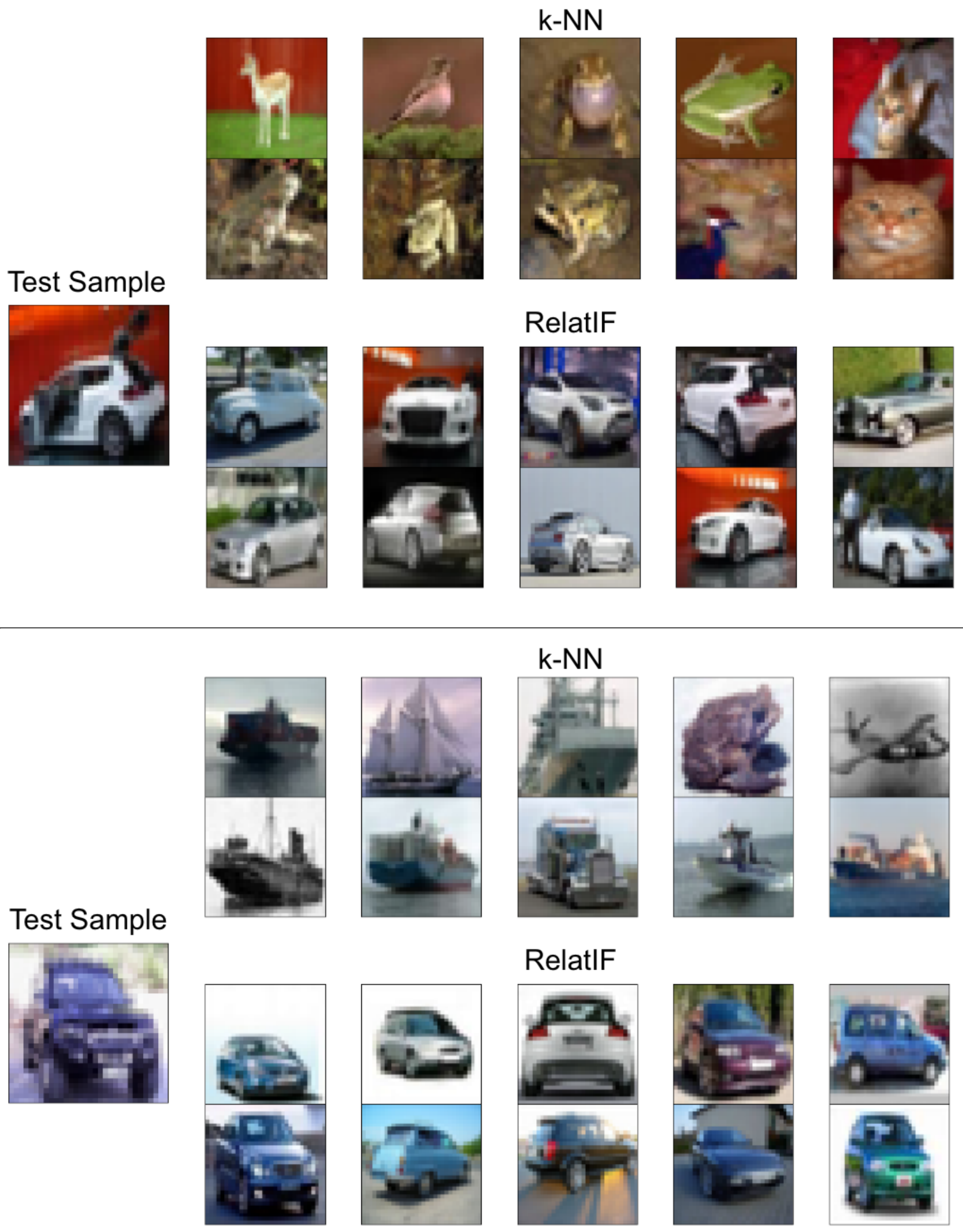


Figure 11: Comparing the k-nearest neighbors (k-NN) to the most (positive) relatively influential samples (RelatIF) for a convolutional neural network trained on CIFAR10. The two test samples were correctly classified by the model. The k-nearest neighbors are similar in pixel composition, but are semantically different from the test sample. They provide no evidence to explain why the model has correctly classified the test sample. Conversely, the samples returned by RelatIF are of the matching class. They suggest the model has been exposed to relevant training data, and has learned something useful.