A Computing Mutual Information $I[Z_I, I]$

For two random variables Z_I and I, where Z_I is dependent on I, we can calculate the mutual information via KL-divergence:

$$I(Z_I, I) = D_{KL}(P(Z_I|I)||P(Z_I)) \tag{5}$$

KL-divergence is defined as

$$D_{KL}(P, Q) = -\int_x P(x) \log \frac{Q(x)}{P(x)} dx \tag{6}$$

which means for calculating KL-divergence we need to integrate over the entire probability space. However, Given two random variables $P \sim \mathcal{N}(\mu_1, \sigma_1)$ and $Q \sim \mathcal{N}(\mu_2, \sigma_2)$ that are Gaussian distributed, the KL-divergence between this two random variables can be calculated in closed form:

$$D_{KL}(P, Q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{7}$$

The bottleneck is constructed by masking the input $Z_I = \Lambda I + (1 - \Lambda)\epsilon_I$, where ϵ_I is the input noise and $\epsilon_I \sim \mathcal{N}(\mu_I, \sigma_I)$. The distribution of Z_I given input I, $P(Z_I|I)$, is thus (Remark 2):

$$P(Z_I|I) \sim \mathcal{N}(\Lambda I + (1 - \Lambda)\mu_I, (1 - \Lambda)^2\sigma_I^2) \tag{8}$$

As explained in Section 3.2, Z_I is conditioned on Z_G by using $Z_I = \Lambda Z_G + (1 - \Lambda)\epsilon$, and $Z_G = \lambda_G I + (1 - \lambda_G)\epsilon_G$, by substitution we have:

$$P(Z_I) \sim \mathcal{N}(\lambda_G \Lambda I + (1 - \lambda_G \Lambda)\mu_I, (1 - \lambda_G \Lambda)^2\sigma_I^2) \tag{9}$$

We are implicitly introducing an independence assumption for the random variable $P(Z_I)$ through our Gaussian masking ($Z_I = \Lambda Z_G + (1-\Lambda)\epsilon$) formulation of $P(Z_I)$. Where each elements of $P(Z_I)$ are constructed by an independent Gaussian masking. A better approximation would not impose such an independence assumption. Please note that in original IBA [19] the same independence assumption exists for $P(Z)$. Since KL-divergence is additive for independent distributions ($D_{KL}(P||Q) = \sum_k D_{KL}(P_k||Q_k)$), we can calculate the KL-divergence of $P(Z_I|I)$ and $P(Z)$ by summing over the KL-Divergence of there elements. Therefore (proposition 3):

$$\begin{aligned}
&D_{KL}[P(Z_{I,k}|I_k)||P(Z_{I,k})] \\
&= \log \frac{(1 - \lambda_{G,k}\Lambda_k)\sigma_{I,k}}{(1 - \Lambda_k)\sigma_{I,k}} + \frac{(1 - \Lambda_k)^2\sigma_{I,k}^2}{2(1 - \lambda_{G,k}\Lambda_k)^2\sigma_{I,k}^2} \\
&\quad + \frac{((\Lambda_k I_k + (1 - \Lambda_k)\mu_{I,k}) - (\lambda_{G,k}\Lambda_k I_k + (1 - \lambda_{G,k}\Lambda_k)\mu_{I,k}))^2}{2(1 - \lambda_{G,k}\Lambda_k)^2\sigma_{I,k}^2} - \frac{1}{2} \\
&= \log \frac{1 - \lambda_{G,k}\Lambda_k}{1 - \Lambda_k} + \frac{(1 - \Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2} \\
&\quad + \frac{(\Lambda_k I_k + \mu_{I,k} - \Lambda_k\mu_{I,k} - \lambda_{G,k}\Lambda_k I_k - \mu_{I,k} + \lambda_{G,k}\Lambda_k\mu_{I,k})^2}{2(1 - \lambda_{G,k}\Lambda_k)^2\sigma_{I,k}^2} - \frac{1}{2} \\
&= \log \frac{1 - \lambda_{G,k}\Lambda_k}{1 - \Lambda_k} + \frac{(1 - \Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2} + \frac{(\Lambda_k I_k - \Lambda_k\mu_{I,k} - \lambda_{G,k}\Lambda_k I_k + \lambda_{G,k}\Lambda_k\mu_{I,k})^2}{2(1 - \lambda_{G,k}\Lambda_k)^2\sigma_{I,k}^2} - \frac{1}{2} \\
&= \log \frac{1 - \lambda_{G,k}\Lambda_k}{1 - \Lambda_k} + \frac{(1 - \Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2} + \frac{(I_k - \mu_{I,k})^2(\Lambda_k - \lambda_{G,k}\Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2\sigma_{I,k}^2} - \frac{1}{2}
\end{aligned} \tag{10}$$

B Derivation of Proposition 4

The derivation is inspired by [40]. Contrary to [40], in this work we derive the exact representation of $I(Z, Y)$ instead of a lower bound. Given X as input, Y as output, Z as bottleneck (masked input), we

assume $Y \leftrightarrow X \leftrightarrow Z$, this means that Y and Z are independent given X. This assumption is fulfilled by masking scheme and data generation process: for Z we have $Z = m(X, \epsilon, \lambda)$ (where function m represents the masking scheme), thus Z depends on X given sampled noise and mask; for Y we have $Y = g(X)$, where function g represents the data generation process for the task. Therefore:

$$p(x, y, z) = p(y|x, z)p(z|x)p(x) = p(y|x)p(z|x)p(x) \tag{11}$$

Thus we can also calculate $p(y|z)$ by:

$$p(y|z) = \frac{p(z, y)}{p(z)} = \int \frac{p(y|x)p(z|x)p(x)}{p(z)} dx \tag{12}$$

And mutual information between Z and Y:

$$\begin{aligned} I(Z, Y) &= \int p(y, z) \log \frac{p(y, z)}{p(y)p(z)} dydz \\ &= \int p(y, z) \log \frac{p(y|z)}{p(y)} dydz \end{aligned} \tag{13}$$

However, calculating $p(y|z)$ requires integral over x, which is intractable. We then apply the variational idea to approximate $p(y|z)$ by $q_\theta(y|z)$ instead. $q_\theta(y|z)$ represents the neural network part after bottleneck, θ indicates parameter of the neural network part.

$$\begin{aligned} I(Z, Y) &= \int p(y, z) \log \frac{p(y|z)}{p(y)} \frac{q_\theta(y|z)}{q_\theta(y|z)} dydz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \int p(y, z) \log \frac{p(y|z)}{q_\theta(y|z)} dydz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \int p(z)(\int p(y|z) D_{KL}[p(y|z)||q_\theta(y|z)]dy)dz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \int p(z) D_{KL}[p(y|z)||q_\theta(y|z)]dz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \mathbb{E}_{z\sim p(z)}[D_{KL}[p(y|z)||q_\theta(y|z)]] \end{aligned} \tag{14}$$

C Derivation of Theorem 5

Proof: Here we want to prove that for per-sample Information Bottleneck under classification task, optimizer of cross entropy loss is also the optimal value for mutual information $I[Y, Z]$. Previous work concludes that the optimizer of $\min \mathbb{E}_{\epsilon\sim p(\epsilon)}[-\log q_\theta(y_{sample}|z)]$ is a lower bound of $I(Y, Z)$ [40]. To ease the effort of readers searching across literatures, we summarize the proof in [40] in Appendix G. We now prove this optimizer is not only the lower bound of $I(Y, Z)$, but also the optimizer for $I(Y, Z)$ under per-sample setting (local explanation setting). Now we consider the second term in Eq. (14) which we neglected during mutual information calculation:

$$\mathbb{E}_{z\sim p(z)}[D_{KL}[p(y|z)||q_\theta(y|z)]] \tag{15}$$

This term equals to zero if $p(y|z) \equiv q_\theta(y|z)$. The local explainable set contains a batch of neighbours of X_{sample}. We make two assumptions on the data points in the local explainable set. One is all data points in the local explainable set have the same label distribution (for an Oracle classifier), as data points are only slightly perturbed from X_{sample}. Another one is that distribution of Z can be considered as equivalent inside local explainable set, i.e. the latent features are equivalent for samples that are in the local explanation set of X (we know this is not true for adversarial samples, however for many samples in the neighborhood which have the same label with X this is true). Now

we approximate $p(y|z)$ by:

$$\begin{aligned} p(y|z) &= \frac{1}{N}\sum_{n=1}^{N}\frac{p(y|x_n)p(z|x_n)}{p(z)} \qquad \text{(local set of sample)} \\ &= \frac{1}{N}\sum_{n=1}^{N}\frac{p(y|x_{sample}+\epsilon)p(z|x_{sample}+\epsilon)}{p(z)} \\ &= \frac{p(y|x_{sample})p(z|x_{sample})}{p(z)} \qquad \text{(use two assumptions above)} \\ &= p(y|x_{sample}) \end{aligned} \tag{16}$$

$p(z|x_{sample}) \equiv p(z)$ because we consider a local dataset, and use the second assumption that Z is equivalent given data in a local set:

$$p(z) = \int p(z|x)p(x)dx \approx \frac{1}{N}\sum_{n=1}^{N}p(z|x_n) = \frac{1}{N}\sum_{n=1}^{N}p(z|x_{sample}+\epsilon) = p(z|x_{sample}) \tag{17}$$

Now $p(y|z) \equiv q_\theta(y|z)$ can be easily proved, for $p(y|z)$:

$$p(y|z) = p(y|x_{sample}) = \begin{cases} 1, & \text{if } y = y_{sample} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

For $q_\theta(y|z)$, the optimizer maximize $\mathbb{E}_\epsilon[log(q_\theta(y_{sample}|z))]$. Then we have:

$$q_\theta(y|z) = \begin{cases} 1, & \text{if } y = y_{sample} \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

Thus they are equivalent, and the KL divergence is zero. As a result, we can remove the inequality in variational step under assumption that we are generating attribution per sample, and assuming local explanation.

D Details and Hyper-parameters of Experiment Setup

D.1 Hyper-parameters and Implementation of Attribution Methods

Most hyper-parameters used for ImageNet experiments are presented in the body of the text. We would like to supplement the setting of generative model. The generative adversarial model for estimating Z_G uses a discriminator with 3 CNN layers followed by 2 fully connected layers, and contains 200 samples in target dataset. During training, we update the discriminator's parameter after every 5 generator updates. To attribute the 4-layer LSTM model trained on IMDB dataset, we insert a bottleneck after the last LSTM layer, the parameter β at this bottleneck is 15. The learning rate for this bottleneck at hidden layer is 5×10^{-5}. The generative adversarial model uses a single Layer RNN as discriminator. For the input bottleneck at embedding space, we use $\beta_{in} = 30$, $lr = 0.5$, and *optimization step* $= 30$.

For Extremal Perturbations, size of the perturbation mask is a hyper-parameter. We set the mask size to be 10% of the image size for EHR experiment, as all images for EHR have bounding box covering less than 33% of the image. For the rest of evaluations of Extremal Perturbation, we use default mask size implemented in public code framework.

Regarding the parameters of Integrated Gradients (IG), we use the default values (proposed in the original paper [2]). The number of integrated points is 50 and the baseline value is 0. We will add the details in the appendix. These values are commonly used and we observe that the method with default parameters performs well in the quantitative experiment (Fig. 5b) as explained.

D.2 Hyper-parameters of InputIBA

In Fig. 7 we observe the effect of β in InputIBA optimization. Similar to IBA [19] it controls the ammount of information passing through the bottleneck. Fig. 8 shows the effect of number of optimization steps in InputIBA. Fig. 9 shows how the attribution map varies while the number of optimizations steps of the generative adversarial model changes.

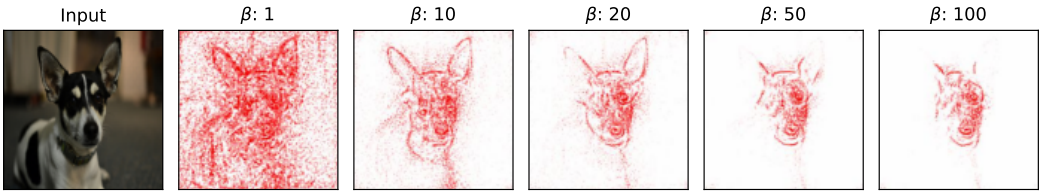

Figure 7: Influence of β of the InputIBA.

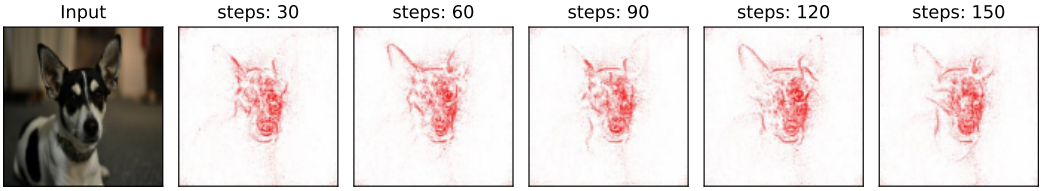

Figure 8: Influence of optimization steps of the InputIBA.

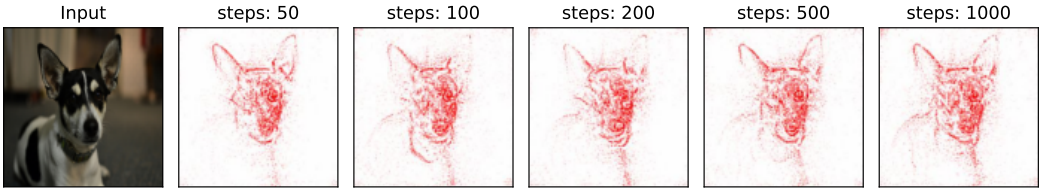

Figure 9: Influence of optimization steps of the generative adversarial model.

Method	Insertion AUC	Deletion AUC
InputIBA	0.710 ± 0.005	0.045 ± 0.001
IBA	0.713 ± 0.004	0.090 ± 0.002
GradCAM	0.703 ± 0.005	0.133 ± 0.003
Guided BP	0.529 ± 0.004	0.132 ± 0.002
Extremal Perturbation	0.676 ± 0.004	0.135 ± 0.003
DeepSHAP	0.430 ± 0.004	0.196 ± 0.003
Integrated Gradients	0.358 ± 0.004	0.210 ± 0.003

Table 3: Mean and the standard error of insertion/deletion AUC for ImageNet dataset. We show the statistical information with standard error in the table.

D.3 Insertion/Deletion

Instead of inserting or deleting one single pixel/token and report the model prediction on modified input at each step, we apply batch-wise pixel/token insertion and deletion. One advantage of batch-wise modification is that the evaluation is accelerated by leveraging the batch processing ability of existing deep learning framework. Also batch-wise modification on input stabilizes the output, as perturb only one pixel or token cannot change the overall information of the input effectively, thus introduces strong deviation on model prediction. On both ImageNet and IMDB dataset, we set the batch size to be 10. For images in ImageNet, we delete pixel by replacing it with black pixel. In insertion test, we insert pixel on a blurred image rather than a black canvas, since insert pixel on black canvas generates stronger adversarial effect than in deletion test. We generate the blurred image by applying Gaussian blur with kernel size equal to 29, standard deviation for Gaussian kernel is 15. For texts in IMDB dataset, both insertion and deletion are performed by replacing the token in text with <unk> token. Insertion/Deletion results on ImageNet dataset and their corresponding standard deviation are presented in Table 3.

D.4 Remove-and-Retrain (ROAR)

In Remove-and-Retrain (ROAR), we divide CIFAR10 dataset into train set, validation set and test set. In training phase, a classifier is trained on the train set with 30 training epochs. After training is complete, we apply the attribution method on trained model for all images in 3 subsets. In retrain phase, for each attribution method, we perturb all 3 subsets based on attribution maps. As a result, we generate 9 perturbed dataset using different perturbation rate (ranging from 10% to 90%, with 10% step). A model is trained from scratch on perturbed train set with 30 training epochs, and we report the final performance of this model on perturbed test set.

E Qualitative Results of Attribution Methods

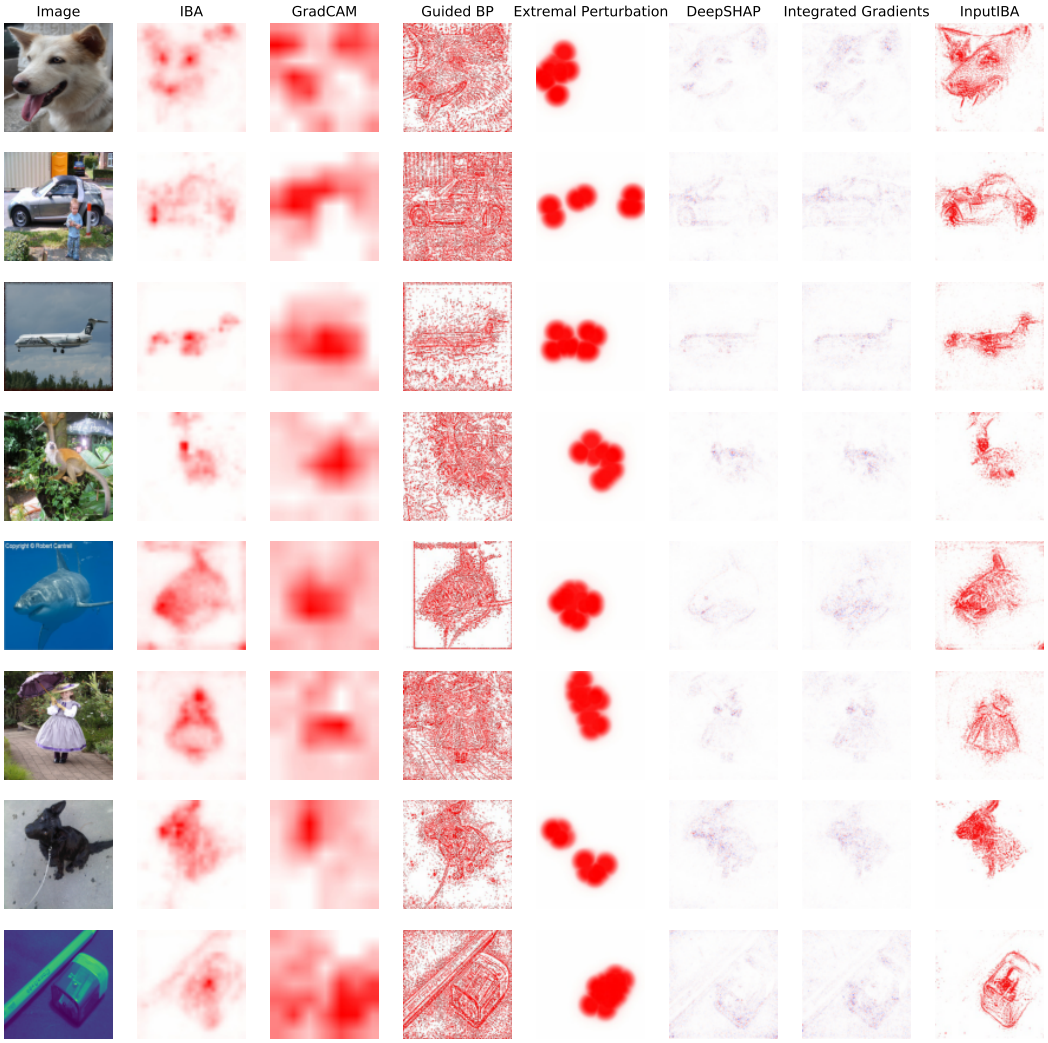

Figure 10: **Qualitative Comparison (ImageNet)**: We conduct qualitative comparision between attribution methods on more sample images form ImageNet validation set.

F The case for EHR (vs. BBox metric in IBA [15])

Here we explain the limitations of *bbox* evaluation metric used in the [15] with three synthetic examples. As illustrated in Fig. 11, we assume an object covers 25% area of an image. We also assume three attribution methods and their attribution maps. As illustrated in Fig. 11a, method (a)

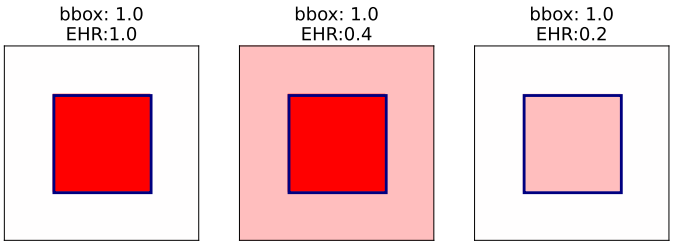

Figure 11: **Synthetic Examples**: We synthesize three attribution maps of an imaginary image. The object is surrounded by a bounding box, which is annotated with a blue box. Values of the *bbox* metric and EHR are also shown on the top of each figure. (a): Attribution scores within the bounding box are 1.0 (the maximal value), outside the bounding box are 0 (the minimal value). (b): Attribution scores within the bounding box are 1.0, outside the bounding box are 0.25. (c) Attribution scores within the bounding box are 0.25, outside the bounding box are 0.

only highlights the pixels within the bounding boxes. In this case, both the *bbox* metric and EHR are 1.0. In Fig. 11b, method (b) additionally highlights the region outside the bounding box, the scores outside the bounding box is lower than that within the bounding box. Thus, the *bbox* metric remains 1.0. However, method (b) is sub-optimal to method (a) as it erroneously highlights the non-object pixels. In Fig. 11c, method (c) only highlights the pixels within the bounding box, but with lower scores. Thus, method (c) is also considered to be sub-optimal to method (a). The *bbox* metric in [15] cannot distinguish these three cases, while EHR can: the EHR of method (b) and (c) are substantially lower than that of method (a).

G Optimization of $I[Y, Z]$ [40]

For easier reference, we provide a summary of derivation of the lower bound of the mutual information $I(Z, Y)$ as done in [40]:

$$\begin{aligned} I(Z,Y) &\geq \int p(y,z) \log \frac{q_\theta(y|z)}{p(y)} dydz \\ &= \int p(y,z) \log q_\theta(y|z) dydz + H(Y) \end{aligned} \tag{20}$$

The second term $H(Y)$ is a constant and has no effect when optimize over mask. We can now further construct our optimization objective as

$$\max_{\lambda} OE(Z,Y) \tag{21}$$

To get the final result, we first expand $p(y, z)$ as $\int p(x, y, z)dx$, and reformulate $p(x, y, z)$ based on Eq. (11), then we construct an empirical data distribution

$$p(x,y) = \frac{1}{N} \sum_{n=1}^{N} \sigma_{x_n}(x) \sigma_{y_n}(y) \tag{22}$$

Lastly, we use reparameterization trick since $Z = f(x, \epsilon)$, and x is not a random variable.

$$p(z|x)dz = p(\epsilon)d\epsilon \tag{23}$$

Combine all tricks together:

$$\begin{aligned} OE(Z,Y) &= \int p(y,z)\log q_\theta(y|z)dydz \\ &= \int p(y|x)p(z|x)p(x)\log q_\theta(y|z)dxdydz \\ &= \frac{1}{N}\sum_{n=1}^{N}\int p(z|x_n)\log q_\theta(y_n|z)dz \\ &= \frac{1}{N}\sum_{n=1}^{N}\int p(\epsilon)\log q_\theta(y_n|z)d\epsilon \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\epsilon\sim p(\epsilon)}[\log q_\theta(y_n|z)] \\ &= \mathbb{E}_{\epsilon\sim p(\epsilon)}[\log q_\theta(y_{sample}|z)] \end{aligned} \tag{24}$$

Summation is removed in the last derivation step, because we assume only one attribution sample. We can write the optimization problem as minimizing a loss function, then the loss function is [40]:

$$L = \mathbb{E}_{\epsilon\sim p(\epsilon)}[-\log q_\theta(y_{sample}|z)] \tag{25}$$

L is the cross entropy loss function for a single sample x_{sample}.