Improving Model Inversion Using Foreign Embeddings

Christopher Roßbach
FAU-Erlangen-Nürnberg

September 16, 2025

Abstract

Model inversion seeks to reconstruct the original input of a neural network from its outputs or intermediate representations. This process is highly ill-posed, especially when reconstructing images from deep feature embeddings, and thus requires strong priors and constraints to yield plausible results. In this work, we investigate optimization-based methods for reconstructing images from their embeddings in various vision model latent spaces. Building on empirical observations, we propose a novel approach that leverages the embeddings of multiple reconstruction candidates across different backbone models to improve reconstruction quality. Our results motivate further work on how incorporating information obtained from foreign backbones can enhance reconstruction fidelity.

1 Introduction

The reconstruction of network inputs from intermediate representations produced by a model can be used to evaluate the privacy risks of sharing such representations. Due to the ill-posed nature of the problem, one may expect that the reconstruction of the input is not possible in practice and thus share such representations without caution. As the private input may contain sensitive or private information, the ability to reconstruct the input from such representations poses a significant privacy risk. Thus, reconstructing the input from intermediate representations can be considered an attack on the privacy of the victim owning the input data. This report investigates techniques that attempt to reconstruct the full model input from a latent representation. We limit ourselves to the task of image reconstruction from feature embeddings given by a pretrained vision backbone. We assume a white-box setting where the attacker has access to the model architecture and parameters as well as the feature embedding of the target image.

We aim to approximate the private image x^* owned by the victim from its embedding $y^* = f(x^*)$. We assume that both the network f and the embedding y^* are known to the attacker (either leaked or purposely shared).

To leverage the information contained in the model f, we employ an optimization-based approach that iteratively refines a reconstruction proposal $\hat{x}$ to minimize the distance between its embedding $f(\hat{x})$ and the target y^*. To address the inherent ambiguity of the inversion process, we propose incorporating additional information obtained from foreign backbones h. Empirically, we observe that embeddings of different reconstruction candidates under a foreign backbone tend to scatter around the true embedding $h(x^*)$. By leveraging this property, we intend to improve reconstruction quality by encouraging the reconstructions to have more similar embeddings under h. Our findings show clear trends in different dimensions of parametrization of the reconstruction process, including the choice of foreign backbone, the method of aggregating embeddings, the number of parallel reconstructions used, and the effect of the data distribution.

2 Methodology

We attempt to reconstruct the private image from its embedding through an optimization process. The problem of reconstructing the private image from its embedding is strongly ill-posed, which forces us to incorporate additional constraints and priors into the optimization process. The optimization problem can be formulated as

$$\min_{\hat{x}} d_f(f(\hat{x}), y^*) + \mathcal{R}(\hat{x}),$$

where $\mathcal{R}$ is a regularization term that encourages the reconstruction to be smooth and natural and d_f is a distance function that is dependent on the used backbone f. For a ResNet backbone we use the mean squared error (MSE) as the distance function while using a cosine based distance for a CLIP encoder.

2.1 The Iterative Base Method

We employ an iterative optimization method to progressively refine the reconstruction. We generally optimize in the image space while resorting to a lower dimensional image space at the beginning of the reconstruction. Successive upscaling encourages the reconstruction of coarse structures before refining details. We take the resolution steps used in [3] and find adding another low resolution step of 32 to be beneficial, resulting in a scale sequence of $n \in \{32, 64, 128, 224\}$ where upscaling is performed at steps 400, 900 and 1800.

At each iteration, we update the reconstruction x in the possibly smaller space $\mathbb{R}^{3 \times n \times n}$ from which we retrieve the full size reconstruction proposal $\mathbb{R}^{3 \times N \times N} \ni \hat{x} = S_N(x)$ by using a differentiable scaling function $S_k : \mathbb{R}^{3 \times n \times n} \to \mathbb{R}^{3 \times k \times k}$. The next reconstruction x is determined by

$$x^{(t)} = S_{n_t}\left(x^{(t-1)} - \eta \frac{\nabla \mathcal{L}(x^{(t-1)}, y^*)}{\|\nabla \mathcal{L}(x^{(t-1)}, y^*)\|_2}\right), \quad (1)$$

where t denotes the iteration step, n_t the resolution at step t, η is the learning rate, and $\mathcal{L}$ the loss function. This process is repeated for 3000 iterations after which no notable improvement was noticed in the experiments. Because the result depends on the initialization $x^{(0)} \sim \mathcal{N}(0, 1)^{3 \times n_0 \times n_0}$, we perform multiple reconstructions in parallel. We denote such a set of equally constructed reconstructions as $X^{(t)} = \{x_0^{(t)}, x_1^{(t)}, \dots, x_{m-1}^{(t)}\}$ where m is the number of parallel reconstructions.

For the baseline method, we define the loss function $\mathcal{L}$ as

$$\mathcal{L}(x, y^*) = \mathcal{L}_f(x, y^*) + \mathcal{R}(S_N(x)), \quad (2)$$

with

$$\mathcal{L}_f(x, y^*) = d_f(f(A(S_N(x))), y^*), \quad (3)$$

where A is a randomized augmentation function as described in Section 2.2, and d_f is a distance function that is dependent on the used backbone f and $\mathcal{R}$ is a regularization term as described in Section 2.3.

2.2 Augmentations

As proposed in [1] we employ a set of differentiable augmentations to improve the robustness of the reconstruction. We expect the embedding to be stable under (slight) scaling, rotation and translation. We incorporate this assumption into the optimization process as a prior by applying a (different) set of augmentations at each step of the optimization. The transformation is thereby chosen randomly $(\text{scale}, \text{rotate}, \text{translate}_h, \text{translate}_v) \sim \mathcal{U}([0.7, 1.5] \times [-30, 30] \times [-0.1, 0.1] \times [-0.1, 0.1])$.

2.3 Regularization

For regularization, we use a term based on total variation as proposed in [4], with the adjustment of using a L_1 distance:

$$\mathcal{R}(x) = g_{\alpha,\beta}\left(\frac{\mathcal{R}_{TV}(x)}{(n-1)^2}\right)$$

with

$$\mathcal{R}_{TV}(x) = \sum_{i,j} |x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|.$$

This term is used to penalize sharp edges in the reconstructed images. Here n refers to the width of the square image, and $g_{\alpha,\beta}$ is a penalty function that balances the contribution of the total variation term and is defined as:

$$g_{\alpha,\beta}(x) = \text{relu}(x - \alpha) + \beta \cdot \text{relu}(x - \alpha)^2.$$

This quadratic dead zone penalty function allows for punishment-free variation in the interval $[-\alpha, \alpha]$ and (for large enough β) limits the total variation to be close to α. The use of this penalty function ensures that this regularization term does not lose importance when adding additional terms to the objective function of the optimization. A parameter selection of $\alpha = 0.3$ and $\beta = 10$ was found to yield good results in the experiments.

2.4 Impact of Inverted Model

The reconstruction method described in Section 2.1 yields visibly different results when using different backbones f. It is important to note that reconstruction for different backbones does not strictly yield better or worse results, but rather reconstructions with qualitatively different details. This effect can be seen in Figure 1.

For the picture of the safe, we note that the reconstruction on a ResNet backbone gives a picture containing structural properties of a safe, while the reconstruction based on the CLIP embedding seems to reproduce the text contained in the lower right corner of the original picture. For the dog picture, the reconstruction based on the CLIP embedding yields a better representation of the dog's features compared to the ResNet-based reconstruction, while the ResNet reconstruction contains patterns of the background. In the reconstruction of the fish picture, the ResNet-based reconstruction focuses on the fish's shape and the hands, while the CLIP-based reconstruction reconstructs the face of the person holding the fish.

From this observation, we conclude that an improvement in reconstruction quality may be achieved

by combining different backbones in the reconstruction process while maintaining the assumption of access to a single embedding.
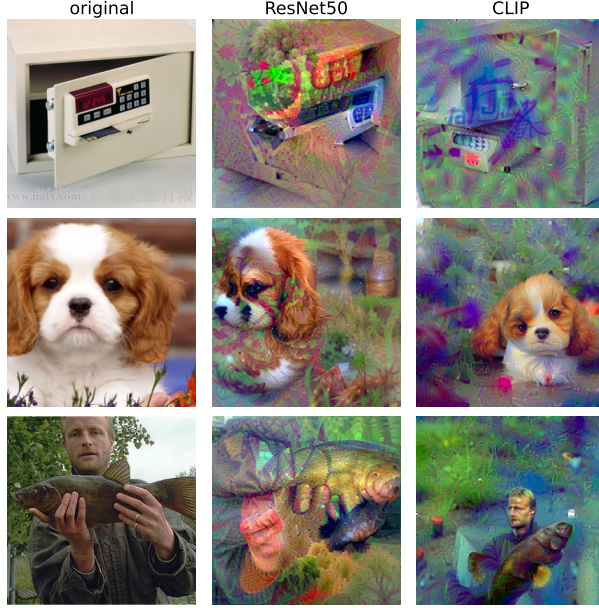

Figure 1: Comparison of reconstructions on ResNet and CLIP backbones.

2.5 Average of Foreign Embeddings

Another important observation is that if we have two different backbones f and h, a private image x^*, and a slightly modified version x_p^* of the private image with $d_f(f(x^*), f(x_p^*)) \approx 0$, then $d_h(h(x^*), h(x_p^*)) \approx 0$. That is, if we modify the private image only slightly, the embeddings produced by different backbones remain similar. However, this behavior does not apply to reconstructions: for a reconstruction $\hat{x}_f$ obtained by the procedure described in 2.1 with respect to the backbone f that satisfies $d_f(f(\hat{x}_f), f(x^*)) \approx 0$, we observe strong differences in the embedding under h, $d_h(h(\hat{x}_f), h(x^*)) \gg d_f(f(\hat{x}_f), f(x^*)) \approx 0$. The same is true for two different reconstructions $\hat{x}_{f,0}$ and $\hat{x}_{f,1}$: while $d_f(f(\hat{x}_{f,0}), f(\hat{x}_{f,1})) \approx 0$, we observe $d_h(h(\hat{x}_{f,0}), h(\hat{x}_{f,1})) \gg 0$.

Furthermore, if we have a set of reconstructions $X_f = \{\hat{x}_{f,0}, \hat{x}_{f,1}, \dots\}$, we observe that the average embedding under h of the reconstructions X_f is closer to the embedding under h of the private image than the average distance of the reconstructions:

$$d_h\left(\overline{h(X_f)}, h(x^*)\right) < \overline{d_h(h(X_f), h(x^*))}.$$

We often observe the even stronger statement, that the averaged embedding is closer to the embedding of the private image than every single reconstruction:

$$\forall \hat{x}_f \in X_f : d_h\left(\overline{h(X_f)}, h(x^*)\right) < d_h(h(\hat{x}_f), h(x^*)).$$

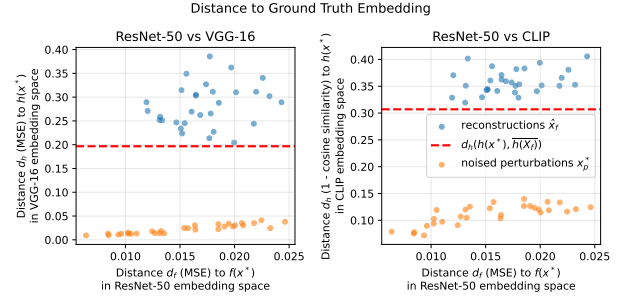This effect can be seen in Figure 2.

Figure 2: Comparison of embedding distances for noised images, reconstructions and averaged reconstruction embeddings for f being a ResNet backbone, and h being a VGG backbone (left) and h being a CLIP backbone (right).

That leads to the intuitive assumption the that embeddings under a foreign backbone h scatter around the embedding of the private image $h(x^*)$ and do not deviate in a specific direction. This idea is illustrated in figure 3.
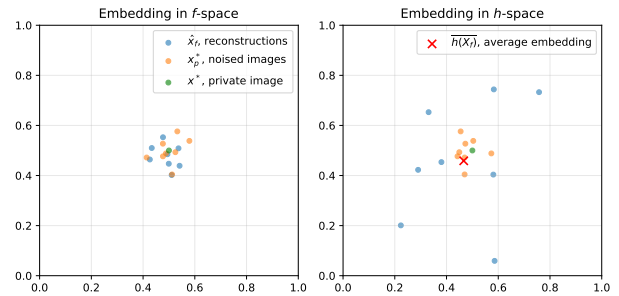
Figure 3: Illustration of the imagined scattering behavior of embeddings under a backbone used for reconstruction f (left) and a foreign backbone h (right). The average embedding in the foreign space is closer to the embedding of the private image than the embeddings of the reconstructions.

2.6 Combining Multiple Backbones

To potentially make use of this we explored multiple methods to incorporate the average in the foreign embedding space during the reconstruction process.

Since most of the used networks apply a ReLU activation as their last non-linearity, simply calculating

3

the arithmetic average in each coordinate leads to a large shift in the shape of the distribution of embeddings. To reduce that shift, we can remove the last ReLU activation from the network. The effect of this can be seen in Figure 4. We perform the experiments on both the modified and the original backbones.
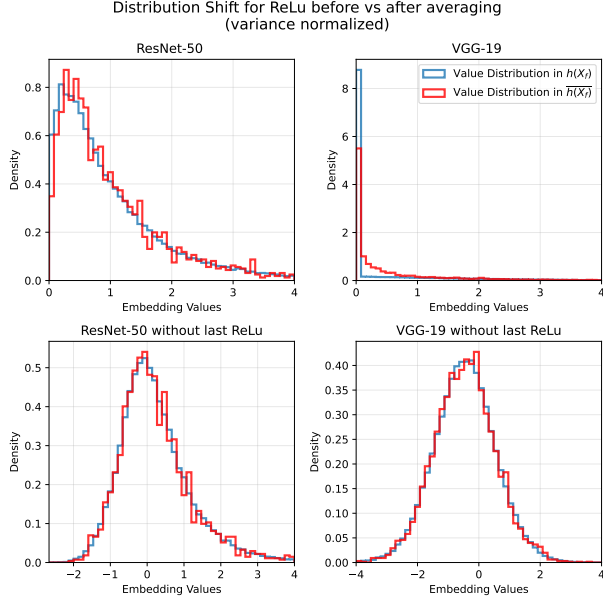

Figure 4: Distribution of embeddings in different backbones. Top: Value distribution under original backbone. Bottom: Value distribution under modified backbone with the last ReLu removed.

We incorporate the average embedding in a foreign space by extending the loss function from equation 2:

$$\mathcal{L}(x, y^*, y_h) = \mathcal{L}_f(x, y^*) + \mathcal{L}_h(x, y_h) + \mathcal{R}(S_N(x)) \quad (4)$$

where $\mathcal{L}_h$ is defined as in equation 3 and y_h is a element in the foreign space. We will choose y_h to either be $\overline{h(X^{(t)})}$, the average embedding of the current reconstructions in the foreign space, or $\overline{h(X_f)}$, the average embedding of reconstructions of previously completed reconstruction run with the loss function from equation 2.

3 Experiments

We conducted a series of experiments to evaluate the effectiveness of the proposed method. We experimented with different choices of the foreign backbone h and the construction of y_h.

For the presented experiments, we fixed f to be a ResNet-50, pretrained on ImageNet without its classification layer. For h, we explored different

ResNet backbones: ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152 (referred to as rn18, rn34, rn50, rn101, rn152), VGG-16 and VGG-19 (referred to as vgg16 and vgg19), and a CLIP ViT-L/14 (referred to as clip). For all of them (except clip), we used a model pretrained on ImageNet without its classification layer and also performed the experiments on the original and the version without the last ReLU activation. For the CLIP model, we used an OpenAI pretrained version as presented in [5]. As metrics for the evaluation of the reconstruction quality, we used the structural similarity index measure (SSIM) [9], the learned perceptual image patch similarity (LPIPS) [8], as well as the cosine similarity to the ground truth CLIP embedding of the image ($\mathcal{L}_{\text{clip}}(\hat{x}, \text{clip}(x))$). The SSIM value is supplemented by LPIPS, which is used to measure perceptual similarity as a human viewer would perceive it, and by the CLIP loss, which measures how well the reconstruction captures the semantic content of the original image.

The calculation of y_h was performed by using $m = 4$ parallel reconstructions and in the two different ways described in Section 2.6: We choose y_h to be the average of the current iteration ($y_h^{\text{avg}} = \overline{h(X^{(t)})}$) or to be a reference vector obtained by independently and previously completed run ($y_h^{\text{ref}} = \overline{h(X_f)}$).

All reported numbers in the tables are the mean over 8 images randomly chosen from the ImageNet validation set except for Table 7 where we used 8 images randomly chosen from the FFHQ dataset [2].

3.1 Influence of the Last ReLU

The effect of removing the last ReLU activation can be seen in Table 1. Despite the rather promising observations in Figure 4, we observe that removing the last ReLU activation generally hurts the reconstruction quality, independent of the metric used.

The reason for this could be that removing the last ReLU activation increases the search space of the optimization problem significantly. This could cause the optimization to get stuck in local minima more often. Another reason could be that the last ReLU activation is an important part of the backbone's learned representation, and removing it degrades the quality of the features extracted by the backbone. This is because during training of the backbone, the exact value of below-zero activations was not relevant and thus not learned, but now they contribute directly to the loss.

3.2 Choice of Foreign Backbone

The choice of the foreign backbone h has a measurable impact on the reconstruction quality. We are especially interested in the effect of the depth of the network on the reconstruction quality. Within the family of ResNets we observe that increasing the depth of the network generally improves the reconstruction quality as can be seen in table 2. This may be due to the fact that deeper networks are able to capture more complex features and thus provide a more informative guidance during the reconstruction process. The same effect can be observed in the family of VGG networks as can be seen in table 3.

3.3 Method of Calculating the Reference Embedding

Whether using the average of the current reconstructions or a reference vector obtained from a previous run has a significant impact on the reconstruction quality, as can be seen in Table 4. For the perceptual LPIPS metric and the CLIP-based metric, we observe a significant improvement over the avg-method when using the ref-method. This may be because in the ref-method, the reference vector can be freely developed in an independent run and thus may contain more complex features, while the avg-method pulls the reconstructions towards an average from the beginning of the optimization, which may discourage exploration of more diverse solutions.

On the other hand, the SSIM metric seems to benefit from the avg-method. This may be due to the same reason: the avg-method discourages complex features and thus reduces the development of strong structural artifacts that may be penalized by the SSIM metric.

For the best performing backbones (rn152, vgg19, clip (each without modification)) and the ref-method we performed additional experiments to evaluate the effect of the number of parallel reconstructions used to calculate y_h. Going from 4 to 8 parallel reconstructions yields an improvement for the VGG backbone while the ResNet performance degrades as can be seen in Table 5. Doubling the parallel reconstructions again from 8 to 16 continues this trend only partially as can be seen in Table 6.

3.4 Effect of the Data Distribution

Incorporating foreign backbones h may be especially beneficial if the private image x^* stems from a different distribution than the data used to train the backbone f. To evaluate the effect of incorporating foreign embeddings on the reconstruction quality when reconstructing images from a different distribution than f was trained on, we performed experiments on images from the FFHQ dataset. In Table 7, we display how much the reconstruction quality improves for the FFHQ dataset when using different foreign backbones. In brackets, we display the difference to the corresponding improvement when reconstructing images from the ImageNet validation set.

Especially for reconstructions using a backbone h that has a different architecture than f (e.g., VGG or CLIP), we observe an improvement in reconstruction quality when reconstructing images from the FFHQ dataset compared to the ImageNet validation set. This may be some form of generalization effect, as the features learned by different architectures may be more general and thus provide more informative guidance during the reconstruction process, especially for the CLIP model where the training data is more diverse.

4 Discussion

4.1 Results

Table 8 shows the results from the best parametrizations for each architecture. That means we chose the deepest version for each network, the ReLU was not removed from the backbones, and we used the reference method to calculate y_h. The results do not show a clear improvement over the base method, but there are some improvements in certain metrics for certain backbones. The fact that using more parallel reconstructions shows only slight improvements or even degrades the performance indicates that the method of simply averaging the embeddings in the foreign space is not optimal. Also, the observed behavior that removing the last ReLU activation generally degrades performance, despite the observed distribution shift in Section 2.6, supports this and motivates further research into better aggregation methods.

Table 7 shows that using foreign backbones seems to be especially beneficial when reconstructing images from a different distribution than the one used to train the backbone f, even when h is not trained on the distribution of x^*. Using a backbone trained on a similar distribution as the one x^* was obtained from may further increase this effect.

4.2 Possible Extensions

This idea can be extended in several directions without diverging from the main concept. Possible extensions regarding the choice of h include:

1. The identity mapping $h(x) = x$ would correspond to averaging the reconstructions in pixel space. That could encourage the reconstruction mechanism to maintain common features in pixel space that were already discovered.

2. Choosing h as an autoencoder trained on a large corpus of images could encourage the reconstructions to maintain features that are common in natural images.

3. If the attacker has a suspicion about the distribution of the private image, h could be chosen as a backbone trained on a similar distribution.

4. If the attacker is only interested in the reconstruction of certain parts of the image, h could be chosen as a backbone trained for that specific task (e.g., face recognition).

5. In the same manner, the backbone f could be appended by a task-specific head c to encourage the reconstruction of certain features, and h can be chosen to be $c \circ f$.

Possible extensions regarding how y_h is calculated include:

1. Instead of using the average embedding in the foreign space, one could use a more robust statistic like the median or a trimmed mean.

2. One could also use a clustering algorithm to identify groups of similar reconstructions and use the centroid of the largest cluster as y_h.

3. For location-sensitive embeddings h (e.g., $h =$ id), using an image registration method like RANSAC-Flow [6] before averaging could be beneficial, as this method has already been shown to be effective for other reconstruction scenarios, as in [7].

Extensions regarding adjustments to the loss and distance functions include:

1. Another extension would be to adjust the distance function d_h to punish certain components differently depending on the agreement of the reconstructions in the foreign space in those components.

2. Combining multiple losses $\mathcal{L}_{h_1}, \mathcal{L}_{h_2}, \dots$ for different backbones $h_1, h_2, \dots$ is a simple but not yet explored extension.

References

[1] Amin Ghiasi, Hamid Kazemi, Steven Reich, Chen Zhu, Micah Goldblum, and Tom Goldstein. Plug-In Inversion: Model-Agnostic Inversion for Vision with Data Augmentations.

[2] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405. IEEE, 2019.

[3] Hamid Kazemi, Atoosa Chegini, Jonas Geiping, Soheil Feizi, and Tom Goldstein. What do we learn from inverting CLIP models?

[4] Aravindh Mahendran and Andrea Vedaldi. Understanding Deep Image Representations by Inverting Them. pages 5188–5196.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR.

[6] Xi Shen, François Darmon, Alexei A. Efros, and Mathieu Aubry. RANSAC-Flow: Generic Two-Stage Image Alignment. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 618–637. Springer International Publishing.

[7] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through Gradients: Image Batch Recovery via GradInversion. pages 16332–16341. IEEE.

[8] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. pages 586–595.

[9] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. 13(4):600–612.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
avg	yes	rn18	0.176 (+0.000)	0.793 (+0.000)	0.796 (+0.000)	0.600 (+0.000)	4
avg	no	rn18	0.176 (−0.001)	0.817 (+0.024)	0.807 (+0.012)	0.595 (−0.004)	4
ref	yes	rn18	0.170 (+0.000)	0.778 (+0.000)	0.786 (+0.000)	0.621 (+0.000)	4
ref	no	rn18	0.168 (−0.002)	0.795 (+0.017)	0.793 (+0.007)	0.603 (−0.018)	4
avg	yes	rn34	0.175 (+0.000)	0.792 (+0.000)	0.794 (+0.000)	0.585 (+0.000)	4
avg	no	rn34	0.173 (−0.002)	0.800 (+0.008)	0.795 (+0.000)	0.576 (−0.009)	4
ref	yes	rn34	0.166 (+0.000)	0.776 (+0.000)	0.791 (+0.000)	0.623 (+0.000)	4
ref	no	rn34	0.166 (−0.000)	0.783 (+0.007)	0.793 (+0.002)	0.606 (−0.017)	4
avg	yes	rn50	0.178 (+0.000)	0.761 (+0.000)	0.787 (+0.000)	0.636 (+0.000)	4
avg	no	rn50	0.178 (+0.000)	0.767 (+0.006)	0.789 (+0.002)	0.628 (−0.007)	4
ref	yes	rn50	0.175 (+0.000)	0.755 (+0.000)	0.779 (+0.000)	0.636 (+0.000)	4
ref	no	rn50	0.175 (+0.001)	0.757 (+0.002)	0.782 (+0.003)	0.640 (+0.004)	4
avg	yes	rn101	0.178 (+0.000)	0.761 (+0.000)	0.778 (+0.000)	0.631 (+0.000)	4
avg	no	rn101	0.178 (−0.000)	0.767 (+0.007)	0.780 (+0.002)	0.618 (−0.013)	4
ref	yes	rn101	0.177 (+0.000)	0.757 (+0.000)	0.776 (+0.000)	0.648 (+0.000)	4
ref	no	rn101	0.174 (−0.004)	0.759 (+0.002)	0.781 (+0.005)	0.636 (−0.013)	4
avg	yes	rn152	0.178 (+0.000)	0.760 (+0.000)	0.777 (+0.000)	0.627 (+0.000)	4
avg	no	rn152	0.176 (−0.001)	0.766 (+0.006)	0.781 (+0.004)	0.619 (−0.008)	4
ref	yes	rn152	0.173 (+0.000)	0.755 (+0.000)	0.775 (+0.000)	0.645 (+0.000)	4
ref	no	rn152	0.177 (+0.003)	0.756 (+0.001)	0.776 (+0.001)	0.643 (−0.002)	4
avg	yes	vgg16	0.176 (+0.000)	0.779 (+0.000)	0.787 (+0.000)	0.616 (+0.000)	4
avg	no	vgg16	0.184 (+0.008)	0.864 (+0.086)	0.847 (+0.060)	0.565 (−0.051)	4
ref	yes	vgg16	0.182 (+0.000)	0.775 (+0.000)	0.788 (+0.000)	0.621 (+0.000)	4
ref	no	vgg16	0.173 (−0.008)	0.803 (+0.029)	0.813 (+0.024)	0.570 (−0.051)	4
avg	yes	vgg19	0.178 (+0.000)	0.767 (+0.000)	0.785 (+0.000)	0.616 (+0.000)	4
avg	no	vgg19	0.175 (−0.003)	0.841 (+0.073)	0.854 (+0.069)	0.563 (−0.052)	4
ref	yes	vgg19	0.181 (+0.000)	0.775 (+0.000)	0.785 (+0.000)	0.634 (+0.000)	4
ref	no	vgg19	0.175 (−0.006)	0.812 (+0.037)	0.810 (+0.026)	0.593 (−0.042)	4

Table 1: Comparison of reconstruction quality with and without the last ReLU activation. The differences, shown in brackets, are calculated with respect to the same backbone and y_h calculation method so that only the effect of the removal of the ReLU is shown. Reference lines are indicated by a gray background.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
avg	yes	rn18	0.176 (−0.002)	0.793 (+0.032)	0.796 (+0.009)	0.600 (−0.036)	4
avg	no	rn18	0.176 (−0.002)	0.817 (+0.050)	0.807 (+0.018)	0.595 (−0.033)	4
ref	yes	rn18	0.170 (−0.005)	0.778 (+0.023)	0.786 (+0.008)	0.621 (−0.014)	4
ref	no	rn18	0.168 (−0.008)	0.795 (+0.038)	0.793 (+0.011)	0.603 (−0.037)	4
avg	yes	rn34	0.175 (−0.003)	0.792 (+0.031)	0.794 (+0.008)	0.585 (−0.051)	4
avg	no	rn34	0.173 (−0.005)	0.800 (+0.033)	0.795 (+0.006)	0.576 (−0.052)	4
ref	yes	rn34	0.166 (−0.009)	0.776 (+0.021)	0.791 (+0.013)	0.623 (−0.012)	4
ref	no	rn34	0.166 (−0.009)	0.783 (+0.026)	0.793 (+0.012)	0.606 (−0.034)	4
avg	yes	rn50	0.178 (+0.000)	0.761 (+0.000)	0.787 (+0.000)	0.636 (+0.000)	4
avg	no	rn50	0.178 (+0.000)	0.767 (+0.000)	0.789 (+0.000)	0.628 (+0.000)	4
ref	yes	rn50	0.175 (+0.000)	0.755 (+0.000)	0.779 (+0.000)	0.636 (+0.000)	4
ref	no	rn50	0.175 (+0.000)	0.757 (+0.000)	0.782 (+0.000)	0.640 (+0.000)	4
avg	yes	rn101	0.178 (+0.000)	0.761 (−0.000)	0.778 (−0.009)	0.631 (−0.005)	4
avg	no	rn101	0.178 (−0.000)	0.767 (+0.000)	0.780 (−0.009)	0.618 (−0.010)	4
ref	yes	rn101	0.177 (+0.003)	0.757 (+0.002)	0.776 (−0.002)	0.648 (+0.013)	4
ref	no	rn101	0.174 (−0.002)	0.759 (+0.002)	0.781 (−0.000)	0.636 (−0.004)	4
avg	yes	rn152	0.178 (−0.000)	0.760 (−0.001)	0.777 (−0.009)	0.627 (−0.009)	4
avg	no	rn152	0.176 (−0.002)	0.766 (−0.001)	0.781 (−0.008)	0.619 (−0.009)	4
ref	yes	rn152	0.173 (−0.001)	0.755 (−0.000)	0.775 (−0.004)	0.645 (+0.010)	4
ref	no	rn152	0.177 (+0.001)	0.756 (−0.001)	0.776 (−0.006)	0.643 (+0.004)	4

Table 2: Comparison of different depths of ResNets. ResNet-50 is chosen as the reference value for the differences. Reference lines are indicated by a gray background.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
avg	yes	vgg16	0.176 (+0.000)	0.779 (+0.000)	0.787 (+0.000)	0.616 (+0.000)	4
avg	no	vgg16	0.184 (+0.000)	0.864 (+0.000)	0.847 (+0.000)	0.565 (+0.000)	4
ref	yes	vgg16	0.182 (+0.000)	0.775 (+0.000)	0.788 (+0.000)	0.621 (+0.000)	4
ref	no	vgg16	0.173 (+0.000)	0.803 (+0.000)	0.813 (+0.000)	0.570 (+0.000)	4
avg	yes	vgg19	0.178 (+0.001)	0.767 (−0.011)	0.785 (−0.003)	0.616 (+0.000)	4
avg	no	vgg19	0.175 (−0.010)	0.841 (−0.024)	0.854 (+0.007)	0.563 (−0.001)	4
ref	yes	vgg19	0.181 (−0.001)	0.775 (−0.000)	0.785 (−0.004)	0.634 (+0.014)	4
ref	no	vgg19	0.175 (+0.001)	0.812 (+0.009)	0.810 (−0.002)	0.593 (+0.023)	4

Table 3: Comparison of different depths of VGGs. VGG-16 is chosen as the reference value for the differences. Reference lines are indicated by a gray background.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
avg	yes	rn18	0.176 (+0.000)	0.793 (+0.000)	0.796 (+0.000)	0.600 (+0.000)	4
avg	no	rn18	0.176 (+0.000)	0.817 (+0.000)	0.807 (+0.000)	0.595 (+0.000)	4
ref	yes	rn18	0.170 (−0.006)	0.778 (−0.015)	0.786 (−0.009)	0.621 (+0.021)	4
ref	no	rn18	0.168 (−0.008)	0.795 (−0.022)	0.793 (−0.014)	0.603 (+0.007)	4
avg	yes	rn34	0.175 (+0.000)	0.792 (+0.000)	0.794 (+0.000)	0.585 (+0.000)	4
avg	no	rn34	0.173 (+0.000)	0.800 (+0.000)	0.795 (+0.000)	0.576 (+0.000)	4
ref	yes	rn34	0.166 (−0.009)	0.776 (−0.016)	0.791 (−0.003)	0.623 (+0.038)	4
ref	no	rn34	0.166 (−0.007)	0.783 (−0.017)	0.793 (−0.001)	0.606 (+0.030)	4
avg	yes	rn50	0.178 (+0.000)	0.761 (+0.000)	0.787 (+0.000)	0.636 (+0.000)	4
avg	no	rn50	0.178 (+0.000)	0.767 (+0.000)	0.789 (+0.000)	0.628 (+0.000)	4
ref	yes	rn50	0.175 (−0.003)	0.755 (−0.006)	0.779 (−0.008)	0.636 (−0.000)	4
ref	no	rn50	0.175 (−0.003)	0.757 (−0.010)	0.782 (−0.007)	0.640 (+0.011)	4
avg	yes	rn101	0.178 (+0.000)	0.761 (+0.000)	0.778 (+0.000)	0.631 (+0.000)	4
avg	no	rn101	0.178 (+0.000)	0.767 (+0.000)	0.780 (+0.000)	0.618 (+0.000)	4
ref	yes	rn101	0.177 (−0.001)	0.757 (−0.003)	0.776 (−0.002)	0.648 (+0.017)	4
ref	no	rn101	0.174 (−0.004)	0.759 (−0.008)	0.781 (+0.001)	0.636 (+0.017)	4
avg	yes	rn152	0.178 (+0.000)	0.760 (+0.000)	0.777 (+0.000)	0.627 (+0.000)	4
avg	no	rn152	0.176 (+0.000)	0.766 (+0.000)	0.781 (+0.000)	0.619 (+0.000)	4
ref	yes	rn152	0.173 (−0.004)	0.755 (−0.005)	0.775 (−0.003)	0.645 (+0.018)	4
ref	no	rn152	0.177 (+0.000)	0.756 (−0.010)	0.776 (−0.005)	0.643 (+0.024)	4
avg	yes	vgg16	0.176 (+0.000)	0.779 (+0.000)	0.787 (+0.000)	0.616 (+0.000)	4
avg	no	vgg16	0.184 (+0.000)	0.864 (+0.000)	0.847 (+0.000)	0.565 (+0.000)	4
ref	yes	vgg16	0.182 (+0.005)	0.775 (−0.004)	0.788 (+0.001)	0.621 (+0.005)	4
ref	no	vgg16	0.173 (−0.011)	0.803 (−0.061)	0.813 (−0.034)	0.570 (+0.005)	4
avg	yes	vgg19	0.178 (+0.000)	0.767 (+0.000)	0.785 (+0.000)	0.616 (+0.000)	4
avg	no	vgg19	0.175 (+0.000)	0.841 (+0.000)	0.854 (+0.000)	0.563 (+0.000)	4
ref	yes	vgg19	0.181 (+0.003)	0.775 (+0.007)	0.785 (−0.000)	0.634 (+0.019)	4
ref	no	vgg19	0.175 (−0.000)	0.812 (−0.029)	0.810 (−0.044)	0.593 (+0.029)	4
avg	no	clip	0.170 (+0.000)	0.764 (+0.000)	0.796 (+0.000)	0.581 (+0.000)	4
ref	no	clip	0.163 (−0.007)	0.768 (+0.004)	0.788 (−0.008)	0.648 (+0.067)	4

Table 4: Comparison of different methods of calculating y_h (avg vs ref). The differences are calculated with respect to the same backbone and relu configuration so that only the effect of the method of calculating y_h is shown. Reference lines are indicated by a gray background.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
ref	yes	rn152	0.173 (−0.000)	0.754 (−0.001)	0.780 (+0.005)	0.643 (−0.003)	8
ref	yes	vgg19	0.176 (−0.005)	0.770 (−0.005)	0.785 (+0.001)	0.631 (−0.003)	8
ref	no	clip	0.164 (+0.001)	0.766 (−0.002)	0.782 (−0.005)	0.655 (+0.007)	8

Table 5: Metrics for 8 parallel reconstructions to calculate y_h. The differences are calculated with respect to using 4 parallel reconstructions.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
ref	yes	rn152	0.171 (−0.002)	0.756 (+0.002)	0.781 (+0.001)	0.641 (−0.002)	16
ref	yes	vgg19	0.177 (+0.001)	0.770 (+0.001)	0.782 (−0.004)	0.633 (+0.001)	16
ref	no	clip	0.162 (−0.002)	0.768 (+0.002)	0.786 (+0.004)	0.662 (+0.007)	16

Table 6: Metrics for 16 parallel reconstructions to calculate y_h. The differences are calculated with respect to using 8 parallel reconstructions.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
ref	yes	rn152	−0.001 (+0.002)	−0.005 (−0.009)	−0.009 (−0.006)	0.002 (−0.003)	4
ref	yes	vgg19	0.004 (+0.000)	0.017 (−0.007)	0.008 (+0.002)	0.001 (+0.007)	4
ref	no	clip	−0.014 (+0.000)	0.009 (−0.009)	0.001 (−0.009)	0.008 (+0.001)	4

Table 7: Comparison of reconstruction quality improvement when reconstructing images from the FFHQ dataset instead of the ImageNet validation set. The values indicate the difference in the corresponding metric compared to the baseline method on FFHQ data. The numbers in brackets indicate the difference to the corresponding improvement when reconstructing images from the ImageNet validation set.

Type	ReLU	h	SSIM ↑	LPIPS VGG ↓	LPIPS Alex ↓	CLIP cos sim ↑	m
baseline	yes	–	0.177 (+0.000)	0.751 (+0.000)	0.778 (+0.000)	0.641 (+0.000)	4
ref	yes	rn152	0.173 (−0.003)	0.755 (+0.004)	0.775 (−0.004)	0.645 (+0.005)	4
ref	yes	vgg19	0.181 (+0.004)	0.775 (+0.024)	0.785 (+0.006)	0.634 (−0.006)	4
ref	no	clip	0.163 (−0.014)	0.768 (+0.017)	0.788 (+0.010)	0.648 (+0.007)	4

Table 8: Comparison of reconstruction quality for the best parametrizations for each architecture. For each of the images 4 reconstructions were performed and their metrics were averaged.