A Analyzing and Visualizing the Results of the Reconstruction Optimization

The analysis of the results of the various reconstruction losses Eqs. (6), (12) and (15), involve verifying and checking which of the training samples were reconstructed. In this section we provide further details on our method for analyzing the reconstruction results, and how we measure the quality of our reconstructions.

A.1 Analyzing the Results of the Reconstruction Optimization

In order to match between samples from the training set and the outputs of the reconstruction algorithm (the so-called "candidates") we follow the same protocol of Haim et al. [2022]. Note that before training our models, we subtract the mean image from the given training set. Therefore the training samples are d-dimensional objects where each entry is in $[-1, 1]$.

First, for each training sample we compute the distance to all the candidates using a normalized L_2 score:

$$d(\mathbf{x}, \mathbf{y}) = \left\| \frac{\mathbf{x} - \mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}} - \frac{\mathbf{y} - \mu_{\mathbf{y}}}{\sigma_{\mathbf{y}}} \right\|_2^2 \tag{16}$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ are a training sample or an output candidate from the reconstruction algorithm, $\mu_{\mathbf{x}} = \frac{1}{d}\sum_{i=1}^{d} \mathbf{x}(i)$ is the mean of $\mathbf{x}$ and $\sigma_{\mathbf{x}} = \sqrt{\frac{1}{d-1}\sum_{i=1}^{d}(\mathbf{x}(i) - \mu_{\mathbf{x}})^2}$ is the standard deviation of $\mathbf{x}$ (and the same goes for $\mathbf{y}, \mu_{\mathbf{y}}, \sigma_{\mathbf{y}}$).

Second, for each training sample, we take C candidates with the smallest distance according to Eq. (16). C is determined by finding the first candidate whose distance is larger than B times the distance to the closest nearest neighbour (where B is a hyperparameter). Namely, for a training sample $\mathbf{x}$, the nearest neighbour is $\mathbf{y}_1$ with a distance $d(\mathbf{x}, \mathbf{y}_1)$, then C is determined by finding a candidate $\mathbf{y}_{C+1}$ whose distance is $d(\mathbf{x}, \mathbf{y}_{C+1}) > B \cdot d(\mathbf{x}, \mathbf{y}_1)$, and for all $j \leq C$, $d(\mathbf{x}, \mathbf{y}_j) \leq B \cdot d(\mathbf{x}, \mathbf{y}_1)$. B was chosen heuristically to be $B = 1.1$ for MLPs, and $B = 1.5$ for convolutional models. The C candidates are then summed to create the reconstructed sample $\hat{\mathbf{x}} = \frac{1}{C}\sum_{j=1}^{C} \mathbf{y}_j$. In general, we can also take only $C = 1$ candidate, namely just one nearest neighbour per training sample, but choosing more candidates improve the visual quality of the reconstructed samples.

Third, the reconstructed sample $\hat{\mathbf{x}}$ is scaled to an image in $[0, 1]$ by adding the training set mean and linearly "stretching" the minimal and maximal values of the result to $[0, 1]$. Finally, we compute the SSIM between the training sample $\mathbf{x}$ and the reconstructed sample $\hat{\mathbf{x}}$ to measure the quality of reconstruction.

A.2 Deciding whether a Reconstruction is “Good”

Here we justify our selection for SSIM=0.4 as the threshold for what we consider as a “good" reconstruction. In general, the problem of deciding whether a reconstruction is the correct match to a given sample, or whether a reconstruction is a “good" reconstruction is equivalent to the problem of comparing between images. No “synthetic" metric (like SSIM, $l2$ etc.) will be aligned with human perception. A common metric for this purpose is LPIPS Zhang et al. [2018] that uses a classifier trained on Imagenet Deng et al. [2009], but since CIFAR images are much smaller than Imagenet images (32×32 vs. 224×224) it is not clear that this metric will be better than SSIM.

As a simple rule of thumb, we use SSIM>0.4 for deciding that a given reconstruction is “good". To justify, we plot the best reconstructions (in terms of SSIM) in Fig. 8. Note that almost all samples with SSIM>0.4 are also visually similar (for a human). Also note that some of the samples with SSIM<0.4 are visually similar, so in this sense we are “missing" some good reconstructions. In general, determining whether a candidate output of a reconstruction algorithm is a match to a training sample is an open question and a problem in all other works for data reconstruction, see for example Carlini et al. [2023] that derived a heuristic for reconstructed samples from a generative model. This cannot be dealt in the scope of this paper, and is an interesting future direction for our work.

Figure 8: Justifying the threshold of SSIM$= 0.4$ as good rule-of-thumb for a threshold for a "good" reconstruction. The SSIM values are shown above each train-reconstruction pair. Note that samples with SSIM> 0.4 (blue) are visually similar. Also some of the samples with SSIM< 0.4 (red) are similar. In general deciding whether a reconstruction is "good" is an open question beyond the scope of this paper.

B Implementation Details

Further Training Details. The models that were reconstructed in the main part of the paper were trained with learning rates of 0.01 for binary classifiers (both MLP and convolutional), and 0.5 in the case of multi-class classifier (Section 4). The models were trained with full batch gradient descent for 10^6 epochs, to guarantee convergence to a KKT point of Eq. (1) or a local minima of Eq. (13). We note that Haim et al. [2022] observed that models trained with SGD can also be reconstructed. The experiment in Appendix G (large models with many samples) also uses SGD and results with similar conclusion, that some models trained with SGD can be reconstructed. In general, exploring reconstruction from models trained with SGD is an interesting direction for future works.

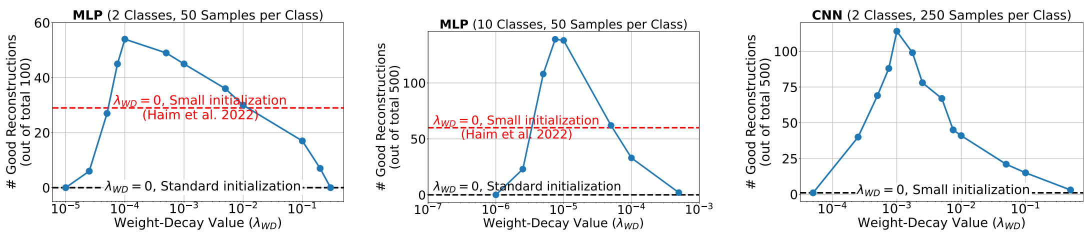
Runtime and Hardware. Runtime of a single reconstruction run (specific choice of hyperparameters) from a model D-1000-1000-1 takes about 20 minutes on a GPU Tesla V-100 32GB or NVIDIA Ampere Tesla A40 48GB.

Hyperparameters of the Reconstruction Algorithm. Note that the reconstruction loss contains the derivative of a model with ReLU layers, which is flat and not-continuous. Thus, taking the derivative of the reconstruction loss results in a zero function. To address this issue we follow a solution presented in Haim et al. [2022]. Namely, given a trained model, we replace in the backward phase of backpropogation the ReLU function with the derivative of a softplus function (or SmoothReLU) $f(x) = \alpha \log(1 + e^{-x})$, where α is a hyperparameter of the reconstruction scheme. The functionality of the model itself does not change, as in the foraward phase the function remains a ReLU. Only the backward function is replaced with a smoother version of the derivative of ReLU which is $f'(x) = \alpha\sigma(x) = \frac{\alpha}{1+e^{-x}}$ (here σ is the Sigmoid function). To find good reconstructions we run the algorithm multiple times (typically 100 times) with random search over the hyperparameters (using the Weights & Biases framework Biewald [2020]). The exact parameters for the hyperparameters search are:

- Learning rate: log-uniform in $[10^{-5}, 1]$
- σ_x: log-uniform in $[10^{-6}, 0.1]$
- $\lambda_{\min}$: uniform in $[0.01, 0.5]$
- α: uniform in $[10, 500]$

B.1 Small Initialisation

Models whose first layer was initialized with a small (non-standard) initialization plays several roles in our paper. These include models that were trained following the approach in Haim et al. [2022], as in Section 4, or comparison of such models to models trained with weight-decay, as discussed in Section 6.1. The models whose results appear in Fig. 2 and Fig. 5 were initialized with a scale of 10^{-3}. After submitting the paper we noticed that the initialization used in Haim et al. [2022] was in fact smaller - 10^{-4}. In order to make a fair comparison, we re-run the baselines shown in Fig. 5 (red-dashed lines). As seen in Fig. 9, the corrected initialisation increase the number of reconstructed samples in the case of the binary classifier (Fig. 9a) and decrease in the case of multiclass classifier (Fig. 9b). In both cases, this does not change the main claim in Section 6.1, that using weight decay terms during training changes the reconstructability and in some cases dramatically increase the number of samples that are vulnerable to reconstruction. We will make sure to update Fig. 5 with the corrected version.



(a) (b) (c)

Figure 9: Corrected version of Fig. 5. The correction of the baselines in red did not affect the claims in Section 6.1.

C Experiments with Different Number of Classes and Fixed Training Set Size

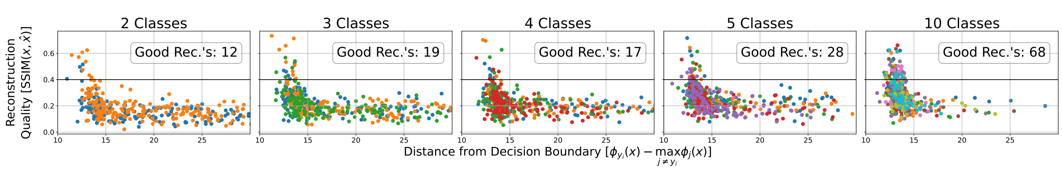


Figure 10: Experiments of reconstruction from models trained on a a fixed training set size (500 samples) for different number of classes. Number of "good" reconstruction is shown for each model.

To complete the experiment shown in Fig. 3, we also perform experiments on models trained on various number of classes ($C \in \{2, 3, 4, 5, 10\}$) and with a fixed training set size of 500 samples (distributed equally between classes), see Fig. 10. It can be seen that as the number of classes increases, also does the number of good reconstructions, where for 10 classes there are more than 6 times good reconstructions than for 2 classes. Also, the quality of the reconstructions improves as the number of classes increase, which is depicted by an overall higher SSIM score. We also note, that the number of good reconstructions in Fig. 10 is very similar to the number of good reconstructions from Fig. 3 for 50 samples per class. We hypothesize that although the number of training samples increases, the number of "support vectors" (i.e samples on the margin which can be reconstructed) that are required for successfully interpolating the entire dataset does not change by much.

D General Losses - More Results

Following the discussion in Section 5 and Fig. 4, Figures 11, 12, 13 present visualizations of training samples and their reconstructions from models trained with L_2, $L_{2.5}$ and Huber loss, respectively.

Figure 11: **Reconstruction using L_2 loss.** Training samples (red) and their best reconstructions (blue) using an MLP classifier that was trained on 300 CIFAR10 images using an L_2 regression loss, as described in Section 5 and Fig. 4.

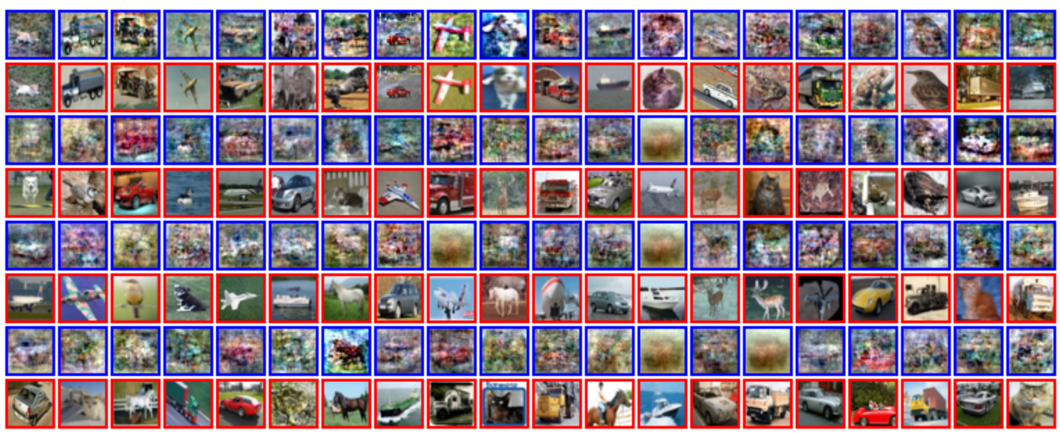


Figure 12: **Reconstruction using $L_{2.5}$ loss.** Training samples (red) and their best reconstructions (blue) using an MLP classifier that was trained on 300 CIFAR10 images using an $L_{2.5}$ regression loss, as described in Section 5 and Fig. 4.

Figure 13: **Reconstruction using Huber loss.** Training samples (red) and their best reconstructions (blue) using an MLP classifier that was trained on 300 CIFAR10 images using Huber loss, as described in Section 5 and Fig. 4.

E Further Analysis of Weight Decay

By looking at the exact distribution of reconstruction quality to the distance from the margin, we observe that weight-decay (for some values) results in more training samples being on the margin of the trained classifier, thus being more vulnerable to our reconstruction scheme.

This observation is shown in Fig. 14 where we show the scatter plots for all the experiments from Fig. 5 (a). We also provide the train and test errors for each model. It seems that the test error does not change significantly. However, an interesting observation is that reconstruction is possible even for models with non-zero training errors, i.e. models that do not interpolate the data, for which the assumptions of Lyu and Li [2019] do not hold.

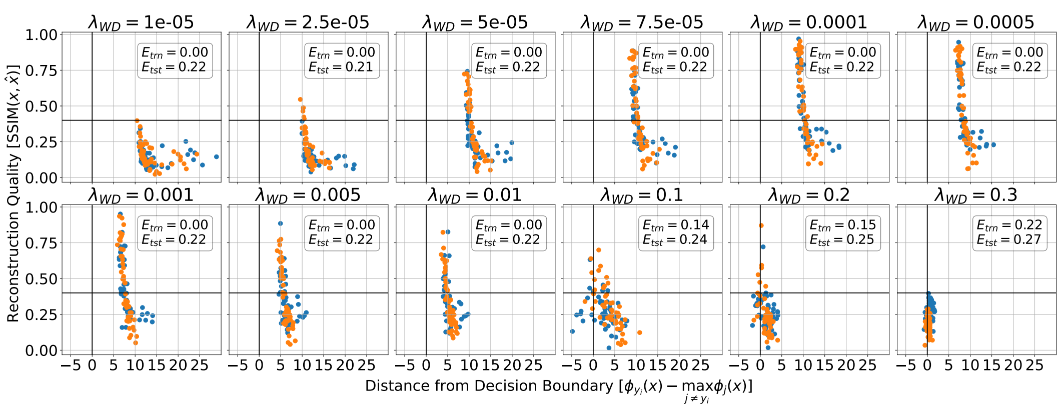


Figure 14: Scatter plots of the 12 experiments from Fig. 5 (a). Each plot is model trained with a different value of weight decay on 2 classes with 50 samples in each class. Certain values of weight decay make the model more susceptible to our reconstruction scheme.

F Convolutional Neural Networks - Ablations and Observations

In this section we provide more results and visualizations to the experiments on convolutional neural network in Section 6.1.

In Fig. 15 we show ablations for the choice of the kernel-size (k) and number of output channels (C_{out}) for models with architecture CONV(kernel-size=k,output-channels=C_{out})-1000-1. All models were trained on 500 images (250 images per class) from the CIFAR10 dataset, with weight-decay term λ_{WD}=0.001. As can be seen, for such convolutional models we are able to reconstruct samples for a wide range of choices.

Note that the full summary of reconstruction quality versus the distance from the decision boundary for the model whose reconstucted samples are shown in Fig. 6, is shown in Fig. 15 for kernel-size 3 (first row) and number of output channels 32 (third column).

Further analysis of Fig. 15. As expected for models with less parameters, the reconstructability decreases as the number of output channels decrease. An interesting phenomenon is observed for varying the kernel size: for a fixed number of output channel, as the kernel size increases, the susceptibility of the model to our reconstruction scheme decreases. However, as the kernel size approaches 32 (the full resolution of the input image), the reconstructability increases once again. On the one hand it is expected, since for kernel-size=32 the model is essentially an MLP, albeit with smaller hidden dimension than usual (at most 64 here, whereas the typical model used in the paper had 1000). On the other hand, it is not clear why for some intermediate values of kernel size (in between 3 and 32) the reconstructability decreases dramatically (for many models there are no reconstructed samples at all). This observation is an interesting research direction for future works.

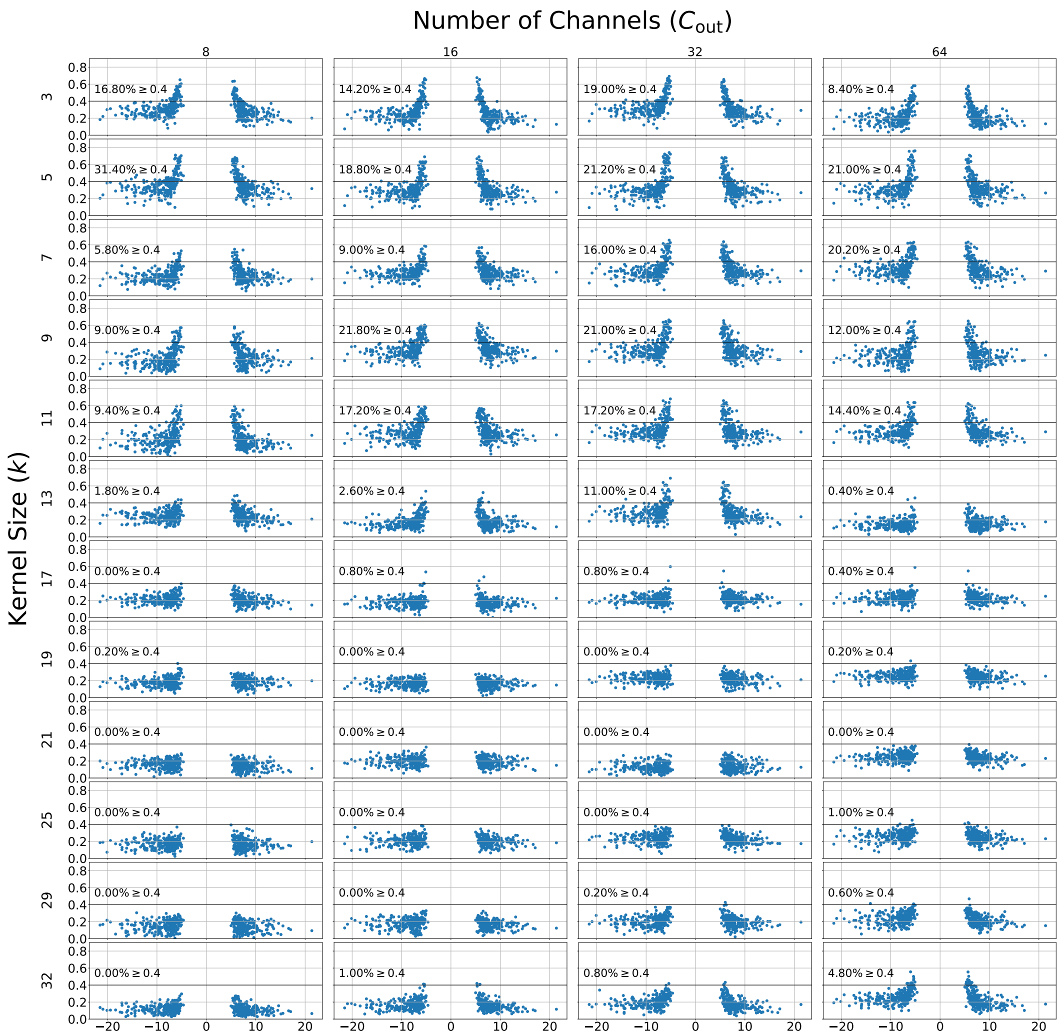


Figure 15: **Ablating the choice of the kernel size and output-channels** for reconstruction from neural binary classifiers with architecture CONV(kernel-size=k,output-channels=C_{out})-1000-1.

Visualizing Kernels. In Haim et al. [2022], it was shown that some of the training samples can be found in the first layer of the trained MLPs, by reshaping and visualizing the weights of the first fully-connected layer. As opposed to MLPs, in the case of a model whose first layer is a convolution layer, this is not possible. For completeness, in Fig. 16 we visualize all 32 kernels of the Conv layer. Obviously, full images of shape 3x32x32 cannot be found in kernels of shape 3x3x3, which makes reconstruction from such models (with convolution first layer) even more interesting.

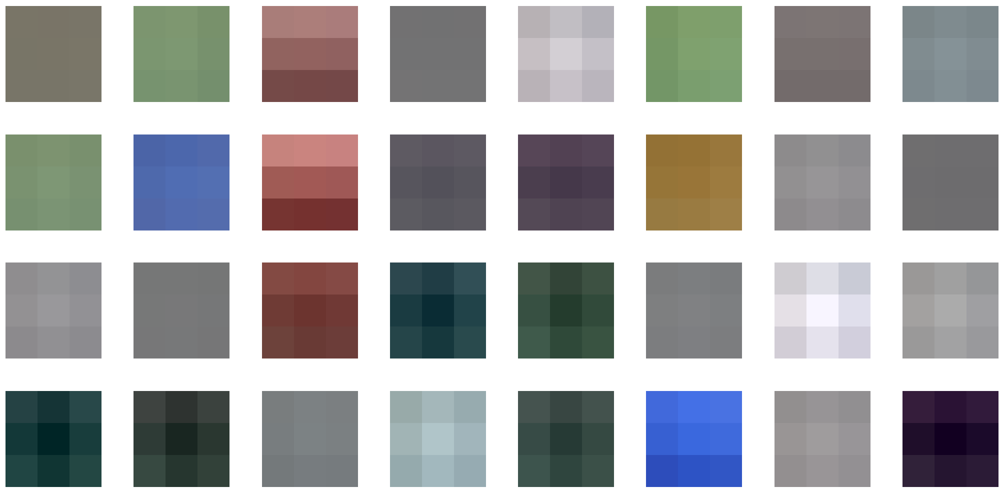


Figure 16: The kernels of the model whose reconstructions are shown in Fig. 6, displayed as RGB images.

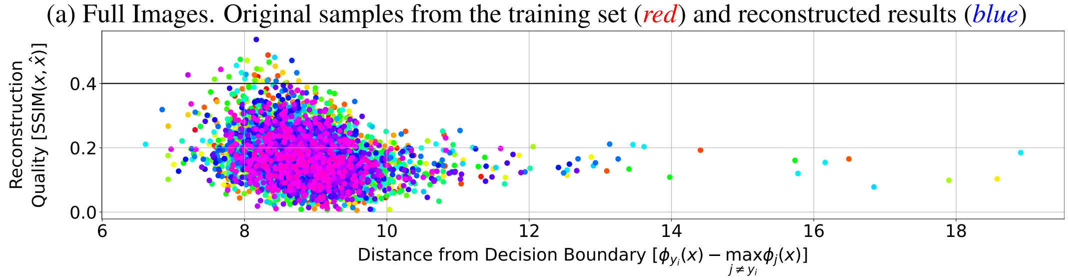
G Reconstruction From a Larger Number of Samples

One of the major limitations of Haim et al. [2022] is that they reconstruct from models that trained on a relatively small number of samples. Specifically, in their largest experiment, a model is trained with only 1,000 samples. Here we take a step further, and apply our reconstruction scheme for a model trained on 5,000 data samples.

To this end, we trained a 3-layer MLP, where the number of neurons in each hidden layer is 10,000. Note that the size of the hidden layer is 10 times larger than in any other model we used. Increasing the number of neurons seems to be one of the major reasons for which we are able to reconstruct from such large datasets, although we believe it could be done with smaller models, which we leave for future research. We used the CIFAR100 dataset, with 50 samples in each class, for a total of 5000 samples.

In Fig. 17a we give the best reconstructions of the model. Note that although there is a degradation in the quality of the reconstruction w.r.t a model trained on less samples, it is still clear that our scheme can reconstruct some of the training samples to some extent. In Fig. 17b we show a scatter plot of the SSIM score w.r.t the distance from the boundary, similar to Fig. 3a. Although most of the samples are on or close to the margin, only a few dozens achieve an SSIM> 0.4. This may indicate that there is a potential for much more images to reconstruct, and possibly with better quality.

(a) Full Images. Original samples from the training set (*red*) and reconstructed results (*blue*)



(b) Scatter plot (similar to Fig. 3) .

Figure 17: Reconstruction from a model trained on 50 images per class from the CIFAR100 dataset (100 classes, total of 5000 datapoints). The model is a 3-layer MLP with 10000 neurons in each layer.