8. Supplementary Material

8.1. Other Privacy Attacks

Besides membership inference attacks, there exists a wide range of privacy attacks against neural networks. Model inversion attacks, first proposed by [14], aim at reconstructing entropy loss: [11, 12], we use the weighted IoU loss and binary cross-entropy loss: For semantic segmentation, where the output values are probability vectors rather than pixel values, we use the cross-entropy loss.

8.2. Detailed Description of Our MIA Algorithm

Our MIA consist of computing the two terms in Eq. (2), i.e. $L_{\text{rec}}$ and $L_{\text{pred}}$ for a given query pair $(x, y)$, where $x$ is an image from the input domain and $y$ is the ground truth from the output domain, using only a black-box access to the victim conditional generation model $V$.

$L_{\text{rec}}$ is computed using the pixel-wise error between the output image predicted by the model, $V(x)$, and the ground truth image $y$, see step 1 in the Algorithm 1. For image translation models, we set the pixel-wise error function, $err$, to be the $L_1$ loss:

$$L_{\text{rec}}(x, y) = \|V(x) - y\|_1$$

(3)

For semantic segmentation, where the output values are probability vectors rather then pixel values, we use the cross-entropy loss:

$$L_{\text{rec}}^\text{seg}(x, y) = -\log(V(x)[y])$$

(4)

In the case of medical segmentation, following Fan et al. [11, 12], we use the weighted IoU loss and binary cross-entropy loss:

$$L_{\text{rec}}^\text{med}(x, y) = L_{\text{IoU}}^w(x, y) + L_{\text{BCE}}^w(x, y)$$

(5)

Defined as:

$$L_{\text{IoU}}^w = 1 - \frac{\sum_{i=1}^{H} \sum_{j=1}^{W} w_{ij}(V(x)_{ij} \cdot y_{ij})}{\sum_{i=1}^{H} \sum_{j=1}^{W} w_{ij}(V(x)_{ij} + y_{ij} - V(x)_{ij} \cdot y_{ij})}$$

(6)

$$L_{\text{BCE}}^w = -\frac{\sum_{i=1}^{H} \sum_{j=1}^{W} w_{ij} \log(V(x)[y]_{ij})}{\sum_{i=1}^{H} \sum_{j=1}^{W} w_{ij}}$$

(7)

Where $H$ and $W$ are the height and width of the query sample, and $w_{ij}$ is the weight of pixel $(i, j)$ and is defined as follows, where $A_{ij}$ represents the area that surrounds the pixel $(i, j)$:

$$w_{ij} = 1 - \frac{\sum_{m,n \in A_{ij}} y_{mn}}{\sum_{m,n \in A_{ij}} 1 - y_{ij}}$$

(8)

$L_{\text{pred}}$ is computed as the average error of a linear regression model, $P$, in predicting pixel values from deep features of the input image.

Our deep features are the activation values in the first 4 blocks of a pre-trained Wide-ResNet50×2 [50]. These features are of sizes 56×56×256, 28×28×512, 14×14×1024, and 7×7×2048. We interpolate all features to size 56×56 using bi-linear interpolation (step 2), and also reduce the output image to 56×56 using bicubic interpolation (step 3). This gives a concatenated feature vector of size 3840 for each pixel $i$ in the 56×56 image ($256+512+1024+2048=3840$). We denote the concatenated feature vector for pixel $i$ as $\psi(i)$.

We randomly select 70% of the pixels as train set, and compute a linear model $P$ to estimate the RGB pixel values $y_{\text{train}}$ from the corresponding deep features $\psi_{\text{train}}(i)$ (step 4). The remaining 30% of pixels will be used as a test split, $\psi_{\text{test}}$ (step 5). I.e. $|\psi_{\text{train}}| = 2195 \times 3840$, $|y_{\text{train}}| = 2195 \times 3$ and $|\psi_{\text{test}}| = 941 \times 3840$, $|y_{\text{test}}| = 941 \times 3$.

The linear regression model $P$, a matrix of size 3840×3, is trained to minimize the error over $\psi \psi_{\text{train}}(i)$ (step 6). $L_{\text{pred}}$ will be the average absolute error over $\psi \psi_{\text{test}}(i)$ (step 7). We found that fitting the linear model to 70% of pixels and measuring the error on the remaining 30% gives better results than just measuring the error of the linear fitting.

We compute $L_{\text{mem}}$ according to Eq. (2) and compare the results with a predefined threshold value $\tau$, such that any pair $(x, y)$ for which is holds that $L_{\text{mem}}(x, y) < \tau$ is denoted as a member of the victim models’ $V$ train set (steps 8-9).

We have experimented with different resize methods (step 3) and found that our attack success rate is not very sensitive to the resize method. Additionally, we evaluated the effect of different train-test partitions (steps 4 & 5) and found that using less than 50% of the image pixels for training the linear regression model results with unstable performance, while all values of 50% or above results in similar attack success rates.

8.3. Parameter Selection

We experimented with different values for the $\alpha$ value in Eq. (2). As can be seen in Fig. 6 $\alpha = 1$ was the best choice over all benchmarks.
Algorithm 1. Membership Inference Attack

Input: Query pair \((x, y)\), victim model \(V\), feature extractor \(F\), scalar \(\alpha\), threshold \(\tau\), error function \(err\)

Output: Membership inference result

1. \(L_{rec} = err(V(x), y)\)
2. \(\psi = F(x) // |\psi| = 56 \times 56 \times 3840\)
3. \(y = \text{resize}(y, 56 \times 56 \times 3)\)
4. \(\{\psi_{train}, y_{train}\} \leftarrow 70\%\ \{\psi, y\}\)
5. \(\psi_{test}, y_{test} = \{\psi, y\} \setminus \{\psi_{train}, y_{train}\}\)
6. Train linear regression \(P\) with \(\psi_{train}, y_{train}\)
7. \(L_{pred} = \frac{1}{N} \sum_{i=1}^{N} ||P \psi_{test}(i) - y_{test}||_1 //N = 941\)
8. \(L_{mem} = L_{rec} - \alpha \cdot L_{pred}\)
9. if \(L_{mem} < \tau\) then
   Return True
else
   Return False

8.4. MIA vs output dimension

As described in Sec. 4.2.1, we evaluated the effect of reducing the output dimension on the accuracy of reconstruction-based MIA. The reduction was achieved by randomly sampling \(N\) output pixels, and using them as the output, where \(N\) ranges from a single pixel and up to 200 pixels. Fig. 7 demonstrates that MIA accuracy indeed scales with the number of output dimensions. Results for Pix2PixHD, UperNet and HRNetV2 are presented in Fig. 2.

8.5. calibration Effect

As can be seen in Tab. 1 using our membership error \(L_{mem}\), Eq. 2, substantially improves the success rates in all of our experiments. As can be seen in Fig. 8 our \(L_{mem}\) can better separate train and test images by a simple threshold compared to the reconstruction error \(L_{rec}\). Results for Pix2PixHD on the Maps2sat and Cityscapes datasets are presented in Fig. 4.

8.6. Human-Supervised Image difficulty score

We compare our self-supervised single-sample predictability error with the human-supervised difficulty score proposed by [41]. In Fig. 9 we present images ranked from easy to difficult using our implementation of the supervised-image difficulty score, for the Cityscapes and Maps datasets. The ranking seems correlated with image sharpness and level-of-detail images. As can be seen in Tab. 3 our score outperforms the human-supervised score. We compare the correlation between the reconstruction error for unseen images to our self-supervised predictability error and the human-supervised scorre.

8.7. Multi-Image predictability error

As discussed in Sec. 4.2.2 we compare our single-sample predictability error to a multi-sample predictability error (MSPS) by training a “shadow” model, sharing the same architecture as the victim model, on auxiliary samples. As can be seen in Tab. 3 when training the MSPS on 100 images, it underperforms our method on Pix2PixHD and the evaluated semantic segmentation models. For the smaller Pix2Pix architecture, MSPS was more successful, obtaining...
Figure 8. The proposed membership error $L_{mem}$ can better separate train and test images by a simple threshold (i.e. a vertical line) compared to the reconstruction error $L_{rec}$. Pix2pixHD for Maps2sat and Cityscapes are presented in Fig. 4.

We also compare our method to the setting were many out-of-distribution but similar sample are available. We trained shadow models on $4K$ samples from the BDD dataset as MSPS for the Cityscapes dataset. As can be seen in Tab. 8 this too underperforms our method. Note that it is rare to have similar datasets with nearly identical labels, such as in the case of BDD and Cityscapes.

8.8. Shadow models

As discussed in Sec. 4.2 we, for the interest of completeness we compare our method with the popular approach of shadow-model classifiers for image translation MIA. For this,
we select $N$ images, denoted as $shadow_{train}$, for training the shadow model. As an upper-bound, the shadow model is given the exactly same architecture as used by the victim model. Another $N$ images, not seen by the shadow model, are set to be $shadow_{test}$. We define a new label for each sample as follows:

$$
label(x) = \begin{cases} 
0, & \text{if } x \in shadow_{train} \\
1, & \text{if } x \in shadow_{test} 
\end{cases}
$$

The classifier $C$ architecture and training procedure are similar to [21]. For each image, we compute the structured loss map between the ground-truth image and the generated image, using $L_1$ as the loss function. At every epoch we randomly crop 15 patches of size $90 \times 90$ from the struc-
Table 9. Comparison between our MIA and the commonly used shadow-model-based classifier attack, using $4K$ train and $4K$ test images from the BDD dataset. Our MIA outperforms while not requiring extra training images.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>Ours ROC</th>
<th>In-Dist ROC</th>
<th>In-Dist Acc.</th>
<th>Out-of-Dist (BDD) ROC</th>
<th>Out-of-Dist Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pix2pix</td>
<td>Maps2sat</td>
<td>85.65%</td>
<td>80.15%</td>
<td>73.4%</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pix2pix</td>
<td>Cityscapes</td>
<td>83.23%</td>
<td>78.68%</td>
<td>67.5%</td>
<td>72.57%</td>
<td>56.16%</td>
</tr>
<tr>
<td>Pix2pixHD</td>
<td>Maps2sat</td>
<td>99.42%</td>
<td>98.63%</td>
<td>93.7%</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pix2pixHD</td>
<td>Cityscapes</td>
<td>99.09%</td>
<td>96.39%</td>
<td>64.0%</td>
<td>95.78%</td>
<td>56.5%</td>
</tr>
</tbody>
</table>

8.9. Memorization

As mentioned in Sec. 5, memorization is the main reason for the success of our method. Fig. 11 shows the accuracy of our method as a function of the number of epochs used for training the victim model, clearly suggesting that memorization is indeed the vulnerability.

8.10. Defenses

In Sec. 5, we discuss the Gauss defense, including other common defenses, against our attack. We evaluated our attack accuracy as a function of different noise STD. Fig. 12 shows that a considerable amount of noise, which corrupts the generated output, is required in order to have a significant effect over our attack success, which is still much better than random guessing (50%). Results for Pix2PixHD, UperNet and HRNetV2 are presented in Fig. 5.

8.11. ImageNet predictability error

Our predictability error relies on learning a mapping between feature vectors to their corresponding pixel values. We use a pre-trained Wide-ResNet50×2 [50], which is trained on the ImageNet dataset. We do not make any assumptions regarding an overlap between the pre-trained model’s training data (i.e. ImageNet) and the data during in the attack. In the scenario in which such an overlap exists, the concern is that the predictability error would lose its credibility.

In order to verify this, we computed the predictability error of a random subset of 1K train images and 1K test images from the ImageNet dataset that received the lowest and highest predictability errors. First row - lowest scored train images. Second row - lowest scored test images. Third row - highest scored train images. Last row - highest scored test images. As can be seen, the predictability error is effective even on images that were used for training the feature extractor.

Figure 11. Examples of images from the ImageNet dataset that received the lowest and highest predictability errors. First row - lowest scored train images. Second row - lowest scored test images. Third row - highest scored train images. Last row - highest scored test images. As can be seen, the predictability error is effective even on images that were used for training the feature extractor.
images, from the ImageNet dataset. As no input-output pairs exist, we trained the linear predictor to predict pixel values from deep features of the same image. We do not observe any significant difference between the two - both share similar mean and std values: (0.0549, 0.018) for the train images and (0.0556, 0.0191) for the test images. A ROCAUC score of 51% further demonstrates that there is no clear difference between the distribution of the predictability error on seen and unseen images.

Fig. 13 further demonstrates this. The first row presents the train images that received the lowest scores, i.e. marked as easy images, and the second row presents the test images with the lowest scores. Both correspond to "plain" images, regardless of whether they are known (train) or unknown (test). The same applies to the Difficult images. The third row presents the highest scored train images and the last row presents the highest scored test images. Both contains complex patterns and high variance. This demonstrates that the predictability error is not affected by the having prior knowledge of the image, and is only measuring the amount of variance and complexity of an image.