Supplementary Material
PrivateEye: In-Sensor Privacy Preservation Through Optical Feature Separation

1. Overview

The supplement is organized in the following order. We first elaborate on our overall algorithm for PrivateEye. We next showcase additional experimental results, including a scenario involving privacy-preservation for 3 tasks (2 binary and 1 multi-class) using a single encoder, along with corresponding visualizations of CAMs and encoded images. We also look at a case study of obfuscating facial identification while recovering other attributes. Finally we explore an alternative loss function that allows for both pushing and pulling of features.

2. Algorithm

Algorithm 1 PrivateEye: Training the Encoder

```
Input: Tasks T1, T2 ∈ T, Classifiers $f_{T1}$, $f_{T2}$, Encoder $f_e(:,\phi)$, Training data with images $x$, T1 labels $\tilde{y_1}$ and T2 labels $\tilde{y_2}$, Anchor points $P_1, P_2$, Anchor loss weight $\gamma$, Learning rate $\eta$
Output: Learned encoder $f_e$ parameters $\phi$
1:  Train classifiers $f_{T1}$, $f_{T2}$ on T1, T2 using cross-entropy loss $\mathcal{H}$
2:  Freeze $f_{T1}$, $f_{T2}$ parameters
3:  for each epoch do
4:      for each $x, y_1, y_2$ in batch do
5:          $x' = f_e(x)$
6:          $x' = normalize(x)$
7:          $(y_1, A_1),\ (y_2, A_2) = f_{T1}(x'),\ f_{T2}(x')$
8:          Get CAMs CAM-T1, CAM-T2          {Eqn 1 and 2}
9:          $\mathcal{L}_{anchor,T1} = loss_{anchor}(\text{CAM-T1}, P_1)$
10:         $\mathcal{L}_{anchor,T2} = loss_{anchor}(\text{CAM-T1}, P_2)$
11:         $\mathcal{L}_{anchor} = \mathcal{L}_{anchor,T1} + \mathcal{L}_{anchor,T2}$
12:         $\mathcal{L} = \mathcal{H}(y_1, \tilde{y_1}) + \mathcal{H}(y_2, \tilde{y_2}) + \lambda\mathcal{L}_{anchor}$
13:         $\phi = \phi - \eta\nabla_\phi\mathcal{L}$
14:     end for
15:     Evaluate model on validation set for T1, T2
16: end for
17: return Return $f_e(:,\phi)$
```

Algorithm 1 outlines our method for training the encoder. We start by pre-training the classifiers for the relevant tasks, then begin training the encoder while keeping the classifiers frozen. For each batch, the image is passed through the encoder and normalized using both a mean and standard deviation of 0.5. The normalized image is then processed by each classifier to obtain the corresponding logits (y_i) and intermediate activations (A_i). These are used to compute the Class Activation Maps (CAMs) for each classifier.

For binary tasks, the CAMs for the positive and negative class labels can be simply combined by adding the positive and negative components of the CAMs before applying ReLU or by taking its absolute value. For multi-class tasks, we take the mean of the CAMs across each relevant logit. The anchor loss is then calculated based on the CAMs and anchor points for each task to facilitate feature separation, along with the cross-entropy loss to ensure high classification accuracy for each task. We backpropagate through the networks using the AdamW [5] optimizer and update the encoder parameters, repeating this process for a fixed number of epochs.

3. Additional Experiments

3.1. Case study: Three Tasks

We demonstrate that our method extends beyond two binary tasks by exploring a scenario with three tasks. Using the CelebA [4] dataset, we choose the following tasks:

1. T1 (smiling?) ∈ {smiling, not smiling}.
2. T2 (gender) ∈ {male, not male}.
3. T3 (hair color) ∈ {black hair, blond hair, brown hair, gray hair}.

We push T1 features to the top-left (TL), T2 to the bottom-right (BR), and T3 to the top-right (TR). We use three ResNet18 models as classifiers, modifying the last fully-connected layer to have one output for T1 and T2, and four outputs for T3 to account for the hair colors. The loss for

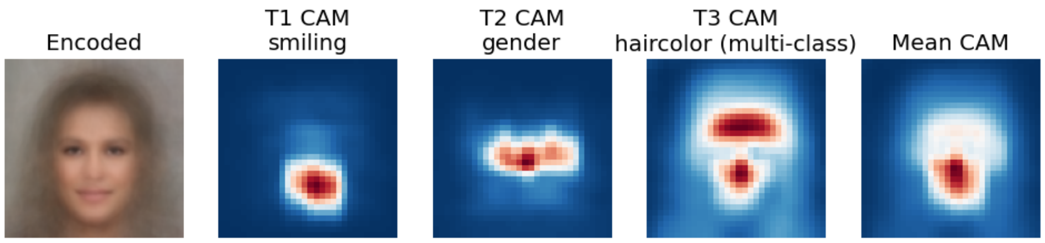

Figure 1. Visualization of the mean encoded images (pre-masking) and CAMs for the three tasks without any encoder, on the pretrained classifiers. Notice how the CAMs are focused around the mouth, face, hair/beard regions for the `T1`, `T2` and `T3` tasks respectively.

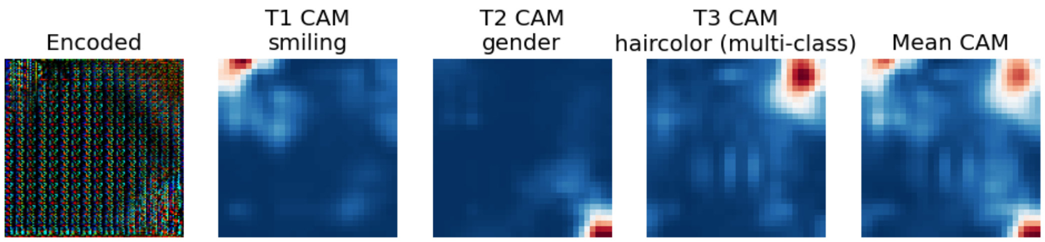

Figure 2. Visualization of the encoded (pre-masking) and CAMs for the three tasks with our learned UNet-tiny encoder, on the pretrained classifiers. Notice how the CAMs have been pushed to the top-left, bottom-right and top-right for `T1`, `T2` and `T3` respectively.

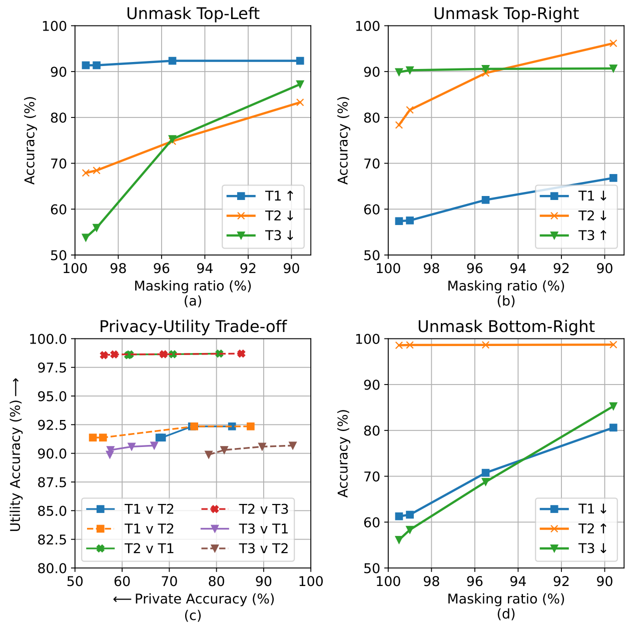

Figure 3. Performance for training classifiers on a learned encoder (UNet-tiny) on the three tasks: `T1` (smiling), `T2` (gender) and `T3` (hair color). We use three different masks where we unmask either the top-left (a), bottom-right (b) and top-right (d) to preserve tasks `T1`, `T2` and `T3` respectively. (d) summarizes the trade-off curves.

`T1` and `T2` is binary cross-entropy, while the loss for `T3` is cross-entropy. Our method can be adapted to any setup for which we can calculate CAMs. We test this using the UNet-tiny encoder and train it according to Algorithm 1 with an additional `T3` objective (cross-entropy and anchor loss) added to the loss function. The input and encoded images are of dimensions $128 \times 128 \times 3$.

Figures 1 and 2 visualize the encoded images and the CAMs for each task. The CAMs, originally 16×16, have been resized to 128×128 using bilinear interpolation for better visualization. The visualizations clearly show that the encoder has effectively pushed the features to the specified corners. Carefully looking at the encoded image also shows that the three corners are more irregular than the rest of the image. We believe this contains information pertaining to the said tasks.

To evaluate the privacy-utility trade-off, we freeze the encoder and train ResNet18 classifiers to evaluate the performance of various combinations of mask by selecting specific masks. For example, a mask which keeps only the top-right part of the encoded image will preserve `T2` accuracy and hinder an attacker from ascertaining information about `T1` and `T3`. Different combinations of masks along with their masking ratios are shown in Fig. 3. We observe that the utility task accuracies are well-preserved across all three masks. At unmask ratios of about 1% (i.e., 99% of the encoded image is masked) we achieve a strong trade-off between utility and privacy tasks, maintaining high utility accuracy and low privacy accuracy. However, some task combinations exhibit less ideal trade-offs. For example, in Fig. 3(b), while we maintain high `T3` (utility) accuracy and low `T1` (privacy) accuracy across the shown masking ratios,

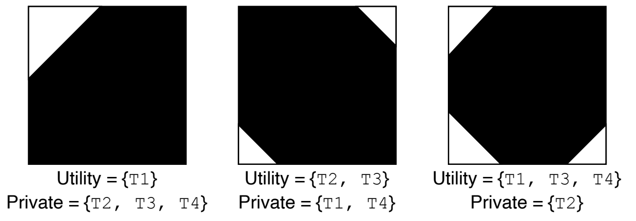

Figure 4. Example of masking four tasks with the encoder trained to push $T1, T2, T3, T4$ to the top-left, top-right, bottom-left and bottom-right respectively. Various masking patterns serve as filters for utility and privacy tasks.

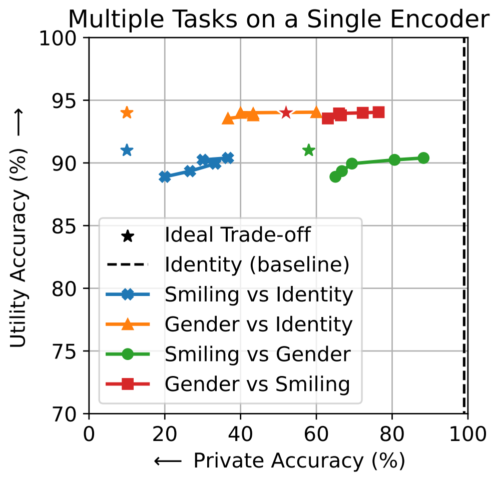

Figure 5. Evaluation of a trained DONN-10-res (10-layer residual DONN) encoder to perform task separation on *smiling*, *gender* and *identity*. Importantly, one set of encoder weights is used for all tasks. The tasks are filtered based on how we perform masking.

an attacker is able to recover T2 (privacy) accuracy by about 20%. We hypothesize that that this is because T3 (hair color) leaks information about T2 (gender). Note that some of the baseline performance for tasks T1 and T2 slightly differs from the main manuscript because we use a subset of the dataset containing hair color labels. Approximately 40% of the dataset lacks hair color labels, which negatively impacts training, even if we set the label as None.

Overall this demonstrates that our encoder once trained for a set of tasks, provides flexibility in selecting the utility and privacy tasks. This is enabled by our ability to control the masking pattern and its associated masking ratios. Fig. 4 showcases an example of some of the possible mask patterns for 4 tasks.

3.2. Case Study: Obfuscating Identities

We now explore our method's capability of handling multiple tasks on a more a practical use case where specific attributes about a person need to be inferred, but the person's identity needs to be kept hidden. To evaluate this, we use the previously trained encoder that learned to separate the tasks of *smiling* and *gender* and test it on the *identity* task. For the *identity* task, we select a subset of 10 identities from the CelebA dataset and train a ResNet18 classifier, achieving a baseline accuracy of 100%. We then evaluate different combinations of these tasks using the trained encoder. To balance *smiling* and *gender or identity*, we mask all but the top-left corner of the encoded image. Similarly, for *gender* and *smiling or identity*, we mask all but the bottom-right corner. Masking ratios between 99% and 90% are used for this evaluation.

Fig. 5 demonstrates the results where we observe some interesting trends. With aggressive masking ratios, we achieve good trade-offs across all tasks. For example, for the *gender* (utility) vs. *identity* (privacy) task, at 99% masking, utility accuracy drops by only $\sim$2%, while private accuracy decreases from 100% to 36%. We observe that *gender* leaks more information about identity than *smiling*. We hypothesize this is because the encoder only needs to push features from a localized region (e.g., mouth) for the *smiling* task but requires features from a larger region (e.g., face/head) for the *gender* task, which inadvertently contains more information about the person's identity.

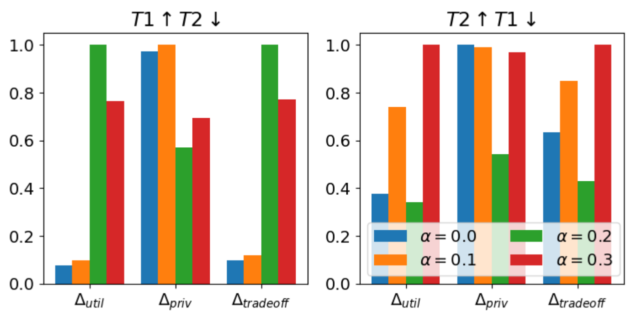

Figure 6. DONN Attract Repel Loss function. Lower is better in all cases.

3.3. Alternative loss function

The loss function in Eq. (6) in the main manuscript pushes features belonging to a task to one corner of the image but does not explicitly push away remnant features from that corner. We enhance the loss function with an alternative *attract+repel* method:

$$\begin{aligned}\mathcal{L}_{anchor,attract,T1} &= loss_{anchor}(L_{CAM,T1},(0,0))\\ \mathcal{L}_{anchor,repel,T1} &= loss_{anchor}(L_{CAM,T2},(0,0))\\ \mathcal{L}_{anchor,attract,T2} &= loss_{anchor}(L_{CAM,T2},(1,1))\\ \mathcal{L}_{anchor,repel,T2} &= loss_{anchor}(L_{CAM,T1},(1,1))\\ \mathcal{L}_{anchor} &= \mathcal{L}_{anchor,attract,T1} + \mathcal{L}_{anchor,attract,T2}\\ &\quad -\alpha(\mathcal{L}_{anchor,repel,T1} + \mathcal{L}_{anchor,repel,T2})\end{aligned} \tag{1}$$

The *attract* component pulls the corresponding features towards its anchor, while the *repel* pushes away features from its anchor. In Eq. (1), we attract `T1` features to (0, 0) (top-left corner) and repel `T2` features from that corner, and similarly for the bottom-right corner (1, 1). The strength of repulsion relative to attraction is controlled by the hyper-parameter α. When masking is performed for utility task `T1`, the unmasked region should now leak less information about `T2`.

Fig. 6 compares the original loss function ($\alpha = 0.0$) and the attract-repel loss across different values of α. We observe different performances on task pairs (`T1`, `T2`) $\in$ (smiling, gender) and (`T1`, `T2`) $\in$ (gender, smiling). For the former, $\alpha = 0$ provides the best trade-off, while for the latter, $\alpha = 0.2$ offers the best trade-off. A better trade-off is indicated by having both low Δ_{util} and Δ_{priv} and an overall lower $\Delta_{trade-off}$. This indicates that for certain task configurations, the repel loss component significantly aids in achieving a better utility-privacy balance.

3.4. Additional Dataset

We setup a privacy-utility task on a grayscale CIFAR-10 dataset [2] with classes $\in$ {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}. The utility task is to perform animal vs. machine classification, and the private task is to perform the typical 10-class CIFAR-10 classification. The backbone we use is a ResNet18 which achieves an accuracy of 99% and 88.4% on the two tasks respectively with no encoders. With 95% masking, we achieve a utility/privacy accuracy of 75.7% and 29.3% respectively with a DONN.

4. Masking Details

4.1. Digital Implementation

As described in Section 3.3.3, we generate static corner-based masked to allow utility features to go through while simultaneously obfuscating private features. The masks are generated using Eq. (2):

$$M_{top-left}, M_{bottom-right} = triu(diag), tril(diag) \quad (2)$$

Here, $triu$ and $tril$ represent the upper and lower triangular matrix operations, with $diag$ as the principal axis controlling the masking strength. All values above the principal axis are set to 1, and the rest are 0. In practice, we generate these masks using NumPy or PyTorch's *tril* or *triu* operation.

4.2. Hardware Implementation

In typical image sensors, a row addresser and a column addresser are placed beside the pixel array to sequentially pass the readout-enable signals to all pixels for a full-sized image readout. However, these addressers can also be easily programmed to change their addressing start/end positions to only read out the pixels within a customized region, i.e., the mask. The mask can either be regular shapes like rectangular or triangular, or irregular shapes, by changing the addressing start/end positions for every row/column of pixels.

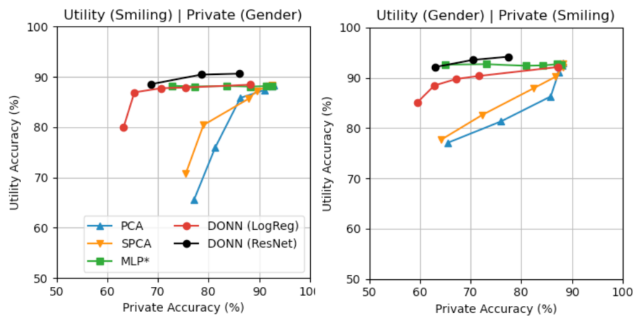

Figure 7. We contrast our method with other dimensionality reduction methods and show that DONN provides better trade-offs.

5. Additional Discussion

5.1. Advantages of DONN

Higher Level of Security. DONNs enhance security through their unique computational approach. Optical processing combined with pixel dropout techniques obfuscates sensitive information before the image is read out, ensuring that by the time data is converted to the digital domain, private features are effectively masked. This preemptive measure secures private information even if the edge system is breached, unlike traditional digital privacy filters which are vulnerable if hacked.

5.2. Limitations of DONNs

Privacy with Linear Models. Despite our encoder being linear, it performs well due to the effective combination of feature separation and masking, which acts as dimensionality reduction by preserving only the most relevant features. Dimensionality reduction techniques like Principal Component Analysis (PCA) and Supervised PCA (SPCA) are known to achieve compressive privacy, balancing both privacy and compression [1,3]. To evaluate our method, we compare it with PCA, SPCA, and a linear projection from a 3-layer MLP. We use the same number of principal components as the masking ratio used for DONN and employ a logistic regression classifier. Fig. 7 shows that while PCA and SPCA have a subpar utility-privacy trade-off, the MLP projection demonstrates a strong trade-off. Our method aligns with these findings, indicating that our mode of dimensionality reduction can effectively enhance privacy.

References

[1] R Vidya Banu and N Nagaveni. Preservation of data privacy using pca based transformation. In *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 439–443. IEEE, 2009. 4

[2] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[3] Sun-Yuan Kung, Thee Chanyaswad, J Morris Chang, and Peiyuan Wu. Collaborative pca/dca learning methods for compressive privacy. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(3):1–18, 2017. 4

[4] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018. 1

[5] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017. 1