
A APPENDIX

A.1 U-NET ARCHITECTURE

In this section, we present our U-Net model configuration in JSON format. We used the following architecture for from-scratch training on MNIST and CIFAR10:

```
{
  "_class_name": "UNet2DModel",
  "_diffusers_version": "0.30.2",
  "act_fn": "silu",
  "add_attention": true,
  "attention_head_dim": 8,
  "attn_norm_num_groups": null,
  "block_out_channels": [
    128,
    128,
    256,
    512
  ],
  "center_input_sample": false,
  "class_embed_type": null,
  "down_block_types": [
    "DownBlock2D",
    "DownBlock2D",
    "AttnDownBlock2D",
    "DownBlock2D"
  ],
  "downsample_padding": 1,
  "downsample_type": "conv",
  "dropout": 0.0,
  "flip_sin_to_cos": true,
  "freq_shift": 0,
  "in_channels": 3,
  "layers_per_block": 2,
  "mid_block_scale_factor": 1,
  "norm_eps": 1e-05,
  "norm_num_groups": 32,
  "num_class_embeds": null,
  "num_train_timesteps": null,
  "out_channels": 3,
  "resnet_time_scale_shift": "default",
  "sample_size": 32,
  "time_embedding_type": "positional",
  "up_block_types": [
    "UpBlock2D",
    "AttnUpBlock2D",
    "UpBlock2D",
    "UpBlock2D"
  ],
  "upsample_type": "conv"
}
```

We used the following architecture for from-scratch training on STL10:

```
{
  "_class_name": "UNet2DModel",
  "_diffusers_version": "0.30.2",
  "act_fn": "silu",
  "add_attention": true,
  "attention_head_dim": 8,
  "attn_norm_num_groups": null,
  "block_out_channels": [
    128,
```

```

054     128,
055     256,
056     512
057 ],
058 "center_input_sample": false,
059 "class_embed_type": null,
060 "down_block_types": [
061     "DownBlock2D",
062     "DownBlock2D",
063     "AttnDownBlock2D",
064     "DownBlock2D"
065 ],
066 "downsample_padding": 1,
067 "downsample_type": "conv",
068 "dropout": 0.0,
069 "flip_sin_to_cos": true,
070 "freq_shift": 0,
071 "in_channels": 3,
072 "layers_per_block": 2,
073 "mid_block_scale_factor": 1,
074 "norm_eps": 1e-05,
075 "norm_num_groups": 32,
076 "num_class_embeds": null,
077 "num_train_timesteps": null,
078 "out_channels": 3,
079 "resnet_time_scale_shift": "default",
080 "sample_size": 96,
081 "time_embedding_type": "positional",
082 "up_block_types": [
083     "UpBlock2D",
084     "AttnUpBlock2D",
085     "UpBlock2D",
086     "UpBlock2D"
087 ],
088 "upsample_type": "conv"
089 }

```

We used the following architecture for fine-tuning on CIFAR10:

```

087 {
088     "_class_name": "UNet2DModel",
089     "_diffusers_version": "0.0.4",
090     "act_fn": "silu",
091     "attention_head_dim": null,
092     "block_out_channels": [
093         128,
094         256,
095         256,
096         256
097     ],
098     "center_input_sample": false,
099     "down_block_types": [
100         "DownBlock2D",
101         "AttnDownBlock2D",
102         "DownBlock2D",
103         "DownBlock2D"
104     ],
105     "downsample_padding": 0,
106     "flip_sin_to_cos": false,
107     "freq_shift": 1,
108     "in_channels": 3,
109     "layers_per_block": 2,
110     "mid_block_scale_factor": 1,
111     "norm_eps": 1e-06,
112     "norm_num_groups": 32,

```



```

108     "out_channels": 3,
109     "sample_size": 32,
110     "time_embedding_type": "positional",
111     "up_block_types": [
112         "UpBlock2D",
113         "UpBlock2D",
114         "AttnUpBlock2D",
115         "UpBlock2D"
116     ]
117 }

```

We used the following architecture for fine-tuning on CelebA:

```

121 {
122     "_class_name": "UNet2DModel",
123     "_diffusers_version": "0.0.4",
124     "act_fn": "silu",
125     "attention_head_dim": null,
126     "block_out_channels": [
127         128,
128         128,
129         256,
130         256,
131         512,
132         512
133     ],
134     "center_input_sample": false,
135     "down_block_types": [
136         "DownBlock2D",
137         "DownBlock2D",
138         "DownBlock2D",
139         "DownBlock2D",
140         "AttnDownBlock2D",
141         "DownBlock2D"
142     ],
143     "downsample_padding": 0,
144     "flip_sin_to_cos": false,
145     "freq_shift": 1,
146     "in_channels": 3,
147     "layers_per_block": 2,
148     "mid_block_scale_factor": 1,
149     "norm_eps": 1e-06,
150     "norm_num_groups": 32,
151     "out_channels": 3,
152     "sample_size": 256,
153     "time_embedding_type": "positional",
154     "up_block_types": [
155         "UpBlock2D",
156         "AttnUpBlock2D",
157         "UpBlock2D",
158         "UpBlock2D",
159         "UpBlock2D",
160         "UpBlock2D"
161     ]
162 }

```

A.2 GENERATED SAMPLES

The generated samples for CIFAR10 (animals) and CelebA (female) are shown in Figure 1 and 2, respectively.



Figure 1: Generated samples from fine-tuned diffusion models on CIFAR10 (animals).



Figure 2: Generated samples from fine-tuned diffusion models on CelebA (female).

A.3 THEORETICAL JUSTIFICATION

In this section, we provide a theoretical explanation of the proposed method from the perspective of maximizing and minimizing the ELBO $\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta) \approx -\ell(\mathbf{x}; \theta)$ defined in Eqs. (7) and (8).

To prevent diffusion models from generating sensitive data, we aim to maximize the ELBO for the normal data distribution $p_{\mathcal{N}}(\mathbf{x})$:

$$\max_{\theta} (1 - \beta) \mathbb{E}_{p_{\mathcal{N}}}[-\ell(\mathbf{x}; \theta)],$$

and minimize it for the sensitive data distribution $p_{\mathcal{S}}(\mathbf{x})$:

$$\min_{\theta} \beta \mathbb{E}_{p_{\mathcal{S}}}[-\ell(\mathbf{x}; \theta)],$$

where β is the ratio of the sensitive data.

First, we focus on maximizing the ELBO for the normal data. As shown in Eqs. (13) and (14), we assume that the unlabeled data distribution $p_{\mathcal{U}}(\mathbf{x})$ can be written as a linear combination of $p_{\mathcal{N}}(\mathbf{x})$ and $p_{\mathcal{S}}(\mathbf{x})$:

$$p_{\mathcal{U}}(\mathbf{x}) = \beta p_{\mathcal{S}}(\mathbf{x}) + (1 - \beta) p_{\mathcal{N}}(\mathbf{x}) \Leftrightarrow (1 - \beta) p_{\mathcal{N}}(\mathbf{x}) = p_{\mathcal{U}}(\mathbf{x}) - \beta p_{\mathcal{S}}(\mathbf{x}).$$

With this assumption, the ELBO for $p_{\mathcal{N}}(\mathbf{x})$ can be rewritten as follows:

$$\max_{\theta} (1 - \beta) \mathbb{E}_{p_{\mathcal{N}}}[-\ell(\mathbf{x}; \theta)] = \max_{\theta} \{ \mathbb{E}_{p_{\mathcal{U}}}[-\ell(\mathbf{x}; \theta)] - \beta \mathbb{E}_{p_{\mathcal{S}}}[-\ell(\mathbf{x}; \theta)] \}.$$

This maximization equals the following minimization:

$$\min_{\theta} (1 - \beta) \mathbb{E}_{p_{\mathcal{N}}}[\ell(\mathbf{x}; \theta)] = \min_{\theta} \{ \mathbb{E}_{p_{\mathcal{U}}}[\ell(\mathbf{x}; \theta)] - \beta \mathbb{E}_{p_{\mathcal{S}}}[\ell(\mathbf{x}; \theta)] \}.$$

This is equivalent to the sum of the second and third terms in Eq. (17) of the proposed method, where $\ell_{\text{BCE}}(\mathbf{x}, 0; \theta) = \ell(\mathbf{x}; \theta)$.

Next, we focus on minimizing the ELBO for the sensitive data. Unfortunately, since $-\ell(\mathbf{x}; \theta)$ is unbounded below, this minimization diverges to negative infinity, leading to meaningless solutions. To solve this issue, we introduce the following upper bound for $-\ell(\mathbf{x}; \theta)$, inspired by the binary classification as shown in Eq. (11):

$$-\ell(\mathbf{x}; \theta) \leq -\log(1 - \exp(-\ell(\mathbf{x}; \theta))).$$

This upper bound is proportional to $-\ell(\mathbf{x}; \theta)$ and approaches zero when $-\ell(\mathbf{x}; \theta)$ tends to negative infinity. With this upper bound, the ELBO for $p_S(\mathbf{x})$ is bounded above as follows:

$$\min_{\theta} \beta \mathbb{E}_{p_S}[-\ell(\mathbf{x}; \theta)] \leq \min_{\theta} \beta \mathbb{E}_{p_S}[-\log(1 - \exp(-\ell(\mathbf{x}; \theta)))].$$

This is equivalent to the first term in Eq. (17), where $\ell_{\text{BCE}}(\mathbf{x}, 1; \theta) = -\log(1 - \exp(-\ell(\mathbf{x}; \theta)))$.

From the above discussion, the proposed method in Eq. (17) can be interpreted as maximizing the ELBO for normal data while minimizing the ELBO for sensitive data. For the former, we use PU learning to approximate the normal data distribution $p_N(\mathbf{x})$, and for the latter, we introduce the upper bound for the ELBO based on the binary classification.

A.4 LIMITATIONS

In this section, we discuss the limitations of the proposed method. Our approach requires that the labeled sensitive data represent the characteristics of all sensitive data. We will explain this using the example of MNIST, where even numbers are considered normal and odd numbers are considered sensitive. If the unlabeled data contain all even and odd numbers, but the labeled sensitive data only include 1, 3, 5, and 7, then the proposed method would generate the digit 9.

To solve this issue, it is necessary to prepare a sufficient variety of labeled sensitive data. In the MNIST example, it would be enough to include the digit 9 in the labeled sensitive data. Similarly, if we want to prevent the generation of a face of a particular person, it would be enough to prepare just a few photos of that person. However, if the sensitive data are more diverse, such as male or female faces, or categories like vehicles or animals, then our approach would likely require a larger amount of labeled sensitive data. In the experiments using STL10 and CelebA, we successfully prevented the generation of sensitive data with 200 labeled sensitive samples. Since the faces, vehicles, and animals included in the training and test sets are different, we believe our approach demonstrated a certain level of generalization ability.

In any case, addressing this issue is an important part of our future work. We plan to explore ways to prevent the generation of sensitive data not covered by the labeled sensitive data, for example, by using supplementary information such as text descriptions.

A.5 WHY EXISTING METHODS ARE INEFFECTIVE

Existing methods assume that all unlabeled data are normal, and maximize the ELBO $\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta)$ in Eq. (7) for the unlabeled data. Let $p_U(\mathbf{x})$, $p_N(\mathbf{x})$, $p_S(\mathbf{x})$ represent the unlabeled, normal and sensitive data distribution, respectively. By using $p_N(\mathbf{x})$ and $p_S(\mathbf{x})$, $p_U(\mathbf{x})$ can be rewritten as follows:

$$p_U(\mathbf{x}) = \beta p_S(\mathbf{x}) + (1 - \beta) p_N(\mathbf{x}),$$

where β is the ratio of the sensitive data. Hence, the ELBO for the unlabeled data can be rewritten as:

$$\max_{\theta} \mathbb{E}_{p_U}[\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta)] = \max_{\theta} \{ \beta \mathbb{E}_{p_S}[\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta)] + (1 - \beta) \mathbb{E}_{p_N}[\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta)] \}.$$

Therefore, if all unlabeled data are assumed to be normal, the ELBO will be maximized even for the sensitive data included in the unlabeled data. This would inadvertently encourage the generation of sensitive data.

Meanwhile, our approach maximizes the ELBO for the normal data by using the unlabeled and sensitive data as follows:

$$\max_{\theta} (1 - \beta) \mathbb{E}_{p_N}[\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta)] = \max_{\theta} \{ \mathbb{E}_{p_U}[\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta)] - \beta \mathbb{E}_{p_S}[\mathcal{L}_{\text{DM}}(\mathbf{x}; \theta)] \}.$$

Therefore, our approach can maximize the ELBO for normal data without using labeled normal data.



Figure 3: Generated samples from fine-tuned stable diffusion models (middle aged man).

Unsupervised	Supervised	Proposed
0.80	0.84	0.99

Table 1: Comparison of non-sensitive rates for fine-tuned stable diffusion models.

A.6 STABLE DIFFUSION EXPERIMENTS

We conducted experiments using Stable Diffusion 1.4¹ with the following experimental settings. The objective was to ensure that specific individuals (in this case, Brad Pitt) are excluded when generating images of middle-aged men using Stable Diffusion. The dataset was prepared as follows:

- Unlabeled data: 64 images of middle-aged men and 16 images of Brad Pitt.
- Labeled sensitive data: 20 images of Brad Pitt.

These images were generated using Stable Diffusion with the prompts “a photo of a middle-aged man” and “a photo of Brad Pitt”. This experimental setup is an extension of (Heng & Soh, 2024) to the PU setting.

Using this dataset, we applied standard fine-tuning (unsupervised), supervised diffusion models as described in Section 3.1 (supervised), and the proposed method to Stable Diffusion 1.4.

As an evaluation metric, we use the non-sensitive rate described in Section 5.2, which represents the ratio of the generated samples classified as normal (non-sensitive) by a pre-trained classifier. For the pre-trained classifier, we use a ResNet-34 fine-tuned on 1,000 images each of middle-aged men and Brad Pitt, generated by Stable Diffusion. The accuracy on the test set for the classification task is 99.75%.

The experimental setup is almost the same as in Section 5.3, with a batch size of 16, a learning rate of 10^{-5} for the proposed method and 10^{-4} for the others, 1,000 epochs, and $\beta = 0.2$ for the proposed method.

We experimentally found that using the max-based loss function described in Section 3.2 for Stable Diffusion models caused divergence, when attempting to maximize $\mathcal{L}_{\mathcal{U}}^{-}(\theta) - \beta \mathcal{L}_{\mathcal{S}}^{-}(\theta)$ until it became non-negative. To prevent this, we instead employed the absolute value-based loss function, as also described in Section 3.2.

The experimental results for generating images of middle-aged men are provided in Figure 3, and non-sensitive rates are listed in Table 1. With the Unsupervised and Supervised methods, attempts to generate images of middle-aged men often result in the generation of Brad Pitt. This issue arises due to the presence of Brad Pitt in the unlabeled data. Meanwhile, the proposed method successfully suppresses the generation of Brad Pitt.

¹<https://huggingface.co/CompVis/stable-diffusion-v1-4>