Supplementary Material: Analysis of Master Vein Attacks on Finger Vein Recognition Systems

Huy H. Nguyen[1], Trung-Nghia Le[1,3,4], Junichi Yamagishi[1], and Isao Echizen[1,2]
[1]National Institute of Informatics, Tokyo, Japan [2]The University of Tokyo, Tokyo, Japan
[3]University of Science, VNU-HCM, Vietnam [4]Vietnam National University, Ho Chi Minh City, Vietnam
{nhhuy,jyamagis,iechizen}@nii.ac.jp

This Supplementary Material is organized as follows. First, we provide the detail of the latent variable algorithm (LVE) used to craft master veins in section 1. Next, we include some additional visualizations of real and generated finger veins in different settings in section 2. Last, we present an ablation study on top-k AdvML attack in section 3.

1. Latent Variable Evolution Algorithm

The LVE algorithm, which is used by the LVE-based methods to generate master veins (visualized in Fig. 3 in the main paper), is described in detail by Alg. 1. Regarding the implementation of the CMA-ES, we used the pycma library[1]. For simplicity, we used its default parameters.

Algorithm 1 Latent variable evolution.

$m \leftarrow 18$ ▷ Population size, default value by the pycma library.
procedure RUNLVE(m, n) ▷ m is population size and n is total number of iterations.
  MasterVeins = $\emptyset$ ▷ Master vein set.
  Scores = $\emptyset$ ▷ and corresponding score set.
  $\mathbf{z} \leftarrow$ rand() ▷ Initialize latent vectors $\mathbf{z} \in \mathbb{R}^m$.
  for n loops **do** ▷ Run LVE algorithm n times.
    $\mathbf{V} \leftarrow p_\theta(\mathbf{z})$ ▷ Generate m vein images V
    $\mathbf{s} \leftarrow 0$ ▷ Initialize scores $\mathbf{s} \in \mathbb{R}^m$.
    for each vein image V_i in $\mathbf{V}$ **do**
      for each vein image D_j in database $\mathbf{D}$ **do**
        $s_i \leftarrow s_i +$ Matcher(V_i, D_j) ▷ Calculate similarity score using matcher.
      $s_i \leftarrow \frac{s_i}{|\mathbf{D}|}$ ▷ Calculate the mean scores.
    $V_b, s_b \leftarrow$ GetBestVeinImage($\mathbf{V}, \mathbf{s}$) ▷ Identify local best master vein image.
    MasterVeins $\leftarrow$ MasterVeins $\cup \{V_b\}$
    Scores $\leftarrow$ Scores $\cup \{s_b\}$
    $\mathbf{z} \leftarrow$ CMA_ES($\mathbf{s}$) ▷ Evolve $\mathbf{z}$ on basis of $\mathbf{s}$.
  $V_{gb}, s_{gb} \leftarrow$ GetBestVeinImage(MasterVeins, Scores) ▷ Identify global best master vein image
  return V_{gb}, s_{gb} ▷ Best master vein and its score.

2. Additional Visualizations of Generated Finger Veins

Additional samples of real finger veins and those generated by the LVE-based methods are shown in Fig. 1. Effects of the number of iterations and filters used by the AdvML method on the quality of adversarial master veins are visualized in Figs. 2 and 3, respectively.

[1]https://github.com/CMA-ES/pycma

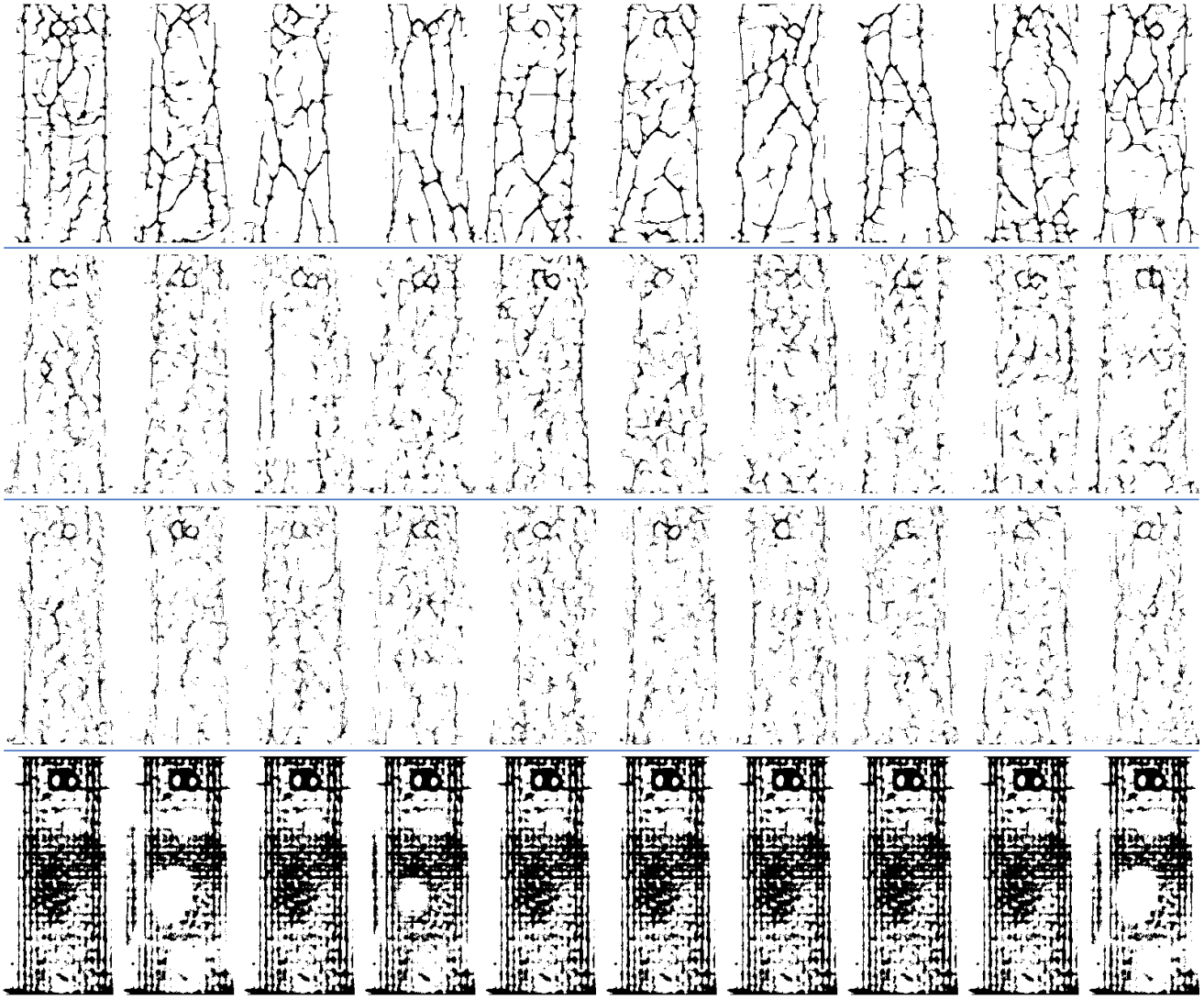
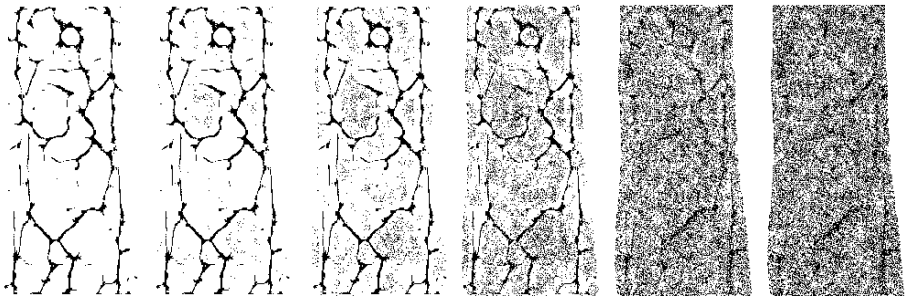
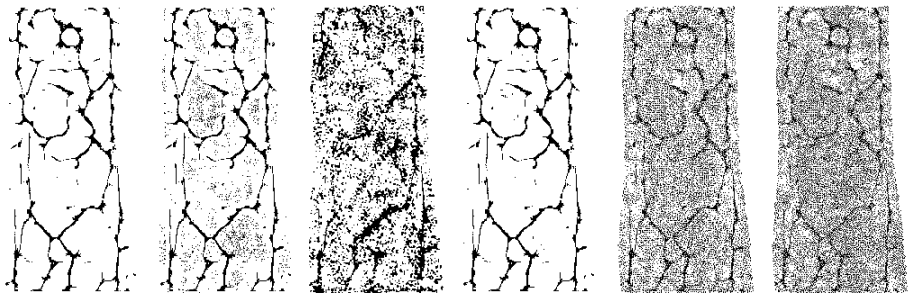


Figure 1. Examples of real finger veins (first row) and those generated by our proposed method (LVE[3], second row), β-VAE (LVE[2], third row), and WGAN-GP (LVE[1], last row). Since latent codes are sampling from noise, generated images have the randomness property. Among synthetic finger veins, those generated by our method had the best quality, while those generated by WGAN-GP do not look like finger veins.



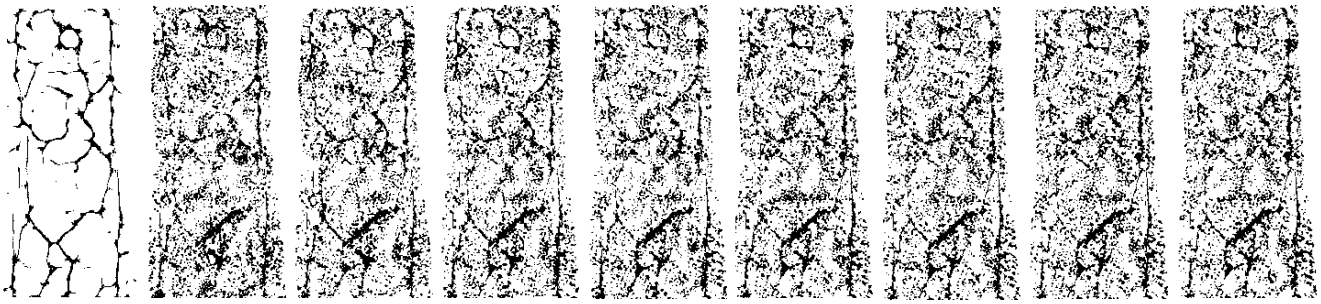
a. Original image | b. 50 iterations | c. 100 iterations | d. 200 iterations | e. 500 iterations | f. 1,000 iterations

Figure 2. Effect of the number of iterations on the quality of adversarial master veins.



a. Original image | b. No filter | c. Gaussian blur | d. Low-pass filter | e. High-pass filter | f. Laplacian filter

Figure 3. Effect of filters on the quality of adversarial master veins.



a. Original image | b. Top-2.5% | c. Top-5% | d. Top-10% | e. Top-20% | f. Top-40% | g. Top-60% | h. Top-80% | i. All labels

Figure 4. Adversarial master veins generated with different k values in top-k-label targeted attack.

3. Ablation Study on Top-k AdvML Attack

Examples of adversarial master veins generated with different k values in the AdvML attack with top-k labels are shown in Fig. 4. There are no significant differences in the amounts of perturbations between them. The relationship between k and FARs is visualized in Fig. 5. There are no significant differences in the FARs, especially when k is in the $[5\%, 60\%]$ range. In practice, it is better to avoid the extreme values of k. If k is too small, its attack ability is limited. If k is too large, it makes the optimization hard to converge.

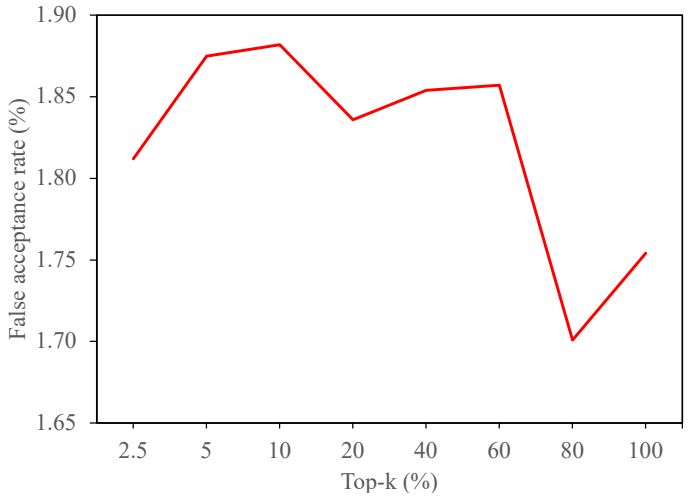


Figure 5. Relationship between k and FARs in top-k-label targeted attack. FARs were calculated using the ResNeXt-50-based recognition system on the training set of the SDUMLA-HMT database.