Finger Vein Spoof GANs: Are they really useful to enhance PAD training?

Tessnim Boulfoul, Valentin Pröpster, Andreas Vorderleitner and Andreas Uhl
Dept. of Artificial Intelligence and Human Interfaces
University of Salzburg (PLUS)
J.-Haringerstr.2, 5020 Salzburg, Austria
Email: andreas.uhl@plus.ac.at

Abstract—**Four traditional GAN-based I2I translation techniques have been employed for the synthesis of biometric finger vein presentation attack instrument (PAI) samples (three public presentation attack datasets have been considered). These synthetic samples have been used to train presentation attack detectors (PAD). This work considers the more realistic setting to augment PAD training sets with synthetic data, instead of entirely replacing real by synthetic samples, as done in earlier work. Our analysis reveals that uninformed usage of synthetic data in the considered context has to be avoided as it can lead to dramatically high APCER values. Instead, we recommend to use an augmentation of training data with synthetic samples only in case of too high BPCER values when training with real data alone.**

I. Introduction

Presentation attacks (PA) are typically conducted by presenting artefacts mimicking real biometric traits (aka "presentation attack instrument" (PAI)) to the biometric sensor to be deceived. Counter-measures to these types of attacks have of course already been developed and are termed "presentation-attack detection (PAD)" or "anti-spoofing" measures [1]. A comprehensive overview of PAD techniques for vascular data can be found in table 14.1 in [2]. More recent examples are e.g. [3] where a targeted fusion of recognition scheme results is used for PAD and [4] where a customised CNN is trained to detect PAI samples. Also, liveness detection measures can be used against PA, typically by analysing near-infrared videos (in the spectral domain of dorsal hand vein videos [5] or by applying a light vision transformer approach in the Gabor domain of finger vein videos [6]).

The last decade has brought forward several publications that presented ways to potentially fool finger vein-based authentication systems. In first attempts, vascular PAIs are generated as easily as printing a previously captured finger vein sample image on a piece of paper or on overhead projector foil and presenting this printout (eventually manually enhanced) to the sensor (see [7] for a review on these techniques). Current public datasets containing PAI sample data are based on this approach while more advanced techniques involving smartphone displays [7] or modelling finger properties using silicone or beeswax [8] have been developed. Obviously, the generation of PAI samples is tedious work: Generating printouts (manually enhanced) or physical models (in various materials, typically with attached printed vascular structures) and subsequent scanning with a target sensor is required to generate the forged sample data. As a consequence, available PAI sample datasets are of moderate size at best [3] which endangers a statistically relevant assessment of associated security risks. As a consequence, it is obvious to consider the synthesis of PAI sample data to provide sufficiently sized datasets. Note that PAI samples are used for two purposes mainly: First, to evaluate the threat posed by such artefacts used in a PA against a particular recognition scheme (vulnerability assessment), and second, to train PAD techniques designed for securing the biometric system [1].

In this work, we focus on the second application case, i.e. training PAD techniques using synthetic data. In earlier work [9], [10], finger vein PAI sample data originally synthesised to conduct a vulnerability assessment [11] has been shown to be appropriate in principle to train a PAD system as well (at least when choosing suited feature extraction schemes and GAN types [9], [10]). The synthetic data has been generated after training well-known classical image-to-image (I2I) translation GAN structures with data from public PAI sample datasets. In subsequent work [12], we have demonstrated that the usage of more recent synthesis methods (i.e. StyleSwin and DDPM) for generating PAI samples delivers competitive but nor superior results, probably caused by the limited amount of training data available.

The work in [9], [10] also reveals certain drawbacks when using synthetic PAI data in PAD training, in particular when training a two-class classifier with bona fide data (real data, class 1) vs. synthetic PAI data (GAN generated samples, class 2). We have found that depending on features and classifier configuration, the PAD classifier may learn two different things: To distinguish, as intended when using a PAD scheme, (i) bona fide samples from PAI samples (based on their visual differences), and to distinguish, as not intended, (ii) real from synthetic data (based on artefacts synthetic data typically contain, e.g. so-called model fingerprints [9]). The assessment of the PAD classifier is done with **real** datasets, i.e. both bona fide and PAI samples are actually acquired by a biometric sensor. In case the PAD classifier has indeed (unintentionally) learned to discriminate real from synthetic data, all data used in assessment is classified into class 1, failing to detect any PAI samples and causing extremely high error rates for these data. The work done so far investigates these effects for a PAD training process which **completely** replaces existing, *real* PAI samples with the synthetic ones in a PAD system [9], [10]. This scenario is not realistic, as real PAI samples are required to generate synthetic ones anyway, so why not using those being available in PAD training ? Thus, the more realistic setting is to augment existing real PAI samples with synthetic ones, i.e. to use a PAI sample training set that is a mix of real and synthetic PAI samples. This paper investigates the effects when applying this approach in finger vein PAD training (and assessment) and tries to answer the question if the usage of synthetic PAI data in a mixed training data set can avoid the pitfalls observed when entirely replacing real PAI data.

The rest of this manuscript is organised as follows: In Section II we define the experimental settings with respect to datasets, employed image synthesis methods and used evaluation metrics, experimental results are presented in Section III. The conclusion and outlook to future work is given in Section IV.

II. Experimental Settings

A. (Deep-Learning based) Synthesis of PAI Samples

A recent survey on synthetic biometric data [13] reveals, that synthetic generation of vascular data, in particular finger vein samples, has hardly been addressed before apart from own prior work. One of the few exceptions is [14], where it was shown that it is indeed possible to generate grey-scale vascular samples (finger vein as well as hand vein data) from corresponding binary features using a learning-based approach (template inversion). An entirely different way for finger vein sample synthesis, relying on a model-based approach, has been demonstrated in [15]. Still, there are a few further examples employing generative AI techniques: [16] proposed a GAN-based synthesis of a finger vein sample dataset based on the prior generation of a vein pattern image, while [17] applied an end-to-end GAN-based sample generation where the samples are used to augment the training set in deep-learning based finger vein recognition. Similarly, also [18] applies a (Cycle)-GAN-based finger vein sample synthesis approach to improve recognition.

In earlier work [11], the following I2I networks have been applied to generate the data used in this work as well: CycleGAN [19], DistanceGAN [20], DRIT [21], and StarGANv2 [22]. The network implementations made available by the authors of the original papers have been used to create the data. Samples have been fed into the networks in full size and slightly resized according to the networks need using bicubic (e.g. CycleGAN) or bilinear (e.g. DistanceGAN) interpolation. Augmentations are done within the network as supported, without any additional external augmentation.

Note that for vulnerability assessment (as originally intended), synthetic samples need to be suited for impersonation (i.e. synthetic PAI samples are confused with real samples of a person enrolled in a database by the recognition approach used by the biometric system), while for PAD training, synthetic PAI samples "only" need to look / behave similar to real PAI samples according to the employed PAD classifiers' perspective. Thus, the data sets synthesised in [11] for vulnerability assessment contain a synthetic PAI sample for each corresponding real PAI sample in the original database.

Data synthesis for these four networks has been done in a five-fold cross validation, i.e. for each configuration, five different network instances have been trained from scratch to generate their share of the final data. Fold construction prevents to have distinct samples of a single subject in both the training and evaluation sets, respectively (thus we separate subjects in training and evaluation data). The detailed description of which parameters have been used for each network can be retrieved from https://wavelab.at/sources/Vorderleitner23b/, based on which the synthetic data can be reproduced.

B. Datasets

The Idiap Research Institute VERA Fingervein Database (VERA) [23] consists of 220 unique fingers captured in 2 sessions from 110 subjects. Each sample has one PAI sample counterpart, which is generated by printing preprocessed samples on high quality paper using a laser printer and enhancing vein contours with a black whiteboard marker afterwards. Images come as full (250×665 pixels) or cropped (150×565 pixels) samples, overall we have 440 PAI samples.

The South China University of Technology Finger Vein Database (SCUT) [24] was collected from 6 fingers of 100 subjects captured in 6 acquisition sessions, overall we have 3600 PAI samples. For PAI sample generation, each finger vein image is printed on two overhead projector films which are aligned and stacked. In order to reduce overexposure, additionally a strong white paper is put in-between the two overhead projector films. Images come as full (640×288 pixels, we use those) or cropped samples of variable size.

The Paris Lodron University of Salzburg Finger Vein Spoofing Data Set (PLUS) [8] uses a subset of the PLUS Vein-FV3 dataset as bona fide samples. For PAI sample generation, principle curvature binarised vein structures from 6 fingers of 22 subjects were printed on paper and sandwiched into a top and bottom made of beeswax. Capturing is done employing two illumination variants (LED and Laser) and using two different levels of vessel thickness. Every sample is of size 192×736 pixels, using data from 3 sessions we have 396 PAI samples.

In Fig. 1 we display a pair of bona fide and PAI sample images, respectively, from each of the considered datasets. Note that for PLUS data, the two samples look rather differently (so a PAD detector should have an easy job to do), while for VERA and SCUT similarity is much higher. The PAI samples look much more blurred (and additionally exhibit larger areas of overexposure in case of VERA).

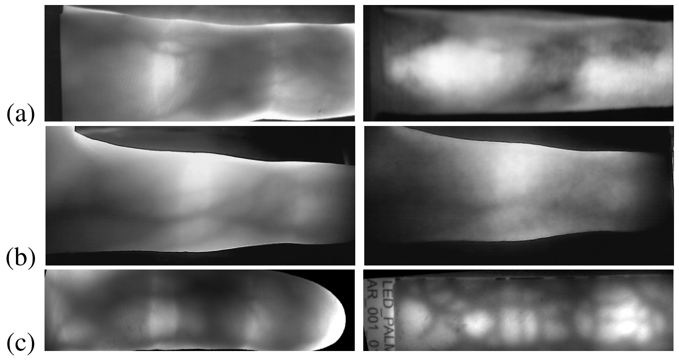

Fig. 1: A pair of bona fide and *real* PAI sample images, respectively, from the (a) VERA (b) SCUT, and (c) PLUS datasets.

C. Evaluation Methodology

Note that as the available datasets are rather limited in size, we always apply a five-fold cross validation for error estimation (folds are constructed by separating subjects to avoid subject related bias). We employ 396 PAI samples from each dataset only for a fair comparison with respect to training among the three datasets. Also, we always use balanced data in classifier training, i.e. both classes consist of the same number of samples and we employ the corresponding bona fide and real/synthetic PAI samples from the same subject.

PAD Testing: As the aim is to simulate "realistic" PAD, in all cases we test the ability of the trained PAD classifier to discriminate between bona fide and *real* PAI samples (as taken directly from the datasets). Thus, while the testing is always done in the same manner (considering real-life application), the training is different according to what we mean to determine.

PAD Training: For "BaseLine" results (BA), we train the classifier to discriminate between bona fide and *real* PAI samples as provided by the datasets, i.e. all real PAI samples are used. In five-fold cross validation this obviously means that for each fold used in testing, the remaining four folds are used in training. To create more challenging settings (to even better motivate the usage of synthetic data), we introduce training scenarios **S1** and **S2** as follows (both scenarios involve real PAI samples only): In S1 we reduce the size of the training set to 50% of the BA training. In five-fold cross validation this means that for each fold used in testing, only two (randomly chosen) folds out of the remaining four folds are used in training. In S2 we further reduce the size of the training set to only 25% of the BA training. In five-fold cross validation this means that for each fold used in testing, only one (randomly chosen) fold out of the remaining four folds is used in training. For assessing the suitedness of synthetic data, we train the classifier to discriminate between bona fide samples and a mixture of real and synthetic PAI samples, respectively. Training scenario **S3** increases the size of the training set to 50% of the BA training by adding synthetic data to the training set used in scenario S2. In five-fold cross validation this is done by replacing one of the two folds used in S1 training by synthetic PAI data (which one is selected is randomly chosen). This also implies that the size of the training sets in S3 and S1 is identical, they differ in their composition only. Consequently, training scenario **S4** increases the size of the training set to 100% of the BA training by adding more synthetic data to the training set used in scenario S3. In five-fold cross validation this is done by replacing three of the four folds used in BA training by synthetic PAI data (which ones are selected is randomly chosen). Finally, training scenario **S5** uses synthetic PAI sample data only but the same training data size as scenarios BA and S4, respectively, and matches the training strategy investigated in earlier work [9], [10].

PAD Assessment: To evaluate the effectiveness of the proposed PAD approach, results are reported in compliance with ISO/IEC 30107-3:2017. Since a presentation attack detection mechanism is a binary classifier, four outcomes are possible: correctly classified as attack (true positives TP), wrongly classified as attack (false positive FP), correctly classified as bona fide (true negative TN) and wrongly classified as bona fide (false negative FN). According to the standard, we report Attack Presentation Classification Error Rate (APCER - proportion of PAI attack presentations incorrectly classified as bona fide presentations) and Bona Fide Presentation Classification Error Rate (BPCER - proportion of bona fide presentations incorrectly classified as presentation attacks)

$$APCER = \frac{FN}{FN + TP} \quad , \quad BPCER = \frac{FP}{FP + TN} \ .$$

PAD Classifier: In order to complement the convolutional neural networks (CNNs) used in earlier work [9], [10], we consider pre-trained CNN architectures with a systematically decreasing number of learn-able parameters (and a lower number of network layers in many cases) to study the corresponding impact on PAD training accuracy: VGG16 [25] with ≈138 mio. parameters, TinyNet-A [26] with ≈ 6.2 mio. parameters, MobileNetV2 [27] with ≈ 3.4 mio. parameters, and SqueezeNet 1.1 [28] with ≈ 1.2 mio. parameters. The models pre-trained on ImageNet have been taken from Kaggle (MobileNet and VGG16), Huggingface (Tinynet) and github (SqueezeNet). For the performance assessment in PAD training, we use the networks in two ways: First, to match the approach followed in [10], we remove the classification network part and use the pre-trained network as a feature extractor, the resulting features are used in a KNN (K-nearest neighbour, K being optimised) classifier to facilitate straightforward comparison to [10]. Second, the classification network part is configured to support binary classification and is "fine-tuned" in the training process by analogy to [9]. Binary cross entropy is used as loss function and with respect to optimisers, the best choice between SGD and Adam optimisation is employed.

III. Experimental Results

For each of the three datasets and the four GAN-types, we present a visual example of a synthetic PAI sample for qualitative analysis in Fig. 2. When comparing to Fig. 1, we observe that the PAI samples from the SCUT dataset are difficult to synthesise properly, as except for the CycleGAN result, the samples lack in clear vascular structure. The DRIT data look rather disappointing overall, as even the PLUS data clearly lack in detail (which is rather prominent and much better generated by the other GAN-types).

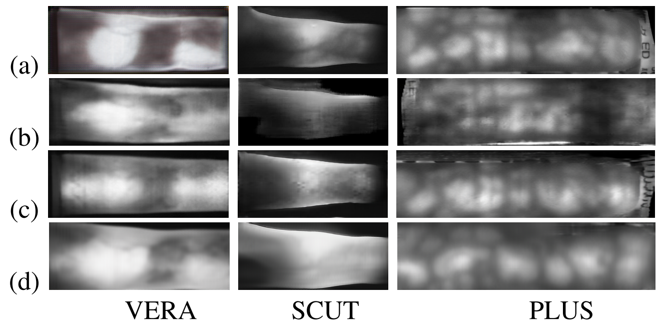

Fig. 2: Example synthetic PAI sample images: (a) CycleGAN (b) DistanceGAN (c) StarGANv2 (d) DRIT

In Table I we present quantitative PAD results in terms of APCER in % obtained for training scenario S2 (lowest amount of data used) for all CNN variants and the three datasets considered. Only APCER values exceeding 0.5% are listed. In the exponent, we provide the order-number(s) of the training scenarios S3 - S5 for which an improvement in terms of APCER as compared to S2 is observed (i.e. the exponent shows, for which training setting adding synthetic data is beneficial). Is the number given in bold, all GANs improve APCER, otherwise only a subset of the GANs used led to an improvement.

We observe that out of 24 configurations (8 PAD classifiers, 3 datasets), only 9 exhibit APCER in excess of 0.5%. From these 9 cases, 5 did not benefit at all from adding additional synthetic training data (no matter how much was added). From those which took benefit, only a single configuration improved for S3, S4, and S5 as well and in most of these cases, only a subset of GANs led to improved results. It is interesting to observe that S5 improved over S2 in two cases only, and this is only seen for a subset of GANs.

TABLE I: Quantitative results: S2 Results exceeding 0.5% APCER, and stages S3 - S5 improving this result when adding synthetic PAI samples (given in the exponent).

Architecture	**VERA**	**SCUT**	**PLUS**
TinyKNN	$9^{\mathbf{345}}$	5	10
Tiny	2	1	4^{34}
SqueezeKNN	–	–	–
Squeeze	–	–	–
MobileKNN	3^{5}	3	–
Mobile	–	–	–
VGGKNN	–	–	2^{34}
VGG	–	–	–

There are 4 PAD classifiers without any APCER value exceeding 0.5%, so overall we observe fairly good results. Fine-tuning the classification network is obviously superior to using the network output as feature vector in KNN-classification. TinyNet-A is clearly the least suited architecture to handle the classification task at hand.

In Table II we present the analogous PAD results in terms of BPCER in % obtained for training scenario S2 (lowest amount of data used) for all CNN variants and the three datasets considered. The results have a different characteristic than those for APCER. We observe that out of 24 configurations, 12 exhibit BPCER in excess of 0.5%. From these 12 cases, all did benefit from adding additional synthetic training data (no matter how much was added). From those which took benefit, a majority improved for S3, S4, and S5 as well and in a majority of these cases, all considered GANs led to improved results. It is also interesting to observe that, contrasting to increased APCER values, S5 improved over S2 in all relevant configurations.

There are 3 PAD classifiers without any BPCER value exceeding 0.5%, so again, overall we observe fairly good results. Fine-tuning the classification network is confirmed to be superior to using the network output as feature vector for KNN-classification. TinyNet-A is also clearly confirmed to be the least suited architecture to handle this classification task. To summarise, wisely chosen CNNs can cope with training scenario S2 quite well, such that the addition of synthetic data to augment training is not required. Higher APCER values are much more difficult to improve by adding additional synthetic training data as compared to higher BPCER values. Using exclusively synthetic PAI samples in training (scenario S5) can hardly improve high APCER values, while this approach works well for high BPCER values.

TABLE II: Quantitative results: S2 Results exceeding 0.5% BPCER, and stages S3 - S5 improving this result when adding synthetic PAI samples (given in the exponent).

Architecture	**VERA**	**SCUT**	**PLUS**
TinyKNN	$5^{\mathbf{345}}$	$16^{\mathbf{345}}$	$3^{\mathbf{35}}$
Tiny	1^{5}	2^{345}	1^{345}
SqueezeKNN	–	$8^{\mathbf{345}}$	–
Squeeze	–	–	–
MobileKNN	$2^{\mathbf{45}}$	$1^{\mathbf{345}}$	$3^{\mathbf{345}}$
Mobile	–	–	–
VGGKNN	$1^{\mathbf{45}}$	$9^{\mathbf{345}}$	–
VGG	–	–	–

In Tables III - V we take a different perspective. For each dataset and training scenario separately, we present APCER and BPCER values exceeding 4.5%. This is done in two groups: BA, S1, and S2 form the first group, S3 - S5 form the second group. A group covers one line and it is only listed in case of at least one training scenario in that group leads to those higher APCER / BPCER values (i.e. the more lines we see, the weaker are the PAD results). If different GANs exhibit distinct behaviour, they are listed separately, if they show the same tendency, a APCER / BPCER range for all of them is provided.

The VERA results shown in Table III exhibit only two BA, S1, S2 groups, i.e. those results generated by TinyKNN. For those, APCER is improved over S2 for all GANs using S3 and S4, but significantly worsened using S5 (for three out of four GANs). On the other hand, BPCER is improved over S2 results for all GANs only in case of S5. For the other groups shown, adding synthetic samples in training worsens the results, partially significantly so (as the first group results are not displayed). S5 results (i.e. training with synthetic PAI samples only) are the worst in many cases. We see more APCER results being listed, and GANs exhibit similar behaviour in many cases. The two VGG-16 variants and MobileNetv2 results do not show up at all.

SCUT results are displayed in Table IV. Similar to VERA results, the TinyKNN results cover both groups and APCER as well as BPCER. Using additional synthetic data worsens the APCER results (in particular the S5 results are very poor), but improves BPCER results. For the other PAD classifier networks, we only observe results for the first group for BPCER, which are obviously improved using additional synthetic data (as those lines are not listed). For APCER, only results for the second group are present, i.e. showing results which are worsened from unproblematic results to partially very poor APCER, especially for the S5 training setting. We see more APCER results being listed, and again GANs exhibit similar behaviour in many cases. MobileNetv2 results do not show up at all.

For the PLUS dataset, results are shown in Table V. It

TABLE III: Quantitative results: VERA (BaseLine and training with five training set variants, S3 - S5 involving a growing number of GAN synthesised PAI samples).

Architecture	**Error**	**Synthesis**	BA	S1	S2	S3	S4	S5
TinyKNN	APCER	–	-	6	9			
		Cycle				-	5	19
		Dist				-	7	10
		DRIT				-	-	20
	BPCER	–	-	-	6			
		Cycle				8	-	-
		DRIT				9	5	5
		Star				9	8	-
Tiny	APCER	All				-	6-8	40-67
SqueezeKNN	APCER	All				-	-	37-97
Squeeze	APCER	All				3-8	-	7-81
MobileKNN	APCER	All				5	8	2-7
	BPCER	Cycle				-	-	9

TABLE IV: Quantitative results: SCUT (BaseLine and training with five training set variants, S3 - S5 involving a growing number of GAN synthesised PAI samples).

Architecture	**Error**	**Synthesis**	BA	S1	S2	S3	S4	S5
TinyKNN	APCER	–	-	6	5			
		All				13-17	19-25	71-99
	BPCER	–	9	14	16			
		All				6-8	-	-
Tiny	APCER	All				10-13	13-20	53-82
SqueezeKNN	APCER	All\Cycle				-	-	57-98
	BPCER	–	-	-	8			
Squeeze	APCER	Dist				23	-	21
		DRIT				21	-	40
		Star				-	-	48
MobileKNN	APCER	All				9	10	10
VGGKNN	BPCER	–	-	9	9			
VGG	APCER	DRIT				7	-	-

is interesting to observe that only APCER lines are present (although the PAI samples are so different from bona fide ones for this dataset), and results for the first group are again only seen for TinyKNN. Adding synthetic data improves these results for S4 only, S3 and S5 lead to very poor results. For the other networks, again S5 leads to very poor results. GANs exhibit similar behaviour in many cases, MobileNet(KNN) results as well as VGG results do not show up at all.

To summarise, the TinyNet architecture turns out to perform very poorly in both configurations, MobileNetv2 with fine-tuned classification network is not listed for any dataset, thus is very stable. We observe more APCER lines being listed (thus indicating poor results), but in most cases the additional synthetic data do not improve previously poor APCER results or worsen results not being problematic on real data only. S5 results are often particularly poor in these cases.

IV. Conclusion

We have found that adding synthetic PAI sample data to PAD training datasets is by no means suited to be applied as a general strategy. First, by a sensible selection of the PAD classifier (e.g. MobileNetV2 is a perfect choice in our case) the PAD errors can be kept low even in case of small training set size. Second, the role synthetic data can play, depends on the nature of the error considered: Higher APCER values are difficult to improve with additional synthetic data in training, and using synthetic data without need often worsens results (the worst case is often seen when relying on synthetic PAI sample data in training exclusively, as done in previous work [9], [12]). On the other hand, high BPCER values can be improved using additional synthetic data in training, often using synthetic data exclusively is a good choice. The type of GAN used for sample synthesis plays a minor role.

References

[1] S. Marcel, J. Fierrez, and N. E. (Eds.), *Handbook of Biometric Anti-Spoofing: Presentation Attack Detection and Vulnerability Assessment*. Springer, 2023.

[2] J. Kolberg, M. Gomez-Barrero, S. Venkatesh, R. Ramachandra, and C. Busch, *Presentation Attack Detection for Finger Recognition*. Cham: Springer International Publishing, 2020, pp. 435–463.

TABLE V: Quantitative results: PLUS (BaseLine and training with five training set variants, S3 - S5 involving a growing number of GAN synthesised PAI samples).

Architecture	Error	Synthesis	BA	S1	S2	S3	S4	S5
TinyKNN	APCER	–	5	6	10			
	APCER	All				12-27	3-5	12-70
Tiny	APCER	Dist				5	6	20
SqueezeKNN	APCER	All\Cycle				-	5	79-100
Squeeze	APCER	All\Cycle				-	-	18-100
VGGKNN	APCER	All				-	-	15-43

[3] J. Schuiki, M. Linortner, G. Wimmer, and A. Uhl, "Attack detection for finger and palm vein biometrics by fusion of multiple recognition algorithms," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 4, no. 4, pp. 544 – 555, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9914644

[4] K. Shaheed, A. Mao, I. Qureshi, Q. Abbas, M. Kumar, and X. Zhang, "Finger-vein presentation attack detection using depthwise separable convolution neural network," *Expert Systems with Applications*, vol. 198, p. 116786, 03 2022.

[5] J. Schuiki and A. Uhl, "Improved Liveness Detection in Dorsal Hand Vein Videos using Photoplethysmography," in *Proceedings of the IEEE 19th International Conference of the Biometrics Special Interest Group (BIOSIG 2020)*, Darmstadt, Germany, 2020, pp. 57–65.

[6] L. Chen, T. Guo, L. Li, H. Jiang, W. Luo, and Z. Li, "A finger vein liveness detection system based on multi-scale spatial-temporal map and light-vit model," *Sensors*, vol. 23, no. 24, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/24/9637

[7] R. Raghavendra and C. Busch, "Presentation attack detection algorithms for finger vein biometrics: A comprehensive study," in *11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS'15)*, 2015, pp. 628–632.

[8] J. Schuiki, B. Prommegger, and A. Uhl, "Confronting a variety of finger vein recognition algorithms with wax presentation attack artefacts," in *Proceedings of the 9th IEEE International Workshop on Biometrics and Forensics (IWBF'21)*, Rome, Italy, 2021, pp. 1–6.

[9] M. Langer, M. Hafner, S. Findenig, A. Radovic, A. Vorderleitner, and A. Uhl, "Difficulties in using synthetic data for presentation attack detection in finger vein recognition: The role of model fingerprints," in *2024 IEEE/IAPR International Joint Conference on Biometrics (IJCB)*, 2024, pp. 1–10.

[10] A. Vorderleitner and A. Uhl, "Finger vein spoof gans: Issues in presentation attack detector training," in *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25)*. Catania, Italy, April 2025: ACM, New York, NY, USA, 2025, pp. 751–758.

[11] A. Vorderleitner, J. Hämmerle-Uhl, and A. Uhl, "Finger Vein Spoof GANs: Can we Supersede the Production of Presentation Attack Artefacts?" in *22th International Workshop on Digital Forensics and Watermarking (IWDW'23)*, ser. Springer LNCS, vol. 14511, Jinan, China, 2023, pp. 109–124.

[12] A. Vorderleitner and A. Uhl, "Finger Vein Spoof GANs: Is Synthesis Using Diffusion or VisionTransformer Superior for Presentation Attack Detector Training?" in *25th International Conference on Digital Signal Processing (DSP 2025)*, 2025, pp. 1–5.

[13] A. Makrushin, A. Uhl, and J. Dittmann, "A survey on synthetic biometrics: Fingerprint, face, iris and vascular patterns," *IEEE ACCESS*, vol. 11, pp. 33 887–33 899, 2023.

[14] C. Kauba, S. Kirchgasser, V. Mirjalili, A. Uhl, and A. Ross, "Inverse biometrics: Generating vascular images from binary templates," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, no. 4, pp. 464–478, 2021.

[15] F. Hillerström, A. Kumar, and R. Veldhuis, "Generating and analyzing synthetic finger vein images," in *Proceedings of the International Conference of the Biometrics Special Interest Group (BIOSIG'14)*, Sep. 2014, pp. 121–132.

[16] H. Yang, P. Fang, and Z. Hao, "A gan-based method for generating finger vein dataset," in *Proceedings of the 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence*, ser. ACAI '20. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3446132.3446150

[17] J. Zhang, Z. Lu, M. Li, and H. Wu, "Gan-based image augmentation for finger-vein biometric recognition," *IEEE Access*, vol. 7, pp. 183 118–183 132, 2019.

[18] W. Yang, C. Hui, Z. Chen, J.-H. Xue, and Q. Liao, "Fv-gan: Finger vein representation using generative adversarial networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2512–2524, 2019.

[19] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2242–2251, iSSN: 2380-7504.

[20] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 752–762.

[21] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European Conference on Computer Vision ECCV'18*, 2018, pp. 36–52.

[22] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition CVPR'20*, 2020, p. 8188–8197.

[23] P. Tome, R. Raghavendra, C. Busch, S. Tirunagari, N. Poh, B. H. Shekar, D. Gragnaniello, C. Sansone, L. Verdoliva, and S. Marcel, "The 1st competition on counter measures to finger vein spoofing attacks," in *International Conference on Biometrics (ICB'15)*, May 2015, pp. 513–518.

[24] X. Qiu, W. Kang, S. Tian, W. Jia, and Z. Huang, "Finger vein presentation attack detection using total variation decomposition," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 2, pp. 465–477, 2018.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Prooceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*. Computational and Biological Learning Society, 2015, pp. 1–14.

[26] K. Han, Y. Wang, Q. Zhang, W. Zhang, C. Xu, and T. Zhang, "Model rubik's cube: twisting resolution, depth and width for tinynets," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. Red Hook, NY, USA: Curran Associates Inc., 2020.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, " MobileNetV2: Inverted Residuals and Linear Bottlenecks ," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2018, pp. 4510–4520.

[28] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360