GAN-BASED FINGER VEIN (FV) IMAGE AUGMENTATION FOR BIOMETRIC AUTHENTICATION

Author : Samuel Dubuis
Supervisor : Hak Gu Kim
Image and Visual Representation Lab
School of Computer and Communication sciences
École Polytechnique Fédérale de Lausanne

Master semester project report
Lausanne, EPFL, January 2021

Acknowledgements

We would like to thank a few people in particular who made this project possible.
First off, Dr. Lambert Sonna who as head of Global ID welcomed me with open arms and created an opportunity for me to work on a topic that was dear to me. He never hesitated to provide me with technical resources or information about his technology.
Thank you also to Hak Gu Kim, even through a complicated semester due to the COVID19, we were able to meet weekly and your input, theoretical and technical, was precious and often necessary.
Finally, thank you to Professor Dr. Sabine Süsstrunk for welcoming me into her lab once again and for supervising me from afar.

Lausanne, 15 January 2021 S. D.

Abstract

Startup Company “Global ID", Swiss startup specialized in cyber-security through high level of security and confidentiality, and fighting against identity thief has developed a 3D finger vein biometric identification system.

To improve and verify the performance of their biometric identification system, they have to get a large number of the finger vein images in person. However it is very cumbersome, time consuming, and labor intensive to collect a large-scale dataset from subjective experiments. To address that, in this project, we will generate a large number of finger vein images from a small number of the real finger vein images using deep generative model. For this purpose, we will study the generative adversarial network (GANs). Furthermore, we will design a GAN-based data augmentation algorithm for synthesizing realistic finger vein images.

GANs are a type of machine learning, more precisely deep learning, framework where two different neural networks compete against each other. The generator tries to produce a result which the discriminator evaluates until a point where the generator is able to fool its adversary.

At first we use a spatial transformer network to improve classification and processing of the data. Then we use a cycleGAN in order to augment the dataset. In the cycleGAN part we have added to our framework an additional loss based on the classification of the finger veins to improve the result images.

Finally the full framework with tunable parameters was implemented, with default values having been pre-optimized.

Contents

List of Figures

List of Tables

1 Introduction

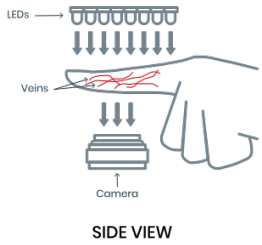
In order to fight against nowadays modern threats like identity theft, data theft or cyber-criminality the need for a tamper-proof identification solution has quite grown over the years. To respond to that need, Global ID startup has developed the first biometric identification system based on the finger veins in three dimensions. The finger veins network of an individual is unique and can not be replicated, thus being a great medium to identify someone.

1.1 Global ID

The Global ID startup, based at the Innovation park at EPFL and founded by Dr. Lambert Sonna, is a Swiss startup specialized in cyber-security. The first idea was to create a system to identify holder of diplomas from emerging countries, especially from Africa. From that idea was born the project of a reliable system of identification based on finger veins.

Fingerprints now have flaws and are becoming easier to forge. There exists possibilities to buy false fingerprints for cheap on black markets, etc... As previously mentioned, finger veins were defined as a great way to identify someone. Finger veins biometric has multiple advantages like : uniqueness of each person's finger veins, its resistance to forgery and replication, particularly as it's hidden from eye sight, and finally it is also untraceable as there is nearly no contact between the user and the scanner.[1] Based on that knowledge, Global ID developed a biometric scanner that takes infrared pictures of a finger at three different angles and recreates a 3D image of the venous network. For the remainder of that report, we will abbreviate finger veins with FV.

Figure 1.1 – Pipeline of the Global ID FV matching



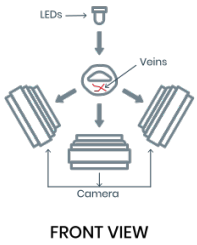


Figure 1.2 – Side and front view of the scanner

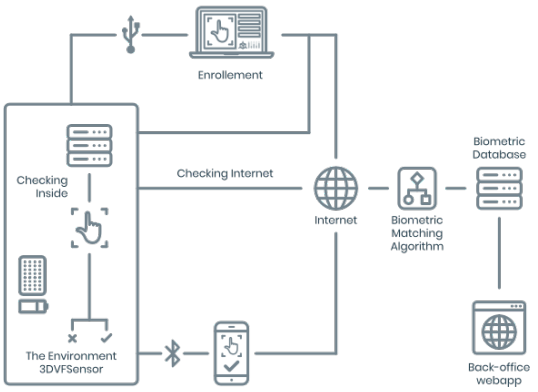


Figure 1.3 – Full framework with servers

This scanner was conceived in partnership with the research institute Idiap in Martigny, Valais, with the HES-SO School and with the EPFL. The images taken by the scanner are end-to-end encrypted and the scanner itself is authenticated on a server, therefore preventing the reconstructed image to be revealed in whole, hiding the identity of the user and allowing to match the finger veins network with the database. This technology was patented by the startup.

The scanner opportunities are various and Global ID has developed a full enrollment and matching process, as shown in figure 1.3. A few examples of usage of this scanner and technology are ID verification in airport, smartphone unlocking, coded doors opening and payment verification.

1.2 Problems and motivation

Having a performing system of matching FV to a database requires that the said database is big enough. The bigger it is, the more data the system has to train on and to perform faster and better.

In the same way, we can define a few problems in this field that became our motivations for this project:

- The importance of diverse training dataset in deep learning for biometric authentication
  - Deep learning methods can demonstrate state-of-the-art results for biometric authentication. However, the performance and the robustness of these methods depend highly on the availability of a diverse training dataset to learn different users' attributes and capturing conditions such as shape or luminance characteristics.

- The difficulty of a large scale data collection for biometric authentication
  - A large scale dataset collection is time consuming and complicated. It requires a variety of biometric patterns from each person, also in different conditions. Scanning by hand hundreds or thousands of fingers would take a lot of time itself already, and now added the different parameters and setup, the job becomes overwhelming.
- The necessity of data augmentation in the domain of finger veins authentication
  - In the particular field of FV images, there are uncontrollable factors such as environmental illumination, light scattering or misplaced fingers in the scanner. Augmenting the data by anticipating those parameters becomes a necessity to counter them.

1.3 Contribution

Having been given the opportunity by Global ID to tackle the problem of augmenting their own dataset, we defined a few points as our contribution to this field and project. Global ID already has a basic dataset consisting of 440 images of 110 subjects, four FV images for each participant. This dataset was produced by the Idiap research institute

1.3.1 Generative Adversarial Network

First would be the actual generation of a large number of FV images from a small starting dataset, with the help of generative adversarial networks (GANs). GANs are a class of machine learning, more particularly deep learning frameworks that consists of a generator and a discriminator. The goal of the generator is to produce results which will be evaluated by the adversary, the discriminator. It works in an unsupervised manner, the generator and the discriminator being updated dynamically until the generator is able to fool the discriminator, resulting in very good outputs to the user if the network worked well.

Based on [2] and [3], we can define precisely the functioning of a GAN. They are generative models that learn a mapping from random noise vector z to an output image y, $G : z \rightarrow y$ [4]. In contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to an output image y, $G : x, z \rightarrow y$. The generator G is trained to produce outputs that cannot be distinguished from “real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator’s “fakes". This training procedure is diagrammed in Figure 1.4, from [5].

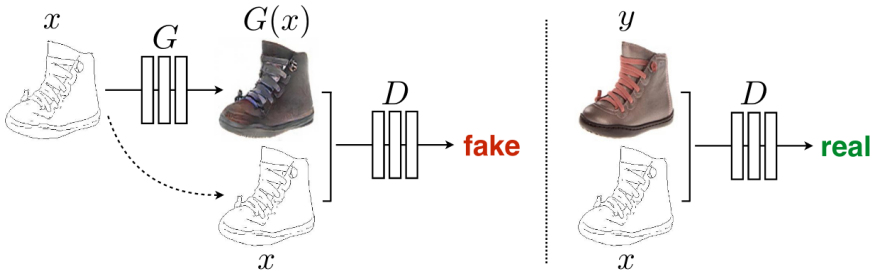


Figure 1.4 – Taken from [5]. Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real edge, photo tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

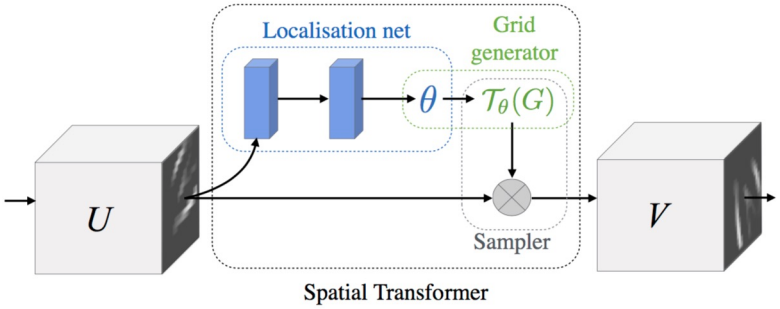


Figure 1.5 – Spatial Transformer Network

1.3.2 Spatial Transformer Network

A second part of our contribution would be a spatial transformer network. The idea behind this particular network is to consider the spatial variation and illumination variation during the FV images acquisition.

The principles of a spatial transformer network, as explained in the original paper [6], is to give neural networks the ability to actively spatially transform feature maps, conditional on the feature map itself, without any extra training supervision or modification to the optimisation process.

Both figures 1.5 and 1.7 show the spatial transformation idea and the framework of the STN. This network is able to learn to automatically apply transformations on distorted images via classification task. The resulting images, spatially transformed will then be inputs of the previously mentioned GAN.

It is also worth mentioning the mathematical aspect of the spatial transformation applied in this network. As explained briefly in the caption of figure 1.7, we are interested by the the sampling grid which is the result of warping the regular grid with an affine transformation $\mathcal{T}_\theta(G)$. The said affine transformation is displayed below in figure 1.6.

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix}$$

Figure 1.6 – Affine transformation $\mathcal{T}_\theta$

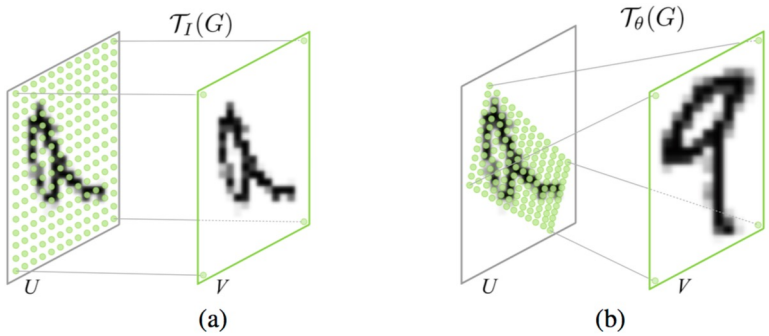


Figure 1.7 – (a) is when the sampling grid is a grid where the transformation parameters used are the identity ones. (b) is the sampling when the transformation parameters used are those of an affine one.

1.3.3 Integration of the algorithm

The final part of our contribution is the full integration of our framework and the algorithm behind it to the startup pipeline. In-depth details are that the augmented data, meaning the images produced from our combination of STN and GAN, will be useful to the startup. Useful means that more than producing decent images, resembling original FV images, they have to be able to help the startup's system to be more efficient and precise. This is achieved by reducing the time taken to match the user to the database, and also reducing the potential errors. We would not want to have errors or false positives when authenticating someone for an access to a restricted area for security reasons, for example.

2 Related work

2.1 Deep learning based FV authentication

There are several existing works on FV image feature representation based on deep learning.

Deep Representation-Based Feature Extraction and Recovering for Finger-Vein Verification by Huafeng Qin and Mounim A. El-Yacoubi [7] proposes a first deep learning model to extract and recover vein features using limited a priori knowledge. In three steps, they first identity regions by segmentation to only process the interesting part, then they use a convolutional neural network for classification of the FV. Finally, they recover and reconstruct potential missing veins of the image. Particularly, they use an interesting way of using and splitting two databases in their work, which can be used in our case.

FV-GAN : Finger Vein Representation Using Generative Adversarial Networks by Wenming Yang, Changqing Hui, Zhiquan Chen, Jing-Hao Xue, and Qingmin Liao [8] proposes the first attempt of FV extraction and verification based on generative adversarial networks. They designed an adversarial training framework and an hybrid loss function. It is really interesting and obviously useful for our work, and hinted us the idea of multiple losses.

However, when taking these articles into consideration, we can notice that most experiments that were conducted had a limited number of FV images. In our work we explicitly want to augment this number of FV images in order to produce better results. The related work here gave us multiple ideas and hints on how to improve their work and our project.

2.2 Data augmentation for FV authentication

Finger-vein Recognition Based on Densely Connected Convolutional Network Using Score-Level Fusion with Shape and Texture Images by Kyoung Jun Noh, Jiho Choi, Jin Seong Hong and Kang Ryoung Park [9] presents a recognition system for FV with a deep learning, fully connected framework.

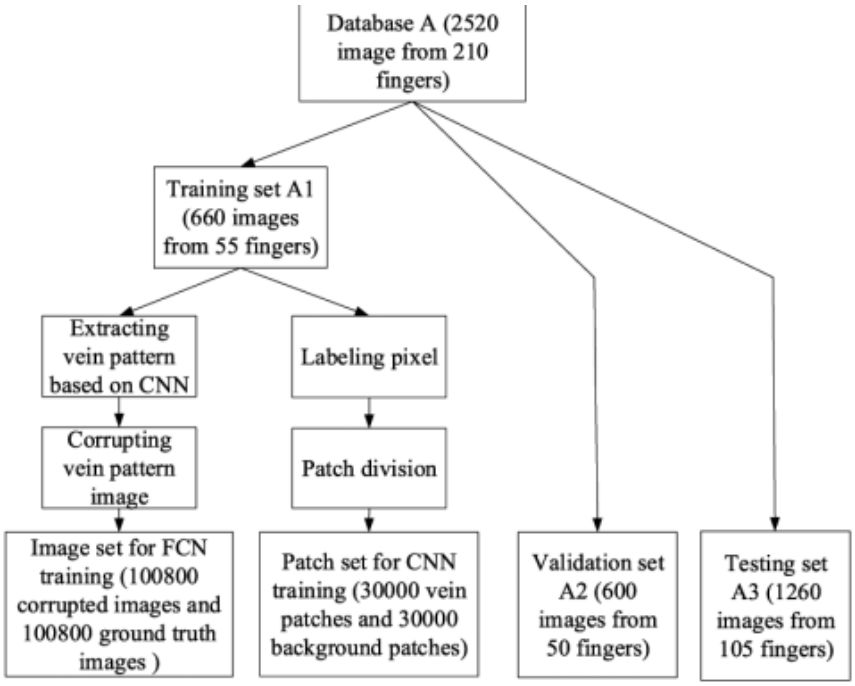


Figure 2.1 – Taken from [7]. Example of database splitting

Finger-Vein Pattern Restoration with Generative Adversarial Network by Shuqiang Yang, Huafeng Qin, Xia Liu and Jun Wang [10] proposes a FV images restoration framework to optimize the authentication systems, based on GANs. They use a segmentation algorithm to extract veins which are then used in the generative model.

Those papers here show ways to improve the authentication systems used for FV such as GAN models and recognition algorithms, or through the restoration of faulty images. However, we found no research on a data augmentation algorithm to improve FV authentication.

2.3 Other useful papers

We took into consideration other articles that we will list here. They brought us more general knowledge, details on FV, on GANs and finally on the system used by the Global ID startup for authentication.

- BioLocker: A Practical Biometric Authentication Mechanism based on 3D Fingervein by F. Betül Durak, Loïs Huguenin-Dumittan, and Serge Vaudenay [11], Global ID paper about their work in the FV biometric authentication.
- Data augmentation using learned transformations for one-shot medical image segmentation by Amy Zhao, Guha Balakrishnan, Frédo Durand, John V. Guttag and Adrian V. Dalca [12].

- Pathological Retinal Region Segmentation From OCT Images Using Geometric Relation Based Augmentation by Dwarikanath Mahapatra, Behzad Bozorgtabar, Jean-Philippe Thiran and Ling Shao [13] which particularly gave us the idea for the use of a spatial transformer network.
- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks by Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A. Efros [14] for the Cycle GAN implementation.
- Spatial Transformer Networks by Max Jaderberg, Karen Simonyan, Andrew Zisserman and Koray Kavukcuoglu [6], first paper written about those transformer networks.

3 Implementation

After having done some research, shown and explained in the previous chapters, we went on to the implementation of the framework that was discussed and decided as the best fit.

At first, we will explain each component separately, and in the last section they will all come together in order to run as one complete framework. For information, the code was written in Python3, and some of the most often used libraries were `TensorFlow`, `Numpy`, `cv2` `TensorLayer` and some others self made or found on the internet.

3.1 Data

In this section we talk about the data part of the project. The databases used, given, acquired and how they were processed.

3.1.1 Databases

After surveying the web and contacting some people in other universities mainly, we collected a few databases in addition of the one we already had access to. We will now present them.

- The Idiap Research Institute VERA Fingervein Database [15]
  - The VERA Fingervein Database for fingervein recognition consists of 440 images from 110 subjects. It was created by the Idiap Reseach Institute and is the main database currently used by the startup Global ID.
- Finger Vein USM (FV-USM) Database [16]
  - The images in the database were collected from 123 volunteers. Every person provided four fingers, for 492 classes of fingers. Finally each FV was taken six times in the first session, and another six times during a second session. This makes a total of 5904 images, or two times 2952, for a training and testing sets.

- Tsinghua University Finger Vein and Finger Dorsal Texture Database (THU-FVFDT) [17]
  - This database consists of three smaller ones. Two were interesting for our project, one consisted on the dorsal texture of fingers which was of no use for us. It totaled 1220 FV images for 610 subjects, divided into two sessions, one for a training set, one for a testing set.

As the original database of the startup, the VERA FV database, isn't big enough and we are planning to augment it, we focused our work on the second one, the Finger Vein USM (FV-USM) Database. The last database was lacking some information and some part of the FV images were of no interest to us, therefore we discarded it.

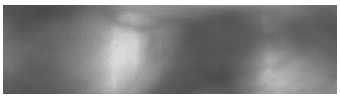


Figure 3.1 – Example image of the Idiap Research Institute VERA Fingervein Database

Figure 3.2 – Example image of the Finger Vein USM (FV-USM) Database

Figure 3.3 – Best example image of the Tsinghua University Finger Vein and Finger Dorsal Texture Database

3.1.2 Data processing

Now we are going to talk about the way we processed this data. As a remainder, we were working with 5904 FV images. They were split into two folders for the two sessions, and in each we had 492 folders for each person with their id, or label, as names and six images in those last folders for the six takes.

We had to create a personalised loader as the data wasn't structured efficiently for us, and that loader would also read the file names to know the labels that will be used later in the classification.

We only took into account one channel, as we work with gray images, when loading images, and we casted them to `float32`. Images were of height 300 pixels and width 100 pixels. We

didn't resize them during the spatial transformer part, but they were resized and zero-padded to 256 by 256 pixels when going into the GAN, as they work better and the settings were pre-made for this size of images.

3.2 Spatial Transformer Network

The first module of the framework we implemented, after the loader, is the Spatial Transformer Network. As cited in [13] and in [6], spatial transformer networks (STN) can help us spatially place and transform our FV images, based on scale, location and orientation.

The STN implementation was inspired by some pre-made code from TensorLayer. They have a module called `SpatialTransformer2dAffine`, that we included in our neural network. This one is described as such :

1. Localisation network
    - Flatten layer for the image input
    - Fully connected layer from flat layer to 20 nodes, tanh activation
    - Dropout of 20%
2. Spatial Transformer module
    - `SpatialTransformer2dAffine` module with 20 nodes as input, and a matrix, image, as output
3. Classifier
    - 2D Convolution layer with 16 filters and ReLu activation, one channel as input
    - 2D Convolution layer with 16 filters and ReLu activation, 16 channels as input
    - Flatten layer
    - Fully connected layer from flat layer to 1024 nodes, ReLu activation
    - Fully connected layer from previous dense layer to 123 nodes, identity activation (123 nodes because there are 123 possible labels with the database we chose)

In order to find the hyper-parameters of our STN, we ran a grid search. As we can see in the tables 3.1, 3.2, 3.3, 3.4 and 3.5, we obtain the best results for the classification after our images have been through the spatial transformer network for 300 epochs, a batch size of 32 and a learning rate, as predicted because it's the same as the authors of the paper [6], of 0.0001. Some plots are also displayed to show the training and validation accuracy and loss evolution in figures 3.4 and 3.5.

Batch size	Learning rate							
	0.0001		0.001		0.01		0.0005	
1	7.902	0.315	4.813	0.008	4.824	0.008	4.812	0.008
4	3.514	0.465	4.812	0.008	4.816	0.008	5.224	0.140
16	4.200	0.303	5.095	0.053	4.814	0.008	6.134	0.241
32	2.476	0.466	4.883	0.135	4.813	0.008	8.067	0.238
64	2.625	0.387	5.134	0.199	4.813	0.007	4.213	0.125

Table 3.1 – Epoch 50 (test loss & test accuracy)

Batch size	Learning rate							
	0.0001		0.001		0.01		0.0005	
1	4.812	0.008	4.813	0.008	4.825	0.008	4.812	0.008
4	2.870	0.610	4.812	0.008	4.816	0.008	6.324	0.031
16	3.208	0.572	4.812	0.005	4.815	0.008	4.862	0.510
32	2.080	0.714	4.812	0.008	4.813	0.008	6.580	0.281
64	1.806	0.693	8.755	0.359	4.813	0.007	4.633	0.214

Table 3.2 – Epoch 100 (test loss & test accuracy)

Batch size	Learning rate							
	0.0001		0.001		0.01		0.0005	
1	4.812	0.008	4.813	0.008	4.825	0.008	4.812	0.008
4	3.706	0.556	4.812	0.008	4.816	0.008	5.732	0.557
16	2.893	0.648	4.812	0.008	4.814	0.008	6.009	0.466
32	2.107	0.662	6.607	0.417	4.814	0.008	4.812	0.008
64	1.482	0.742	9.604	0.272	4.813	0.007	5.767	0.483

Table 3.3 – Epoch 150 (test loss & test accuracy)

Batch size	Learning rate							
	0.0001		0.001		0.01		0.0005	
1	5.292	0.553	4.813	0.008	4.824	0.008	4.812	0.008
4	4.185	0.578	4.812	0.008	4.816	0.008	4.812	0.008
16	2.534	0.637	4.812	0.008	4.814	0.008	4.812	0.008
32	1.793	0.721	5.042	0.516	4.813	0.008	3.739	0.594
64	1.587	0.75	4.812	0.007	4.813	0.007	4.221	0.547

Table 3.4 – Epoch 200 (test loss & test accuracy)

Batch size	Learning rate							
	0.0001		0.001		0.01		0.0005	
1	4.812	0.008	4.812	0.008	4.824	0.008	4.812	0.008
4	4.444	0.637	10.430	0.009	4.816	0.008	4.812	0.008
16	2.085	0.720	4.812	0.008	4.814	0.008	12.221	0.203
32	**1.770**	**0.765**	4.812	0.008	4.813	0.008	8.723	0.380
64	1.784	0.737	4.813	0.007	4.813	0.007	4.579	0.586

Table 3.5 – Epoch 300 (test loss & test accuracy)

Figure 3.4 – Training/Validation loss and accuracy over training of STN - Bad results (Batch size 16, epochs 50, learning rate 0.01)

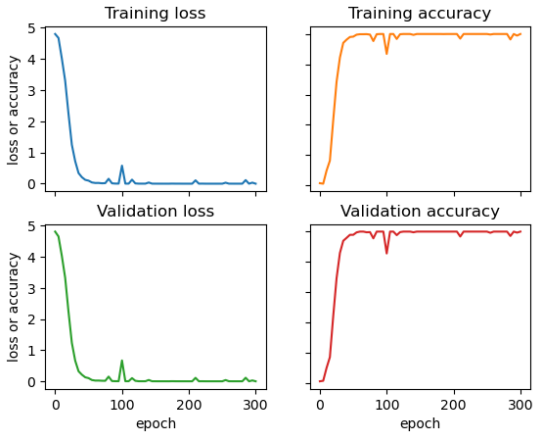


Figure 3.5 – Training/Validation loss and accuracy over training of STN - Good results (Batch size 32, epochs 300, learning rate 0.0001)

During the training of the STN, following a parameter `print_freq`, we display the results of the training loss and the training accuracy. At that time, we also save some sample spatially transformed FV images. See figure 3.6.

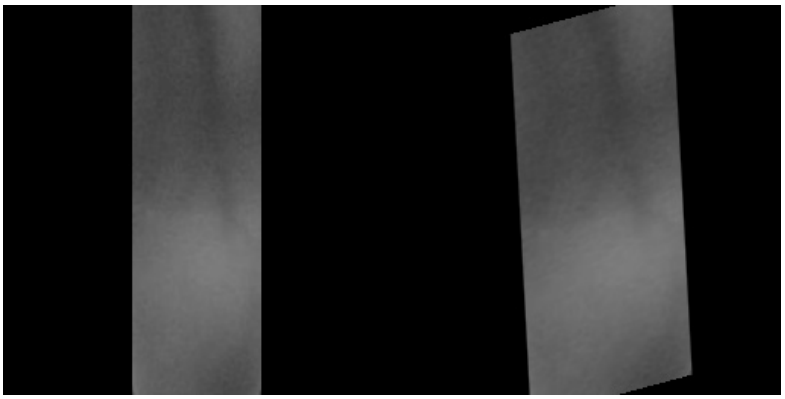


Figure 3.6 – Before and after spatial transformation of FV image - As can be seen, the image is slightly angled as the network must have find it the best spatial transformation for this type of images.

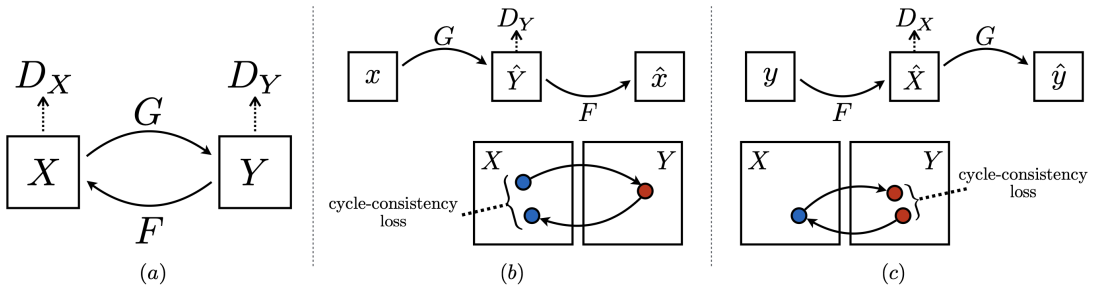


Figure 3.7 – Figure taken from [14]. CycleGAN concept and mapping

The final part of the STN is the application of the spatial transformation to the full data. Folders are created in a way that the next module, the GAN can consider it best, and each image is passed through the STN and saved in those said folders.

3.3 CycleGAN

The generative adversarial network module was the main time taking part of the implementation of our project. Running a full training to achieve comparable results often took a couple days.

We chose the model of cycleGAN [14] as a generator.

3.3.1 Model

Our approach is based on cycleGAN [14] and the associated code, information were also from [2]. The intuition behind the cycleGAN paper is the addition of a cycle-consistent loss, which enables the pairing of an inverse mapping $F : B \rightarrow A$ with the mapping $G : A \rightarrow B$ found in the classic GAN mode. This allows to enforce $F(G(A)) \approx A$ (and vice versa).

In practice, it means that the network learns the mapping from domain A to domain B at the same time as the mapping from B to A. Therefore, it needs two generators and two discriminators, making it quite a heavy GAN to run.

The figure 3.7 shows that the model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and the associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs that are indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, [14] introduces two cycle consistency losses that capture the intuition that if CycleGAN translates from one domain to the other and back again, it should arrive where it started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$.

The full objective function used is

$$\mathscr{L}(G,F,D_X,D_Y) = \mathscr{L}_{GAN}(G,D_Y,X,Y) + \mathscr{L}_{GAN}(F,D_X,Y,X) + \theta\mathscr{L}_{cyc}(G,F) \tag{3.1}$$

where $\mathscr{L}_{GAN}$ is the adversarial loss, and $\mathscr{L}_{cyc}$ is the cyclic loss described before.

3.3.2 Our usage

Our definition of the cycleGAN consisted of multiple layers. It was implemented with two ResNets of nine blocks for both generators, and two convolutional networks for both discriminators. When the images for both domains were loaded, they were pre-processed. CycleGAN works best with 256 by 256 pixels images as explained earlier. Therefore we resize them, but to not loose information by distortion, we decide to zero pad them to the side so that the ratio of content remains the same. They're also normalized.

In this project, to improve the generation of FV images, we contributed by adding another loss to those already implemented, the cyclic loss and normal adversarial one. Our idea was to compute the classification loss from the original image and the generated image and use it as addition to the two previously mentioned in equation 3.1. We use the classifier from the STN, after having removed the zero-padding.

$$\mathscr{L}(G,F,D_X,D_Y) = \mathscr{L}_{A2B\_GAN} + \mathscr{L}_{B2A\_GAN} + \theta\mathscr{L}_{cyc}(G,F) + \delta(\mathscr{L}_{A2B\_C} + \mathscr{L}_{B2A\_C}) \tag{3.2}$$

where $\mathscr{L}_{A2B\_GAN}$ is the adversarial loss, $\mathscr{L}_{cyc}$ and finally $\mathscr{L}_{A2B\_C}$ is the new classification loss.

Finally, we can run the cycleGAN training with two domains loaded, and those specific parameters that we considered optimal as the authors had already had computed them in [14]:

- Batch size of 5
- Epochs 200
- Learning rate 0.0002
- Cycle loss weight 10
- Classification loss weight 0.1

As in the spatial transformer network, we print the results of the losses at a certain frequency and we also save some samples at those times, that we can see in figures 3.8. A is the original image, in domain A. B is the style we want to apply to the image, still an original image in domain B. A2B means image in domain A transformed to domain B. The rest is self explanatory. As we can see, we wanted to change the illumination of our FV images in order to augment them, and when focusing on image A2B we see that results are quite good.

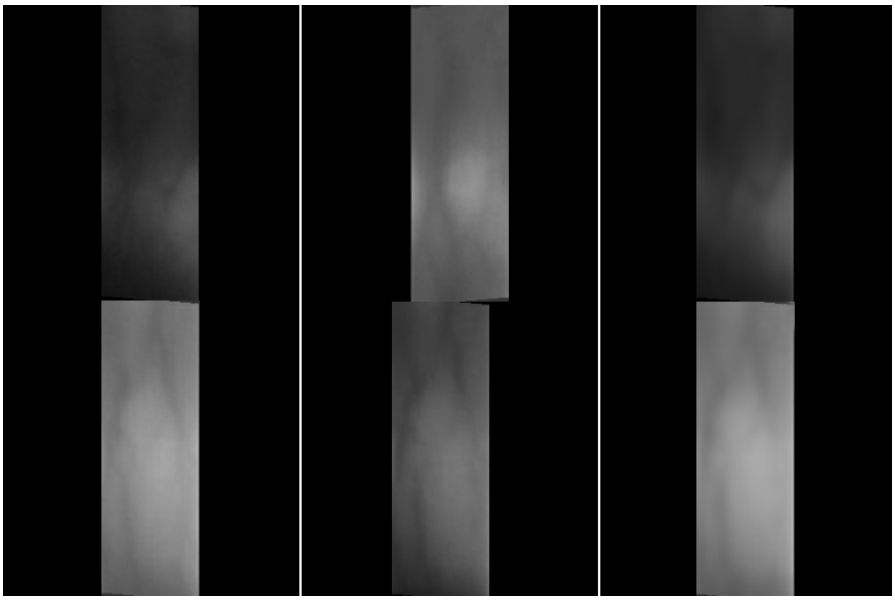


Figure 3.8 – Sample images during training of cycleGAN
Top A-A2B-A2B2A
Bottom B-B2A-B2A2B

3.4 Full framework

Now that we know about individual modules, we can explain the full concept of our framework. We can see in figure 3.9 how exactly it works.

First we begin by loading the images which will go trough the spatial transformer network. To spatially transformed images then go through a function called `create_domainB` in charge of, as named, creating the domain B of the cycle GAN. For this, we select 100 images at random and through TensorFlow function, adjust their brightness to a brighter level. However it can be adjusted to any value for further augmentation of the data. Those illumination transformed FV images are put in their respective folders for the cycleGAN to work.

Then the cycleGAN starts training. As explained, it generates fake images that the discriminator tries to classify as real or fake until the discriminator is fooled. We have added a supplementary loss in order to improve further the generated results, and particularly so that these results can be useful for classification later on.

The final part is to reload the full database and make it go through the cycleGAN generator to get a secondary database. When combined, we get an augmented database, which we train and test on the same first classifier taken from the STN.

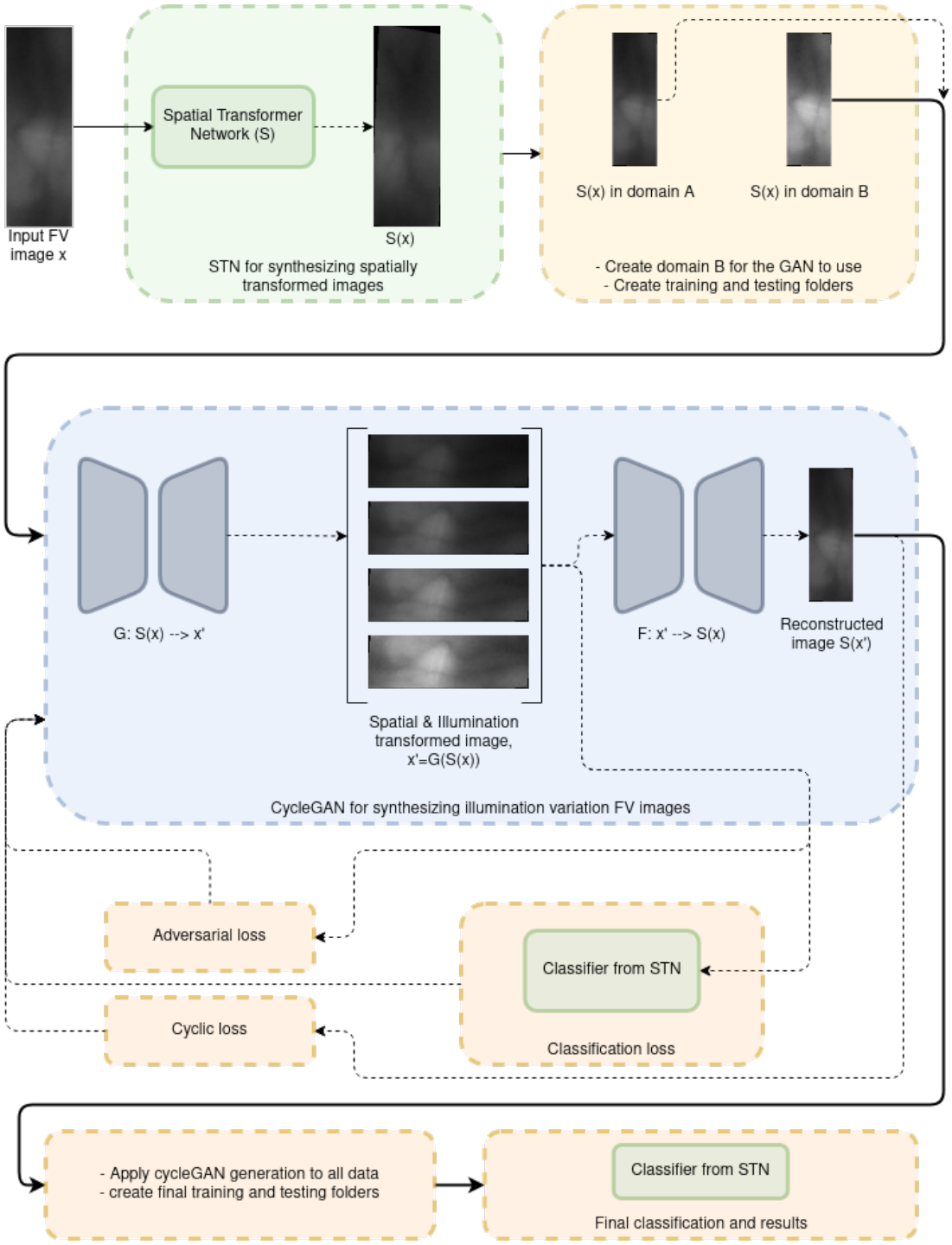


Figure 3.9 – Schema of the whole framework of the project

4 Results

In this chapter we will talk about the results we got with the framework previously explained. Some important results were already displayed in the implementation chapter, under the Spatial Transformer Network section. We achieved the best classification accuracy (0.765) after the FV images went through the STN with parameters batch size 32, learning rate 0.0001 and 300 epochs. See table 3.5. We can also take a look at the training accuracy and loss during STN training, figure 3.5.

The next result we got was the classification loss and accuracy after having generated new images and trained our classifier with them. Those results varies obviously depending on the run and on the style the generator learns to apply. However, we generally get an improvement of 15 to 10%, computed after multiples runs. For example, classification accuracy went from 0.625 to 0.714, 0.640 to 0.704, 0.681 to 0.715 and the best improvement was 0.576 to 0.736, an improvement of 27%! We have also noticed that augmenting the database simply with STN output also augment the accuracy.

An another important result to mention is the creation of a fully functional code, and an implemented generative adversarial networks.

Finally and as result of the project's main goal, we have been able to double the size of the database used as input. We create the domain B of the cycleGAN generator by modifying the FV image illumination. Therefore, by changing the value of illumination transform, we could once again double the database size to improve accuracy during testing. We could also apply different kind of basic transformation, without any knowledge to augment the database like rotation, flipping and scaling of the FV images.

5 Conclusion

In this final chapter we conclude our master semester project. We have seen that our full framework achieve satisfactory results and that these are, most important, useful to improve classification for authentication.

It is worth noting a few problem encountered during implementation, and to finalize, discuss some openings and further ideas for the future of this project.

5.1 Problems encountered

The first main problem was that this project occurred during the Covid19 worldwide pandemic. It was sometimes very difficult to discuss, exchange ideas or points of view and organize virtual meeting. This pandemic also influenced greatly one aspect of the project that was supposed to happen near the end, when the implementation was achieved. That goal was to conduct a data collection. This meant setting up a desk, with a scanner provided by the Global ID startup. This would have complemented their already existing database, and would have given a lot of FV images to train our framework on. But as social interactions became impossible, this never happened.

Another problem, however less annoying, was access to computational resources. As a student of EPFL and working for the IVRL, we had access to clusters of GPUs that allowed us to run our models and train them. However, those clusters were sometimes crowded, sometimes down and it would cancel already ongoing training or at least slow them down.

5.2 Openings and ideas

Potential ideas and further openings for this project we thought of will be discussed here.

First, the main addition that could improve the cycleGAN generator is the addition of yet another loss. The Global ID startup has its own finger veins extractor. It could be used in the

framework (figure 3.9) in parallel to the classification loss we implemented during this project. This additional loss could help further improve the details of the FV images generated, making them always better for authentication.

Another idea we had was the use of a different generative adversarial network instead of cycleGAN. The cycleGAN main concept, the cyclic loss, should be kept. But instead of working on a single domain to transform the images, starGAN [18][19] allows the parallel transformation toward five different domains. Therefore, instead of augmenting by two the size of the database, we could potentially augment it by five in single runs.

Another improvement in the training of the full framework would be the use of multiple databases. As seen in the Related Work chapter, [7] proposes to use a couple databases, and from these create training inputs for the STN and for the GAN, as well as mixing in order to create the validation and testing set. It is a point worth remembering to augment the database, at least during training, and then use the actual useful database for testing, for example.

Finally, the full code can be further optimized. As of yet, it works but improvements can still be made to obtain better results in classification or in generation. Also tweaking parameters and the code could make the generator act more stably, we sometimes happened to have GAN failures. GAN failures happen when the generator finds a way to fool the discriminator, and all images then generated look the same and are most of the time useless.

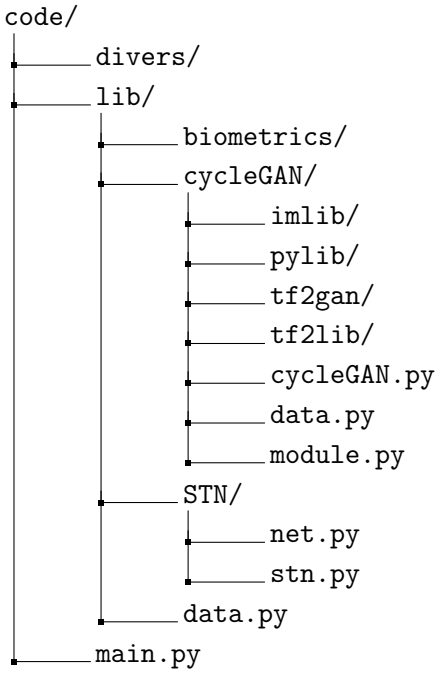
A Run the framework

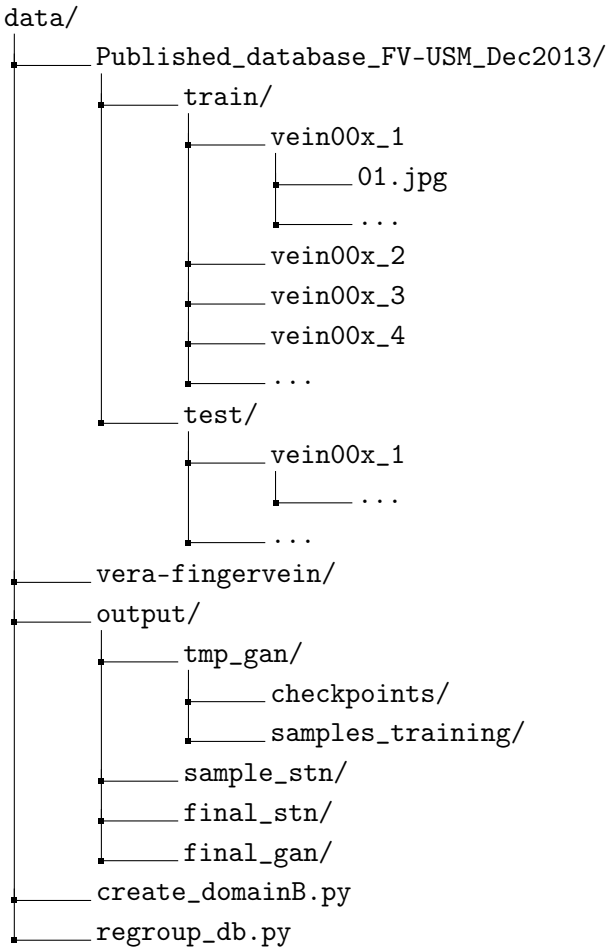
The framework was written in Python3. The code can be found and downloaded at : https://github.com/samdubuis/ma-semester-project

Here are the dependencies that were used and must be installed. : `tensorflow`, `opencv-python`, `tensorlayer`, `os`, `numpy`, `argparse`, `pathlib`, `matplotlib` and `tqdm`.

A.1 Files organization

The framework is organized by multiple folders. The interesting ones for running it are `code/` and `data/`. Below is a representation of useful folders and files, as the GitHub repository also contains documents helpful for the writing of this report, as well as tests and ideas :





The `code/` folder contains all of the implementation. The `lib/` folder contains the library written. `biometrics/` is the folder with the FV extraction code from the startup, not added to the framework. `cycleGAN/` contains all of the code for the cycleGAN part, the file `cycleGAN.py` will be called from a main file to run it all. Finally, the `STN/` folder contains, as named, the file describing the spatial transformer network and the `stn.py` file which will be also called from a main file.

In the `data/` folder there are the folders contains the actual databases, and two files that helped for processing all of the images.

A.2 Database adaptation

In order to have a functional framework, the database should be a folder in the `data/` folder, containing itself two folders `train/` and `test/`. The loader will probably have to be updated as it was made to work with the [16] database (`Published_database_FV-USM_Dec2013/`). The loader is found in the `data.py` file under the `lib/` folder. The loader available takes the directory path and outputs a TensorFlow dataset of images and labels. The values are

hard-coded and therefore should be modified to work with any database. Of course, the new loader must be added and replace the actual one over the files.

The FV-USM database can be downloaded and processed by following the [16].

A few modification must be made when working with a personal database.

- In the `net.py` file
  - at line 15, the `in_channels` value must be the width multiplied by the height of the images
  - at line 20, the `out_size` value must also be adapted to height and width of image
  - at line 26, the same as at line 15 must be done
  - at line 27, the `n_units` must be the number of classes or labels

A.3 Running the code

The `main.py` file is the one that's ran. It is ran with the following code input, while being in the `code/`folder :
`python3 main.py`

The help is displayed with :
`python3 main.py -h`

And this is the ouput:

```
1   -h, --help                show this help message and exit
2   --phase PHASE             Train or test ? (default: train)
3   --data_dir DATA_DIR       Directory path of data (default: ../data/)
4   --data_name DATA_NAME     Data name (default: Published_database_FV-USM_Dec
5                             2013)
6   --output_dir OUTPUT_DIR   Directory where samples are stored (default: output/)
7   --brightness BRIGHTNESS   Value for brightness adjustment when creating second
8                             domain (default: 0.2)
9   --stn_epoch STN_EPOCH     Number of epochs for the STN (default: 300)
10  --lr_stn LR_STN           Learning rate for the STN (default: 0.0001)
11  --batch_size_stn BATCH_SIZE_STN
12                            Batch size for STN (default: 32)
13  --print_freq PRINT_FREQ   Frequency of printing info, the higher the less
14                            (default: 10)
15  --batch_size_gan BATCH_SIZE_GAN
16                            Batch size for GAN (default: 5)
17  --gan_dir GAN_DIR         Directory path for the GAN part (default: tmp_gan)
18  --resize RESIZE           Size to which images are resized and then used
19                            (default: 256)
20  --gan_epoch GAN_EPOCH     Number of epochs for the GAN (default: 200)
```

The main arguments that can and should be modified to use with one's database are :

- `-data_name` the name of the database folder in `data/`
- `-brightness` the value, between 0 and 1, that changes the illumination transformation of the images
- `-print_freq` the value at which information of the STN is displayed, the higher the value, the rarer the information

Details of ongoing training and evaluation will be printed in the terminal. At the end, the newly generated images will be found under `data/output/final_gan/` as well as other useful information and partial samples in `data/output/`, especially checkpoints for the GAN that takes long to train. If the line 74 `cycleGAN.run()` is commented in the main file and there exist checkpoints, the GAN training will be skipped, saving plenty of time.

Bibliography

[1] A. H. Mohsin, A. A. Zaidan, B. B. Zaidan, O. S. Albahri, S. A. Bin Ariffin, A. Alemran, O. Enaizan, A. H. Shareef, A. N. Jasim, N. S. Jalood, M. J. Baqer, A. H. Alamoodi, E. M. Almahdi, A. S. Albahri, M. A. Alsalem, K. I. Mohammed, H. A. Ameen, and S. Garfan, "Finger vein biometrics: taxonomy analysis, open challenges, future directions, and recommended solution for decentralised network architectures," *IEEE Access*, vol. 8, pp. 9821–9845, 2020. DOI: 10.1109/ACCESS.2020.2964788.

[2] M. Bickel, S. Dubuis, and S. Gachoud, *Multiple generative adversarial networks analysis for predicting photographers' retouching*, 2020. arXiv: 2006.02921 [cs.CV].

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, 2016. arXiv: 1611.07004 [cs.CV].

[6] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, *Spatial transformer networks*, 2016. arXiv: 1506.02025 [cs.CV].

[7] H. Qin and M. El Yacoubi, "Deep representation-based feature extraction and recovering for finger-vein verification," *IEEE Transactions on Information Forensics and Security*, vol. PP, pp. 1–1, Mar. 2017. DOI: 10.1109/TIFS.2017.2689724.

[8] Y. Wenming, 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. PP, pp. 1–1, Mar. 2019. DOI: 10.1109/TIFS.2019.2902819.

[9] K. Noh, J. Choi, J. Hong, and K. Park, "Finger-vein recognition based on densely connected convolutional network using score-level fusion with shape and texture images," *IEEE Access*, vol. PP, pp. 1–1, May 2020. DOI: 10.1109/ACCESS.2020.2996646.

[10] S. Yang, H. Qin, X. Liu, and J. Wang, "Finger-vein pattern restoration with generative adversarial network," *IEEE Access*, vol. PP, pp. 1–1, Jul. 2020. DOI: 10.1109/ACCESS.2020.3009220.

[11] F. Durak, L. Huguenin-Dumittan, and S. Vaudenay, "Biolocker: a practical biometric authentication mechanism based on 3d fingervein," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 453, 2020.

[12] A. Zhao, G. Balakrishnan, F. Durand, J. V. Guttag, and A. V. Dalca, *Data augmentation using learned transformations for one-shot medical image segmentation*, 2019. arXiv: 1902.09383 [cs.CV].

[13] D. Mahapatra, B. Bozorgtabar, J.-P. Thiran, and L. Shao, *Pathological retinal region segmentation from oct images using geometric relation based augmentation*, 2020. arXiv: 2003.14119 [eess.IV].

[14] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, 2017. arXiv: 1703.10593 [cs.CV].

[15] M. Vanoni, P. Tome, L. El Shafey, and S. Marcel, "Cross-database evaluation with an open finger vein sensor," in *IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications (BioMS)*, Rome, Italy, Oct. 2014. [Online]. Available: http://publications.idiap.ch/index.php/publications/show/2928.

[16] M. S. Mohd Asaari, S. A. Suandi, and B. A. Rosdi, "Fusion of band limited phase only correlation and width centroid contour distance for finger based biometrics," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3367–3382, 2014, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.11.033. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417413009536.

[17] W. Yang, C. Qin, and Q. Liao, "A database with roi extraction for studying fusion of finger vein and finger dorsal texture," in *Biometric Recognition*, Z. Sun, S. Shan, H. Sang, J. Zhou, Y. Wang, and W. Yuan, Eds., Cham: Springer International Publishing, 2014, pp. 266–270, ISBN: 978-3-319-12484-1.

[18] Y. Choi, M.-J. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: unified generative adversarial networks for multi-domain image-to-image translation," *CoRR*, vol. abs/1711.09020, 2017. arXiv: 1711.09020. [Online]. Available: http://arxiv.org/abs/1711.09020.

[19] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, *Stargan v2: diverse image synthesis for multiple domains*, 2019. arXiv: 1912.01865 [cs.CV].