Use of autoencoders for fingerprint encoding and comparison

Wouter Pool

Abstract—**Deep learning methods for fingerprint comparison are starting to outperform classic, minutiae-based fingerprint comparison methods. This research explored the possibility of deep learning fingerprint comparison in two steps. First, an autoencoder was used to create a latent vector representation of a fingerprint segment. Three different autoencoders were created, one with a loss function which focuses more on fingerprint minutiae, one which emphasizes frequency components that normally occur less often in autoencoder reconstruction and one which is trained with normal mean square error. Secondly, a fully connected neural network was used to perform fingerprint comparison using the latent vectors of the autoencoders. The fingerprint comparison results show that despite some limitations in the training process and fingerprint data, it is possible to compare fingerprints using the latent space. It also shows that the classification results will improve with the different loss functions implemented in this paper. The most effective method was increasing the loss of the autoencoder at the minutiae locations.**

keywords: *Autoencoder, Deeplearning, neural network, fingerprint comparison, fingerprint recognition*

Introduction

Fingerprint recognition is currently one of the most used biometric modalities[1]. It finds its application in law enforcement, crime scene analysis and national security, but also in access to personal devices[2]. Its popularity depends on factors like the ease of acquisition, acceptance of acquisition and the amount of biometric data per individual.

To compare 2 fingerprints, most algorithms make use of minutiae comparison. Minutiae are easily distinguishable features of the fingerprint. Three examples of minutiae relevant to this paper can be found in Figure 1. The lines in the fingerprint are referred to as ridges and the empty space between them as valleys.

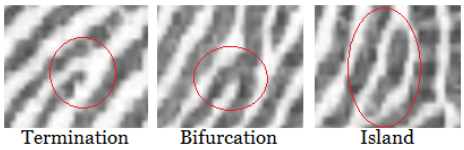

Figure 1: An example of a few minutiae types

For fingerprint comparison, the minutiae locations are extracted from fingerprints and a minutiae comparison algorithm is applied to determine the probability of a fingerprint match. This can, for example, be done by using the 12 Locard rule [3], which states that if an image is of high quality it needs 12 corresponding minutiae to determine a fingerprint match. For a lower number of minutiae, between 8 and 12, a fingerprint match depends on a number of factors, for example the uniqueness of the minutiae[3].

An alternative option for fingerprint comparison, which is explored in this paper, is to first use an unsupervised learning algorithm to create a latent representation of the fingerprint segment. This latent representation is a compressed version of the original image. This compression is lossy, meaning that not all information at the input of the autoencoder will be present at the output. Which information is lost depends on the compression method. The method proposed in this paper uses an autoencoder[4] to create a latent representation.

An autoencoder is a type of neural network that can encode a large part of the data in a small latent space representation[4]. It uses a target which is equal to the input of the neural network, meaning that it tries to recreate the input image at the output. The autoencoder consists of an encoder part, a latent representation and a decoder part as can be seen in Figure 2.

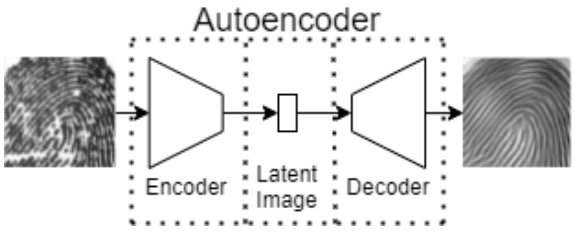

Figure 2: The inner workings of an autoencoder.

The encoder creates a bottleneck halfway through the network, which forces the autoencoder to create a compressed version of this input image, which is called the latent representation or latent vector.
Minutiae locations and orientations are crucial for classical fingerprint comparison methods[3], so effort will be made to ensure that the minutiae locations are present in the latent space of the autoencoder. It is difficult to confirm if minutiae locations are present in the latent space. Therefore the assumption is made that if these minutiae are reconstructed in the output of the autoencoder they will also be present in the latent space.
This leads to the first research question: *To what extent can identifying information of a fingerprint be preserved and reconstructed by an autoencoder?* Hereby the focus lies on the minutiae, but other identifying information like general shape and ridge width might also be important for fingerprint classification.
Assuming a method is found which will properly translate the minutiae locations to the output of the autoencoder the question remains whether it is possible to extract these properties from the latent space for fingerprint comparison. This leads to the second research question. *To what extent can the information in the latent space of the autoencoder be used for fingerprint comparison?* This paper will first cover research question one under the header "Preservation of identifying information". Secondly, it will discuss research question two under the header "Classification in the latent space". Afterwards a joined conclusion will be drawn and some suggestions for future work will be provided.

Materials

For this research a python 3.7.3 environment was used with pytorch 1.7.0.
The dataset that was used for this research was the MCYT dataset[5]. It contains fingerprints of all 10 fingers of 330 individuals, resulting in a total of 3300 unique fingerprints. Of each fingerprint 24 different images were taken, 12 with a c-mos sensor and 12 with an optical sensor.

Preservation of identifying information

The following section will cover the research question *To what extent can identifying information of a fingerprint be preserved and reconstructed by an autoencoder?* It will first describe relevant literature. In the next section an explanation will be provided of how the autoencoder will be implemented and trained. Afterwards the three different loss functions which will create three different autoencoders will be discussed. Finally the results of the different loss functions will be shown and discussed.

Related work

Recently, several deep learning methods have been developed to replace parts of the fingerprint classification process. A common part to replace is the minutiae extraction process [6][7][8].
One of those works, which bears some resemblance to this paper, is from A. Sankaran et al.[9]. They trained two different autoencoders on small segments of fingerprints, one only on segments with minutiae and the other on segments without minutiae. Afterwards they trained two binary classifiers on the latent spaces of these autoencoders to determine if they contained minutiae or not. With this method they achieved a rank 10 identification accuracy comparable to manual identification rate[9].
K.M. Sagayam et al. [10] researched the next step in the classification process. They developed a

neural network to compare two sets of minutiae, in which they achieved an accuracy of 99% on a small testset, despite a very small trainset.
In recent years deep learning applications have been developed which completely replace the fingerprint comparison process.
B. Pandya et al. [11] made a fingerprint comparison process for a small amount of identities. They trained a neural network to recognize thinned fingerprints and assign them to their unique class with an accuracy of 98.21% using only 800 fingerprint images (8 images per fingerprint).
J. Ezeobiejesi and B. Bhanu[12] used a Restricted Boltzmann Machine to compare two fingerprint images and get a latent representation of fingerprint segments to train a fully convolutional neural network to generate a comparison score. With this method they achieved a rank-1 identification score of 81.35%. However, the fingerprint segments they compared were only 32x32 pixels, since they focused on latent fingerprints.
The method proposed in this paper will distinguish itself in that it creates an autoencoder with an input segment which is significantly larger than the work mentioned above. It will also focus on direct fingerprint comparison using the latent space, which in itself seems to be an unexplored field.

A lot of research has been published about autoencoders. Autoencoders have been developed for various tasks, for example denoising autoencoders for audio [13] and images [14][15][16], optical flow estimation [17] and dimensionality reduction[18][19]. However, the application of autoencoders on large fingerprint segments seems to be a relatively unexplored field.N. Ichimura et al.[20] implemented a Laplacian filter bank to increase the performance of their autoencoder, which will be one of the loss functions researched in this paper for preservation of fingerprint identifying information in the autoencoder.
J. Springenberg et al.[21] proved that in convolutional networks it is more efficient to use stride instead of pooling without efficiency loss. This motivates the design decision to use stride instead of pooling for the autoencoder presented in this work.

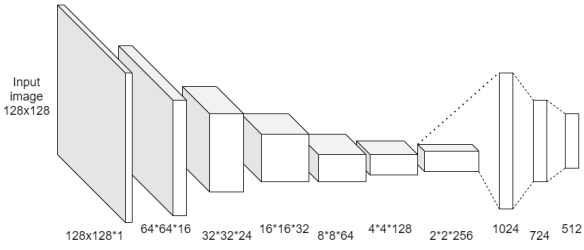

Figure 3: Network architecture of the encoder part of the autoencoder

Network architecture and training process

The autoencoder model proposed in this paper is 16 layers deep, the encoder consists of the first 8 layers and can be found in Figure 3. The decoder consists of the last 8 layers and is a mirrored version of the encoder. The first four layers consist of convolutional layers with filter size 7x7 using padding and a stride of 2. The 5th layer uses a filter of 3x3 with padding and a stride of 2. The 6th layer uses a filter size of 3x3 without padding and a stride of 1. Afterwards the data is flattened for the 7th layer, which is a fully connected layer bringing the data from 1024 to 724 parameters. The 8th layer is a fully connected layer which brings it from 724 to 512 parameters. The output of the 512 parameter layer will be considered the latent space. All the layers use a Relu[22] activation function.
The middle part of the autoencoder consists of fully connected layers. Earlier experimentation on smaller segments showed that this resulted in a better reconstruction compared to a fully convolutional neural network. It was not possible to use a fully connected network, since it would contain too many parameters to fit inside the memory of a GPU when scaling to 128x128 segments.
Furthermore, the first (and last) convolutional layer contain many filters. The filters in these layers formed ridge valley patterns. By having more filters in these layers the fingerprint could be reconstructed more accurately, since more possible patterns could be created.
The network was trained on images of 128x128 pixels with values scaled between 0 and 1. It contained a section of a fingerprint which was

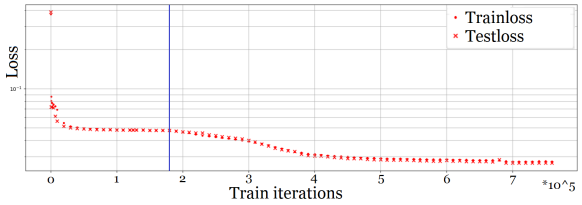

Figure 4: The effect of switching between different optimizers during the training process. The blue line indicates the moment which the optimizer switches from SGD to Adam.

selected around the fingerprint core. This segment is approximately one fourth of an average fingerprint size[23] with the exception of the thumb, which is larger. Each trainbatch contained 6 images from the dataset. To increase the effective size of the trainset each image was rotated with an interval of 15 degrees from -90 to 90 degrees and each images was mirrored resulting in 26 training images per image in the dataset. Each trainbatch contains 156 images, one epoch of the trainset contains 547 iterations of trainbatches.

The training process started with a stochastic gradient descent optimizer with a learning rate (LR) of 0,01. When the decrease in loss was less than 4 percent over the last 150000 iterations, the training switched to Adam optimizer[24] with a LR of 0.01 for the remainder of the training. The effect of the optimizer switch on the loss can be seen in Figure 4. This switch was necessary to guarantee results during the training process of the autoencoder and proved superior in loss and image quality to using exclusively Adam or only SGD.

Loss functions

Autoencoders trained with mean square error (MSE) loss have a tendency to miss high frequency components in their reconstruction, as can be seen in the middle image in Figure 10. Fingerprint minutiae contain more high frequency components, so minutiae locations might disappear or become blurry.

Two methods are implemented to preserve the minutiae locations at the output of the autoencoder. Next to these two methods a network is trained with a MSE loss between the output and the target. This network is used as a reference

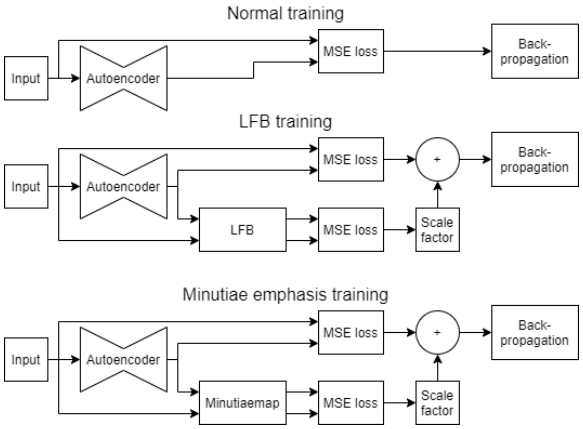

Figure 5: Block diagram of the different train processes which are implemented to perserve minutiae locations

and will be referred to as the Normal trained network. An overview of its train process can be found in the first block diagram in Figure 5.

The first method to preserve minutiae locations is to emphasize frequencies normally less present in the reconstructions. A Laplacian filter bank (LFB)[20] has a band pass property which is used to increase the focus on high frequency components which are normally less present in the reconstruction. The overall filter of the Laplacian filter bank is given by:

$$h(x,y) = -\frac{1}{2\pi\sigma^4}(1 - \frac{x^2+y^2}{2\sigma^2})e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

where x and y are the coordinates of the filter and σ is a scale factor, used for scaling the length of the filter and thus letting it focus on a different frequency. For σ the values 0.6, 0.8, 1, and 1.2 are used.

During the training process the LFB training method combines multiple loses. First, the standard MSE loss is calculated in the same way as in the Normal training method. Afterwards, the output and target are filtered with the Laplacian filter bank, which consists of the four filters described earlier, each with a different scale. Following this, the MSE is calculated between the filtered outputs and their targets. This loss is scaled with a factor 2, to make the proportional loss of the LFB comparable to that of the normal training method. Finally, the losses are added and back propagation is applied to the total loss. An overview can be found in the second block

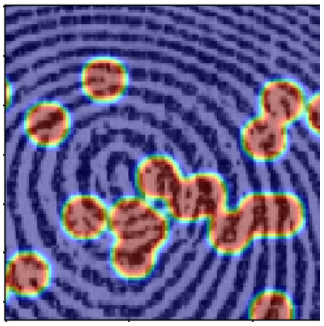

Figure 6: An example of the minutiae map overlaying an fingerprint segment

diagram in Figure 5.

The second method to preserve the minutiae locations at the output of the autoencoder, is to increase the loss of the autoencoder during the train process at the minutiae locations. This forces the autoencoder to better reconstruct minutiae in order to decrease the overall loss.

To emphasize the minutiae locations, the output of the autencoder and the target are first multiplied by a minutiae map. This Minutiae map has a value of 1 at minutiae locations, and a declining value around the minutiae until it has reached 0 on locations where there are no minutiae.

The minutiae map is created by finding the minutiae locations of the training set using a minutiae extractor[25]. A map consisting of zeros is created in which the locations of the minutiae are set to 1. This map is dilated by a morphological sphere with a diagonal of 21 pixels and filtered with a Gaussian blur filter with a sigma of 9 in both the x and y directions. An example of a minutiae map displayed over a fingerprint image can be found in Figure 6. The minutiae emphasis training combines two losses similar to the LFB training method. First, the standard MSE loss is calculated in the same way as in the Normal training method. Secondly, the output and target of the autoencoder are multiplied by the minutiae map and the loss is calculated between these filtered outputs and targets. This loss is scaled with a value of 2 to make the minutiae locations more important and added to the standard MSE loss calculated before. An overview can be found in the third block diagram in Figure 5.

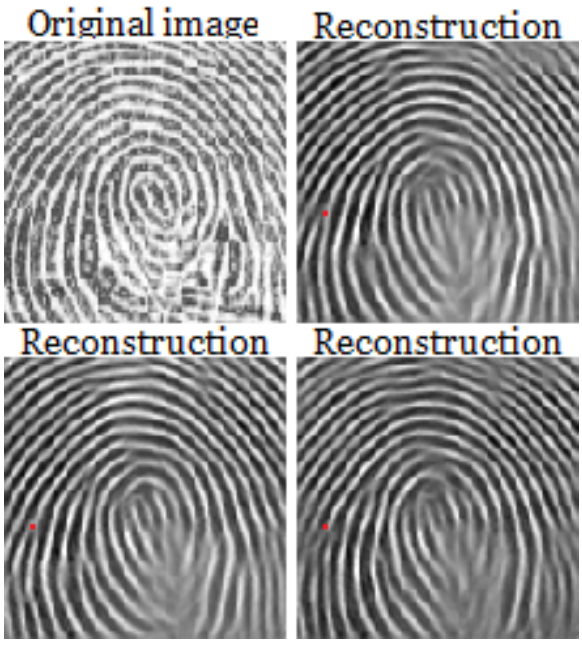

Figure 7: An example fingerprint segment (left upper corner) and 3 reconstructions of 3 different images of the same fingerprint segment using an Minutiae emphasis trained autoencoder. In the 3 reconstructions a minutiae is marked which is reconstructed differently each time.

Results and discussion

Figure 7 shows the reconstruction of the three different images of the same fingerprint using the minutiae emphasis trained autoencoder. The core segment of the fingerprint is reconstructed differently in all of the three images. Figure 8 shows that the Laplacian filter bank provides a more similar core reconstruction over the different images. This is because the band pass property of the LFB creates a preference for certain frequencies, which results in a more consistent reconstruction using these frequencies. However, the core was incorrectly reconstructed in all the instances in Figure 8. Figure 9 shows that the reconstruction of the core improves if the core contains less minutiae and more straight lines.

A possible explanation for the incorrect core reconstruction is that the core is only a small part of the fingerprint segment, containing many minutiae and curving and varying ridges. Instead of using a lot of resources to properly

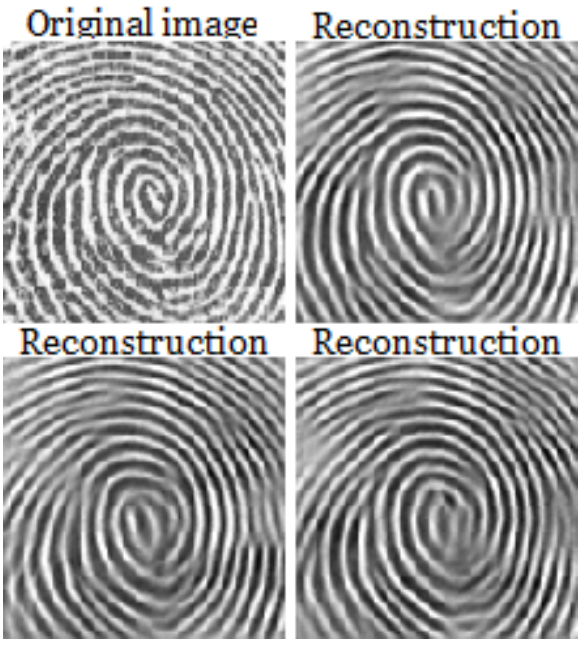

Figure 8: An example fingerprint segment (left upper corner) and 3 reconstructions of 3 different images of the same fingerprint segment using an LFB trained autoencoder

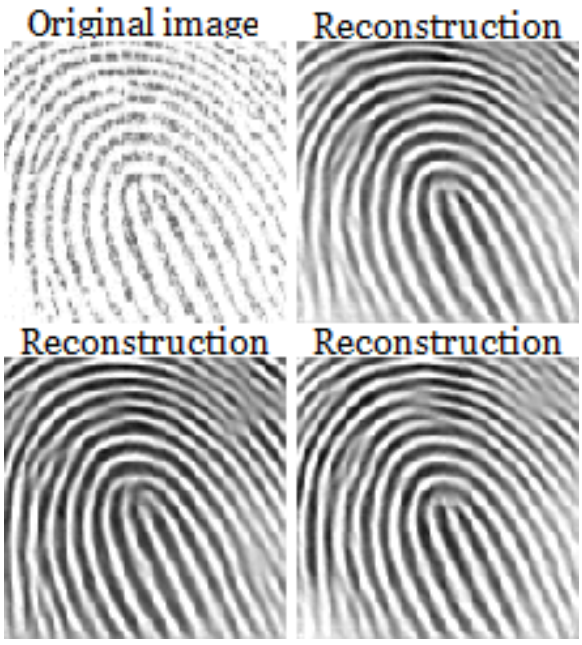

Figure 9: An example fingerprint segment (left upper corner) and 3 reconstructions of 3 different images of the same fingerprint segment using an normal trained autoencoder

reconstruct a fingerprint core, the loss can more easily be minimized by focusing on the other parts of the fingerprint. If necessary, part of the core can be generated, instead of reconstructed. Another option is that the latent space is not large enough to get an accurate representation of the entire image, so by not properly encoding the core the latent space can still encode the rest of the segment.

Despite the faulty core reconstruction the minutiae around the core are reconstructed properly most of the time. However, the type and exact location of minutiae may vary, which can be seen when looking left of the core of the reconstructions in Figure 8 or at the marked minutiae in Figure 7. These changes in minutiae types might not be a problem since the location of the minutiae will still be encoded in the latent space. This means that classification on the latent space might not be hindered by these changing minutiae types.

As can be seen in the fourrier transforms in Figure 10 the autoencoder trained with the Laplacian filter bank contains more high frequency information. The images generated by the LFB are sharper than the normal training method. This is most clearly visible in images with a small ridge valley distance, as can be seen in Figure 11 where two minutiae are marked which are not reconstructed with the normal training method.

Furthermore, Figure 11 shows that the autoencoder has developed some interesting reconstruction properties. The scar located in the lower left part of the image is removed in the reconstructions. The downside is that the normal and minutiae training methods have repaired this scar by filling the space with minutiae. The LFB trained method has made a different reconstruction and fixed the segment without generating minutiae, likely due to its preference for certain frequencies.

The difference between the autoencoders regarding the minutiae outside of the core is hard to determine, since the reconstructions of the same autoencoder vary too much. All autoencoders are able to reconstruct factors such as the general shape of the fingerprint. The

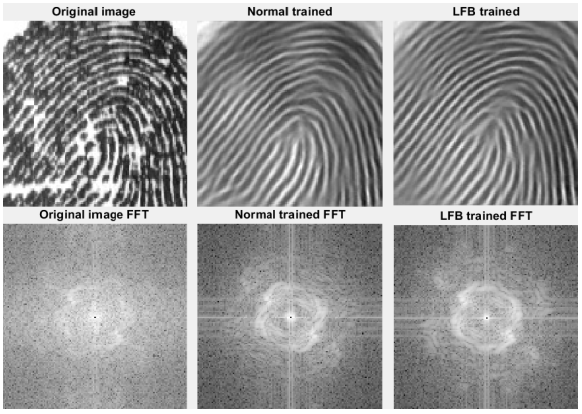

Figure 10: The input, output of an trained autoencoder and output of a LFB trained autoencoder and their Fourier transforms

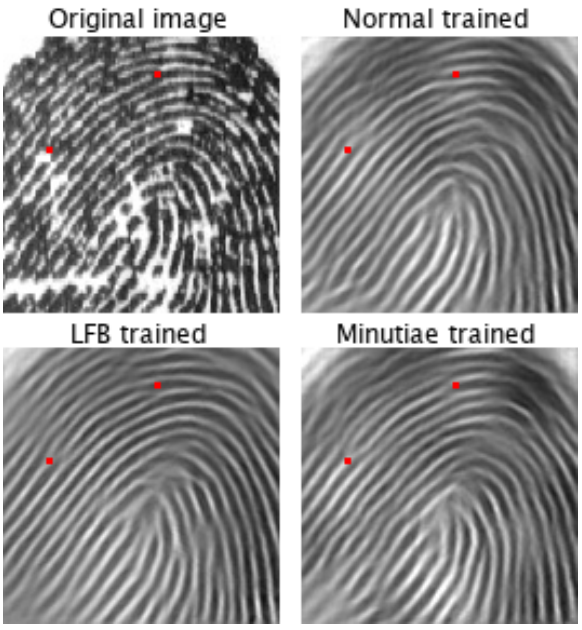

Figure 11: An example of the reconstruction process for the different trained autoencoders. The same location near a minutiae is marked over the 3 different reconstructions.

various ridge widths proved to be no problem for the reconstruction, is demonstrated when comparing Figure 11 and Figure 9.

Classification in the latent space

The following section will cover the research question: *To what extent can the information in the latent space of the autoencoder be used for fingerprint comparison?* Firstly, it will shortly describe some literature which is used in this section. Secondly, it will explain how the different latent images of the trained autoencoders were evaluated. It will describe how the neural network classifier, which is one of the evaluation methods, was trained and implemented. Lastly, the results will be discussed.

Related work

Some research has been performed to increase understanding of the latent space of autoencoders. T. Spinner et al. [26] tried to compare the latent spaces of autoencoders and variational autoencoders using a t-distributed stochastic neighbor (T-SNE) analysis. They concluded that it is hard to change the latent space of a normal autoencoder, since you need luck to find a value in which the decoder knows a reconstruction.
Several published studies have made use of a fully connected neural network for latent classification. For example, the aforementioned studies by A. Sankaran et al. [9] used it for their minutiae detector and K.M. sagayam et al. [10] used it for their minutiae comparison. In addition, when a convolutional network is applied for fingerprint comparison they tend to make use of some fully connected layers at the end of their neural network to do the final classification step[11][27]. However, since the classification results are very dependent on the latent vector creation these results cannot be compared to this work.
Furthermore, the work of S. Roweis and G. Hinton[28] and L. van der Maaten[29] is applied in this section to visualize the latent space using their work regarding the t-distributed stochastic

neighbor embedding for visualizing the latent space. For the classification process the T-SNE analysis could be used when there are a limited number of identities. However, with the number of unique identities in the dataset this does not seem like a viable option for the classification problem presented in the paper.
Lastly, a t-test[30] will be performed to determine the statistical significance and confidence interval (CI) of the difference between the several fingerprint comparison runs.

Latent creation and comparison

To generate the latent spaces, the fingerprint sections around the core were used as input for the encoder part of the three autoencoders that were trained as described in the previous section. Each fingerprint segment resulted in a 512 parameter long vector which is considered the latent space.
A T-SNE algorithm was applied to a eight identities to identify if the identities could be mapped close to each other. A classifier was trained on the identities to determine if it is viable to use the latent space for classification.

Training method and network architecture

For classification of the latent space a neural network was developed with an input of a 1024 factor containing the 2 latent spaces of the fingerprints, which were both 512 parameters long. The networks contains 1 hidden layer with 2048 neurons and a single output neuron which was trained to be 1 in case of a fingerprint match and 0 otherwise. The network was trained with stochastic gradient descent. 15 identities were used as testset and the other 315 as trainset. The classifiers were trained multiple times with a different initialization of the neural network, to check if the difference between the runs was statistically significant.

Results and discussion

The results of the T-SNE analysis in Figure 12, 13 and 14 provide an insight in the information in the latent space. All three training methods show indications of grouping of the latent space in the T-SNE analysis. However, the identities are also grouped based on fingerprint sensors used in the

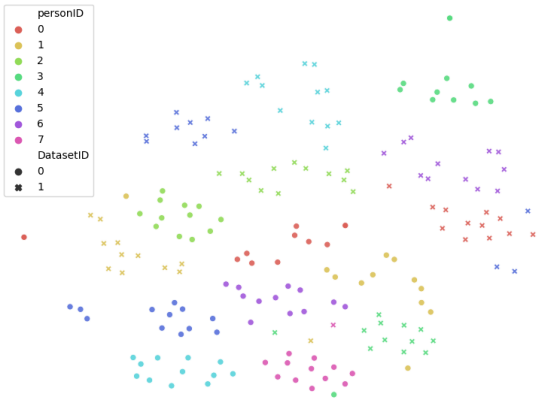

Figure 12: The T-SNE results of some latent space images generated by the normal trained autoencoder

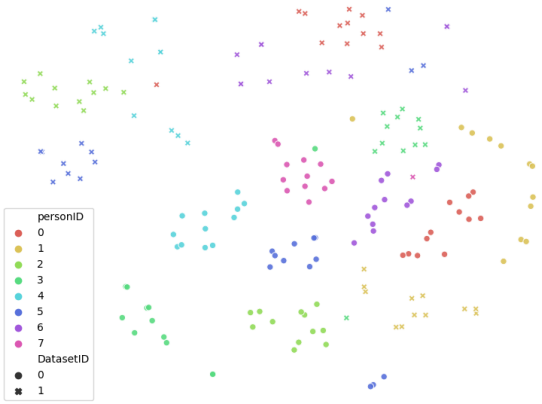

Figure 13: The T-SNE results of some latent space images generated by the LFB trained autoencoder

dataset, for which it was not trained. In addition, there are several outliers with different causes. Some were caused by an error during fingerprint capture, which resulted in partial white space around the fingerprint core. Others were caused by an error in the fingerprint core detection software, which detected a core in the wrong place.

The wrongly detected fingerprint core segments have a significant impact on the classification process. When comparing two fingerprint segments, the part of the fingerprint which is similar between the two segments is smaller and not located in the same part of the image. If

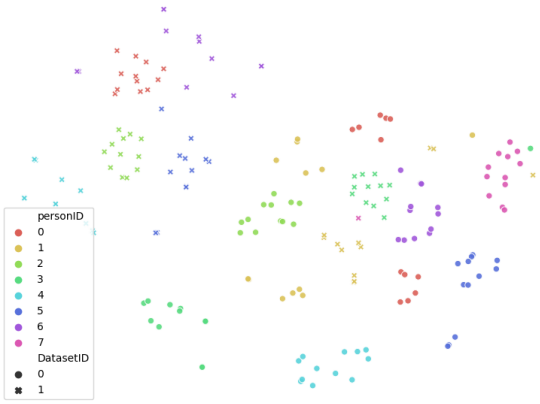

Figure 14: The T-SNE results of some latent space images generated by the Minutiae trained autoencoder

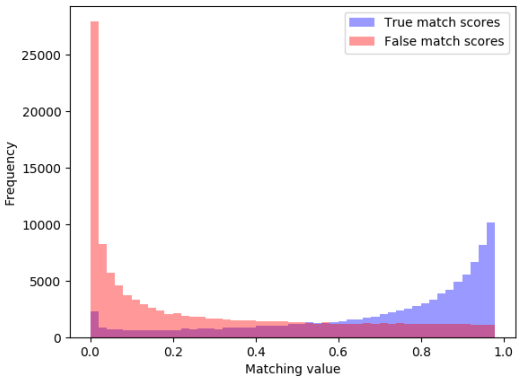

Figure 15: The classification scores of run 2 for the normal trained autoencoder

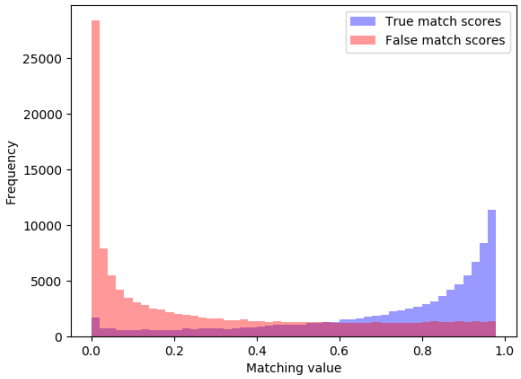

Figure 16: The classification scores of run 2 for the LFB trained autoencoder

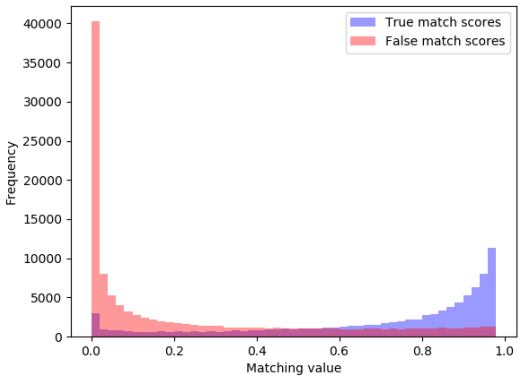

Figure 17: The classification scores of run 2 for the Minutiae trained autoencoder

the core is inaccurately detected even further away. there might not be any overlap between the fingerprint segments.
When looking at the classification scores of the testset for the trained classifiers in Figure 15, 16 and 17, these misaligned fingerprint segments can explain the peak in the true match scores with a comparison value of 0.

Another factor which influences the classification scores is the amount of unique information in a fingerprint segment. Since this research used only a segment of the fingerprint, the amount of identifying information is limited. Since the core reconstruction leaves room for improvement this means that the classification mainly depends on minutiae and other fingerprint identifying information located outside of the core but inside the segment. This is generally in between 5 and 10 minutiae. Comparing this to, for example, the Locard 12 rule[3] it means that proper identification is not always possible with this small amount of properly reconstructed minutiae.

When looking at the equal error rate (EER) of the trained classifiers in Table I, it shows that the minutiae training method performs better than the normal training or LFB training method. Table II shows that the differences between the training methods are statistically significant. This difference would suggest that the minutiae locations are still important in the classification process. Since the minutiae training

Table I: The equal error rate (EER) of the trained classifier for differently trained autoencoders over multiple runs

EER	Run 1	Run 2	Run 3	Run 4
Normal training	0.230	0.232	0.236	0.230
LFB training	0.228	0.228	0.227	0.226
Minutiae training	0.212	0.212	0.208	0.209

Table II: The p values, the mean difference and confidence interval (CI) when comparing the different loss functions

	p value	mean (95% CI)
Normal vs LFB	0.0190	0.00475 (0.00110-0.00840)
LFB vs Minutiae	<0.0001	0.01700 (0.01422-0.01978)
Normal vs Minutiae	<0.0001	0.02175 (0.01747-0.02603)

method focuses more on the minutiae, they will likely occur more often in the latent space, which improves the classification. This difference can also be observed in the ROC curves of the 3 classifiers in Figure 18.
Furthermore, Table I shows that the classifier using the LFB training method results in a slightly better EER in comparison to the normal training method. This difference is less visible when looking at the ROC curves in Figure 18, where the LFB trained and Normal trained curves lie very close to each other. The difference is small, but statistically significant as demonstrated in Table II. This would suggest that the autoencoder profits from training methods other than the MSE loss training methods and from the additional high frequency information forced in the reconstruction.

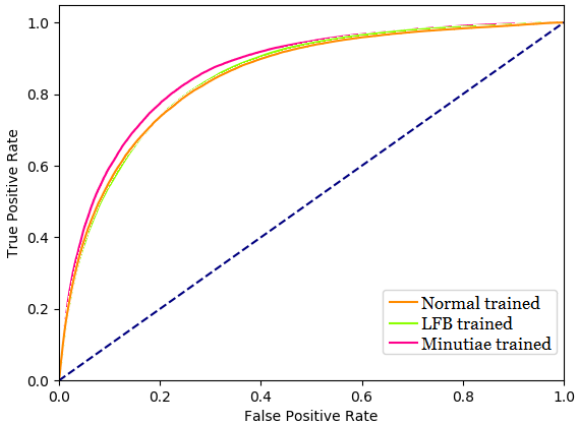

Figure 18: The combined ROC curves of the three trained classifiers

Conclusion

The first research question was: *To what extent can identifying information of a fingerprint be preserved and reconstructed by an autoencoder?* The autoencoder that was developed could reconstruct most of the fingerprint minutiae located outside of the core of the fingerprint. In addition, factors such as ridge width, overall shape and illumination differences were reconstructed in the output of the encoder. A limitation of the autoencoder was that the core segment of the fingerprint seemed too complex to be properly reconstructed in various instances.

The second research question was: *To what extent can the information in the latent space of the autoencoder be used for fingerprint comparison?* Despite the limitations of the fingerprint segment size, the imperfect core reconstruction and the misaligned fingerprint segments, the classifier achieved an EER of 0.208.
If the aforementioned problems would be solved and the entire fingerprint would be used, it is suspected that using the latent space of an autoencoder could result in a fingerprint matching system with competitive performance compared to modern day fingerprint matching algorithms.

Future work

For future work, the research regarding the first question could be improved. Some uncertainties remain regarding the preservation of identifying information of a fingerprint and the reconstruction by an autoencoder. More insight in this question could be achieved by applying a minutiae extractor to the input and output of the autoencoder, so an understanding can be created on how the minutiae locations are preserved when training the autoencoder.
Furthermore, the reconstruction of the core should be improved. This could possibly be done by increasing the loss at the location of

the core or perhaps by increasing the size of the latent space. If the LFB were to be used, the core reconstruction could be improved by finding out exactly which frequency components are necessary for a proper reconstruction.
The classification process can likely be improved by using a larger fingerprint segment, as this will result in more fingerprint identifying information being included in the image. This information will also be transferred to the latent space. In addition, it is recommended to improve the reconstruction of the fingerprint core to further increase results of the classification process.
Furthermore, the classification results could be improved by allowing the encoder to continue training during the training of the classifier. This will allow the encoder to start focusing on different parts of the fingerprint segment which perhaps contain more important identifying information. Another area for further development which arose during this research is the use of different optimizers during the training phase of the autoencoder. By switching between optimizers, the autoencoder achieved significantly better results in comparison to only using Adam optimizer or stochastic gradient descent. Further research into this topic is recommended to gain insight into why this works and how it could be used to improve the performance of neural networks in the future.

References

[1] D. THAKKAR, "What makes fingerprint the most popular biometric modality?," 3 2019.

[2] NEC, "The top 9 common uses of biometrics in everyday life," 7 2020.

[3] P. Kingston, C.R.: Kirk, "La regle des 12 points dans l'identification par les empreintes: historique et valeur," *Revue internationale de police criminelle*, vol. 20(186), pp. 62–69, 1965.

[4] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[5] J. Ortega-Garcia, J. Fierrez, D. Simon, J. Gonzalez, M. Faundez-Zanuy, V. Espinosa, A. Satue, I. Hernáez, J. Igarza, C. Vivaracho-Pascual, D. Escudero, and Q. Moro-Sancho, "Mcyt baseline corpus: a bimodal biometric database. iee proc vis image signal process spec issue biom internet," *IEE Proceedings - Vision Image and Signal Processing*, pp. 395 – 401, 12 2003.

[6] L. N. Darlow and B. Rosman, "Fingerprint minutiae extraction using deep learning," pp. 22–30, 2017.

[7] W. Leung, S. Leung, W. Lau, and A. Luk, "Fingerprint recognition using neural network," *Neural Networks for Signal Processing Proceedings of the 1991 IEEE Workshop*, pp. 226–235, 1991.

[8] "B. gour, t. bandopadhyaya, and s. sharma, "fingerprint feature extraction using midpoint ridge contour method and neural network," international journal of computer science and network security, vol. 8, no. 7, pp. 99–103, 2008,"

[9] A. Sankaran, P. Pandey, M. Vatsa, and R. Singh, "On latent fingerprint minutiae extraction using stacked denoising sparse autoencoders," pp. 1–7, 2014.

[10] K. M. Sagayam, D. N. Ponraj, J. Winston, J. Yaspy, D. E. Jeba, and A. Clara, "Authentication of biometric system using fingerprint recognition with euclidean distance and neural network classifier," *Int. J. Innov. Technol. Explor. Eng*, vol. 8, no. 4, pp. 766–771, 2019.

[11] B. Pandya, G. Cosma, A. A. Alani, A. Taherkhani, V. Bharadi, and T. McGinnity, "Fingerprint classification using a deep convolutional neural network," in *2018 4th International Conference on Information Management (ICIM)*, pp. 86–91, 2018.

[12] J. Ezeobiejesi and B. Bhanu, "Patch based latent fingerprint matching using deep learning," pp. 2017–2021, 2018.

[13] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder.," in *Interspeech*, vol. 2013, pp. 436–440, 2013.

[14] I. L. Y. B. P.-A. M. Pascal Vincent, Hugo Larochelle, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research 11*, pp. 3371–3408.

[15] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, pp. 341–349, 2012.

[16] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 241–246, 2016.

[17] V. Patraucean, A. Handa, and R. Cipolla, "Spatio-temporal video autoencoder with differentiable memory," *CoRR*, vol. abs/1511.06309, 2015.

[18] R. Hinton, G. E.; Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507.

[19] D. Del Testa and M. Rossi, "Lightweight lossy compression of biometric patterns via denoising autoencoders," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2304–2308, 2015.

[20] N. Ichimura, "Spatial frequency loss for learning convolutional autoencoders," 06 2018.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2015.

[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," pp. 807–814, 2010.

[23] J. C. Wu, "Statistical analysis of widths and heights of fingerprint images in terms of ages from segmentation data," 10 2008.

[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[25] Utkarsh-Deshmukh, "Fingerprintfeatureextraction." https://github.com/Utkarsh-Deshmukh/Fingerprint-Feature-Extraction, 2021.

[26] T. Spinner, J. Körner, J. Görtler, and O. Deussen, "Towards an interpretable latent space : an intuitive comparison of autoencoders with variational autoencoders," in *Proceedings of the Workshop on Visualization for AI Explainability 2018 (VISxAI)*, 2018.

[27] L. Jiang, T. Zhao, C. Bai, A. Yong, and M. Wu, "A direct fingerprint minutiae extraction approach based on convolutional neural networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 571–578, 2016.

[28] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Stochastic neighbor embedding with gaussian and student-t distributions: Tutorial and survey," 2020.

[29] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[30] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908.