# Fingerprint Image Restoration Using the U-Net Deep Learning Model

Albert Heurlin de Oliveira,
Elin Stenbäcken

Master's thesis
2026:E2


Lund Institute of Technology
Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

# Abstract

This thesis investigated the use of convolutional neural networks (CNNs) for image restoration of fingerprint images, focusing on the reconstruction from raw sensor images to denoised fingerprint images, suitable for matching. Fingerprint images from sensors often introduce noise and artifacts that inhibit the matching process. The goal was to develop a CNN-based approach capable of removing artifacts such as moiré patters that degrade fingerprint quality, thereby restoring the fingerprint image. A U-Net architecture was used as a baseline model and extended with several architectural modifications such as Convolutional Block Attention Modules (CBAM), gated skip connections and dilated convolutions. In addition, the effects of different batch sizes and learning rates for the ADAM optimizer were evaluated. The models were trained with synthetic fingerprint data, including a dataset augmented with image transformations to increase the training dataset size. The results show that CNN-based restoration can surpass traditional ISP pipelines, especially when training and test domains are well aligned. Learning rate selection held significant importance, with $10^{-4}$ consistently yielding the lowest False Reject Rates (FRR). Data augmentation improved robustness and frequently reduced FRR relative to identical models trained on non-augmented data. However, generalization to different datasets remained limited, highlighting the need for a training dataset representative of diverse data. The most promising model architecture for restoration was U-Net, which is the simplest of all the proposed models. Since the training dataset was very limited in size, the risk of overfitting was high, and it seemed that the more complex models tended to overfit relatively quickly. The simple U-Net, in contrast, seemed to generalize the best, and had more consistent performance on the testing data than every other model. Overall, it seems that CNN-based restoration methods offers a promising approach to reconstruct fingerprints from raw sensor images, and that their role and performance could be increased with further development.

# Populärvetenskaplig sammanfattning

När biometriska metoder används för autentisering är det viktigt både att processen är korrekt, och går snabbt. Givetvis ska en obehörig användare inte kunna passera autentiseringsprocessen, och givetvis är det fördelaktigt om autentiseringen inte tar lång tid. Vid biometrisk fingeravtrycksigenkänning finns flertalet faktorer som försvårar matchningen. Brist på ljus kan göra fingeravtrycksbilden för mörk, och därmed ha bristande information. När fingret trycks mot skärmen kan artefakt er som moirémönster upkomma. Moirémönster är ett slags interferensmönster som bildas av överlappet mellan displayens pixelmönster och sensorns elektrodmönster.

Medan traditionella matchnings-metoder fungerar bra, är de ofta specifika till särskilda typer av sensorer och användarfall. Således är de inte generella nog att fungera bra under nya förhållanden, utan kräver anpassningar beroende på fall. En stor del av dessa anpassningar tar form av förbearbetning av fingeravtrycksbilderna, innan matchningen utförs. Under arbetet har traditionella bearbetningsmetoder jämförts med metoder baserade på neurala nätverk. Som grund för arbetet har en modell kallad U-Net använts, och flera utbyggnader av detta nätverk har utvärderats.

Att träna neurala nätverk att bearbeta och restaurera bilder på fingeravtryck är oerhört utmanande. Det finns en stor variation av artefakter och förhållanden som påverkar bildens kvalitet. Till exempel ger ett ”normalt” fingeravtryck en väldigt annorlunda bild än ett fuktigt fingeravtryck. Fokus i det här arbetet har varit på normala fingeravtryck. Även då är det utmanande att utveckla stabila, bra modeller. Variationer som huruvida bilden är ljus eller mörk, lågupplöst eller högupplöst, eller om fingret täcker sensorn eller inte, gör det svårt att träna en alltäckande modell.

Genom att använda den matchningsmetod som finns på Precise Biometrics var det enkelt att jämföra olika modellers kvalitet.

# 1 Preface

This thesis was carried out in the fall of 2025 at Precise Biometrics, Lund in collaboration with the Centre for Mathematical Sciences at LTH.

We would like to express our gratitude to the team at Precise Biometrics for their support, guidance and encouragement. Special thanks to Johan Windmark and Ellen Åström for their invaluable help and feedback throughout. We would also like to thank our supervisor Anders Heyden for his continuous advice and support during this thesis.

# 2 Table of Contents

# Contents

# 3 Introduction

Biometric security has become part of everyday life during the 2000s. Biometric security offers several advantages over traditional security measures like passwords. Since no two people share the same fingerprint, face and iris, every person's biometric is unique to them. Unlike passwords, it is impossible (or at the very least way more difficult) to phish someone's biometric credentials, than a password [10]. Because of this, today around 81% of phones have biometric security enabled [7]. With today's prominence of biometric security, convenience has become a major part of biometric solutions' goals. Biometric security does not only have to be safe, i.e. not accepting unauthorized users (so called impostors), but also be able to quickly verify the user.

Traditional measures for biometric fingerprint security typically involves an image signal processor (ISP), which analyses the fingerprints' characteristics such as local ridge frequencies, local ridge orientation and gradients [13]. More contemporary approaches include deep neural networks to increase prominence of relevant fingerprint features, or increase efficiency for low-resolution images[6].

## 3.1 Purpose

In this thesis, several artificial intelligence models were developed and evaluated based on their ability to extract fingerprints from a noisy sensor image. The motivation for this work was to investigate whether CNN-based restoration could compare with, or even outperform, traditional ISP pipelines. To evaluate the models, a reference ISP and biometric matcher from Precise Biometrics was used. Thus it was possible to get a clear comparison of how the models perform relative to more traditional methods.

# 4 Methodology

## 4.1 Theory

### 4.1.1 Image Restoration

Image restoration aims to recover a clean image from a degraded version caused by noise, blur, or other artifacts. Unlike image enhancement where the goal is to enhance the image quality and visual improvement, restoration aims to reconstruct the original image as closely as possible. In this sense, image restoration is about reversing the degradation process and recovering the original image from a distorted image. This involves modeling the degradation mathematically. A common way to describe image degradation is through the following model:

$$g(x, y) = h(x, y) * f(x, y) + n(x, y), \tag{1}$$

where $g(x, y)$ is the degraded image, $h(x, y)$ the degradation process and $f(x, y)$ the original image. The additive noise is represented by $n(x, y)$. The goal of the restoration is to estimate $f(x, y)$ from the known $g(x, y)$.

### 4.1.2 Traditional Image Restoration Methods

Classical image restoration methods rely on mathematical models of both noise and image statistics. One of the most common traditional approaches is a low-pass filter.

The purpose of a lowpass filter is to remove high frequency noise and preserve smooth variations in intensity. Given an input image $I(x, y)$, the process begins by computing its Fourier Transform:

$$F(u, v) = \mathcal{F}\{I(x, y)\}. \tag{2}$$

Most important image structures in the original image are assumed to be contained in low-frequency components while the noise signal is typically modeled as having stronger high-frequency content. Thus, a lowpass filter filter $H(u, v)$ suppresses high-frequency noise while preserving low-frequent image content. In the frequency domain, $F(u, v)$ is multiplied by a filter $H(u, v)$ with large values for low frequencies and small for high frequencies, to attenuate high frequencies and preserve low-frequency components:

$$G(u, v) = H(u, v)F(u, v). \tag{3}$$

Applying the inverse Fourier Transform yields the restored image $g(x, y)$ :

$$g(x, y) = \mathcal{F}^{-1}\{G(u.v)\} \ . \tag{4}$$

This result represents the reconstructed image aquired by applying a lowpass filter in the frequency domain[20].

### 4.1.3 Artificial Neural Networks

Artificial neural networks are machine learning models that are heavily inspired by the neurological structure and function of the human brain[1]. The models consist of several layers containing one or more nodes (see Figure 1), mimicking the neurons in human brains. The first layer is the input layer, followed by one or more hidden layers, and finally an output layer. To produce its output, every node computes a weighted sum of the inputs it receives, adds a bias term and applies an activation function. These weights and biases are learned over time, using an appropriate loss function. When the output layer is reached, this loss function quantifies the error between the predicted output and the true output which is then used to adjust the weights during training[9].

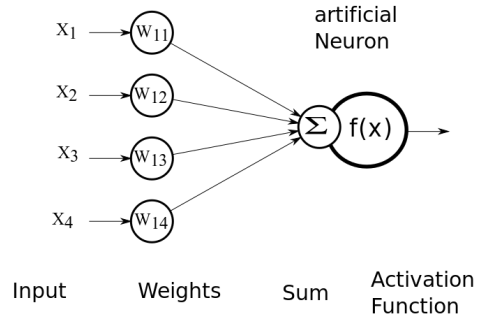



Figure 1: Illustration of a node in a neural network.

### 4.1.4 Feedforward Neural Networks

A feedforward neural network (FNN) is a subclass of artificial neural networks. The "feedforward" refers to the fact that information flows uni-directionally through the network; there are no feedback connections where the output from a node is fed into an earlier node.

A feedforward network can be characterized as a composition of several different functions. For example, the chain

$$f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}))), \tag{5}$$

defines a neural network with three layers, $f^{(1)}$ being the first (input) layer, $f^{(2)}$ a second (hidden) layer, and finally $f^{(3)}$ being the output layer[8]. Thus, a general FNN with $N-2$ hidden layers can be expressed as

$$f(\mathbf{x}) = f^{(N)}(f^{(N-1)}(...f^{(2)}(f^{(1)}(\mathbf{x})))).$$

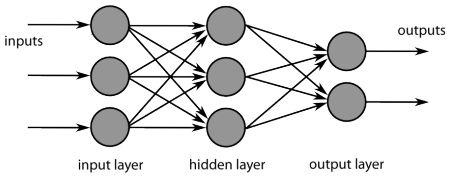



Figure 2: Illustration of a feedforward neural network. Each layer can be interpreted as a function $f^{(k)}$. Image courtesy of Chrislb via WikiMedia Commons

### 4.1.5 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized type of neural network that is suitable for two-dimensional inputs and are commonly used for

image-related tasks. In CNNs, convolution operations involve sliding small filters (kernels) over the input to extract local features, including shapes, edges, and textures. Mathematically, for a 2D input $I$ and a 2D kernel $K$, the convolution output $S$ for position $\{i, j\}$ can be written:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)\, K(m,n), \tag{6}$$

where $m$ and $n$ are the height and width of the kernel. The resulting feature map $S$ is then passed into a non-linear activation function, for instance ReLU or Sigmoid, which enables the network to model complex patterns [8].

During training, the network predicts an output $\hat{\mathbf{Y}}$ and compares it to the ground truth $\mathbf{Y}$ using a loss function. The gradient from the loss is computed through backpropogation and used to update the weights in the model. The weights $\boldsymbol{w}$ are updated iteratively and optimized using gradient descent:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\, \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}_t) \ , \tag{7}$$

where $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}_t)$ is the gradient of the loss function with respect to the weights at iteration $t$ and $\eta$ is the learning rate [3]. The learning rate can be manually set or adaptively changed during training by the Adam optimizer.

#### 4.1.6 Adam Optimizer

The Adaptive Moment Estimator (Adam) optimizer was first proposed in 2015 [11] and is today one of the most widely used optimization techniques for training deep neural networks. It combines AdaGrad and RMSProp to adaptively learn individual learning rates for each parameter, by using first and second moment estimates of the gradient.

At each iteration $t$, Adam updates the parameters $w_t$ by computing first and second moment estimates of the gradient. The gradient of the loss function $\mathcal{L}$, based on the previous parameters $w_{t-1}$ is given by:

$$g_t = \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}_{t-1}) \tag{8}$$

The first moment estimate $m_t$ and second moment estimate $v_t$ are then updated according to:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \tag{9}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2, \tag{10}$$

where $\beta_1$ and $\beta_2$ are exponential decay rates for the first and second moment. Both $m_t$ and $v_t$ are initialized as zero vectors, which causes them to be biased toward zero, particularly during the initial iteration steps. The Adam optimizer therefore applies bias correction to the first and second moment estimates:

$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)} \ , \tag{11}$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)} \ . \tag{12}$$

Once the first and second moment estimates are computed, the parameters $w_t$ can be updated:

$$w_t = w_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{13}$$

where $\alpha$ is the learning rate and $\epsilon$ is a small positive constant added to the denominator to avoid division by zero during parameter updates.

### 4.1.7 The U-Net Model

Deep convolutional networks have been popular for image-restoration tasks. One of the most popular architectures is the U-Net, which follows an encoder-decoder design, and was first proposed in 2015 [15]. The encoder layers extracts high-level features through convolution and pooling layers, while the decoder layers restores the image spatial resolution to reconstruct the output image. Skip connections link the encoder layers feature maps with their corresponding decoder layers via concatenation. This is useful to preserve high resolution features that would otherwise be lost during the downsampling process. Each encoder step consists of two $3 \times 3$ convolutions followed by a rectified linear unit (ReLU), and a two-dimensional max pooling operation. ReLU is defined as

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x \leq 0 \end{cases},$$

which returns the non-negative part of the argument[4]. The max pooling operation consists of an $n \times n$ kernel that "slides" over the input matrix. For each covered $n \times n$ pool, a max operation is performed and the result propagated. This results in a downsampling in the input from $I \in \mathbb{R}^{H \times W}$ to $I' \in \mathbb{R}^{H/n \times W/n}$. Additionally a stride length is used to define how large steps the kernel takes during "sliding". Padding may be used if the shape of the input image is not divisible by $n$.

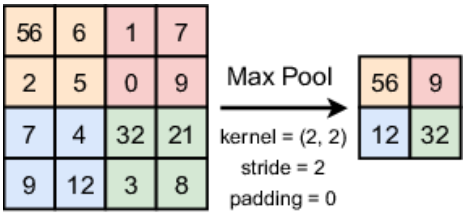



Figure 3: Example of a max pooling operation. Image courtesy of Shubhi Shukla et al.

The decoder includes a transposed convolution followed by two $3 \times 3$ convolutions, each activated by a ReLU function. A final $1 \times 1$ convolution produces the reconstructed image[15].

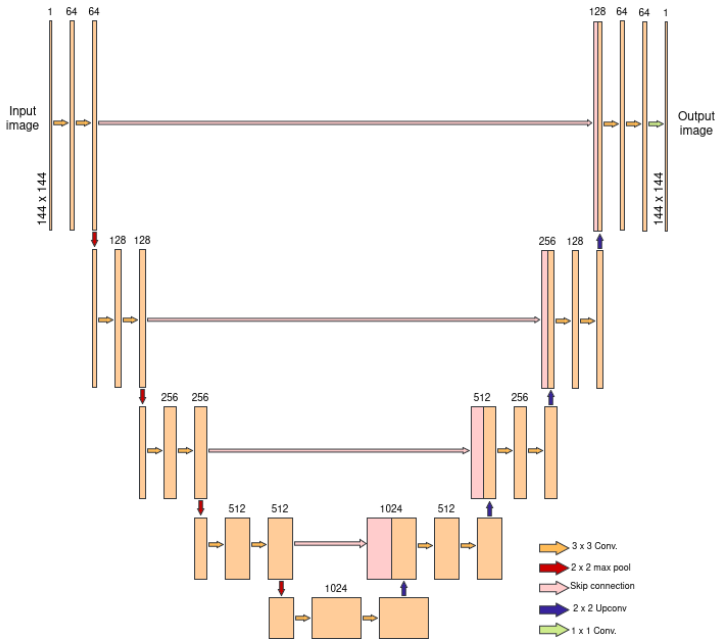



Figure 4: A simple U-net model, without any add-ons.

#### 4.1.8 Convolutional Block Attention Module

Attention mechanisms have played an important role in improving deep learning models. They are based on the human visual system, which has the ability to selectively focus on the salient parts of a scene and ignore the insignificant ones. In a similar manner, the network is guided to focus on the most informative parts of the input. In CNN's, all spatial regions of an image are treated as equally important. However, if the informative regions vary across the image, this becomes suboptimal. Since different areas of the image contribute unequally to the prediction objective, adding attention mechanisms will highlight essential features while diminishing the influence of irrelevant ones. The *Convolutional Block Attention Module* (CBAM) is a type of attention module that uses two sequential attention mechanisms to enhance feature maps. The convolutional attention mechanism was first introduced in 2018 [18] and consists of a *Channel Attention Module* and a *Spatial Attention Module*.

The Channel Attention Module (CAM) refines the feature map by assessing each channels importance, highlighting the significant ones, and suppressing the less relevant ones. Average and max pooling operations are independently applied to the given input $F \in \mathbb{R}^{C\times H\times W}$, producing two channel descriptors:

$$F^c_{\text{avg}} = \text{AvgPool}(F) \quad F^c_{\text{max}} = \text{MaxPool}(F), \tag{14}$$

where both $F^c_{\text{avg}}$ and $F^c_{\text{max}}$ are of size $\mathbb{R}^{C\times 1\times 1}$. To obtain the channel attention map $M_c$, the feature maps are forwarded into a shared multi-layer perceptron containing two convolutional layers. The first layers reduces the number of channels by a factor of $r$, known as the reduction ratio, while the second layer

restores the channel dimension:

$$\mathrm{MLP}(F^c) = W_1(\mathrm{ReLU}(W_0(F^c))), \tag{15}$$

where $W_0 \in \mathbb{R}^{\mathbb{C}/r \times \mathbb{C}}$, and $W_1 \in \mathbb{R}^{\mathbb{C} \times \mathbb{C}/r}$. The two outputs of the MLP are summed and passed through a sigmoid activation function to yield the channel attention map:

$$M_c(F^c) = \sigma(W_1(\mathrm{ReLU}(W_0(F^c_{\mathrm{avg}}))) + W_1(\mathrm{ReLU}(W_0(F^c_{\mathrm{max}})))), \tag{16}$$

where $\sigma$ refers to the sigmoid activation function. The sigmoid activation function is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \ ,$$

which has the range $\sigma(x) \in (0, 1)$ [18].

While the Channel Attention Module focuses on which of the feature channels are most important, the Spatial Attention Module (SAM) learns which spatial regions contain the most informative parts. Firstly, average and max pooling operations are applied along the channel-dimension and produce two 2D maps $F^s_{\mathrm{avg}} \in \mathbb{R}^{1 \times H \times W}$ and $F^s_{\mathrm{max}} \in \mathbb{R}^{1 \times H \times W}$. The maps are then concatenated to identify the most significant spatial regions. The concatenated feature map is forwarded into a convolutional layer to obtain the spatial attention map $M_s$. For a given input map $F \in \mathbb{R}^{C \times H \times W}$ it can be described:

$$\begin{aligned} M_s &= \sigma(f^{7\times7}([(\mathrm{AvgPool}(F); \mathrm{MaxPool}(F)])) \\ &= \sigma(f^{7\times7}([F^s_{\mathrm{avg}}; F^s_{\mathrm{max}}])), \end{aligned} \tag{17}$$

where $\sigma$ refers to the sigmoid activation function and $f^{7\times7}$ denotes a convolution-operation with a $7 \times 7$ kernel [18]. Given an input feature map $F \in \mathbb{R}^{C \times H \times W}$, the resulting channel attention map $M_c$ is a 1D vector $M_c \in \mathbb{R}^{C \times 1 \times 1}$ and the 2D spatial attention map $M_s \in \mathbb{R}^{1 \times H \times W}$. The attention maps are then combined sequentially to rescale the input feature map according to the learned attention weights:

$$\begin{aligned} F' &= M_c(F) \odot F \\ F'' &= M_s(F') \odot F', \end{aligned} \tag{18}$$

where $\odot$ refers to element-wise multiplication and $F''$ is the final feature map[18].

#### 4.1.9 Gated Skip Connections

The original U-Net architecture uses skip connections to pass information from the encoder layers to the decoder layers. Gated skip connections filters the features from the encoder, with the aim that the decoder only receives the useful features. The features of the encoder can contain noise and irrelevant spatial details that negatively affect the performance of the network if they are passed on to the decoder [14]. A variant of gated skips was proposed in *Image Demoiréing in RAW and sRGB Domains* in 2024[19]. They introduce a Gated Feedback Module (GFM) embedded in the skip connections to remove moiré patterns from RAW features.

When a feature map $F_{in} \in \mathbb{R}^{H \times W \times C}$ is passed into the Gated Feedback Module, a point-wise and depth-wise convolution is consequently applied to the feature map to extract the channel and local spatial features. The features are split along the channel dimension to obtain $F_{\text{gate}}$ and $F_{\text{content}}$. These operations can be expressed as:

$$[F_{\text{gate}}, F_{\text{content}}] = \text{DConv}(\text{PConv}(F_{\text{in}}))$$

In the GFM, the gate feature map $F_{\text{gate}}$ is first passed through a GELU (Gaussian Error Linear Unit) activation function:

$$F'_{\text{gate}} = \text{GELU}(F_{\text{gate}}). \tag{19}$$

The activated gate feature map is then point-wise multiplied with the content feature map $F_{\text{content}}$ to adjust its features. The adjusted output is then added to the input feature map $F_{\text{in}}$ and processed through a point-wise convolution, producing the final output of the Gated Feedback Module:

$$F_{out} = \text{PConv}((F'_{\text{gate}} \odot F_{\text{content}}) + F_{\text{in}}), \tag{20}$$

where $\odot$ denotes the point-wise multiplication[19].

### 4.1.10 Dilated Convolutions

The receptive field is the part of the input image that may affect the value of a specific neurons output. Figure 5 illustrates the receptive fields in a CNN for two different neurons $x_2$ and $x_1$. Shallow layers have smaller receptive fields, capturing local detail, while deeper layers have larger receptive fields, capturing global features. For an image restoration task, it is crucial that the network incorporates information from both small and large receptive fields [12].

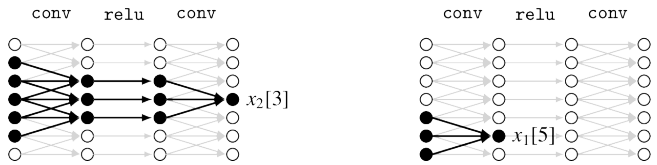



Figure 5: Example of receptive fields for two neurons $x_2$ and $x_1$ in a CNN. Image created by Antonio Torralba, Phillip Isola, and William T. Freeman

.

One way to increase the receptive field is to increase kernel size or add more convolutions and pooling operations, but this is computational costly [12]. Instead dilated convolutions, also know as atrous convolutions, have been proposed. Spacings within the kernel are introduced to get a broader view of the previous layers feature map. Hence, the receptive field increases without reducing the resolution. Different from a regular convolution, the dilated convolution with dilation rate $r$ can be defined:

$$y[i] = \sum_k x[i + r \cdot k] w[k], \tag{21}$$

where $x$ is the input, $w$ is the convolution kernel, $r$ is the dilation rate, and $y$ is the output feature map[5].

#### 4.1.11 Structural Similarity Index Measure

A method to evaluate the structural similarity between images called *Structural similarity index measure* (SSIM) was first proposed in 2004 [16]. Here, the similarity measure is defined as

$$S(\mathbf{x}, \mathbf{y}) = f(l(\mathbf{x}, \mathbf{y}), c(\mathbf{x}, \mathbf{y}), s(\mathbf{x}, \mathbf{y})), \tag{22}$$

where $\mathbf{x}$ and $\mathbf{y}$ are two nonnegative image signals. Thus the similarity measure consists of the three components luminance ($l$), contrast ($c$) and structure ($s$) as well as a combination function $f$. The luminance is defined as

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}.$$

Here $C_1$ is included for stability when the denominator is close to zero, and is defined as

$$C_1 = (K_1 L)^2,$$

with $K_1 \ll 1$ being a small constant, and $L$ the dynamic range of the pixel values, commonly either 1 or 255. Further, $\mu_x$ and $\mu_y$ are the estimated mean intensities of the image signals, given by

$$\mu_x = \frac{1}{N}\sum_i^N x_i. \tag{23}$$

The contrast is defined similarly to luminance

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

with $C_2 = (K_2 L)^2$, where $K_2 \ll 1$. Here $\sigma_x$ and $\sigma_y$ is an estimation of the image signal contrast, given by

$$\sigma_x = \left(\frac{1}{N-1}\sum_i^N (x_i - \mu_x)^2\right)^{1/2}. \tag{24}$$

The structure comparison is computed as

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

where

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y). \tag{25}$$

The three functions are then combined by

$$f(l(\mathbf{x}, \mathbf{y}), c(\mathbf{x}, \mathbf{y}), s(\mathbf{x}, \mathbf{y})) = l(\mathbf{x}, \mathbf{y})^\alpha \cdot c(\mathbf{x}, \mathbf{y})^\beta \cdot s(\mathbf{x}, \mathbf{y})^\gamma$$

with $\alpha > 0, \beta > 0$ and $\gamma > 0$. These exponents control the relative impact of the three factors. In the original paper these exponents are set to $\alpha = \beta = \gamma = 1$

and $C_3 = \frac{C_2}{2}$. With this in consideration, the final form of the SSIM index becomes

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \tag{26}$$

It is important to note that the SSIM index, when used for image quality assessment, is more useful if applied locally to small "patches" of the images, rather than globally. For example, the statistical features $\mu_x$ (23), $\sigma_x$(24) and $\sigma_{xy}$ (25) may be computed in a local $n \times n$ square window, or "patch". A drawback of doing this is that undesirable artifacts may arise. In the original article[16], an $11 \times 11$ circular-symmetric Gaussian weighting function with standard deviation of 1.5 samples was used. That is $\mathbf{w} = \{w_i | i = 1, 2, ..., N\}$ which is then normalized according to

$$\sum_{i=1}^{N} w_i = 1.$$

Following this, the equations for the statistical features $\mu_x$ (23), $\sigma_x$(24) and $\sigma_{xy}$ (25) has to be modified accordingly. Thus,

$$\mu_x = \sum_{i=1}^{N} w_i x_i, \tag{27}$$

$$\sigma_x = \left( \sum_{i=1}^{N} w_i (x_i - \mu_x)^2 \right)^{1/2}, \tag{28}$$

$$\sigma_{xy} = \sum_{i=1}^{N} w_i (x_i - \mu_x)(y_i - \mu_y). \tag{29}$$

Since the SSIM is a window wise measurement, given $M$ windows, $M$ measurements will be computed. It is useful to get a single measurement per image (or rather pair of images). To accomplish this, simply a mean of the SSIM is calculated. Thus the final metric is reached, named MSSIM:

$$\text{MSSIM}(\mathbf{x}, \mathbf{y}) = \frac{1}{M} \sum_{i=1}^{M} \text{SSIM}(\mathbf{x}_i, \mathbf{y}_i) \tag{30}$$

#### 4.1.12 Mean Absolute Error

The Mean Absolute Error (MAE) defined as

$$\text{MAE}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - x_i| \tag{31}$$

is a common metric during deep learning model training. It is simply an arithmetic average of the sum of the absolute difference between two images. In equation (31), $\mathbf{x}$ and $\mathbf{y}$ are two images, and $x_i$ and $y_i$ refer to the pixel values

of said images. Thus the MAE in this case measures the absolute mean pixel difference [17]. Another common metric is root mean square error (RMSE):

$$\text{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2} \ .$$

However, for training a CNN, using MAE as a loss function can be preferable. Due to the quadratic growing penalties of RMSE, one outlier in a batch can govern the total loss over a batch, which can be undesirable.

## 4.2 Dataset

Precise Biometrics has a vast collection of fingerprint images that the models were trained on. Five datasets containing images from different data collection occurrences were extracted and served as a raw, pre-processed dataset.

### 4.2.1 Image extraction

Out of the five datasets, two were reserved as test sets. All images in the five datasets were passed through the reference ISP (Image Signal Processor). For every image, three different levels of ISP-processed images were saved; a raw image, a denoised image and the estimated background (i.e. noise) of the image. The saved images coming from the five selected dataset-folders made up a database here called the primary dataset, and were used to create a so called "synthetic dataset".

### 4.2.2 Synthetic dataset

The training data generation consisted of

1. Overtraining a model on the primary dataset images
2. Defining a metric for grading a model output image
3. Setting a threshold for what constitutes a high-quality image
4. Saving images deemed high-quality
5. Creating synthetic degraded images from the high-quality images.

For overtraining, a simple U-Net model without any add-ons was used. The defined metric was similar to SSIM, and consisted of a combination of coherence, contrast and brightness. The idea behind this first training stage was to convert low-quality images into cleaner, higher-quality images, which the final model could learn from more effectively.

The so called coherence measure was chosen because it provides a reliable measurement of the local orientation consistency, which is an essential property in fingerprint analysis. Areas with consistent ridge flow display high coherence, while noisy or irregular regions produce low coherence, making this an efficient measurement to measure fingerprint image quality.

Firstly, the horizontal and vertical gradients were computed by a Sobel filter. Since orientation coherence is a local measure, the gradients were split into $n \times n$ non-overlapping blocks. The horizontal and vertical gradients at pixel $p$ are given by $G_x(p)$ and $G_y(p)$ respectively. The local orientation coherence $C$ of block $B$ was then calculated according to:

$$C = \frac{\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}}{\max(10^{-6}, G_{xx} + G_{yy})},$$

where

$$G_{xx} = \sum_{p \in B} G_x(p)G_x(p), \quad G_{yy} = \sum_{p \in B} G_y(p)G_y(p), \quad G_{xy} = \sum_{p \in B} G_x(p)G_y(p)$$

[2]. The max operator in the denominator ensures that division by zero is avoided. The global coherence score used for the evaluation is obtained by averaging the local coherence values over all blocks:

$$C_{global} = \frac{1}{N} \sum_{N} C_{local}.$$

For the contrast term, the $N \times N$ pixel images were subdivided into $n \times n$ pixel grids. For each such grid the mean light intensity and light intensity standard deviation was computed:

$$\mu_I(\mathbf{X}) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j} \ ,$$

$$\sigma_I(\mathbf{X}) = \sqrt{\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (x_{i,j} - \mu_I(\mathbf{X}))^2} \ ,$$

where $\mathbf{X}$ is an $n \times n$ subgrid. A local contrast value was then computed as:

$$c(\mathbf{X}) = \frac{\sigma(\mathbf{X})}{\max(10^{-6}, \mu(\mathbf{X}))} \ ,$$

where the max operator is used to avoid any possible zero-division. Finally the contrast for a full image $\mathbf{I}$ was simply defined as the mean of the contrasts of its subdivisions:

$$c(\mathbf{I}) = \sum_{k=1}^{(\frac{N}{n})^2} c(\mathbf{X}_k) \ .$$

Here it is assumed that $n|N$.

Finally, the brightness term was calculated as a weighted mean intensity value. The average brightness of the entire image was calculated as

$$\mu(\mathbf{I}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} I_{i,j} \ .$$

This was then weighted with the gaussian function

$$f(x) = e^{-\frac{(x-0.5)^2}{(2\sigma^2)}}$$

with $\sigma = 0.3$. Thus

$$b(\mathbf{I}) = f(\mu(\mathbf{I})) = e^{-\frac{(\mu(\mathbf{I})-0.5)^2}{(2\cdot 0.3^2)}} \quad .$$

The purpose of the gaussian was to promote images that were not completely black or completely white. It was desirable to get images with both brighter areas and darker areas (i.e. the ridges and valleys of the fingerprint). Of course the brightness term on its own also rates completely gray (i.e. pixel value 0.5) images highly. Ideally the contrast term and suitable threshold counteracts this.

The final quality score was then computed as

$$q(\mathbf{I}) = \frac{1}{2}C_{global}(\mathbf{I}) + \frac{1}{4}c(\mathbf{I}) + \frac{1}{4}b(\mathbf{I}) \quad .$$

The threshold for selecting good images was chosen by trial and error. It was gradually increased from 0.25 until the selected images looked the best. For threshold values set to higher 0.55 the images stopped getting visually better, and instead just got fewer in number. Thus this was the final threshold value used. This resulted in a training dataset of 2787 images. This dataset contained a mix of model-outputs from the over-trained model and denoised images from the primary dataset. Both of the images' qualities were measured, and the higher-quality one was chosen.

### 4.2.3 Augmentations

Since the training dataset was relatively small, a simple image-flipping augmentation was implemented. Before an image was deliberately degraded it would be copied and flipped either horizontally or vertically.

### 4.2.4 Image degradation

The degradation algorithm used a clean, high-quality image and degraded it. The decisions for how an image was degraded was merely a result of observing the raw images and adding similar attributes. The degradation consisted of three steps:

1. Adding background
2. Darkening the image
3. Adding a speck of light to the image border

The adding of the background is there simply to simulate the noise that appear from sensor inference. Some images were notably darker, which is why some images were also darkened. Finally, in some images a speck of light appears somewhere around the fingerprint. In order to simulate actual data as close as possible, all of these attributes were used.

Step 1 consisted of choosing two of the background images that was saved from the reference ISP and blending them together:

$$B = \alpha B_1 + (1 - \alpha)B_2, \quad \alpha \sim U(0, 1) \ ,$$

where $U(a, b)$ denotes a continuous uniform distribution in the interval $[a, b]$. This was then combined with the clean image $I$ to conclude the first step of the degradation:

$$I^{(1)} = \beta I + (1 - \beta)B, \quad \beta \sim U(0.10, 0.18).$$

Every generated image had a 15% chance of being darkened, as this was about the proportion observed in the raw dataset. When applied, darkening is calculated simply as:

$$I^{(2)} = \gamma I^{(1)} \quad \gamma \sim U(0.30, 0.50)$$

If an image was darkened, there was also a 50% chance it would get a speck of light added along one of the image's four borders at random. An initial light speck was defined as:

$$L = l\mathbf{1}, \quad l \sim U\{30, ..., 40\}, l \in \mathbb{N}$$

where

$$L \in \mathbb{N}^{s \times s}, \quad s \sim U\{200, ...225\},$$

that is a square matrix, with all entries equal to $l$. To achieve the radial falloff of a point light source, a gaussian falloff mask was used. Two matrices of coordinates over $L$ was defined as

$$X_{i,j} = -\lfloor \frac{(s-1)}{2} \rfloor + i, \quad i, j \in \{0, 1, ..., s-1\}$$

$$Y_{i,j} = -\lfloor \frac{(s-1)}{2} \rfloor + j, \quad i, j \in \{0, 1, ..., s-1\}$$

The light falloff over $L$ was then computed as

$$M_{i,j} = \exp\big(-\frac{X_{i,j}^2 + Y_{i,j}^2}{2\sigma^2}\big).$$

Here, $\sigma = 5$, as a result of trying several values, and choosing the one which resulted in most similar light artifacts as in the raw data. This falloff mask was normalized

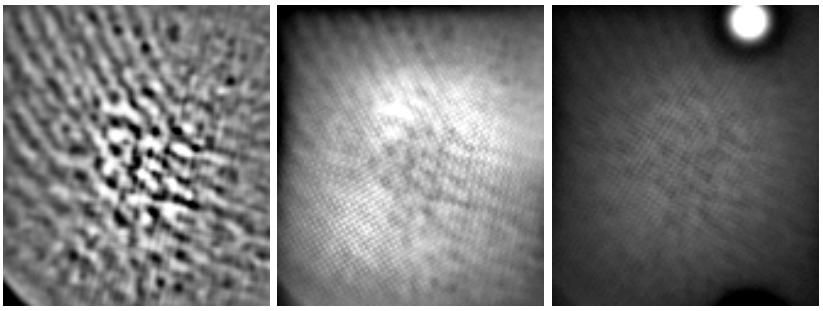
$$\tilde{M}_{i,j} = \frac{M_{i,j}}{\max(M_{i,j})}.$$

The final light was computed

$$\hat{L} = L \odot \tilde{M}$$

and added on top of the degraded $I^{(2)}$ image in the chosen spot.

This process was done for the 2787 high-quality images, resulting in a dataset of noisy (degraded) images and corresponding ground-truths, that were used in model training. Below is a sequence of images, demonstrating a fingerprint's path through the degradation process.



(a) (b) (c)

Figure 6: An example degradation process. Figure (a) is the image to be degraded. (b) has been degraded by adding backgrounds, i.e. noise. (c) shows the darkening of an image, and the addition of a light speck.

## 4.3 Model training

### 4.3.1 Preprocessing

All images went through several steps of preprocessing before being used for training, validation or testing, to ensure they were clean and appropriately sized. Assume an input image of size $m \times n$ with 8 bit unsigned integer pixel values, i.e. in the range [0, 255]. First the image's pixel values are transformed to a floating point value in the range [0, 1]. They were then cropped to the necessary input size $144 \times 144$ pixels. The cropping was done in the images' center, in an attempt to preserve the most interesting regions of the fingerprints. Lastly, a lowpass filter was applied (with appropriate threshold) to get rid of high frequency noise that mutes the signal of interest. The noise varies somewhat depending on several factors, such as sensor type, amount of pressure on the screen etc. For some images the noise appears as concentric circles and others as a grid-like pattern (see Figure 6(b) for example of grid-like noise). The frequency and intrusiveness of the noise also varies. However, for a specific dataset the noise is relatively uniform. Therefore per-dataset lowpass thresholds were manually tuned through experimentation. This was a matter of trying different thresholds for the different datasets, and picking the lowest threshold that still removed the high frequency noise.

### 4.3.2 Training Setup

The neural networks in this work were trained in a supervised learning setup, where each degraded image was paired with its corresponding clean ground truth image. Training was performed on mini-batches, where each batch was passed through the network to produce the restored images. The difference between the predicted output and target image was measured by a chosen loss function (see Section 4.3.4).

All models were trained using the Adam optimizer with default parameters ($\beta_1 = 0.9, \beta_2 = 0.999$). To monitor the models performance a validation loss was calculated on the validation set after each epoch. If the validation loss did not reduce in 10 epochs, training was stopped in order to prevent overfitting.

### 4.3.3 Models

The fundamental model that was trained and evaluated was a simple U-Net model, as described in section 4.1.7. The following models were also based on the U-Net model, but with one or several additional add-ons, that is convolutional block attention modules (Section 4.1.8) dilated convolutions (Section 4.1.10) and gated skip-connections (Section 4.1.9). Finally, a smaller U-Net network was also trained with and without add-ons. Instead of the five encoder layers, the smaller models had four encoder layers, and 256 channels in the bottleneck instead of 1024. A complete list of models are listed below.

- U-Net
- U-Net-CBAM
- U-Net-CBAM-DC
- U-Net-GS
- U-Net-GS-CBAM
- U-Net-256-GS
- U-Net-256-GS-CBAM

In the list above, CBAM denotes that the model had a convolutional block attention module with reduction ratio 8, located in the U-Net's bottleneck, i.e. in the final encoder layer. DC means that the U-Net used a dilated convolution (Section 4.1.10) in the bottleneck. The GS-suffix denotes that the model used gated skip connections (Section 4.1.9) for all skip connections.

All of the listed models were trained with batch size 16 and 8, as well as learning rate $10^{-3}$, $10^{-4}$ and $10^{-5}$. It is also important to note that every model used a sigmoid activation function in the output layer to ensure stable pixel values. Additionally, every model was trained both using the basic synthetic dataset, and an augmented synthetic dataset as described in section 4.2.3. Thus a total of $2 \cdot 3 \cdot 2 \cdot 7 = 84$ models were trained and evaluated.

### 4.3.4 Loss function

The loss function used was a combination of MSSIM (30) and MAE (31). The window size used for the MSSIM was $9 \times 9$ pixels. As opposed to MAE, the MSSIM does in fact not measure how two images differ, but on the contrary how similar they are. Keeping equation (7) in mind, it is desirable that the loss function is 0 when the output is identical to the input. In that case, it is no longer desirable for the weights to update, as a minima has been reached[1]. Instead of using MSSIM as a loss function, the following is used:

$$\text{MSSIM}_{\text{LOSS}}(\mathbf{x}, \mathbf{y}) = 1 - \text{MSSIM}(\mathbf{x}, \mathbf{y}). \tag{32}$$

This will be in the desired range $[0, 1]$ and

$$\text{MSSIM}_{\text{LOSS}}(\mathbf{x}, \mathbf{x}) = 0, \forall \mathbf{x}.$$

[1] Of course there is a risk that it is a local minimum.

Additionally, $\text{MSSIM}_{\text{LOSS}}(\mathbf{x}, \mathbf{y}) = 1$ when the structural properties between $\mathbf{x}$ and $\mathbf{y}$ differ the most[2]. With both MAE and $\text{MSSIM}_{\text{LOSS}}$ defined, the final loss function $\mathcal{L}$ was defined as

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{y}) &= \alpha \cdot \text{MSSIM}_{\text{LOSS}}(\mathbf{x}, \mathbf{y}) + (1 - \alpha) \cdot \text{MAE}(\mathbf{x}, \mathbf{y}) \\ &= \alpha \cdot \left(1 - \frac{1}{M} \sum_{i=1}^{M} \text{SSIM}(\mathbf{x}_i, \mathbf{y}_i)\right) + (1 - \alpha) \cdot \frac{1}{N} \sum_{j=1}^{N} |y_j - x_j|\end{aligned} \tag{33}$$

where $\alpha = 0.85$. This value was acquired through trial and error. The weighting factor $\alpha = 0.85$ prioritize structural preservation while still maintaining a contribution from the pixel-wise loss. The SSIM term encourages the network to preserve structural information, such as edges and textures, which is vital in tasks such as fingerprint enhancements. The MAE loss, on the other hand, ensures the model does not drift too far from the ground truth, ensuring similarity in pixel intensity and applies a penalty to fine reconstruction errors. This balance between maintaining structural information and penalizing pixel differences encourages the network to produce outputs that are both visually and numerically accurate. The MSSIM function used was from the `pytorch_msssim` library in which $\alpha = \beta = \gamma = 1$, and $K_1 = 0.01, K_2 = 0.03$.

## 4.4 Model Evaluation

For every model three sets of images were saved. These were:

- The predicted (output) images
- The ISP-denoised images
- The ISP-denoised images with a lowpass filter applied, where the lowpass filter is specifically selected for each dataset.

These images were then put through the rest of the ISP before they were evaluated using the existing reference fingerprint matcher from Precise Biometrics. For a matching attempt two images $I_{\text{input}}$ and $I_{\text{truth}}$ are used, where $I_{\text{input}}$ is evaluated to verify that it is the same finger as in $I_{\text{truth}}$. During the matching, $I_{\text{input}}$ may deliberately be another finger than in $I_{\text{truth}}$, a so called impostor. For such cases, a rejection is expected. Given this, a false accept rate (FAR) and false reject rate (FRR) was calculated and plotted against each other. This process was done for all three image sets for each model.

All trained models were first evaluated with few impostors, and the models that looked the most promising - i.e. the lowest increase in FRR when FAR decreased - were chosen to be evaluated further. For this further evaluation, 200.000 impostors were used.

[2]What this actually means in the context of fingerprints is, unfortunately, not trivial.

# 5 Results

Here, the result of model training and evaluation is presented. Below are graphs plotting the FRR against FAR for some of the trained models. A complete list of model training statistics can be found in appendix A.

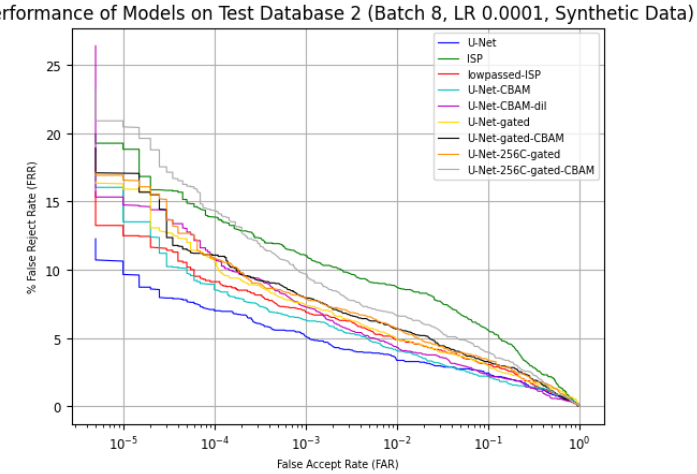



Figure 7: Matching evaluation for all models with batch size 8 and learning rate $10^{-4}$, as well as the ISP and lowpassed-ISP output.

The exact FRR percentages per FAR of the five best models on test Dataset 2, ISP-output and lowpassed-ISP-output were:

| Model \ FAR | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ |
|---|---|---|---|---|---|
| **U-Net** BS=8, LR=$1 \times 10^{-4}$ | 9.66 | 7.00 | 5.12 | 3.37 | 2.44 |
| **U-Net** BS=16, LR=$1 \times 10^{-4}$ | 10.41 | 7.66 | 5.72 | 3.84 | 2.26 |
| **U-Net-GS** BS=8, LR=$1 \times 10^{-5}$, Flip Aug. | 11.03 | 7.18 | 4.41 | 2.88 | 1.33 |
| **U-Net-256C-GS** BS=8, LR=$1 \times 10^{-4}$, Flip Aug. | 11.08 | 7.09 | 5.28 | 3.37 | 1.99 |
| **U-Net-CBAM-dil** BS=8, LR=$1 \times 10^{-5}$, Flip Aug. | 11.87 | 8.89 | 5.58 | 3.63 | 1.95 |
| **Lowpassed ISP** | 12.85 | 8.99 | 7.18 | 5.05 | 3.10 |
| **ISP** | 18.17 | 12.85 | 11.03 | 9.00 | 6.27 |

Table 1: False Rejection Rate (FRR) at different False Acceptance Rates (FAR) for the top five performing models on test Dataset 2.

A complete list of model evaluation statistics can be found in appendix B. The

same models as in figure 7 were evaluated on Dataset 1 and presented in figure 8.

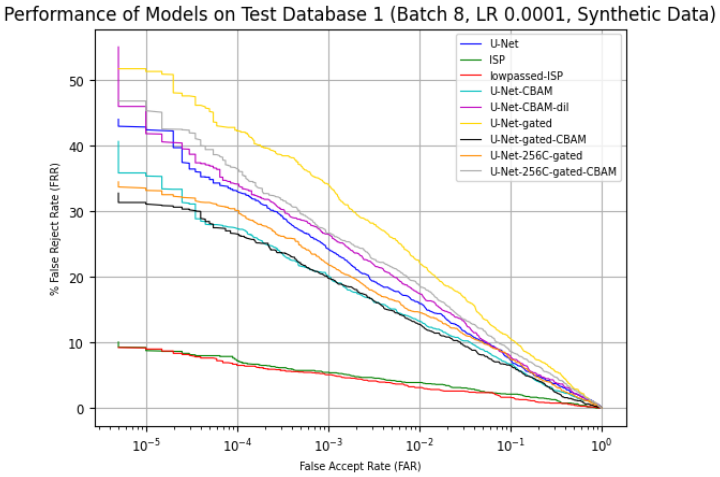
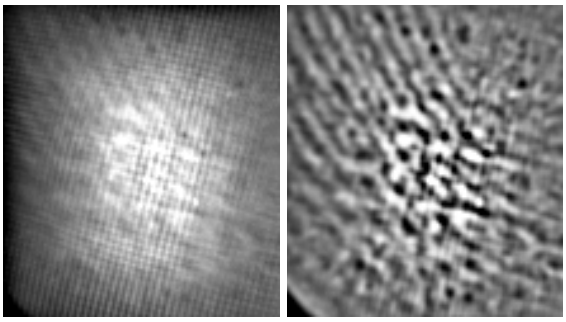



Figure 8: Matching evaluation for all models with batch size 8 and learning rate $10^{-4}$, as well as the ISP and lowpassed-ISP output.

| Model \ FAR | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ |
|---|---|---|---|---|---|
| **U-Net-CBAM** BS=16, LR=$1 \times 10^{-3}$ | 19.26 | 16.30 | 10.46 | 7.23 | 3.86 |
| **U-Net-256C** BS=16, LR=$1 \times 10^{-5}$ | 22.94 | 18.91 | 12.95 | 8.60 | 4.28 |
| **U-Net-256C-GS-CBAM** BS=8, LR=$1 \times 10^{-5}$. | 23.29 | 19.86 | 13.16 | 8.24 | 3.48 |
| **U-Net-GS-CBAM** BS=16, LR=$1 \times 10^{-3}$, Flip Aug. | 24.90 | 18.08 | 12.00 | 7.23 | 3.26 |
| **U-Net-GS-CBAM** BS=16, LR=$1 \times 10^{-3}$, | 27.21 | 20.55 | 13.89 | 8.55 | 4.27 |
| **Lowpassed ISP** | 7.47 | 6.05 | 4.15 | 2.85 | 1.60 |
| **ISP** | 12.63 | 9.96 | 7.35 | 4.99 | 2.90 |

Table 2: False Rejection Rate (FRR) at different False Acceptance Rates (FAR) for the top five performing models on test Dataset 1.

An example of a restored fingerprint is presented below.



(a) (b)

Figure 9: A raw fingerprint image (a) and the corresponding restored fingerprint image (b).

# 6 Discussion

For e.g. mobile phones using biometric security, clearly it is optimal if the owners biometry gets accepted every time, while an impostor gets rejected every time. Thus for biometric matching it is desirable to have FRR and FAR as low as possible. As is clear from the plots in section (5), the problem lies in attaining both low FRR and FAR simultaneously. In real-life scenarios it is much more acceptable with a higher FRR than a high FAR. After all it is preferable that a device is resistant to unauthorized access than it admitting the owner first try every time. Thus for this use case, we value a low FAR higher than a low FRR. For most of the discussed points, the "FRR value" refers to the FRR value at FAR $= 10^{-5}$. However, it is important to note that the 200.000 impostors that were used during evaluation (Section 4.4) is probably too few to get a very good estimate at FAR $= 10^{-5}$. Thus the estimate at FAR $= 10^{-4}$ is also relevant, since it is likely a better, less noisy estimate.

## 6.1 Model Evaluation Metrics

During model evaluation, a model was considered well-performing if it achieved a low FRR value. The goal was to produce models that performed better than the ISP-output, and hopefully lowpassed-ISP-output as well. As shown in tables B10, B11, B12, and B13, these usually perform better with regards to FRR at FAR $= 10^{-5}$. Considering that the aim is to improve the existing ISP prediction, test losses between the models' predictions and ISP output are not a primary evaluation criterion when determining the best-performing model. However a significantly elevated test loss may be concerning. Since the chosen loss function uses a combination of SSIM and MAE, a high test loss may indicate large structural differences between the prediction and ISP output. This is undesirable for fingerprint images, since any alteration of the fingerprint patterns can compromise their integrity and reliability. The objective is to enhance the fingerprints, not to modify them.

From looking at the validation losses in comparison with the models' FRR, there is no clear correlation between low validation loss and low FRR. It is

very possible that the loss function poorly reflects what constitutes a good reconstructed image from a biometrical matching perspective. It is possible the MAE-error component to the loss function causes this mismatch. Among the images from the most successful model in terms of FRR, many of them contain significantly less information to the human eye. A common trait among these is that several areas of the fingerprint has become far brighter. Since the MAE punishes for differing pixel values, perhaps the images that exhibit this trait (and thus would match better) gets punished.

It is also important to note that the ISP itself is tunable with certain parameters. While this was somewhat explored, no particular parameter setup that significantly affected any FRR-scores was attained. However, it is not impossible that the default-values are specified in such a way that the ISP expects images of a certain characteristic, and that further exploration of these settings could improve the matching process.

## 6.2 Test Datasets

From the plots and tables in section (5) it is clear that all models perform better on Dataset 2 than Dataset 1, while the ISP and lowpassed ISP perform consistently well on both datasets. Verily, the images in Dataset 1 differ a lot from the images that were collected for the synthetic (training) dataset (Section 4.2.2). The fingerprint images in Dataset 1 contain far fewer features with relevant information, i.e. distinct ridges and valleys, than those in the synthetic dataset. In addition, most images in the synthetic dataset are of fingerprints that cover most of the image, while Dataset 1 have images only partially covered by the finger. It is not unlikely that the models hallucinate in the empty areas (i.e. the background) of such images, which might inhibit the matching process.

## 6.3 Learning Rate

Changing the learning rate yielded drastically different results. With learning rate $10^{-3}$, some models (e.g. U-Net) fail completely, and just output completely black images which then crash the matching process. While training U-Net, it almost always got stuck at or around validation loss 0.868. This is likely an indicator that the learning rate $10^{-3}$ was too high. The ADAM optimizer updates the weights according to equation 13. The second moment estimate $\hat{v}_t$ is proportional to the gradient, a small gradient over time yields a small second moment estimate. Since ADAM scales the step size by $\frac{1}{\sqrt{\hat{v}_t}+\epsilon}$, a too small $\hat{v}_t$ can result in an unexpectedly large weight update when using a learning rate of 0.001, which can potentially cause the model to overshoot the optimum. Some models (e.g. U-Net-CBAM and U-Net-256C-gated-CBAM) still reach decent performance with learning rate 0.001 and in certain cases out-perform the ISP.

The learning rate $10^{-4}$ seems to be the most stable among the three tested learning rates, with fewer outliers across models. Models trained with this learning rate often achieves lower validation losses and generally a higher number of epochs until early stopping, compared to the models with learning rate set to $10^{-3}$. This suggests that models with a higher learning rate often fail to converge or collapse to trivial solutions, indicating that a learning rate set to $10^{-4}$ better balances convergence speed and stability.

Models generally converged slowly with the lowest learning rate $10^{-5}$, often requiring more epochs than higher learning rates. Despite longer trainings, the validation loss was frequently higher than for $10^{-4}$, signifying underfit. The patience window was set to 10 for models in tables B10, B11, B12 and B13 which might have been too short for such a small learning rate. When weights are updated in very small steps, escaping a local minimum may take more than 10 epochs, causing premature early stopping. Overall, no learning rate was the best across all models and datasets. The optimal learning rate depended on model architecture, batch size and dataset characteristics.

## 6.4 Batch Size

The results (Tables B12 and B10) indicate both batch size 8 and 16 may lead to instability, where a higher learning rate ($10^{-3}$) causes some models to collapse to a trivial solution, regardless of batch size. Neither of the batch sizes produce consistently better false reject rate, instead model architecture and chosen learning rate primarily influence performance. Across the trainings, validation losses were only marginally lower for batch size 8 than for batch size 16, and this difference was not consistent across models and learning rates. For test Dataset 2 however, all model architectures achieved lower FRR than ISP output for at least one learning rate when trained with batch size 8, whereas with batch size 16, only five out of seven models fall below the baseline FRR. It can also be observed that the model which performs best for a given learning rate, typically performs the best across both batch sizes. This indicates the limited influence batch size has on the relative ranking between models.

## 6.5 Model architectures

Adding more complexity to U-Net (e.g. CBAM, gated skip connections and dilated convolution) sometimes improved FRR on Dataset 2, particularly for learning rates $10^{-3}$ and $10^{-5}$. On Dataset 1, the model with the lowest FRR for a given learning rate and batch size was frequently not the simple U-Net. However, while some improvements did occur, complex architectures more often worsened performance. Although architectural modifications may reduce FRR, the U-Net model achieves the lowest FRR across all learning rates and batch sizes, indicating its ability to reconstruct high quality fingerprints. Thus, increased architectural complexity does not translate to better performance, and simpler U-Net architecture may be more reliable. With a small training dataset, adding complexity increases the models' risk of overfitting which leads to poor generalization to unseen data. Therefore, the U-Net model may be more suitable when training data is limited.

## 6.6 Augmentations

The FRR values for Dataset 2 was generally lower when training included data augmentation in the form of image flips, especially for lower learning rates. At learning rate $10^{-4}$ and $10^{-5}$, many architectures exhibited reduced FRR when trained with flips. Particularly for $BS = 8$, where models trained on augmented data outperforms their non-augmented counterpart and ISP baseline. Increasing the dataset size reduced the number of model collapses at learning rate

$10^{-3}$ and produced more stable and less variable FRR values for lower learning rates. For Dataset 1, introducing augmentations during training has a neutral effect on FRR. For certain architectures and learning rates, data augmentation reduced the FRR, whereas for others it resulted in higher FRR. Overall, data augmentation yielded inconsistent results on this dataset and did not show the clear performance improvement observed with Dataset 2.

The validation losses for learning rate $10^{-3}$ did see some improvement, which can't be said for learning rate $10^{-4}$ nor $10^{-5}$. Despite this validation loss decrease for learning rate $10^{-3}$, these models did not achieve a better FRR-value than their non-augmentation trained counterparts. However, the models that crashed without augmentations, provide stable outputs when trained with augmentations.

## 6.7 Future Work

There is a lot of experiments that could be attempted, and changes that could be made to (hopefully) increase model performance further. The most important changes would likely be to change the quality metric used in selecting images for the synthetic dataset (4.2.2) to better reflect what images perform well during matching, and recreate the synthetic dataset. The synthetic dataset is relatively small ($\sim$2500 images) which is likely too small to train a general well-performing model. Thus it could be a good idea to lower the quality threshold in order to increase the synthetic dataset size. The current synthetic training dataset includes fingerprints that cover most of the images, however, further model improvement may benefit from having a wider variety of fingerprint images, such as fingerprints covering only part of the images.

Given the experiments with different patience windows, it is not unlikely that many models were in fact under-trained. While the dataset size does not seem large enough to incentivize a much larger patience window, it was observed that even when the model had trained for 3000 epochs (same order of magnitude as number of images), the model did not seem to over-train (see Figures 10 and 11. It would be interesting to observe how long a model has to train before it over-trains, to better customize and appropriate the number of training epochs.

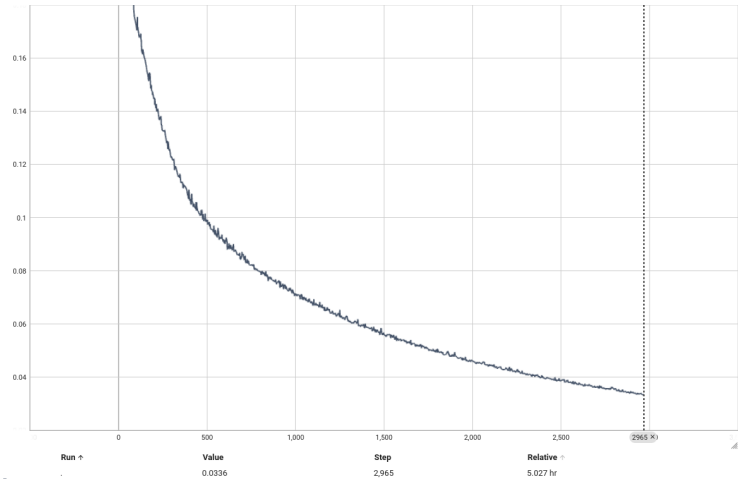



Figure 10: Train loss per epoch for U-Net-256C.

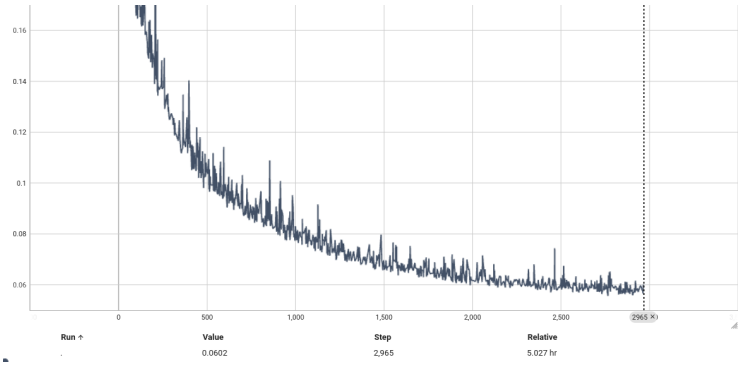



Figure 11: Validation loss per Epoch for U-Net-256C.

The loss function (Section 4.3.4) could likely be improved to better reflect what image properties improve the matching process. It would be interesting to decrease the proportion of the loss function that is made up of MAE, to decrease punishment for individual pixel brightness fluctuation. Similarly, the luminance component of SSIM may also contribute to the predicted images being more uniform in terms of brightness. Perhaps a version of SSIM that does not utilize this component (or at least decreases its impact) would increase model performance.

# 7 Conclusion

This thesis investigated the use of CNN-architectures for fingerprint image restoration to improve biometric verification performance at low FAR. Multiple architectural variants were trained on synthetically generated data and evaluated on two independent test datasets. The results show CNN-based architectures can outperform the ISP pipeline under well-suited conditions. In particular, for Dataset 2, multiple U-Net architectures yield lower FRR than

the ISP outputs, especially at learning rate $10^{-4}$. This indicates that learned restoration can preserve fingerprint details better than the current ISP. In addition, adding a lowpass to the ISP outputs consistently achieved lower FRR.

Learning rate was found to be the most influential hyperparameter. A learning rate of of $10^{-4}$ provided the best trade-off between convergence stability and fingerprint reconstruction quality. Higher learning rates often resulted in unstable trainings and a degenerate solution. The batch size hyperparameter had less impact, with batch size 8 occasionally leading to marginally lower FRR compared to batch size 16.

Despite the improvements on Dataset 2, the results on Dataset 1 suggests limited generalization. For this dataset, the ISP consistently performed better than the proposed CNN models. Although trainings on the augmented synthetic data improved reconstruction for the majority of models, it was insufficient to make the synthetic training data fully representative of the real fingerprint data in Dataset 1. This suggests that fingerprint and sensor characteristics significantly affect restoration performance.

Increasing the model complexity did not consistently improve matching performance. Instead, the U-Net model proved to be the most stable and reliable. This may stem from the fact that the training dataset was small, which leads to more complex models being susceptible to overfitting, potentially performing poorly when encountering new data.

Overall, the results indicate that a CNN-based approach is promising for fingerprint restoration, improving matching performance when training and testing data are well aligned. However, the strong dependence on dataset characteristics demonstrates the need for more representative and varied training data before such models can seriously challenge traditional ISP pipelines.

# A Model Tables: Validation and Test Losses

| Model | BS | Num Ep. | Val. Loss | Test: Dataset 1 | Test: Dataset 2 |
|---|---|---|---|---|---|
| UNet | 16 | 5 | 0.868 | 0.769 | 0.797 |
| UNet-CBAM | 16 | 3 | 0.236 | **0.452** | 0.410 |
| UNet-CBAM-DC | 16 | 91 | 0.117 | 0.512 | 0.395 |
| UNet-GS | 16 | 8 | 0.868 | 0.769 | 0.797 |
| UNet-GS-CBAM | 16 | 41 | 0.144 | 0.500 | 0.412 |
| Unet-256C-GS | 16 | 73 | 0.0717 | 0.517 | 0.366 |
| Unet-256C-GS-CBAM | 16 | 116 | **0.0562** | 0.545 | 0.379 |
| UNet-256C | 16 | 58 | 0.0696 | 0.525 | **0.349** |
| UNet | 8 | 85 | 0.0786 | 0.497 | 0.369 |
| UNet-CBAM | 8 | 66 | 0.124 | 0.519 | 0.408 |
| UNet-CBAM-DC | 8 | 28 | 0.868 | 0.769 | 0.797 |
| Unet-GS | 8 | 88 | 0.141 | **0.490** | 0.428 |
| UNet-GS-CBAM | 8 | 4 | 0.868 | 0.770 | 0.797 |
| Unet-256C-GS | 8 | 78 | 0.0672 | 0.544 | 0.369 |
| Unet-256C-GS-CBAM | 8 | 105 | **0.0566** | 0.556 | **0.353** |

Table 3: Models trained with learning rate $10^{-3}$ on dataset without augmentations.

| Model | BS | Num Ep. | Val. Loss | Test: Dataset 1 | Test: Dataset 2 |
|---|---|---|---|---|---|
| UNet | 16 | 6 | 0.868 | 0.550 | 0.417 |
| UNet-CBAM | 16 | 78 | 0.112 | 0.537 | 0.403 |
| UNet-CBAM-DC | 16 | 73 | 0.109 | 0.533 | 0.396 |
| UNet-GS | 16 | 92 | 0.143 | 0.490 | 0.418 |
| UNet-GS-CBAM | 16 | 18 | 0.159 | **0.476** | 0.393 |
| Unet-256C-GS | 16 | 86 | 0.106 | 0.532 | 0.396 |
| Unet-256C-GS-CBAM | 16 | 85 | **0.0652** | 0.567 | **0.357** |
| UNet | 8 | 1 | 0.247 | **0.479** | 0.426 |
| UNet-CBAM | 8 | 62 | 0.123 | 0.522 | 0.396 |
| UNet-CBAM-DC | 8 | 89 | 0.110 | 0.536 | 0.391 |
| Unet-GS | 8 | 9 | 0.787 | 0.689 | 0.713 |
| UNet-GS-CBAM | 8 | 3 | 0.762 | 0.645 | 0.672 |
| Unet-256C-GS | 8 | 96 | **0.0710** | 0.548 | **0.367** |
| Unet-256C-GS-CBAM | 8 | 111 | 0.0989 | 0.528 | 0.387 |

Table 4: Models trained with learning rate $10^{-3}$ on dataset with augmentations.

| Model | BS | Num Ep. | Val. Loss | Test: Dataset 1 | Test: Dataset 2 |
|---|---|---|---|---|---|
| UNet | 16 | 90 | 0.0454 | **0.553** | **0.328** |
| UNet-CBAM | 16 | 105 | **0.0377** | 0.554 | 0.340 |
| UNet-CBAM-DC | 16 | 112 | 0.0387 | 0.566 | 0.350 |
| UNet-GS | 16 | 70 | 0.0455 | 0.607 | 0.345 |
| UNet-GS-CBAM | 16 | 51 | 0.0632 | 0.597 | 0.359 |
| Unet-256C-GS | 16 | 74 | 0.0861 | 0.574 | 0.372 |
| Unet-256C-GS-CBAM | 16 | 63 | 0.0864 | 0.561 | 0.393 |
| UNet-256C | 16 | 123 | 0.0693 | 0.570 | 0.348 |
| UNet | 8 | 62 | 0.0428 | 0.576 | 0.339 |
| UNet-CBAM | 8 | 96 | 0.0395 | 0.573 | **0.330** |
| UNet-CBAM-DC | 8 | 70 | **0.0382** | 0.573 | 0.341 |
| Unet-GS | 8 | 99 | 0.0466 | 0.604 | 0.353 |
| UNet-GS-CBAM | 8 | 95 | 0.0388 | **0.545** | 0.334 |
| Unet-256C-GS | 8 | 70 | 0.0746 | 0.597 | 0.376 |
| Unet-256C-GS-CBAM | 8 | 72 | 0.0719 | 0.552 | 0.355 |

Table 5: Models trained with learning rate $10^{-4}$ on dataset without augmentations.

| Model | BS | Num Ep. | Val. Loss | Test: Dataset 1 | Test: Dataset 2 |
|---|---|---|---|---|---|
| UNet | 16 | 57 | 0.0451 | 0.552 | 0.343 |
| UNet-CBAM | 16 | 86 | 0.0405 | 0.520 | **0.327** |
| UNet-CBAM-DC | 16 | 124 | **0.0389** | 0.523 | 0.335 |
| UNet-GS | 16 | 81 | 0.0479 | 0.533 | 0.350 |
| UNet-GS-CBAM | 16 | 79 | 0.0455 | 0.541 | 0.342 |
| Unet-256C-GS | 16 | 85 | 0.0735 | 0.577 | 0.363 |
| Unet-256C-GS-CBAM | 16 | 192 | 0.0533 | **0.516** | 0.350 |
| UNet | 8 | 54 | 0.0452 | 0.557 | 0.339 |
| UNet-CBAM | 8 | 68 | **0.0395** | 0.539 | 0.337 |
| UNet-CBAM-DC | 8 | 99 | 0.0409 | **0.474** | **0.331** |
| Unet-GS | 8 | 85 | 0.0488 | 0.580 | 0.351 |
| UNet-GS-CBAM | 8 | 92 | 0.0439 | 0.539 | 0.336 |
| Unet-256C-GS | 8 | 55 | 0.0691 | 0.588 | 0.361 |
| Unet-256C-GS-CBAM | 8 | 124 | 0.0528 | 0.540 | 0.363 |

Table 6: Models trained with learning rate $10^{-4}$ on dataset with augmentations.

| Model | BS | Num Ep. | Val. Loss | Test: Dataset 1 | Test: Dataset 2 |
|---|---|---|---|---|---|
| UNet | 16 | 44 | 0.124 | 0.556 | 0.384 |
| UNet-CBAM | 16 | 131 | **0.0875** | 0.547 | 0.384 |
| UNet-CBAM-DC | 16 | 89 | 0.0883 | 0.592 | **0.357** |
| UNet-GS | 16 | 68 | 0.102 | 0.600 | 0.362 |
| UNet-GS-CBAM | 16 | 44 | 0.130 | 0.575 | 0.394 |
| Unet-256C-GS | 16 | 73 | 0.170 | 0.568 | 0.361 |
| Unet-256C-GS-CBAM | 16 | 139 | 0.149 | 0.550 | 0.417 |
| UNet-256C | 16 | 100 | 0.149 | **0.535** | 0.412 |
| UNet | 8 | 109 | 0.0697 | 0.609 | 0.342 |
| UNet-CBAM | 8 | 104 | **0.0678** | 0.603 | **0.332** |
| UNet-CBAM-DC | 8 | 93 | 0.0730 | 0.616 | 0.338 |
| Unet-GS | 8 | 95 | 0.0731 | 0.598 | 0.339 |
| UNet-GS-CBAM | 8 | 109 | 0.0692 | 0.604 | 0.354 |
| Unet-256C-GS | 8 | 162 | 0.127 | 0.579 | 0.420 |
| Unet-256C-GS-CBAM | 8 | 109 | 0.133 | **0.554** | 0.413 |

Table 7: Models trained with learning rate $10^{-5}$ on dataset without augmentations.

| Model | BS | Num Ep. | Val. Loss | Test: Dataset 1 | Test: Dataset 2 |
|---|---|---|---|---|---|
| UNet | 16 | 114 | **0.0702** | 0.625 | 0.368 |
| UNet-CBAM | 16 | 99 | 0.0731 | 0.632 | 0.360 |
| UNet-CBAM-DC | 16 | 69 | 0.0934 | 0.617 | 0.368 |
| UNet-GS | 16 | 57 | 0.0885 | 0.608 | **0.342** |
| UNet-GS-CBAM | 16 | 80 | 0.0791 | 0.603 | 0.349 |
| Unet-256C-GS | 16 | 97 | 0.159 | **0.571** | 0.407 |
| Unet-256C-GS-CBAM | 16 | 171 | 0.122 | 0.597 | 0.415 |
| UNet-256C | 16 | - | - | - | - |
| UNet | 8 | 89 | 0.0600 | 0.573 | 0.359 |
| UNet-CBAM | 8 | 91 | 0.0649 | 0.610 | 0.369 |
| UNet-CBAM-DC | 8 | 153 | 0.0486 | 0.578 | **0.338** |
| Unet-GS | 8 | 156 | **0.0480** | 0.555 | 0.350 |
| UNet-GS-CBAM | 8 | 60 | 0.0746 | 0.617 | 0.345 |
| Unet-256C-GS | 8 | 85 | 0.131 | 0.595 | 0.394 |
| Unet-256C-GS-CBAM | 8 | 181 | 0.0997 | **0.553** | 0.388 |

Table 8: Models trained with learning rate $10^{-5}$ on dataset with augmentations.

| Model | BS | Num Ep. | Val. Loss | Test: Dataset 1 | Test: Dataset 2 |
|---|---|---|---|---|---|
| UNet | 16 | 1493 | 0.0394 | 0.548 | 0.349 |
| UNet-CBAM | 16 | 1243 | 0.0359 | 0.585 | 0.345 |
| UNet-CBAM-DC | 16 | 1773 | 0.0353 | 0.549 | 0.345 |
| UNet-GS | 16 | 1248 | 00392 | 0.574 | **0.343** |
| UNet-GS-CBAM | 16 | 1800 | 0.0369 | 0.562 | 0.350 |
| Unet-256C-GS | 16 | 2766 | 0.0556 | 0.566 | 0.361 |
| Unet-256C-GS-CBAM | 16 | 3531 | 0.0539 | 0.595 | 0.365 |
| UNet-256C | 16 | 3434 | 0.0500 | **0.534** | 0.362 |

Table 9: Models trained with learning rate $10^{-5}$ with patience window 200.

# B Model Tables: False Reject Rate

| Model | BS | lr0.001 | lr0.0001 | lr0.00001 |
|---|---|---|---|---|
| ISP | 0 | 17.5 | 16.57 | 17.46 |
| Lowpassed-ISP | 0 | 13.25 | 11.34 | 13.11 |
| UNet | 16 | *null* | **10.41**✓ | 30.26 |
| UNet-CBAM | 16 | 30.66 | 15.68✓ | 51.00 |
| UNet-CBAM-DC | 16 | 29.24 | 18.83 | **17.81** |
| UNet-GS | 16 | *null* | 13.78✓ | 19.32 |
| UNet-GS-CBAM | 16 | 27.12 | 12.67✓ | 57.47 |
| Unet-256C-GS | 16 | 18.65 | 20.43 | 26.10 |
| Unet-256C-GS-CBAM | 16 | 16.53✓ | 22.23 | 29.29 |
| UNet-256C | 16 | **14.04**✓ | 14.26✓ | 31.15 |
| ISP | 0 | 18.17 | 18.17 | 18.17 |
| Lowpassed-ISP | 0 | 12.85 | 12.85 | 12.85 |
| UNet | 8 | 21.71 | **9.66**✓ | 17.94✓ |
| UNet-CBAM | 8 | 31.59 | 13.51✓ | **16.66** ✓ |
| UNet-CBAM-DC | 8 | *null* | 14.75✓ | 19.01 |
| Unet-GS | 8 | 42.36 | 15.91✓ | 21.44 |
| UNet-GS-CBAM | 8 | *null* | 17.07✓ | 17.50✓ |
| Unet-256C-GS | 8 | 22.64 | 16.57✓ | 32.70 |
| Unet-256C-GS-CBAM | 8 | **16.04**✓ | 20.47 | 31.72 |

Table 10: The False Reject Rate (FRR) for Dataset 2 for each model trained with synthetic data. Yellow marker denotes the best model per learning rate for each model. Checkmark denotes the models that out-perform baseline (ISP).

| Model | BS | lr0.001 | lr0.0001 | lr0.00001 |
|---|---|---|---|---|
| ISP | 0 | 18.17 | 18.17 | 16.44 |
| Lowpassed-ISP | 0 | 12.85 | 12.85 | 13.69 |
| UNet | 16 | 34.69* | **11.96** ✓ | 16.30* ✓ |
| UNet-CBAM | 16 | **22.42** * | 12.85* ✓ | 15.53* ✓ |
| UNet-CBAM-DC | 16 | 25.65* | 14.40 *✓ | 22.64 |
| UNet-GS | 16 | 31.32* | 13.69* ✓ | 17.59* |
| UNet-GS-CBAM | 16 | 23.66* | 21.80 | **14.53*** ✓ |
| Unet-256C-GS | 16 | 25.96 | 13.60*✓ | 27.91 |
| Unet-256C-GS-CBAM | 16 | 23.30 | 19.98* ✓ | 28.09* |
| ISP | 0 | 19.14 | 17.72 | 19.89 |
| Lowpassed-ISP | 0 | 13.25 | 11.34 | 10.90 |
| UNet | 8 | 31.01 | 12.49 ✓ | 13.60 *✓ |
| UNet-CBAM | 8 | 26.27* | 15.86 ✓ | 15.24* ✓ |
| UNet-CBAM-DC | 8 | 25.61* | 13.69* ✓ | 11.87* ✓ |
| UNet-GS | 8 | null | 14.93* ✓ | **11.03** *✓ |
| UNet-GS-CBAM | 8 | null | 12.54 *✓ | 13.69* |
| Unet-256C-GS | 8 | **21.75*** | **11.08***✓ | 22.91* |
| Unet-256C-GS-CBAM | 8 | 24.06 | 19.48* | 25.79* |

Table 11: The False Reject Rate (FRR) for Dataset 2 for each model trained with augmented synthetic data.Yellow marker denotes the best model per learning rate for each model. Checkmark denotes the models that out-perform baseline (ISP). Models marked with * show a reduced FRR when trained on augmented data.

| **Model** | **BS** | **lr0.001** | **lr0.0001** | **lr0.00001** |
|---|---|---|---|---|
| ISP | 0 | 9.42 | 13.10 | 12.39 |
| Lowpassed-ISP | 0 | 8.83 | 9.19 | 7.41 |
| UNet | 16 | *null* | **31.10** | 38.59 |
| UNet-CBAM | 16 | **19.26** | 38.47 | 45.11 |
| UNet-CBAM-DC | 16 | 29.34 | 35.27 | 44.99 |
| UNet-GS | 16 | *null* | 41.02 | 47.00 |
| UNet-GS-CBAM | 16 | 27.21 | 49.91 | 53.35 |
| Unet-256C-GS | 16 | 32.42 | 40.84 | 42.62 |
| Unet-256C-GS-CBAM | 16 | 38.00 | 42.09 | 27.33 |
| Unet-256C | 16 | 38.47 | 39.48 | **22.94** |
| ISP | 0 | 8.65 | 8.65 | 8.65 |
| Lowpassed-ISP | 0 | 8.95 | 8.95 | 8.95 |
| UNet | 8 | **30.47** | 42.44 | 55.84 |
| UNet-CBAM | 8 | 32.31 | 35.39 | 48.73 |
| UNet-CBAM-DC | 8 | *null* | 41.85 | 52.87 |
| Unet-GS | 8 | 36.22 | 51.33 | 50.32 |
| UNet-GS-CBAM | 8 | *null* | **31.12** | 54.51 |
| Unet-256C-GS | 8 | 41.26 | 33.19 | 28.45 |
| Unet-256C-GS-CBAM | 8 | 32.84 | 45.34 | **23.29** |

Table 12: The False Reject Rate (FRR) for Dataset 1 for each model trained with synthetic data. Yellow marker denotes the best model per learning rate for each model. Checkmark denotes the models that out-perform baseline (ISP).

| **Model** | **BS** | **lr0.001** | **lr0.0001** | **lr0.00001** |
|---|---|---|---|---|
| ISP | 0 | 11.74 | 8.65 | 10.97 |
| Lowpassed-ISP | 0 | 8.95 | 8.95 | 8.89 |
| UNet | 16 | 27.38* | 37.70 | 65.97 |
| UNet-CBAM | 16 | 29.46 | 36.51* | 55.48 |
| UNet-CBAM-DC | 16 | 33.67 | **32.31*** | 55.78 |
| UNet-GS | 16 | 36.69* | 36.99 * | 47.90 |
| UNet-GS-CBAM | 16 | **24.90*** | 41.96* | 57.44 |
| Unet-256C-GS | 16 | 33.67 | 40.60* | **27.74*** |
| Unet-256C-GS-CBAM | 16 | 45.28 | 33.61* | 30.82 |
| ISP | 0 | 12.21 | 12.03 | 9.19 |
| Lowpassed-ISP | 0 | 8.83 | 8.77 | 8.42 |
| UNet | 8 | 29.52* | 33.91 | 51.75* |
| UNet-CBAM | 8 | **27.98*** | 36.42* | 52.87 |
| UNet-CBAM-DC | 8 | 34.32* | **27.62*** | 51.92 * |
| UNet-GS | 8 | null | 36.63* | **42.91*** |
| UNet-GS-CBAM | 8 | null | 29.52* | 48.84* |
| Unet-256C-GS | 8 | 41.49 | 41.26 | 30.05 |
| Unet-256C-GS-CBAM | 8 | 34.79 | 42.20* | 45.82 |

Table 13: The False Reject Rate (FRR) for Dataset 1 for each model trained with augmented synthetic data. Yellow marker denotes the best model per learning rate for each model. Checkmark denotes the models that out-perform baseline (ISP). Models marked with * show a reduced FRR when trained on augmented data.

# References


[1] Luca Barillaro. Artificial neural networks. In Shoba Ranganathan, Mario Cannataro, and Asif M. Khan, editors, *Encyclopedia of Bioinformatics and Computational Biology (Second Edition)*, pages 141–145. Elsevier, Oxford, second edition edition, 2025.

[2] A. M. Bazen and Sabih H. Gerez. Directional field computation for fingerprints based on the principal component analysis of local gradients. In *Proceedings of ProRISC 2000, Workshop on Circuits, Systems and Signal Processing*, pages 215–222, Veldhoven, 2000.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] Jason Brownlee. A gentle introduction to the rectified linear unit (relu). `https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/`, 2020. Accessed: 2025-12-03.

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[6] Luke Nicholas Darlow and Benjamin Rosman. Fingerprint minutiae extraction using deep learning. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 22–30, 2017.

[7] Duo Security. The 2022 duo trusted access report. `https://www.scribd.com/document/683316650/the-2022-duo-trusted-access-report`, 2022. Accessed: 2025-12-15.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] IBM. What are convolutional neural networks? `https://www.ibm.com/think/topics/convolutional-neural-networks`, n.d. Accessed: 30 October 2025.

[10] Keyless Technologies. Why biometric authentication is the future of identity security. `https://keyless.io/blog/post/why-biometric-auth-is-the-future-of-identity-security`. Accessed: 2025-12-15.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[12] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[13] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer, New York, NY, 2nd edition, 2009.

[14] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[16] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[17] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30:79–82, 2005.

[18] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[19] Shuning Xu, Binbin Song, Xiangyu Chen, Xina Liu, and Jiantao Zhou. Image demoireing in raw and srgb domains, 2024.

[20] Huiyu Zhou, Jiahua Wu, and Jianguo Zhang. Digital image processing: Part i, 2010. PDF available online.

Master's Theses in Mathematical Sciences 2026:E2
ISSN 1404-6342

LUTFMA-3602-2026

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/