Master thesis

Deblurring of Cell Images Using Generative Adversarial Networks

Cajsa Olofsson and Ida Wagnström

June 26, 2019

Supervisors:
Martin Almers, CellaVision AB
Niels Christian Overgaard, LTH

LUNDS UNIVERSITET

Abstract

The digital microscopes that CellaVision produces today use a mechanical auto focus when capturing cell images. This requires the camera objective to be vertically positioned with a precision of 0.4 μm in order for the objects in the images to be considered properly focused. Because of the small distances, the system is sensitive to vibrations and errors due to mechanical imperfections are hard to avoid. In order to replace the mechanical auto focus in the system, it would be desirable to digitally transform unfocused images to appear focused after they were captured.

In this thesis we investigate if it is possible to transform an unfocused cell image to a sharp one using generative adversarial networks (GANs). A data set of 10 786 sharp images were collected together with blurry images captured at different distances from the optimal focus, using CellaVision's systems. From implementing and comparing three already defined GANs - pix2pix, PAN and deblurGAN - we saw that pix2pix performed best for our problem. Moving on with pix2pix, we added losses and changed the network structure to improve the result. To make the network faster, we also changed the architecture of the generator network. We ended up with three different GANs, whereof one met the time requirement of transforming a 360×360 image in 70 ms on a CPU.

Generally, all of the three final networks managed to sharpen all of the blurry images to some extent, but not always to the desired focus. The best of the networks was able to successfully transform around 90% of images captured in the interval -0.8 to 0.8 μm from the optimal focus. The fastest network had the poorest performance of them, but still managed to successfully transform 78% of the blurry images from the same interval.

Acknowledgements

First, we wish to thank our supervisor at the department of Mathematics at LTH, Niels Christian Overgaard, for all the meetings and discussions that has helped us throughout the project.

We would like to thank CellaVision for giving us the opportunity to do this master thesis and for providing us with the necessary tools. We also wish to extend our gratitude to the employees of CellaVision for welcoming us and making us feel like a part of the team. The enjoyable breakfast time each morning with freshly baked bread has made our mornings fun and productive.

A huge appreciation goes to our supervisor at CellaVision, Martin Almers, for his enthusiastic, valuable and constructive guidance during the planning and development of this thesis. His generosity with his time and his many ideas has been much appreciated, and the results would not have been as good without his unwavering support. Additionally, we would like to thank last year's thesis workers, Jesper Jönsson and Emmy Sjöstrand, for the invaluable insight they provided about their previous work on GANs [12].

Special thanks to Olof Englund for the Git support and to Carl Brishammar for various tips during the project. At last, we would like to thank our families and friends for the encouragements they have given us all through our education.

Contents

1 Introduction **6**
- 1.1 Aim . . . 8
- 1.2 Image blurring and deblurring . . . 8
- 1.3 Focus measure . . . 10

2 Data **11**
- 2.1 Preprocessing of data . . . 14
  - 2.1.1 Finding the ground truth . . . 14
  - 2.1.2 Alignment . . . 16
- 2.2 Data sets . . . 17
- 2.3 Augmentation of data . . . 17

3 Deep learning **18**
- 3.1 Convolutional neural network (CNN) . . . 18
  - 3.1.1 Convolution . . . 19
  - 3.1.2 Non linearity . . . 19
  - 3.1.3 Pooling . . . 19
  - 3.1.4 Fully connected layer . . . 19
  - 3.1.5 Batch normalization . . . 19
- 3.2 Generative adversarial network (GAN) . . . 20
  - 3.2.1 Binary cross entropy loss . . . 20
- 3.3 Conditional generative adversarial network (CGAN) . . . 21
- 3.4 Wasserstein generative adversarial network (WGAN) . . . 22
- 3.5 U-net . . . 22
- 3.6 Residual network (ResNet) . . . 23

4 Method **24**
- 4.1 Pix2pix . . . 24
- 4.2 Perceptual adversarial network (PAN) . . . 26
- 4.3 DeblurGAN . . . 27
- 4.4 Training details . . . 29

5 Metrics for evaluation **30**
- 5.1 Mean squared error (MSE) . . . 30
- 5.2 Peak signal to noise ratio (PSNR) . . . 30
- 5.3 Structural similarity index (SSIM) . . . 30
- 5.4 Focus measure - ratio (FR) . . . 31
- 5.5 Focus measure - pixelwise (FPW) . . . 31
- 5.6 Visual assessment . . . 31

6 Results **32**
6.1 Pix2pix vs PAN vs deblurGAN . . . 33
6.2 Quality of data . . . 35
6.2.1 Amount of data . . . 35
6.2.2 Augmentation of data . . . 36
6.3 Additional losses . . . 36
6.3.1 Focus loss . . . 36
6.3.2 VGG loss . . . 37
6.3.3 Distance loss . . . 38
6.4 Two inputs . . . 40
6.5 Faster network . . . 41
6.6 Reaching convergence . . . 43
6.7 Unbalanced networks in pix2pix . . . 49

7 Discussion and conclusion **52**
7.1 Quality of results . . . 52
7.2 Model selection . . . 54
7.3 Potential risks with the method . . . 55
7.4 Faster network . . . 55
7.5 The effectiveness of GAN . . . 56
7.6 Quality of data . . . 57
7.7 Conclusions . . . 57
7.8 Future work . . . 57

Appendices **61**

A Tables for evaluation **61**

B Predictions **68**

List of notations

Table 1: Recurring notations used in the project.

WBC		White Blood Cell
DC-1		The CellaVision microscope used to obtain the images.
Focus level (FL)		The distance from the optimal position between the camera and the glass.
Stack		A set of images of the same cell of different focus levels.
PSF		Point Spread Function
F		Focus measure
CPU		Central processing unit
GPU		Graphics processing unit
CNN		Convolutional Neural Network
GAN		Generative Adversarial Network
CGAN		Conditional Generative Adversarial Network
WGAN		Wasserstein Generative Adversarial Network
GP		Gradient Penalty loss
G	=	Generator network
D	=	Discriminator network
X	=	(List of) blurred image(s)
Y	=	(List of) sharp image(s)
$\hat{Y}$	=	(List of) generated image(s) = $G(X)$
MSE		Mean Squared Error
PSNR		Peak Signal to Noise Ratio
SSIM		Structural Similarity Index
FR		Focus measure - Ratio
FPW		Focus measure - Pixelwise
SR		Success Rate

Chapter 1

Introduction

The analysis of blood is an integral part of modern health care and it is an important tool for doctors to discover diseases and confirm diagnoses. Traditionally, blood analyses are carried out manually by highly trained experts using optical microscopes to examine blood samples, which is both time consuming and labour intensive. Hospitals always seek to deliver better and more reliable results, reduce costs, speed up testing and improve productivity while also taking on an increasing number of samples. CellaVision produces digital microscopes and supporting software that replaces conventional microscopy to aid pathologists in their analyses and in this way improve the efficiency and quality of blood analysis.

Blood contains three different types of cells: red blood cells (RBC), white blood cells (WBC) and platelets (PLT) [3]. The red blood cells have the function of transporting oxygen via the blood system and platelets stop bleeding. The white blood cells are a part of the immune system and can in turn be classified in five different subcategories for healthy blood. Some diseases affect the production of blood in a way that causes the cells to develop differently. Because of this, there are more subcategories of white blood cells when considering non-healthy blood. By examining whether the proportion of the cell classes of the white blood cells deviates from the normal, it can give an indication of e.g. a bacterial infection or an inflammation. Examining the morphology of the white blood cells and finding anomalies, such as immature cells, can indicate diseases, such as lymphoma and leukemia.

CellaVision's main analysis is the differential count of white blood cells. By using CellaVision's system, one removes the manual labour of finding cells in a sample. To analyze a blood sample, an operator feeds the sample to the system which then proceeds to capture images of the cells. By using a pretrained artificial neural network, the system then proposes classifications of the white blood cells. The classifications together with the images of the white blood cells, are presented to the user on a screen. The biomedical scientist analyzes the presented images that were captured and the proposed classifications from the system to then make a final decision on the classification of the white blood cells.

The digital microscopes that CellaVision produces today uses an auto focus system which requires around 0.4 μm precision in order for the objects in the images to be considered properly focused. The auto focus consist of moving the objective of the camera vertically, capturing images with 0.2 μm intervals and determining the optimal focus using a so-called focus measure. Because of the small distances, the system is sensitive to vibrations and errors due to mechanical imperfections are hard to avoid. This makes it difficult to design a system that quickly reaches the required precision. In order to set lower requirements on the mechanics of the system, it would

be desirable to digitally transform unfocused images to appear focused after they were captured. Since CellaVision's system is a real time system, it requires the transformation to be fast enough to not affect the speed of the system substantially. Such a transformation could result in that cheaper mechanics could be used and that the speed of the system could be increased.

The auto focus takes about 70 ms to run and is applied a number of times during the analysis of a sample. The system begins by scanning for what is called a monolayer (an area where the red blood cells are spaced with little to no overlap), after which the auto focus is applied. This area is photographed with a magnification of $20\times$ and a resolution of 1920×1200 pixels. The system then proceeds to photograph the white blood cells within this area, now with a magnification of $100\times$ and a resolution of 360×360 pixels. In this step, the auto focus must be applied repeatedly to refocus for the individual cells. A digital sharpening of unfocused images could eventually be applied both when photographing the larger 1920×1200 scan image and the smaller 360×360 WBC images.

Today there is no method in place to digitally make unfocused images focused. However, as the last step in CellaVision's systems a normalization is applied to improve the appearance of the images when presented to the user. It is made up of different filters to strengthen the spectral content in the images. Examples of these filters being applied to cell images can be seen in Figure 1.1. Here we can see that the details of the cells are more distinguishable and the contrast between the different parts of the images have increased. Moreover the colors are normalized to be more visually pleasing. As can be seen in Figure 1.2, where the normalization is applied to some unfocused images, the filters does not correct the focus of images. These algorithms are just used in the very final step when the images are being presented to the operator, but to actually increase focus digitally as a part of the analysis, we need to develop a new method.

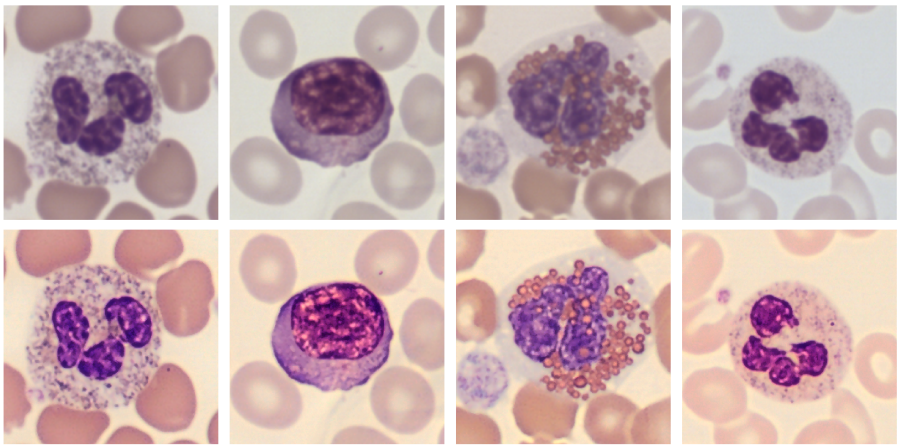


Figure 1.1: Some examples of normalization applied to different images of white blood cells. The upper row contains cell images before and the lower row contains cell images after CellaVision's DC-1 normalization filters are applied.

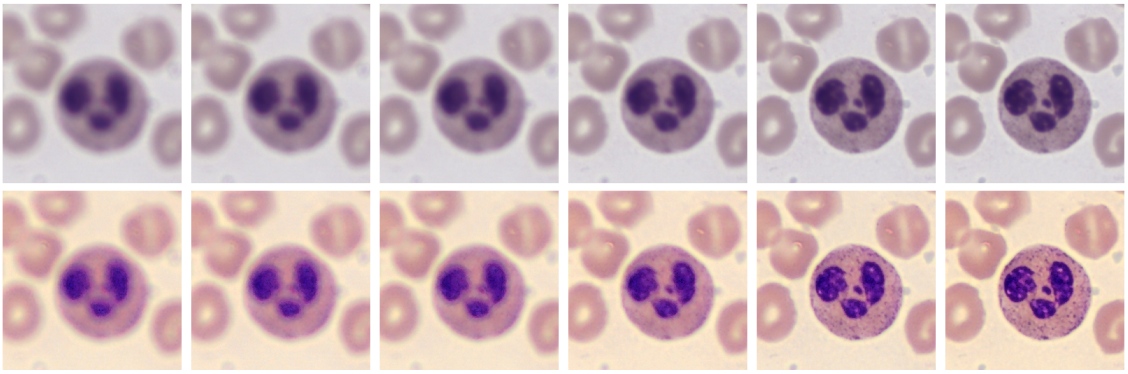


Figure 1.2: A figure showing images of a white blood cell at different focus levels before (upper row) and after normalization (lower row). It is visible that even though the normalization strengthen the spectral content, it does not make the unfocused images more focused.

1.1 Aim

The aim of this project is to develop a generative adversarial network or another deep learning method to reconstruct a focused image using only one or two unfocused images. This network could then be applied to CellaVision's systems in order to reduce the requirements on the hardware when it comes to capturing focused images, as it could in part be done digitally. For the method to be useful, it needs to be accurate as well as time efficient, as it should be applied to a real time system. Today it takes about 70 ms for the system to capture a focused image using auto focus and therefore it would be desirable if the transformation would meet that time requirement. Since CellaVision's systems is not equipped with a GPU, the requirement should preferably be fulfilled on a CPU. The aim can be summarized in the following questions:

- Is it possible to make a transformation on CellaVision's unfocused images so that they become sharp by using primarily generative adversarial networks?
- Is it possible to do the transformation in 70 milliseconds or less on a CPU, so that it does not affect the speed of the system significantly?
- How far away from the optimal focus can one capture an image so that the transformation to sharpness works?

1.2 Image blurring and deblurring

The sharpness or blurriness of an image is a concept central to this thesis. There are essentially two types of image blurring; motion blur and focus blur. Motion blur occurs when the object of a photo is moving during the taking of the photo, resulting in a blurred image where the object looks smeared. Focus blur is the effect of taking a photo with the wrong distance between the lens of the camera and the object of the photo. An image that is blurred due to bad focus is smoothed, which leads to loss of details and less defined edges.

When modelling a blurry image, one often uses convolution [5]. The blurry image X is approximated as a latent sharp image Y convolved with some kernel h.

$$X = h * Y + n, \tag{1.1}$$

where n is some noise.

The convolution kernel h is called a point spread function (PSF), and contains information of the way the image is blurred. In the case of motion blur, the PSF mimics the path that the camera was moved in relation to the object while the image was captured. An example of this is shown in Figure 1.3. By convolving with such a PSF, each pixel is a mixture of the pixels along the camera trajectory from that point. In focus blur on the other hand, the PSF corresponds to the smudge effect of the pixels. Here the PSF should be symmetric and have a width that corresponds to the dependencies of the pixels. This is often approximated as a Gaussian kernel, as shown in Figure 1.4.

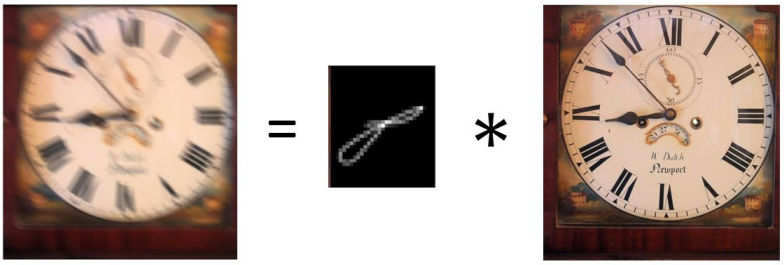


Figure 1.3: An example of motion blur presented by Kupyn et al. in [14]. To the left is the blurred image X which is produced by convolving a point spread function h with a sharp image Y, as pictured on the right.

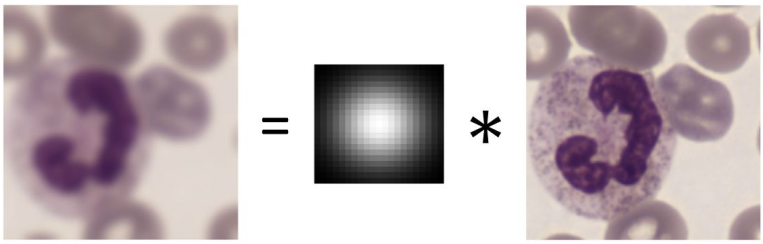


Figure 1.4: An example of focus blur. To the left is the blurred image X which is produced by convolving a point spread function h with a sharp image Y, as pictured on the right.

The aim of this project is to reconstruct a sharp image from an image with focus blur. If the unfocused image was simply a convolution between the sharp image and a point spread function, i.e.

$$X = h * Y,$$

then the sharp image could be retrieved by moving the problem to the frequency domain. By computing the Fourier transformation $\mathcal{F}$ of the PSF and the blurred image the sharp image could then be completely reconstructed as

$$Y = \mathcal{F}^{-1}(\mathcal{F}(X)/\mathcal{F}(h)).$$

Unfortunately, this is not the case in a real life problem. Due to information loss as a result of noise and the mechanics of the camera, it is not that simple to fully recover the sharp image. This

operation is in general also mathematically ill-posed and numerically unstable.

There are methods to approximate a deconvolution kernel, i.e. g in $\hat{Y} = g * X$. This kernel can be applied to the blurry image in order to reconstruct the sharp image. However, there are several problems with traditional deconvolution methods, for example the resulting images often have significant unwanted artifacts. Another drawback of using this type of method is that it is often both hard to approximate the kernel and the parameters needs to be tuned manually [18]. This makes them inefficient to adapt to new settings and data. In our case of focus blur the PSF would have to be estimated separately for different distances from the optimal focus. This is one of the reasons to that we intend to use machine learning to solve the problem in this project.

1.3 Focus measure

Since sharpness is not an absolute measure, but a somewhat subjective property, there is no truly objective way of telling whether an image is sharp or not. Because it is not an definitive state, a way of ranking sharpness of images must be defined. To do this, CellaVision has defined a focus measure that is used in their systems to determine how the camera should be positioned for the optimal focus.

The measure aims to detect edges and information that is visually important in the image, which is done by convolving with kernels that detect high frequency changes in the image in the x- and y-directions. The kernels used for this are

$$\omega_x = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad \omega_y = \omega_x^T.$$

A given RGB image X contains three color channels, i.e. $X = (X_R, X_G, X_B)$. These filters are applied to the green channel of the image, due to that this channel contains the most information in cell images. Also because the objective can have problems focusing all the channels on the same focus plane due to axial chromatic aberration and the green channel is the channel that has the focus plane in the middle of the three channels [16]. Applying the filter on the green channel of the image, squaring it and taking the mean of the pixel values of the filtered images gives the focus measure as

$$F(X) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (\omega_x * X_G)_{i,j}^2 + (\omega_y * X_G)_{i,j}^2,$$

where M, N are the number of columns respectively rows of the image and X_G is the green channel of the image.

The focus measure is used in CellaVision's systems during the auto focus step to determine what distance between the objective and the object that gives the best focus. In this project it was also used to find the most focused image and to assess the quality of the results.

Chapter 2

Data

The data used for this project is microscopy images primarily featuring white blood cells. The images were collected using CellaVision's DC-1 system, which photographs blood samples with 100× magnification. The setup is illustrated in Figure 2.1. After locating a monolayer, the system was set to move the camera over this area and stop at up to 200 positions per slide. The number of positions depended on how many cells the system was able to locate within the monolayer and could therefore be less than 200 for some slides. Our data was collected using four different DC-1 systems and from 120 slides. The diversity of cells and staining is illustrated in Figure 2.2, where a number of cells are shown. These were collected from different slides and using different systems.

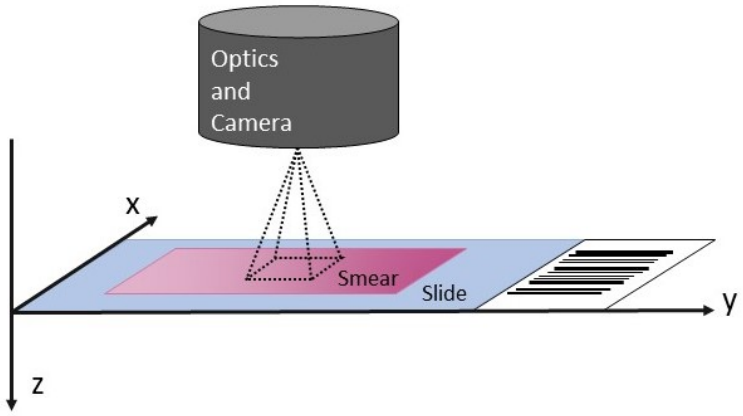


Figure 2.1: An illustration of the setup for data gathering. The slide consists of a rectangular glass with a bar-code in the end and a drop of blood smeared out to a thin layer, which is stained with some type of staining to make the cells distinguishable. The system identifies an area from which it gathers images at different (x, y) positions.

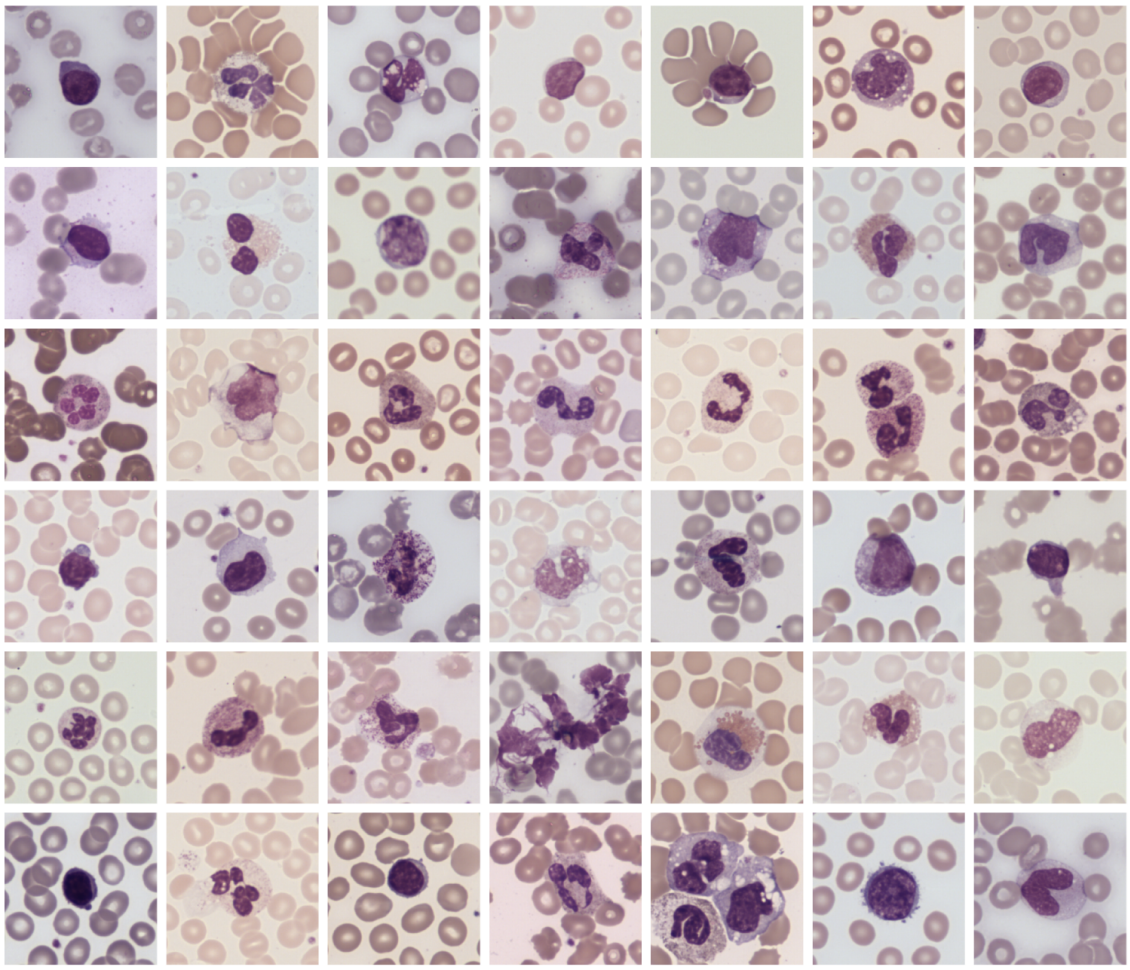


Figure 2.2: Some examples of different cell images, showing the diversity in the data that was used regarding illumination, staining, cell appearance etc.

For any (x, y) position, the blood samples were photographed at different distances along the z-axis to achieve different planes of focus. In addition to the focused image, 16 unfocused images were collected for each cell. These were captured with intervals of 0.2 micrometers from the ideal focus. This resulted in a stack of images with distances 0, ± 0.2, ± 0.4, ± 0.6, ± 0.8, ± 1.0, ± 1.2, ± 1.4 ,± 1.6 micrometers from the expected focus point as determined by the system. In total, around 11 000 different stacks were gathered. The images in a stack are RGB images of size 401×401 pixels. An example stack is shown in Figure 2.3. As can be seen, the images captured in the range -0.2 μm to 0.4 μm are very similar and can all be considered focused. The reason for that 0.4 μm and not -0.4 μm are considered focused is because the focus is not linear and the image is affected more when moving in the negative direction than the positive direction.

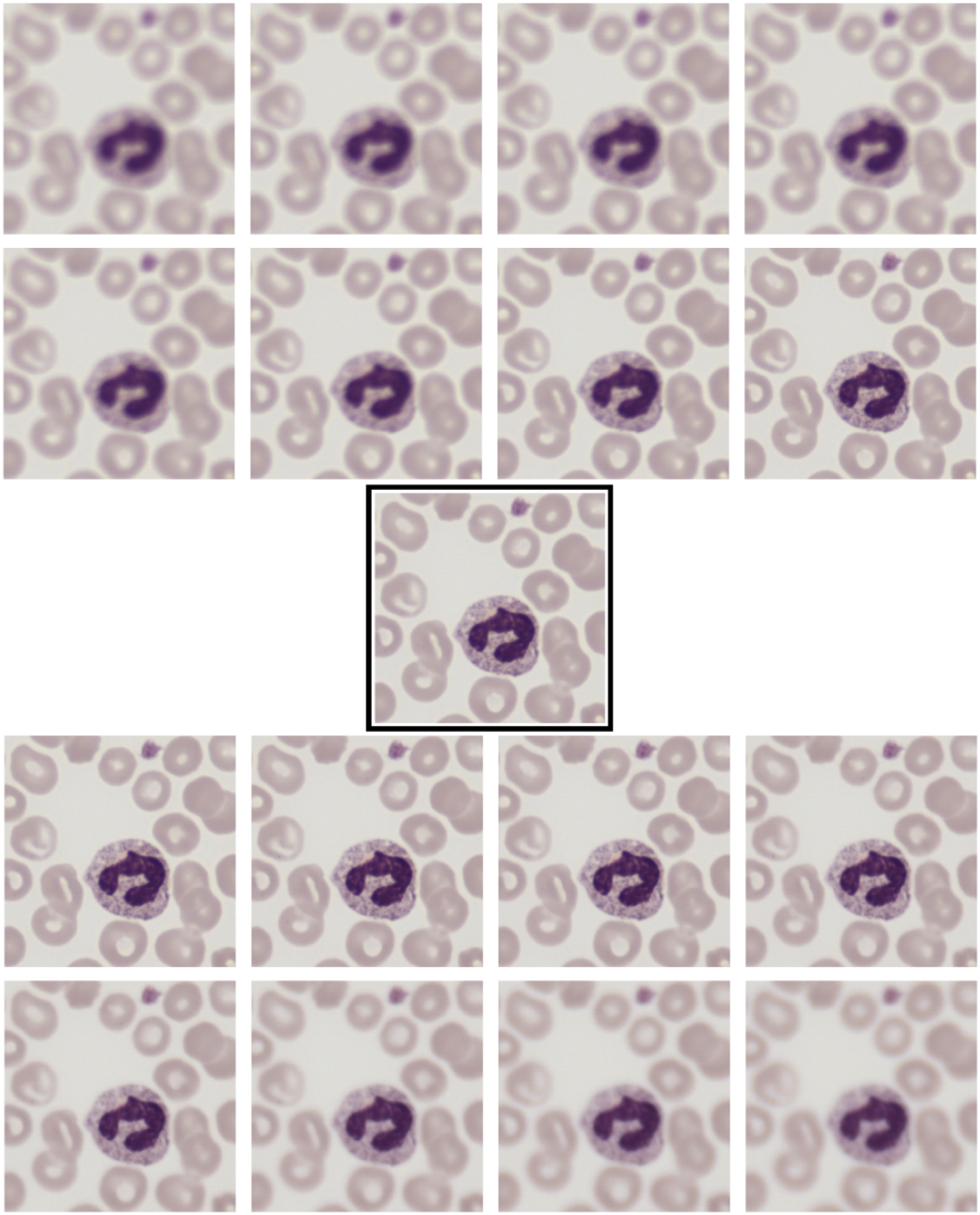


Figure 2.3: An example stack. From the top left: the images are captured with distances $-1.6, -1.4, ..., 1.4, 1.6$ μm from the best focus. The image marked with a black box is, according to the DC-1 system using the focus measure, the one with the best focus.

2.1 Preprocessing of data

In order to speed up the training of the networks, the images were cropped to a smaller size of 256×256 pixels before they where fed to the network. Before an image was cropped, the nucleus of the white blood cell in the image was located by applying a threshold regarding the darkness in the image and then finding the largest of those dark segments. When the center of the white blood cell was found, a normal distributed randomness in the x- and y-direction determined where the image was cropped. In this way, the white blood cell could appear both in the center of the cropped images and near the edges and corners. This was done because we wanted the neural network to only learn how to focus the image and not where the cells were located in the image.

2.1.1 Finding the ground truth

A blood smear will have a certain three dimensional structure, since it contains blood cells of different sizes (see Figure 2.4). These differences in depth will cause the optimal focus to fluctuate over the different regions of an image, since the distance between the objective and the object differs. When cropping the images, the number of cells present in the image will change from when the data was gathered, which could affect the optimal focus. For instance, if the number of red blood cells decreased in relation to the number of white blood cells, the mean depth of the slide would increase and the optimal focus would be moved further away from the slide. Because of this and mechanical problems, the sharpest image must be found again after gathering and cropping the images.

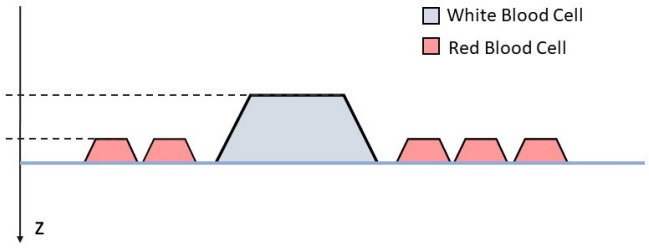


Figure 2.4: A sketch of how the depth in the samples differs. Here we see that the white blood cell will be most in focus at one distance, and the red blood cells at another.

To choose the target for a deblurring transformation, for every stack we needed to determine which image was the sharpest. This was done by applying the focus measure, F, introduced in section 1.3 to the images and finding the one with the highest value. Among the images X_k, $k = 1, ..., 17$, the sharp image Y was determined by

$$Y = \max_k F(X_k).$$

When running a stack through the focus measure, we got a focus curve showing the focus measure for each of the images. This curve should have a somewhat steady incline and decline with the peak at 0. If the curve peaks at 0, it means that the optimal focus is the same as the one found during the gathering of the images. Figure 2.5 shows the focus curve for the example stack shown in Figure 2.3. We see here that the peak is located at 0, meaning that the image with the best focus is the one marked in Figure 2.3. We can also note that the curve is smooth, meaning that there are no sudden changes in the sharpness of the images.

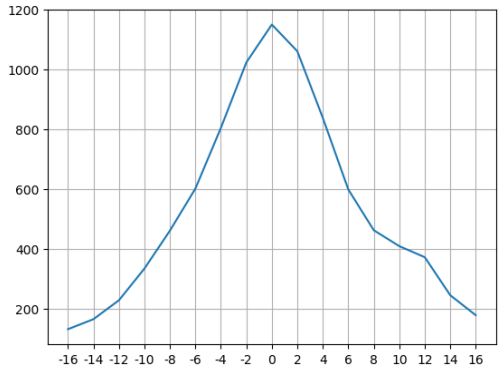


Figure 2.5: The focus curve for the stack shown in Figure 2.3. The x-axis shows the distance in 0.1 μm in z-direction from the optimal focus.

If the curve is not centered around 0, but rather offset in some direction, it means that there now is a new optimal focus point for the cropped images. In this case, the sharp image is redefined by the peak of the focus curve so that all focus levels are shifted backward or forward along the z-axis.

In order for all stacks to contain the same set of focus levels, we redefined the range of the stacks. We chose to only keep stacks with focus curves peaking in -0.2, 0, 0.2, 0.4 μm, since the number of stacks with optimal focus outside of this interval were few enough to be discarded as outliers. By doing this, some of the uttermost focus levels disappeared and the range of the resulting stacks became -1.4 μm to 1.2 μm from the optimal focus. The focus curve of such an offset stack is shown in Figure 2.6. Here the optimal focus is reached at $+0.6$ μm from the original optimal focus, leading us to discard this stack.

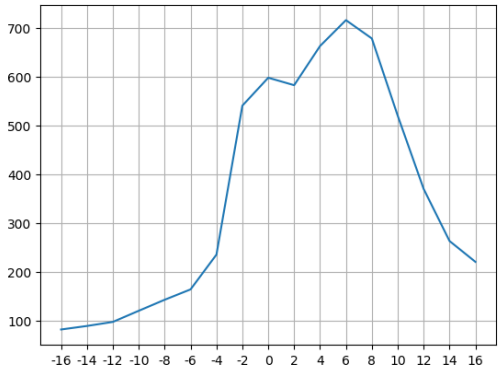


Figure 2.6: A shifted focus curve having the optimal focus at 0.6 μm. The x-axis shows the distance in 0.1 μm in z-direction from the optimal focus. One can also notice that the system has a big jump in focus between -0.4 μm and -0.2 μm.

Another issue that can occur with a focus curve is irregularities. These could be an effect of the different depths of blood cells or the three dimensional texture of a single cell, which could cause the focus to be optimal at one level for some cells and another level for others. The irregularities could also be a result of mechanical problems. An example of this can be seen in Figure 2.7, where

the curve is jagged. Due to mechanical issues the images are not captured at the correct distances, leading the focus measures to be unpredictable. A part of the corresponding stack is shown in Figure 2.8. Here we can see that the image captured at -0.6 μm is nearly as sharp as the sharpest one, as indicated by the curve.

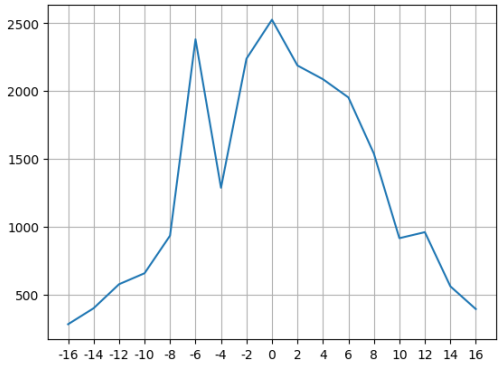


Figure 2.7: A focus curve showing one of the problems with disturbances and lack of accuracy in mechanics affecting the focus of the cell. The x-axis shows the distance in 0.1 μm from the optimal focus. As one can see -0.6 μm is the second most focused image.

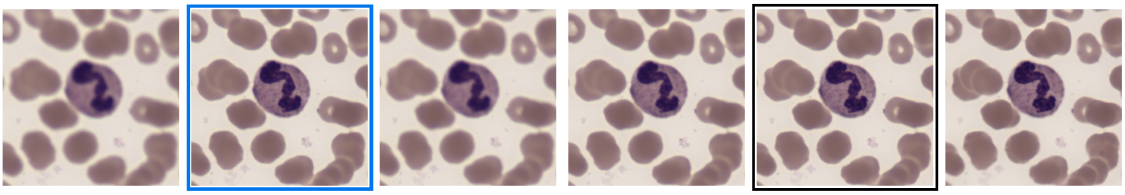


Figure 2.8: The images that corresponds to the focus curve in Figure 2.7 that are captured with distances (from the left) -0.8, -0.6, -0.4, -0.2, 0, 0.2 μm in z-direction from the optimal focus. The image marked with a black box is, according to the focus measure, the one with the best focus and the image marked with a blue box is the second most focused image.

2.1.2 Alignment

To be able to train a network on paired images, i.e. one sharp and one blurry image, the images in a stack needed to be aligned properly. If not, the network could learn patterns in the displacement of the object, that are not relevant to the task at hand. Since the images were gathered in sequence, there could occur some changes in the positioning of the camera and the blood sample due to mechanical errors or external disturbances. To avoid this, the images were aligned digitally after they were captured.

To achieve subpixel accuracy of the alignment, we started by interpolating the images to be 3 times their original size, using bicubic interpolation. We then found the displacement that gave the highest cross correlation for each of the images in a stack. This displacement was then applied to the image, making it aligned to the sharp image. The images were then scaled down to their original size.

2.2 Data sets

After discarding some of the images in the preprocessing step, 10 786 stacks remained. These were divided into a training set, a validation set and a test set. Both the validation and the test sets were collected on separate DC-1 systems, containing 863 and 727 stacks respectively. The rest of the data, 9196 stacks collected on two systems, were used for training.

2.3 Augmentation of data

Augmentation was introduced in the training of the network in order to make the model more robust to changes in the data. The aim of the augmentation was to prepare the model for differences in staining of the samples and differences resulting from using different systems when retrieving images. The changes in coloration were made through adjustments in the contrast, brightness, white balance, saturation and staining of the images. The augmentation also included a rotation to make sure that the model would not learn any irrelevant patterns in how the cells are oriented in the images.

The augmentation was applied randomly when loading data for training the model, but not for testing or validation. To get an understanding of the augmentation used, one can see Figure 2.9 which shows some example of augmentation of five different cell images.

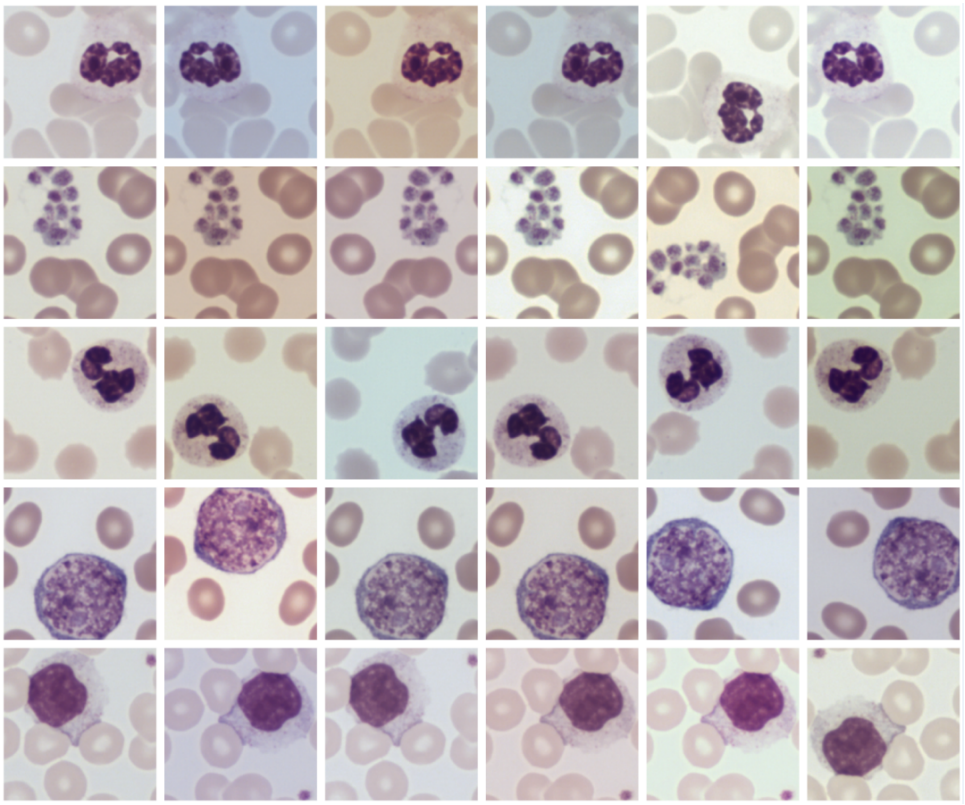


Figure 2.9: Here one can see five different augmentations of five different cells. The first columns contains the original cell and the other columns contains a random augmentation of the cell.

Chapter 3

Deep learning

A deep learning network can be described as a composite function f that, given some input X and parameters θ, produces an output

$$\hat{Y} = f(X, \theta).$$

The θ parameter represents the network weights that are used to regulate the behaviour of the network. The tuning of these weights is done iteratively, in a process that is referred to as training the network. Given some batch of training data (X_i, Y_i), for $i = 1, ..., n$, this process can be described as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, E[\mathcal{L}(Y_i, f(X_i, \theta))] = \underset{\theta}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i}^{n} \mathcal{L}(Y_i, f(X_i, \theta)),$$

where $\mathcal{L}$ is some loss function that is to be minimized on the batch and Y_i is a given ground truth. In our project the minimization of $\mathcal{L}$ is done using an iterative optimization algorithm, which in our project is carried out using an Adam optimizer. After training the network, the finished model is given by $\hat{Y} = f(X, \hat{\theta})$.

3.1 Convolutional neural network (CNN)

A Convolutional Neural Network (CNN), as presented in [13], is a deep learning algorithm which takes in an input image and learns what is important information in the image and what is not, according to the loss function. With the information in the image, it creates features representing different dependencies and importance and these are used by the network to create the desired output. In this way all the information in the image can be used by the network and not only some features from the image as for a regular neural network. A CNN consists of four main operations:

- Convolution,
- Non linearity,
- Pooling or sub sampling
- Fully connected layer,

which will be explained more detailed in further reading.

3.1.1 Convolution

The convolution step is the step where a filter/kernel of a certain size operates over the image to catch different local dependencies between pixels in the image. When the filter operates over the image, a matrix multiplication is performed between the filter and the different part of the image to create a feature map. The bigger size of the filter, i.e. the bigger receptive field, the bigger dependencies are being caught. The size of the resulting feature map depends on three different parameters. One parameter that affect the size of the feature map is the depth of the convolution, i.e. the number of filters in the convolution operation. The larger number of filters, the more feature maps are created. The stride is also an affecting parameter, which is the number of pixels by which we slide our filter over the image; having a larger stride will result in smaller feature maps. The last thing that affects the size is zero-padding, which means that it add zeros around the image in the convolution step so that the corners includes in the right way. Adding zero-padding is called wide convolution and results in larger feature maps.

3.1.2 Non linearity

The non linearity step is performed to each feature map after it has been created, by having a non-linear activation function operating over the feature map. The non-linear function is often either ReLU or leaky ReLU. ReLU, stands for rectified linear unit and it extinguishes all the negative values as

$$f(x) = \max(0, x).$$

A problem with this is that it can get stuck, because of the zero values. Leaky ReLU can fix this by instead having the form

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha x & \text{otherwise,} \end{cases}$$

where α is a small positive scalar. ReLU corresponds to choosing $\alpha = 0$. The non linearity step is done due to that most of the real-world data which the CNN is about to learn would be non-linear.

3.1.3 Pooling

Spatial pooling or sub sampling can be done after the convolutional and the non linearity step. It reduces the dimensionality of each feature map but retains the most important information. There are many different types of spatial pooling such as max, average, sum, etc. In our case, this will not be used due to that we have an image to image transformation and therefore we do not want to reduce or loose information.

3.1.4 Fully connected layer

After all the layers of convolutions, non-linearity and pooling, a fully connected layer (multilayer perceptron) can be applied to learn non-linear combinations of the high-level features so that we get the correct output. This is often the part where it learns to classify the image from the features that have been extracted. When having an image to image transformation, it is often not used.

3.1.5 Batch normalization

Batch normalization is often applied in between convolutional layers in order to stabilize the training of the network. The normalization is applied through fixing the mean and variance of all inputs to the layer. In this way, all inputs will have equal influence on the weight update of the network. Thus, the training will be less dependant on the individual inputs and more stable.

3.2 Generative adversarial network (GAN)

A generative adversarial network (GAN) is a machine learning structure that was first described by Ian Goodfellow et al. in [6]. A GAN consists of two CNNs; a generator G and a discriminator D. The idea of the GAN is for these two networks to train against each other in an adversarial manner. The generator is trained to produce data samples, $\hat{Y} = G(Z, \theta) = G(Z)$, that could realistically belong to the training data, given some noise Z. The discriminator's task is then to distinguish which data samples are real, and which are generated by the generator, by outputting a probability $P = D(\tilde{Y}, \theta) = D(\tilde{Y})$, where $\tilde{Y}$ is either a real or generated sample. The structure is illustrated in Figure 3.1.

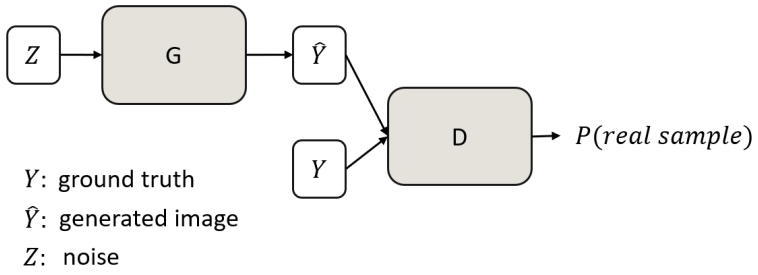


Figure 3.1: A GAN. The generator is given noise and produces a generated image. The discriminator is given an image, generated or a real sample, and gives a probability of the image being a real sample.

In a GAN, both the G and D are trained along side each other, getting better at their respective tasks. In the end, after the training is done, the generator can be used to generate realistic samples as a stand alone CNN. In that way, the discriminator is only used as a tool in the training process, and is omitted in the final product.

3.2.1 Binary cross entropy loss

The idea of the generative adversarial network is to have two networks working against each other. Practically, this is carried out by an adversarial loss. The adversarial loss is used to check whether the generator can fool the discriminator. This is the core of the GAN framework, as it is this loss that creates the adversarial behaviour of the network. This is a loss that is used to train both the generator and the discriminator, as one of them tries to minimize it and the other maximize it. The adversarial loss is traditionally a binary cross entropy loss. For when training the discriminator, the binary cross entropy is defined as

$$\mathcal{L}_{BCE_D} = -E[\log D(Y)] - E[\log(1 - D(\hat{Y}))],$$

where the $D(Y)$ should be maximized and $D(\hat{Y})$ minimized. When training the generator however, we want the discriminator to fail and so we flip the labels

$$\mathcal{L}_{BCE_G} = -E[\log D(\hat{Y})] - E[\log(1 - D(Y))].$$

Since the second term is not affected by the generator, this can be omitted during training. The binary cross entropy used for the generator is then

$$\mathcal{L}_{BCE_G} = -E[\log D(\hat{Y})].$$

In addition to an adversarial loss, many GANs have other losses, e.g. a perceptual loss or a pixelwise comparison loss that are used to train the generator. A perceptual loss is comparing the input and output on a feature level, which is achieved by letting the images pass through a CNN and picking a certain output of a layer in the network. The pixelwise comparison loss on the other hand, compare the images as they are. The reason behind using these kinds of losses is to ensure that the generated image resembles the target image. This prevents the generator from fooling the discriminator with nonsense.

3.3 Conditional generative adversarial network (CGAN)

In this project, we will use conditional GANs [17]. Instead of noise, this version of a GAN gets some relevant information X as input to the generator; $G(X) = \hat{Y}$, as illustrated in Figure 3.2. This information is also passed to the discriminator along with the generated or real image. For example, by conditioning on a blurry image, we can force the generator to produce a sharp image that is similar to the blurry one. Without conditioning, the network can simply generate the exact same image every time and always fool the discriminator. In other words, we need to include conditioning in order to be sure that the generator aims to produce the correct sharp image and not just any sharp image.

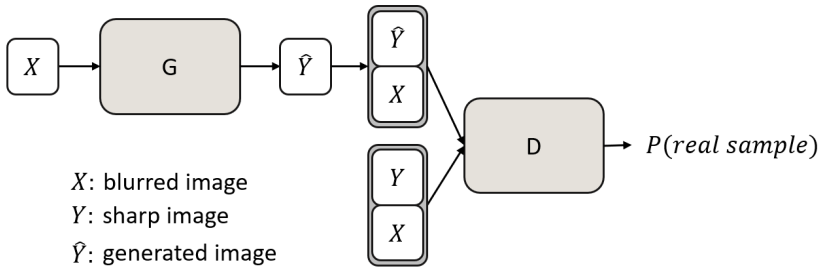


Figure 3.2: A conditional GAN. Both the generator and the discriminator are now given a condition X, containing some relevant information.

By including conditioning we get new binary cross entropy loss functions with dependencies:

$$\mathcal{L}_{BCE_D} = E[\log D(Y|X)] + E[\log(1 - D(\hat{Y}|X))], \tag{3.1}$$

$$\mathcal{L}_{BCE_G} = -E[\log D(\hat{Y}|X)]. \tag{3.2}$$

3.4 Wasserstein generative adversarial network (WGAN)

The Wasserstein GAN was first presented in [1]. This type of GAN is not as sensitive to imbalance between the generator and the discriminator as an ordinary GAN. The losses of a WGAN are

$$\mathcal{L}_{wass,D} = -E[D(Y)] + E[D(\hat{Y})]$$

$$\mathcal{L}_{wass,G} = -E[D(\hat{Y})].$$

A requirement for the WGAN is that it imposes a so called Lipschitz constraint. This implies that the norm of the gradient of the discriminator does not exceed 1 anywhere. In [19] they enforce this by clipping the gradient to be within an interval $[-1, 1]$. However, as described in [7] this method can cause a range of issues, such as vanishing gradients and not using the networks full capacity. Instead, they propose a gradient penalty term on the gradient norm. The gradient penalty loss is given by

$$\mathcal{L}_{GP} = E[(\|\nabla_{\tilde{Y}} D(\tilde{Y})\|_2 - 1)^2],$$

where $\tilde{Y}$ is a randomly weighted average of Y and $\hat{Y}$. This ensures stable training that is robust to the choice of generator architecture.

3.5 U-net

A U-net is a type of encoder-decoder network. This is a convolutional neural network that is made up of a encoder, i.e. a downsampling part that translates the input to features, and a decoder, i.e. an upsampling part that translates these features to higher level information. In image transformation, an encoder-decoder architecture comes in handy, as we want to input an image, extract relevant features, and use these features to produce a new image.

In a U-net, there is the additional benefit of having skip connections from layers in the encoder to corresponding layers of the decoder, see Figure 3.3. This allows the network to pass general information directly from the encoder to the decoder, without passing this information via the feature layers.

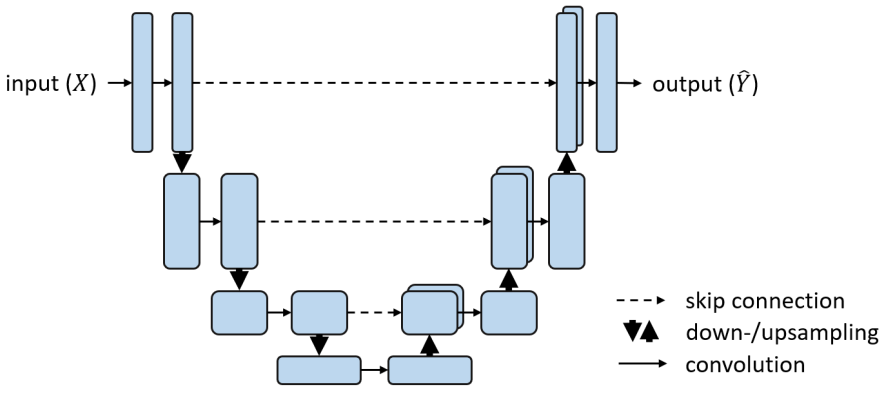


Figure 3.3: A U-net structure. The downsampling is done by convolution with stride larger than 1, or in some cases pooling. The upsampling is done with transposed convolution. The skip connections concatenate the encoder layer to the decoder layer, making the decoder layer deeper.

3.6 Residual network (ResNet)

The ResNet structure is presented in [8]. The building blocks of this type of network contains skip connections between layers that are close to one another. These layers are called resblocks, one is illustrated in Figure 3.4, where we can see that the input X is put through some layers f. The output is then given by the summation of $f(X)$ and the input, i.e. $f(X) + X$. Since the input is given at this point, the network only learns the residual $f(X)$. This idea is called residual learning.

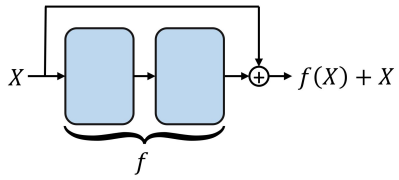


Figure 3.4: A Resblock. A skip connection passes the block input over the convolutional layers to be added to the output of the convolution to give the block output.

Chapter 4

Method

Now looking into our specific problem with transforming images from unfocused to focused, we used conditional generative adversarial networks (CGAN). As described in [10], these networks have previously been used in a wide range of image-to-image translations. We began by implementing three different CGANs structures that have previously been described in [10], [22] and [14], called pix2pix, PAN and deblurGAN.

4.1 Pix2pix

The pix2pix network was proposed by Isola et al. [10] to be used for different types of image translations. It has been used for a wide variety of problems, showing promise to be a very adaptable model.

The pix2pix generator is illustrated in Figure 4.1. It is a U-net, i.e. it contains multiple skip layers from the encoder layers to the decoder. The encoder layers are of the form convolutional layer, batch normalization (except on the first layer) and a leaky ReLU activation with slope 0.2. The decoder blocks differ from the encoder blocks in the fact that they have transposed convolution instead of convolution, as well as an ordinary ReLU instead of a leaky one. The first three decoder blocks have a dropout factor of 0.5, meaning that the nodes randomly are being ignored by a probability of 50%. All convolutions of the U-net have kernel size 4×4 and stride 2, and the number of filters in each block can be seen in Figure 4.1. After the U-net there is one final convolutional layer with 3 filters that maps the data to a RGB image again, followed by a tanh activation function.

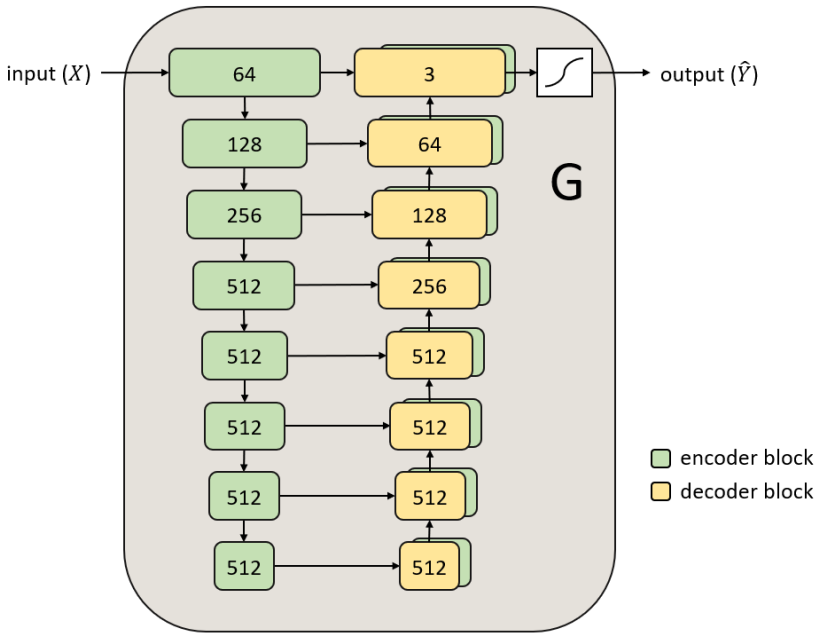


Figure 4.1: An illustration of the architecture of the pix2pix generator.

The discriminator in the pix2pix network can be seen in Figure 4.2. This discriminator is a patch-GAN, which means that the discriminator classify smaller patches of the image (in this case 70×70 pixels) and then outputs the mean of the patch probabilities. This allows the discriminator to only look locally when classifying, and not find dependencies between pixels that are far away from each other.

The convolutions in the discriminator have kernel size 4×4, with a stride 2 for the first three layers and stride 1 for the last two. Each of the first four layers have a leaky ReLU activation with slope 0.2 and the last layer has a sigmoid activation.

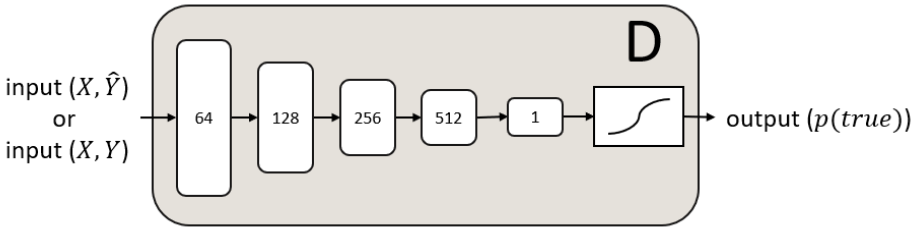


Figure 4.2: An illustration of the architecture of the pix2pix discriminator.

For the adversarial loss pix2pix uses a binary cross entropy, defined in (3.1) and (3.2). In addition, it also uses the L_1-norm of the difference between the generated image and the target image when training the generator.

$$\mathcal{L}_{L_1} = E[\|Y - \hat{Y}\|_1]$$

The losses are then combined by summation and loss weights λ. The losses for the generator and the discriminator are

$$\mathcal{L}_G = \lambda_{B_G}\mathcal{L}_{BCE_G} + \lambda_{L_1}\mathcal{L}_{L_1}$$
$$\mathcal{L}_D = \lambda_{B_D}\mathcal{L}_{BCE_D}.$$

4.2 Perceptual adversarial network (PAN)

The perceptual adversarial network PAN was presented in [22], as a network that can perform a range of image transformations. It consists of an image transformation network T and a discriminative network D. The image transformation network T is trained to synthesize the real samples given some prior information, and is in this report referred to as a generator G according to standard GAN notation.

The generator of a PAN is illustrated in Figure 4.3 and is composed of an encoding and a decoding part. The encoder layers are of the form convolutional layer, batch normalization and leaky ReLU activation with slope 0.2. The convolution is done with kernel size 3×3 and stride 2. The decoder layers contains deconvolution, batch normalization and regular ReLU activation. In the deconvolution the kernel size is 4×4. After the last layer of the decoder a tanh activation is applied.

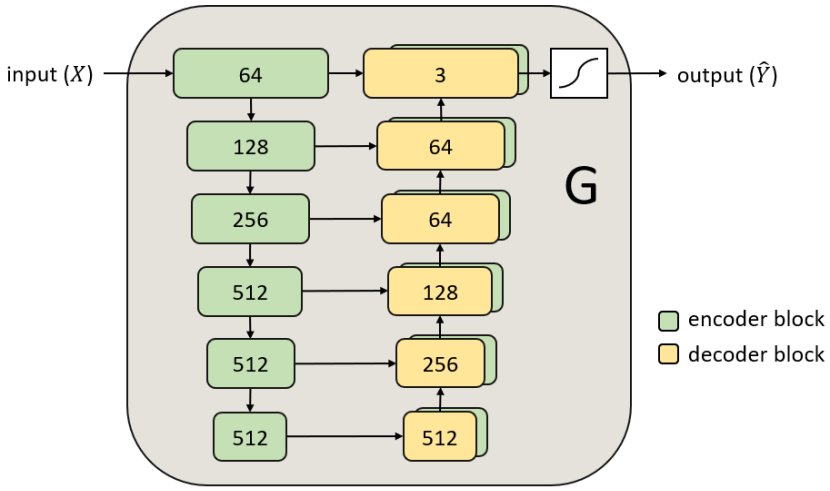


Figure 4.3: An illustration of the architecture of the PAN generator.

The discriminative network D of a PAN is shown in Figure 4.4. It is also a CNN that consists of convolution, batch normalization and leaky ReLU activation. The convolutions are done with kernel size 3×3 and the stride is 1 for layers 1, 3, 5 and 7, and the stride is 2 for layers 2, 4, 6, 8 and 9. After the last convolutional layer, the output is flattened and fed through a sigmoid activation, resulting in a probability. Hidden layers of the network D are utilized to evaluate the perceptual adversarial loss described below.

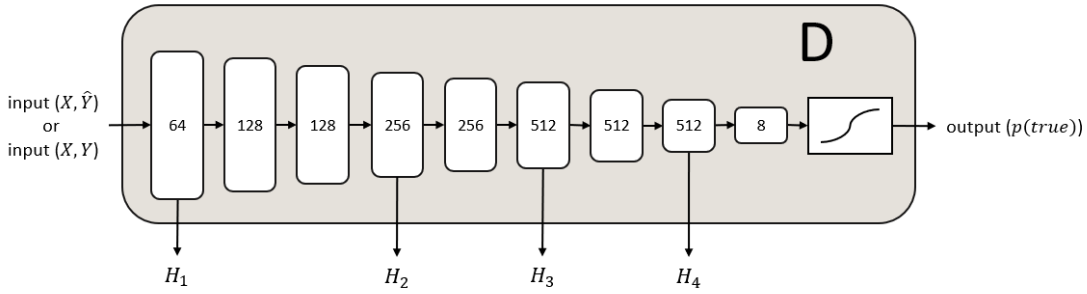


Figure 4.4: An illustration of the architecture of the PAN discriminator.

In addition the a binary cross entropy as the adversarial loss, PAN also includes a perceptual loss. This loss is based on outputs of intermediate layer of the discriminator network. We denote the output of layers 1, 4, 6, and 8 of the discriminator as H_1, H_2, H_3 and H_4 respectively. These are used to compute the perceptual adversarial losses $\mathcal{L}_D$ for the discriminator and $\mathcal{L}_G$ for the generator:

$$P_i(Y, \hat{Y}) = \|H_i(Y) - H_i(\hat{Y})\|_1$$

$$\mathcal{L}_{perc,G} = \sum_{i=1}^{4} \lambda_i P_i(Y, \hat{Y}), \tag{4.1}$$

$$\mathcal{L}_{perc,D} = \max\left(0, \left[m - \sum_{i=1}^{4} \lambda_i P_i(Y, \hat{Y})\right]\right), \tag{4.2}$$

where λ_i are the weights of the different layers and the variable m is a positive margin. Combining these with the adversarial loss, we get the total loss of the PAN:

$$\mathcal{L}_G = \theta \mathcal{L}_{BCE_G} + \mathcal{L}_{perc,G}, \tag{4.3}$$

$$\mathcal{L}_D = \theta \mathcal{L}_{BCE_D} + \mathcal{L}_{perc,D}, \tag{4.4}$$

where θ is the weight of the BCE-loss.

4.3 DeblurGAN

The deblurGAN was presented in [14] as a network that removes motion blur from images. The network aims to estimate the difference between a sharp and a blurred image, and to then remove this difference from the blurred image.

The deblurGAN generator contains two strided convolution blocks with stride 0.5, nine residual blocks (Resblocks) and two transposed convolution blocks. An illustration of the generator is shown in Figure 4.5. Each Resblock consists of a convolution layer, instance normalization layer, and ReLU activation. Instance normalization is in practice the same thing as batch normalization with batch size 1. Dropout regularization with a probability of 0.5 is added after the first convolution layer in each Resblock. In addition, it introduces a global skip connection, from the input to the last layer.

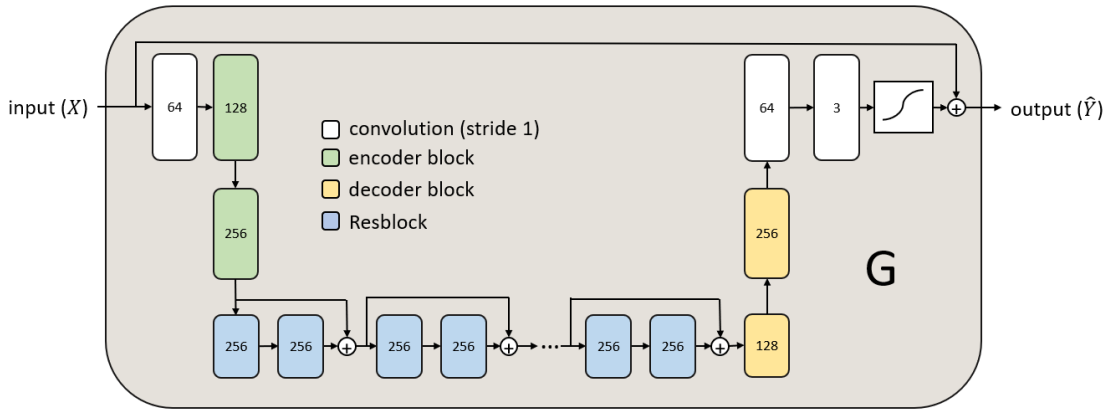


Figure 4.5: An illustration of the architecture of the deblurGAN generator.

The architecture of the deblurGAN discriminator network is identical to the discriminator in pix2pix, see Figure 4.6. All the convolutional layers except the last are followed by instance normalization layer and Leaky ReLU with $\alpha = 0.2$.

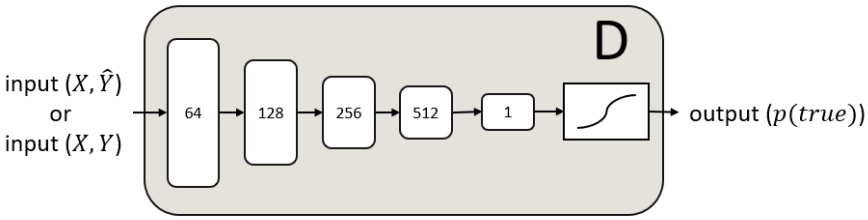


Figure 4.6: An illustration of the architecture of the deblurGAN discriminator.

The deblurGAN loss is made up of a Wasserstein loss with gradient penalty and a perceptual loss. The perceptual loss is calculated using a pretrained network, VGG-19. The VGG-19 network is a CNN trained to extract features on a large data set [20]. By using a pretrained net, we get a very efficient feature extraction that is trained to differentiate between vastly different images. The loss is given by

$$\mathcal{L}_{VGG} = \frac{1}{MNC} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=1}^{C} (\phi(Y)_{i,j,k} - \phi(\hat{Y})_{i,j,k})^2$$

where M, N and C are the number of rows, columns and number of filters of the feature maps ϕ. The feature maps ϕ are extracted using the VGG-19 net at layer $\text{conv}_{3,3}$, $\phi(Y) = \text{VGG-19}_{\text{conv}_{3,3}}(Y)$.

The loss functions of the generator and the discriminator are then given by

$$\mathcal{L}_G = \lambda_{wass,G} \mathcal{L}_{wass,G} + \lambda_{VGG} \mathcal{L}_{VGG}$$

$$\mathcal{L}_D = \lambda_{wass,D} \mathcal{L}_{wass,D} + \lambda_{GP} \mathcal{L}_{GP}$$

4.4 Training details

All networks were trained using an Adam optimizer with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and with learning rate 0.0002. The batch size was set to 4 for pix2pix and PAN. Since the deblurGAN uses instance normalization, the batch size was set to 1 in this case. When training the deblurGAN we used a training ratio of 5:1, since this was proposed in [14]. This means that the discriminator was trained 5 times more often than the generator. For the other networks the training ratio was 1:1. The loss weights for the different networks are shown in tables 4.1-4.3. They were determined by tests which gave an understanding in how they affected the performance and what were reasonable weights for the different networks.

Table 4.1: Loss weights for training pix2pix.

	λ_B	λ_{L_1}
G	1	10
D	1	-

Table 4.2: Loss weights for training PAN.

	θ	λ_1	λ_2	λ_3	λ_4
G	2	100	1	1	1
D	10	5	1.5	1.5	5

Table 4.3: Loss weights for training deblurGAN.

	λ_{wass}	λ_{GP}	λ_{VGG}
G	1	-	100
D	1	1	-

Chapter 5

Metrics for evaluation

In order to assess the quality of the generated images we used a set of evaluation metrics. When evaluating the images, they should be compared to the corresponding ground truth images, i.e. the sharp image in the given stack. The comparison should aim to compare both the general appearance, such as color and motif, as well as the details we aim to reconstruct in the blurry image.

5.1 Mean squared error (MSE)

To compare the images pixel by pixel we used mean squared error, MSE. This is a pixelwise metric, that does not account for spatial structures in the image, and so is not a very good tool to measure details. It does, however, tell us a lot about whether the images compared resemble each other in color and overall appearance. The MSE is given by

$$\text{MSE}(Y, \hat{Y}) = \frac{1}{MNC} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=1}^{C} [Y_{i,j,k} - \hat{Y}_{i,j,k}]^2,$$

where $\hat{Y}$ and Y are image to be compared and M, N and C are the number of columns, rows and color channels respectively.

5.2 Peak signal to noise ratio (PSNR)

The peak signal to noise ratio (PSNR) builds on the MSE metric. In addition to comparing the images, it also takes the maximum pixel value into account. This scales the metric to the dynamic range of the image.

$$\text{PSNR}(Y, \hat{Y}) = 10 \cdot \log_{10} \left(\frac{\text{MAX}_Y^2}{\text{MSE}} \right),$$

where MAX_Y is the maximum pixel value of the sharp image Y.

5.3 Structural similarity index (SSIM)

The structural similarity index SSIM is described in [24] as a evaluation metric that, in contrast to the MSE and PSNR, evaluates the image quality as perceived by the human eye. The index is

calculated as

$$\text{SSIM}(Y, \hat{Y}) = \frac{(2\mu_Y\mu_{\hat{Y}} + c_1)(2\sigma_{Y\hat{Y}} + c_2)}{(\mu_{\hat{Y}}^2 + \mu_Y^2 + c_1)(\sigma_{\hat{Y}}^2 + \sigma_Y^2 + c_2)},$$

where $\mu_{\hat{Y}}$ is the average of $\hat{Y}$, $\sigma_{\hat{Y}}^2$ is the variance of $\hat{Y}$, $\sigma_{Y\hat{Y}}$ is the covariance of Y and $\hat{Y}$. The c_1 and c_2 variables are meant to stabilize the division. They are given by $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$, where L is the dynamic range of the pixel values (in our case 255) and the constants are given from [24] as $k_1 = 0.01$ and $k_2 = 0.03$.

5.4 Focus measure - ratio (FR)

In order to asses whether the focus has been improved in the generated image, we once again used the focus measure described in section 1.3. This metric was calculated by the ratio of the focus measure applied to the sharp and the generated image

$$\text{FR}(Y, \hat{Y}) = \frac{F(\hat{Y})}{F(Y)}.$$

The optimal value for this metric is 1, which means that the generated image is as sharp as the ground truth. A lower value indicates that the image has not been sufficiently sharpened. If this metric reaches a value higher than 1, it indicates that the network over-corrects the focus, resulting in a generated image with higher focus value than the sharp image. This is probably the case when the network make up dots, lines or other things in the image that does not correspond to the sharp image.

5.5 Focus measure - pixelwise (FPW)

The last metric we used was also based on the focus measure in section 1.3, this time as a pixelwise comparison between the focus level of the images Y and $\hat{Y}$. The metric is defined given by

$$\text{FPW}(Y, \hat{Y}) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} |(\omega_x * Y_G)_{i,j}^2 - (\omega_x * \hat{Y}_G)_{i,j}^2| + |(\omega_y * Y_G)_{i,j}^2 - (\omega_y * \hat{Y}_G)_{i,j}^2|,$$

where M, N are the number of columns respectively rows, Y_G is the green channel of the sharp image and $\hat{Y}_G$ is the green channel of the generated image.

5.6 Visual assessment

In order to choose the network that generates the sharpest images from a visual standpoint, we needed to include a visual assessment in our evaluation. The metrics described above were necessary in the development of the network, as they were used to choose how we proceeded with the alternations to the network. However, because the aim of the project is to make sharp images, and as this is a rather subjective property, we needed to look at the images to determine what is visually perceived as the best result.

Chapter 6

Results

In the evaluation phase of the project, we began by comparing the networks presented in Chapter 4 to each other. From there we moved on with the best performing network and tried to enhance it by tuning parameters, adding losses and changing the architecture. Lastly we tried to make the generator network more time efficient, in order to make them applicable to CellaVision's real time requirements. The method of trying different changes to the network means training many networks. Due to time constraints on the project, the networks could not be trained until fully converged. Because of this, the number of iterations they where trained for was fixed for every network in a certain comparison, in order to be able to evaluate the networks and continue with the best one. Even though the networks converge in different time, it in some way gave them the same conditions, since the time it takes to train a network to convergence is one important factor.

In Table 6.1 the evaluation metrics are applied to the test set to get a mean score for each focus level, to be able to compare further networks. As expected, images captured at the optimal focus gives the best result, as this is the ground truth. The scores presented in the rest of the tables in the chapter are the mean of all focus levels, as the last row in this table.

Table 6.1: Evaluation scores for input images X, calculated as a mean of all images of a focus level in the test set. The best score for each metric is marked in bold. At the bottom is the mean score of all the focus levels.

	MSE	SSIM	PSNR	FR	FPW
-1.4 μm	41.48	0.969451	32.10	0.1344	1279
-1.2 μm	36.10	0.976575	32.71	0.1815	1229
-1.0 μm	29.95	0.983287	33.55	0.2558	1128
-0.8 μm	22.87	0.989277	34.77	0.3763	953.0
-0.6 μm	15.32	0.994050	36.57	0.5511	710.4
-0.4 μm	8.561	0.997072	39.10	0.7573	443.9
-0.2 μm	4.623	0.998454	41.56	0.9273	244.6
0.0 μm	**0.0**	**1.0**	**50**	**1.0**	**0.0**
0.2 μm	4.317	0.998551	41.82	0.9312	233.5
0.4 μm	6.241	0.997901	40.31	0.7914	378.4
0.6 μm	9.904	0.996576	38.38	0.6310	571.7
0.8 μm	15.15	0.994389	36.53	0.4828	761.0
1.0 μm	21.10	0.991275	35.03	0.3582	923.3
1.2 μm	27.04	0.987128	33.92	0.2585	1054
Mean	**17.33**	**0.990999**	**37.60**	**0.5455**	**707.8**

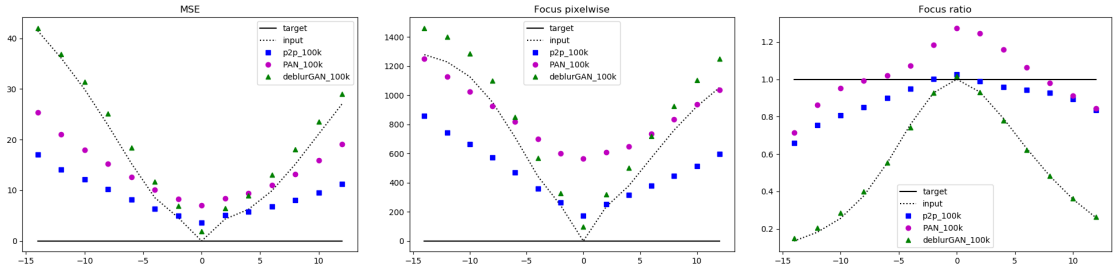
6.1 Pix2pix vs PAN vs deblurGAN

We compared pix2pix, PAN and deblurGAN when they were trained for 100 000 iterations each. The evaluation scores for the three networks are shown in Table 6.2. Here we see that the pix2pix network is superior in all metrics except the focus ratio, which gives a somewhat better result for the PAN network. However, due to the pixelwise focus measure being better for the pix2pix this indicates that the focus has been enhanced by the PAN in a way that does not correspond to the ground truth image. Compared to the evaluation of the input images (Table 6.1), the MSE is lower for both pix2pix and PAN. This suggests that the overall resemblance has been improved for these images. This is not the case for the deblurGAN, where the MSE is instead higher for the output than the input.

In Figure 6.1 the MSE, FPW and FR scores are plotted for each focus level along with the corresponding values for the input images. Here we see that both the pix2pix and PAN have flattened the curves for all three metrics, meaning they have improved the focus of the images taken far away from optimal focus. The images captured at the optimum, in the interval -0.2 μm to 0.4 μm, are somewhat worse after going through the networks. The deblurGAN largely follows the scores for the inputs, but gives a bit worse results overall. Looking at the focus ratio, we see that the PAN is more peaked than the pix2pix. This implies that the sharpening of the pix2pix is more even across the different focus levels.

Table 6.2: Evaluation scores for pix2pix, PAN and deblurGAN. The evaluation is made on the test set after 100 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix	**8.805**	**0.997212**	**39.20**	0.8921	**472.9**
PAN	13.93	0.993035	37.11	**1.019**	843.9
DeblurGAN	19.53	0.991029	36.55	0.5520	851.5



(a) MSE (b) Focus pixelwise (c) Focus ratio

Figure 6.1: Three plots showing the evaluation scores at the different focus levels for the metrics MSE, FPW and FR for networks trained for 100 000 iterations. The black solid line is the target, the black dashed line is the evaluation of the input, the blue squares is the evaluation of pix2pix, the purple dots is the evaluation of PAN and the green triangles is the evaluation of deblurGAN.

The networks were also compared by looking at the generated images using their respective generators trained for 100 000 iterations. In Figure 6.2 we can see the generated images given a blurry image captured at focus level 1.2 μm and -0.8 μm from focus respectively. We can see that both

the pix2pix and the PAN have begun to sharpen the image significantly, while the output of the deblurGAN is still as blurry as the input image. Comparing the pix2pix and PAN outputs, we can see that the details are more clear in the pix2pix image. This corresponds well to the MSE scores, that indicates that the deblurGAN result should be least similar to the ground truth, and that pix2pix result should be most similar.

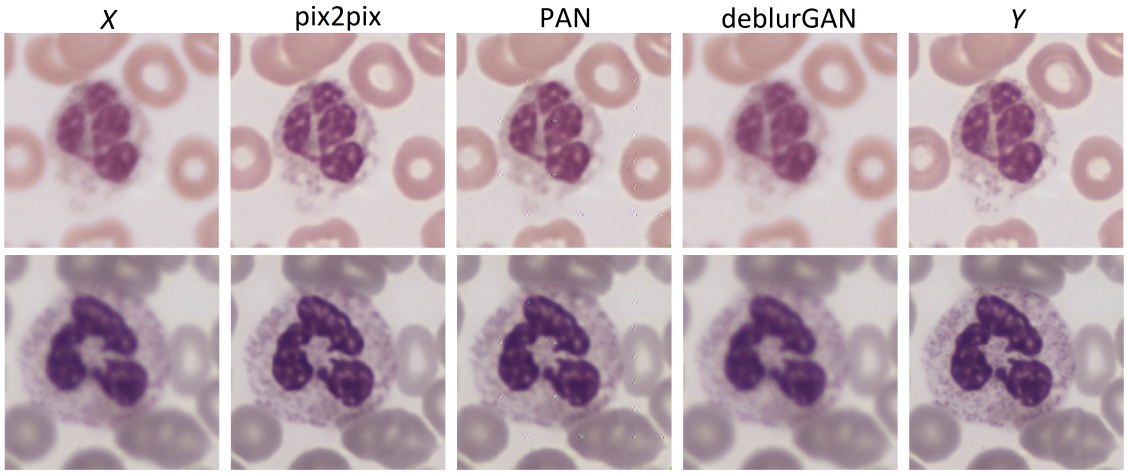


Figure 6.2: Predictions made with the three networks trained for 100 000 iterations each. To the left is the blurry images and to the right is the sharp ground truth images. The blurry images are captured at distances 1.2 μm and -0.8 μm from optimal focus. In the middle are the images generated using pix2pix, PAN and deblurGAN respectively.

It should also be noted that all outputs from our PAN has periodical artifacts, as marked in Figure 6.3. The placements of these artifacts are the same for all predictions done with the PAN that was trained for 100 000 iterations, as well as for predictions done at different stages in the training. This implies that they would probably not disappear by further training. GANs in general has in many cases given results with artifacts, which due to [23] could depend on the batch normalization and could therefore may be removed by removing the batch normalization.

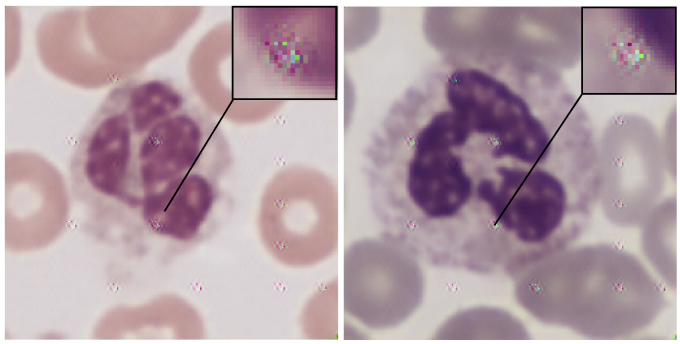


Figure 6.3: Predictions made with PAN trained for 100 000 iterations. The generated images have artifacts as marked in the images. The placements of the artifacts are constant.

In addition to producing a good result, the network should be able to do the transformation in a short time. This aspect of the networks is compared in Table 6.3, where the generators are timed on a Intel Xeon E5-1620 3.5 GHz CPU and a NVidia GeForce GTX 1050 Ti GPU for a 360×360 image and a 1920×1200 image. This is the time it takes to perform the transformation on the images of these sizes. The fastest generator is that of the PAN, which is to be expected as this is the smallest generator network. Correspondingly, the deblurGAN generator is the largest and slowest. On the GPU, both pix2pix and PAN manages to transform the small WBC images in the required amount of time, i.e. under 70 ms. However, none of the networks meet this demand on the CPU. One can also conclude that the transformations of the large scan images are pretty far away from meeting the real time requirements in all cases.

Table 6.3: The prediction time for the generator of pix2pix, PAN and deblurGAN both for the small WBC image (360×360) and for the large scan image (1920×1200). The networks are timed on a Intel Xeon E5-1620 3.5 GHz CPU and a NVidia GeForce GTX 1050 Ti GPU. The best score for each metric is marked in bold.

	360×360		1920×1200	
	CPU (s)	GPU (s)	CPU (s)	GPU (s)
Pix2pix	0.49	0.053	7.9	0.55
PAN	**0.32**	**0.028**	**5.2**	**0.41**
DeblurGAN	3.6	0.14	72	3.8

Lastly we can note that pix2pix is easier to adapt than the other networks. In pix2pix there are relatively few parameters to tune, for example due to having fewer loss weights than PAN. It is also faster to train. One training iteration on the GPU takes about 0.8 seconds for pix2pix, 2.1 seconds for PAN and 2.2 seconds for deblurGAN. This allows us to test more changes to the network in the tuning stage.

To summarize, we see that the pix2pix and PAN networks both do some work on sharpening the images. The pix2pix produced better results overall than the PAN, and the PAN had trouble with artifacts. When it comes to time efficiency, the PAN was the fastest of the generators. However, both pix2pix and PAN were fast enough on a GPU for transforming a 360×360 image and none of the networks where fast enough on a CPU. The PAN and deblurGAN networks are slower to train and have more hyperparameters, making them harder to adapt. Due to all this, we decided not to move on with PAN or deblurGAN, leaving us with the pix2pix network.

6.2 Quality of data

Before we continued to train and develop the pix2pix network, we did some evaluation of the data quality in order to see how the amount and the augmentation of data contributed to the result.

6.2.1 Amount of data

The data was gathered using four DC-1 systems, which we call systems 1, 2, 3 and 4. To investigate the effect of using more or less training data and the quality of the data from the different systems, we compared two pix2pix networks. For this, we had to arrange the data in another way and therefore non of the results correspond to the pix2pix result in Table 6.2. One network is trained on a small data set containing 863 stacks gathered using system 1 and the other networks is trained on a larger data set containing 10 059 stacks from systems 1, 2 and 3. Both systems where then evaluated on 727 stacks from the fourth system. The results of the evaluation are shown in Table

6.4. It is clear that the additional data has contributed to the training of the generator. The fact that we used data from a larger number of systems should lead to the resulting network being more adaptable to new data and more robust to outliers.

Table 6.4: Evaluation scores for pix2pix trained with the small data set respectively the large data set. The evaluation is made on the test set after 100 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Small data set	8.879	0.997325	39.06	1.097	545.8
Large data set	**7.644**	**0.997794**	**39.88**	**0.9550**	**440.1**

6.2.2 Augmentation of data

In order to see if the changes in contrast, brightness, white balance, saturation and staining of the images resulted in a more robust network, we investigated what happened when training the network with or without the augmentation. The result can be seen in Table 6.5. There one can see that network trained with augmentation gave better scores than that trained without it, and so the augmentation had a positive effect on the training. The augmentation seems to have an especially big impact on the MSE and FPW metric.

Table 6.5: Evaluation scores for pix2pix trained with and without augmentation on the training data. The evaluation is made on the test set after 100 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Without augmentation	9.344	0.996897	39.10	**0.9073**	517.1
With augmentation	**8.805**	**0.997212**	**39.20**	0.8921	**472.9**

6.3 Additional losses

The original pix2pix network has two losses; an adversarial binary cross entropy loss and an L_1 loss. A way to tweak the generator is to introduce more losses when training it. By adding losses we can decide what aspects of the ground truth images the generator should focus on mimicking in the transformation. Below we present three new losses and their results.

6.3.1 Focus loss

We defined a focus loss based on the focus measure described in section 1.3, that was added to the generator loss with a weight λ_{focus}. The loss was calculated by applying the edge detection filters on the generated image $\hat{Y}$ and the sharp image Y and comparing them in the same way as the FPW metric described in section 5.5, i.e. $\mathcal{L}_{focus} = \text{FPW}(Y, \hat{Y})$. The meaning of this loss was to tell the generator that the difference in focus between Y and $\hat{Y}$ was something it should train to minimize. The result of different weights of the focus loss can be seen in Table 6.6.

The loss weights where chosen from how much the losses contributed to the total loss. We started out by letting $\lambda_{focus} = 0.02$, since this meant that it contributed as much as the L_1 loss to the total loss. Looking at the table, one can see that it gave better result for FR and FPW while the other metrics became worse than the original pix2pix. In order to see if this was a legitimate weight loss,

we investigated what happened when picking the weight ten times smaller, i.e. $\lambda_{focus} = 0.002$. This meant that the focus loss did not contribute as much to the total loss, which probably is the reason for the FR and FPW becoming worse than before. Further investigating a 5 times larger weight, i.e. $\lambda_{focus} = 0.1$, resulted in that the FPW became better than for the original pix2pix but worse than the network with focus loss $\lambda_{focus} = 0.02$.

The reason to that the focus loss was introduced from the beginning, was to make the generator penalize not just the direct difference between the images but also the difference in focus between them. Watching Table 6.6, one can conclude that having the weight $\lambda_{focus} = 0.02$, resulted in best result in FPW and therefore was the one that best fulfilled the aim. However, it seems like the focus loss did not result in better performance in MSE, SSIM and PSNR than the original pix2pix. Because of this, we discarded this loss and moved onto testing others.

Table 6.6: Evaluation scores for pix2pix with the focus loss with different weights. The evaluation is made on the test set after 200 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix	**6.788**	**0.997891**	**40.48**	0.9125	428.7
Focus loss $\lambda_{focus} = 0.002$	7.069	0.997799	40.29	0.8945	433.7
Focus loss $\lambda_{focus} = 0.02$	8.882	0.997306	39.10	0.9515	**417.5**
Focus loss $\lambda_{focus} = 0.1$	8.817	0.997469	39.23	**0.9672**	422.2

6.3.2 VGG loss

We also tried applying a loss based on a layer of a VGG network, now to the pix2pix network instead of the deblurGAN. For this loss we used the VGG-16 network and not the VGG-19 used before. The choice of the VGG network was based on [21], where it is concluded that the VGG-16 produces better results than VGG-19 despite being a smaller network. The VGG-16 network is made up of 6 blocks, where 5 of them contains 2-3 convolutional layers each, followed by max pooling between each block. The last block is made up of three fully connected layers. The network is illustrated in Figure 6.4.

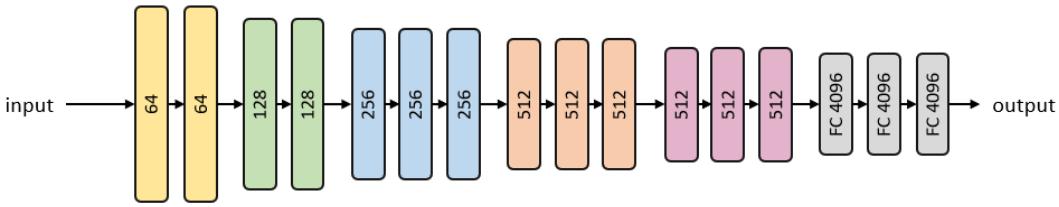


Figure 6.4: The VGG-16 network. Each color represents a separate block of the network and between each block is a maxpooling layer.

We also changed the way the loss was calculated from the deblurGAN VGG loss. Instead of using the same loss function, we used a style reconstruction loss used in [11]. This was because the style reconstruction loss penalize differences in style, i.e. colors, textures, common patterns, etc. This means that generating an image $\hat{Y}$ that minimizes the style reconstruction loss preserves stylistic features from the target image Y, which in our case included the sharpness of the image. The loss

can be described as

$$\mathcal{L}_{SRL} = ||G_j^\theta(Y) - G_j^\theta(\hat{Y})||_2^2,$$

where $G_j^\theta(Y)$ is the Gram matrix, a $C \times C$ matrix described by

$$G_j^\theta(Y) = \frac{1}{MNC} \sum_{i=1}^{M} \sum_{j=1}^{N} \phi(Y)_{i,j,c} \phi(Y)_{i,j,c'},$$

and M, N and C are the number of rows, columns and number of filters of the feature maps $\phi(Y)$.

The result for three different layers and loss weights are presented in Table 6.7. The loss was implemented with the layers at the end of block 3, 4 and 5 of the VGG-16. The loss weights were at first tuned to match the influence of the VGG loss to that of the L_1 loss, with weights $\lambda_{VGG} = 0.05$, 0.5, 10 for block 3, 4 and 5 respectively. Among these losses, the one based on the output of the fifth block produced the best result. However, the original pix2pix was still superior in all aspects except the pixelwise focus measure. To see if a larger influence of the VGG loss would give better results, we increased the weight on the loss to five times its size, $\lambda_{VGG} = 50$. As seen in the table, this gave us worse results, implying that the loss weight should not be increased. One could also have tested to decrease the loss weight, but since the result of the best layer was very similar to the original pix2pix, we did not proceed with testing more weight losses. Due to the results being worse than for pix2pix, we did not continue using this loss.

Table 6.7: Evaluation scores for pix2pix with the VGG loss with different network layers and weights. The evaluation is made on the test set after 200 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix	**6.788**	**0.997891**	**40.48**	0.9125	428.7
VGG loss, layer 3, $\lambda_{VGG} = 0.05$	15.69	0.996128	36.33	**0.9621**	527.3
VGG loss, layer 4, $\lambda_{VGG} = 0.5$	7.510	0.997879	39.89	0.8883	429.5
VGG loss, layer 5, $\lambda_{VGG} = 10$	7.101	0.997871	40.21	0.9138	**425.7**
VGG loss, layer 5, $\lambda_{VGG} = 50$	8.152	0.997770	39.50	0.9256	429.9

6.3.3 Distance loss

Next we defined a distance loss, which was based on a network by CellaVision that is trained to classify how far from the optimal focus a cell image is captured. This network was made up of five convolutional layers with intermediate max pooling layers and the last three layers are fully connected. The network was illustrated in Figure 6.5. The loss was defined from letting Y and $\hat{Y}$ go through the network, picking the difference in output from certain layers, as for the VGG network.

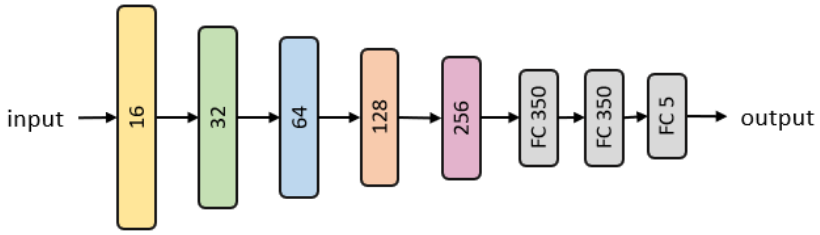


Figure 6.5: The distance classification network. Each color represents a separate block of the network and between each block is a maxpooling layer.

The loss was calculated by

$$\mathcal{L}_{distance} = \|\psi_i(Y) - \psi_i(\hat{Y})\|_1,$$

where $\psi_i(Y)$ is the feature map generated by the pretrained networks ith layer. The idea behind this loss is to train the network on features that is relevant to recognize the level of focus of the image, so that this information can be used in the deblurring. The loss is tested with different layers and weights, and are evaluated after 200 000 iterations of training. The results are shown in Table 6.8 for losses using the feature map from the third, fourth and fifth convolutional layer. The weights are tuned to match the amplitude of the loss to that of the L_1 loss, in order to give both losses equal influence on the training, giving $\lambda_{distance} = 50$, 50 and 10 for layers 3, 4 and 5 respectively.

Using the fourth and fifth layers only seems to worsen the results in comparison to the original pix2pix. However, using the third layer produces better results than the original pix2pix in all metrics. To see if a larger influence of the distance loss would further better the results, we tried increasing the weight loss to twice its size, i.e. $\lambda_{distance} = 100$. This resulted in a worse result in all metrics. We also tested a lower weight of $\lambda_{distance} = 10$ for the third layer in order to see if the distance loss contributes too much, but this only slightly increased the results for the focus ratio and non of the other metrics. Because of this we can say that the distance loss based on layer 3 with weight $\lambda_{distance} = 50$ is the best option and the one we will continue with. From now on, this network will be called *pix2pix-dist*.

Table 6.8: Evaluation scores for pix2pix with the distance loss with different network layers and weights. The evaluation is made on the test set after 200 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix	6.788	0.997891	40.48	0.9125	428.7
Distance loss, layer 3, $\lambda_{distance} = 10$	6.502	0.997952	40.71	0.9164	416.8
Distance loss, layer 3, $\lambda_{distance} = 50$	**6.433**	**0.997999**	**40.72**	0.8905	**413.6**
Distance loss, layer 3, $\lambda_{distance} = 100$	6.9634	0.997877	40.36	0.8975	422.5
Distance loss, layer 4, $\lambda_{distance} = 50$	7.2970	0.997823	40.09	0.8999	424.2
Distance loss, layer 5, $\lambda_{distance} = 10$	9.416	0.997554	38.78	**0.9240**	456.7

6.4 Two inputs

Now when having concluded that the distance loss makes the transformation even better, one can think of changing other things to this network. A simple change is the number of unfocused images that are used to produce a sharp one. By allowing the system to capture two consecutive photos at different random focus levels and input them to the network, we increase the amount of information passed to the system. The new GAN structure with two inputs is illustrated in Figure 6.6.

The network with one and two inputs are compared in Table 6.9. The comparison can not be done in the same way as before, due to that more information is passed to the one with two inputs. Therefore, they where compared by the most focused image being sent to the system, meaning that the focus levels -1.4 μm and 1.2 μm are not contained in the comparison.

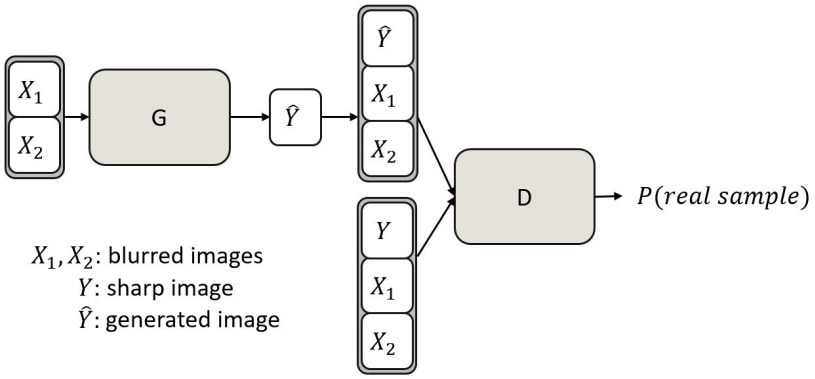


Figure 6.6: A conditional GAN with two inputs. Both the generator and the discriminator are now given two conditions X_1 and X_2.

Table 6.9: Evaluation scores for pix2pix with one and two blurry images as input to the generator. The evaluation is made after 200 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix-dist one input	5.698	0.998281	41.16	0.9061	375.2
Pix2pix-dist two inputs	**5.638**	**0.998301**	**41.22**	**0.9311**	**364.4**

From Table 6.9, one can conclude that the pix2pix-dist with two inputs performs a bit better than that with one input, but it is a pretty small difference in the result. In addition to the digital change of letting the network use two inputs, the approach of using two images calls for some changes in the mechanics of the system as well. It would need to snap two pictures of every cell. The information gained by doing this needs to be weighed against the extra time it takes to capture the second image. Since it is two different methods, both in a mechanical and a digital aspect, to use two images for the transformation, we chose to continue with both of them to see what happens when they reach full convergence. The network with two inputs is from now on called *pix2pix-dist-2inputs*.

6.5 Faster network

Since CellaVision's system is a real time system, it would be desirable that the transformation could be accomplished in no more time than it takes to perform the auto focus mechanically, i.e. 70 ms on a CPU, to not remarkably increase the time it takes to analyze a slide in today's system. This threshold could be set on the transformation time of a 360×360 WBC image or 1920×1200 scan image, depending on the application. As can be seen in section 6.1 in Table 6.3, the original pix2pix network meets this demand when transforming a 360×360 WBC image on a GPU, but not on a CPU, and not when transforming a larger image.

In order to make the transformation faster, we needed to reduce the complexity of the generator. To evaluate how different changes in the generator affect the time it takes to transform an image, we took the time of different changes. The changes involved reducing the number of blocks in the generator, reducing the number of filters in all the blocks through the generator and reducing the kernel size of each block. The reduction of the number of filters was done by dividing the number of filters in every layer by the same factor. One could go even further with these changes, by reducing them even more. However, this would make the generator lose to much complexity.

The results for the changes can be seen in Table 6.10, 6.11 and 6.12. Here one can see that even though the time reduces remarkably when reducing the number of blocks and the kernel size it turns out that reducing the number of filters reduces the time the most. Reducing the number of filters is the only change that meets the time requirement as a stand alone change for the small WBC image on a CPU. For the scan image, no changes meet the requirement on either a CPU or a GPU. We also tried to combine some of the changes to fulfill the criteria for a WBC image on a CPU. We tried a number of different combinations to see how they affected the results. The transformation times for the combinations that met the time criteria are shown in Table 6.13, and their evaluation scores are shown in Table 6.14.

Table 6.10: The prediction time for the generator of pix2pix-dist when reducing the number of blocks for the small WBC image (360×360) and for the large scan image (1920×1200). The networks are timed on a Intel Xeon E5-1620 3.5 GHz CPU and a NVidia GeForce GTX 1050 Ti GPU. The scores that fulfills the time criteria is marked in bold.

	360×360		1920×1200	
	CPU (s)	GPU (s)	CPU (s)	GPU (s)
16 blocks (original)	0.49	**0.053**	7.9	0.55
14 blocks	0.47	**0.047**	8.0	0.55
12 blocks	0.45	**0.040**	7.8	0.53
10 blocks	0.42	**0.033**	7.4	0.52
8 blocks	0.36	**0.027**	6.6	0.46

Table 6.11: The prediction time for the generator of pix2pix-dist when reducing the number of filters in all the layers through the generator for the small WBC image (360×360) and for the large scan image (1920×1200). The number of filters specified in the table is that of the first layer in the generator. The networks are timed on a Intel Xeon E5-1620 3.5 GHz CPU and a NVidia GeForce GTX 1050 Ti GPU. The scores that fulfills the time criteria is marked in bold.

	360×360		1920×1200	
	CPU (s)	GPU (s)	CPU (s)	GPU (s)
64 filters (original)	0.49	**0.053**	7.9	0.55
32 filters	0.19	**0.022**	3.0	0.30
16 filters	0.087	**0.017**	1.5	0.21
12 filters	**0.063**	**0.015**	1.1	0.20
8 filters	**0.046**	**0.014**	0.85	0.19

Table 6.12: The prediction time for the generator of pix2pix-dist when reducing the kernel size for the small WBC image (360×360) and for the large scan image (1920×1200). The networks are timed on a Intel Xeon E5-1620 3.5 GHz CPU and a NVidia GeForce GTX 1050 Ti GPU. The scores that fulfills the time criteria is marked in bold.

	360×360		1920×1200	
	CPU (s)	GPU (s)	CPU (s)	GPU (s)
Kernel size 4×4 (original)	0.49	**0.053**	7.9	0.55
Kernel size 3×3	0.35	**0.034**	5.9	0.44

Table 6.13: The prediction time for the generator when having a combination of the changes above that fulfills the requirements for the WBC image (360×360). The networks are timed on a Intel Xeon E5-1620 3.5 GHz CPU and a NVidia GeForce GTX 1050 Ti GPU. The scores that fulfills the time criteria is marked in bold.

	360×360		1920×1200	
	CPU (s)	GPU (s)	CPU (s)	GPU (s)
16 filters + 8 blocks	**0.070**	**0.014**	1.2	0.20
16 filters + kernel 3×3	**0.065**	**0.015**	1.2	0.21
18 filters + 10 blocks + kernel 3×3	**0.069**	**0.014**	1.3	0.21
20 filters + 8 blocks + kernel 3×3	**0.070**	**0.013**	1.2	0.21

The result after 200 000 iterations when having applied the changes to the generator that fulfilled the time criteria for a WBC image on a CPU can be seen in Table 6.14. There one can see that all the fast generators perform worse in all metrics but the FR compared to the pix2pix-dist with no changes. It is a bit tricky to say exactly how the reductions of filter, blocks and kernels affect the result when watching the table of combinations. However, the best performing of the smaller networks is the pix2pix-dist with 16 filters and 8 block. This is the network we chose to continue with as the fast network and from now on it is called *pix2pix-dist-fast*. An illustration of the generator of this network can be seen in Figure 6.7.

Table 6.14: Evaluation scores for the pix2pix-dist and the pix2pix-dist with the different changes in the generators that fulfilled the time criteria. The evaluation is made after 200 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix-dist original	**6.433**	**0.997999**	**40.72**	0.8905	**413.6**
12 filters	10.13	0.997066	38.51	**0.9069**	499.1
16 filters and 8 blocks	9.226	0.997254	39.00	0.9030	479.2
16 filters and kernel 3×3	13.21	0.996874	37.19	0.8343	508.8
18 filters, 10 blocks and kernel 3×3	9.343	0.997082	38.99	0.8793	493.6
20 filters, 8 blocks and kernel 3×3	9.802	0.997035	38.80	0.8792	508.6

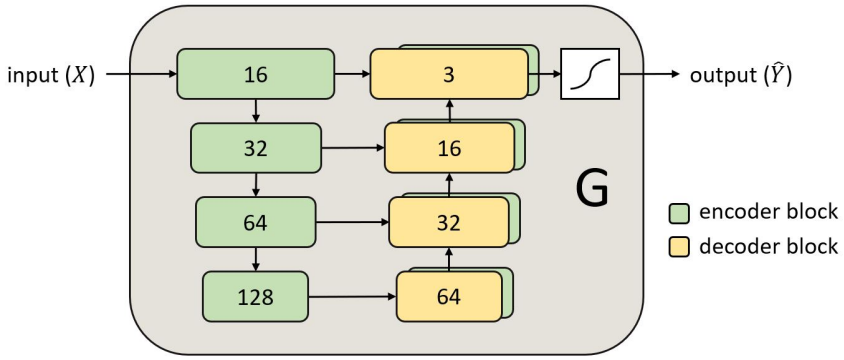


Figure 6.7: An illustration of the generator of pix2pix-dist-fast. As one can see, the blocks has been reduced to only contain 8 blocks and the filter size in the first block has been reduced to 16.

6.6 Reaching convergence

In order to get the final results when the networks pix2pix-dist, pix2pix-dist-2inputs and pix2pix-dist-fast had reached full convergence, they were trained until the validation loss did not decrease anymore. This happened for 900 000 iterations for pix2pix-dist, 1 000 000 iterations for pix2pix-dist-2inputs and 600 000 iterations for pix2pix-dist-fast. The result can be seen in Table 6.15 and 6.16. There one can see that pix2pix-dist outperforms pix2pix-dist-fast even after they have fully converged. For the pix2pix-dist-2inputs one can see that it has better mean scores than pix2pix-dist but it is in general a very small difference between the scores, just as has been concluded before for 200 000 iterations (Table 6.9). Notice that the reason for that pix2pix-dist has better scores in Table 6.16 than in Table 6.15 is that the focus levels -1.4 μm and 1.2 μm are not included.

Table 6.15: Evaluation scores for the pix2pix-dist and the pix2pix-dist-fast. The evaluation is made after 900 000 iterations respectively 600 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix-dist	**5.820**	**0.998218**	**41.22**	**0.9157**	**385.6**
Pix2pix-dist-fast	7.670	0.997545	39.97	0.9038	459.3

Table 6.16: Evaluation scores for the pix2pix-dist and the pix2pix-dist-2inputs. The evaluation is made after 900 000 iterations respectively 1 000 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix-dist	5.117	0.998483	41.67	0.9337	346.6
Pix2pix-dist-2inputs	**4.996**	**0.998551**	**41.71**	**0.9399**	**334.6**

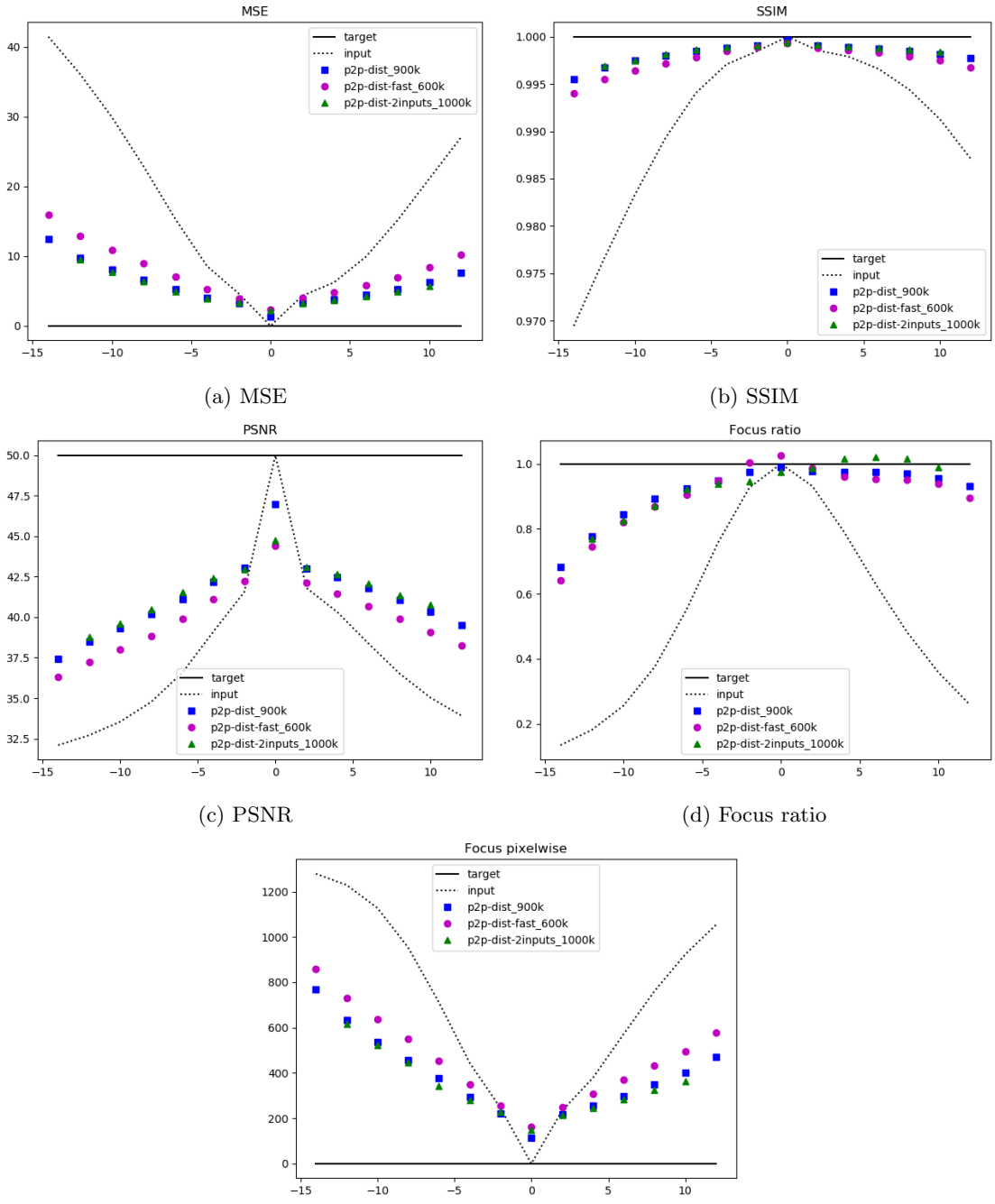
In Figure 6.8, the score for MSE, SSIM, PSNR, FR and FPW has been plotted for each focus level for each of the networks. There one can see the same pattern as in the tables, i.e. that pix2pix-dist-2inputs is best, closely followed by pix2pix-dist and that pix2pix-dist-fast is the poorest of them. One interesting thing is that pix2pix-dist seems to be best close to the optimal focus. It also seems like pix2pix-dist-fast diverge more from the others far away from the focus. Generally, one can notice that the curve has flattened out relative to the input for all of the networks, which was part of the the aim.

In Figures 6.9-6.14, we present the results from some transformations using the final networks together with the MSE and FPW scores. Here one can see that in general, the transformations produce good results. Starting with Figures 6.9-6.12, we see examples of severely blurry images being sharpened. The output of the pix2pix-dist network looks to be the sharpest and is very close to the target image. The $\hat{Y}$ of pix2pix-dist-2inputs is somewhat less sharp, which can be seen when looking at the high frequency details. The image generated by the fast network is the least sharpened of the three, but is still significantly sharper than the input image. This also reflects the score of MSE and FPW for most of the images. Notice however that the scores can not directly be compared between images visualizing different cells.

Figure 6.13 shows an example of a ground truth image being sent through the networks. Here we can see that the outputs of all the generators are equally sharp, and are indistinguishable from the ground truth. This implies that there are no disadvantages of applying the network, no matter the focus level of the input. By looking at the score of the generated images, one can conclude that pix2pix-dist is the best of them, but it is not sure that this reflects all the images that are sharp before they are sent to the network.

An example of a poorly focused image is shown in Figure 6.14. Though the outputs are sharper than the input, as can be seen in the edges, it does not reach the desired quality. In the input image, many details are lost within the WBC. These are to a small degree reconstructed in the generated images, with the pix2pix-dist output being the best.

Even more predictions can be seen in Appendix B.



(a) MSE

(b) SSIM

(c) PSNR

(d) Focus ratio

(e) Focus pixelwise

Figure 6.8: Five plots showing the evaluation scores at the different focus levels for the metrics MSE, SSIM, PSNR, FR and FPW for the three final networks, i.e. pix2pix-dist, pix2pix-dist-fast and pix2pix-dist-2inputs. The black solid line is the target, the black dashed line is the evaluation of the input, the blue squares is the evaluation of pix2pix-dist, the purple dots is the evaluation of pix2pix-dist-fast and the green triangles is the evaluation of pix2pix-dist-2inputs.

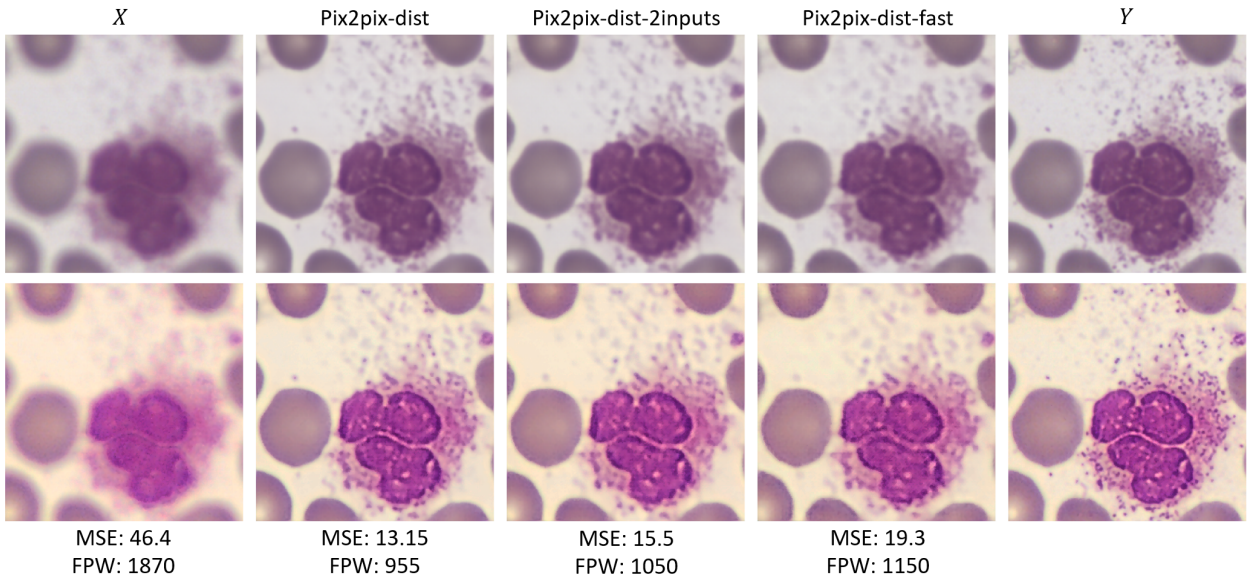


Figure 6.9: Results of transformations of a cell image taken at -1.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

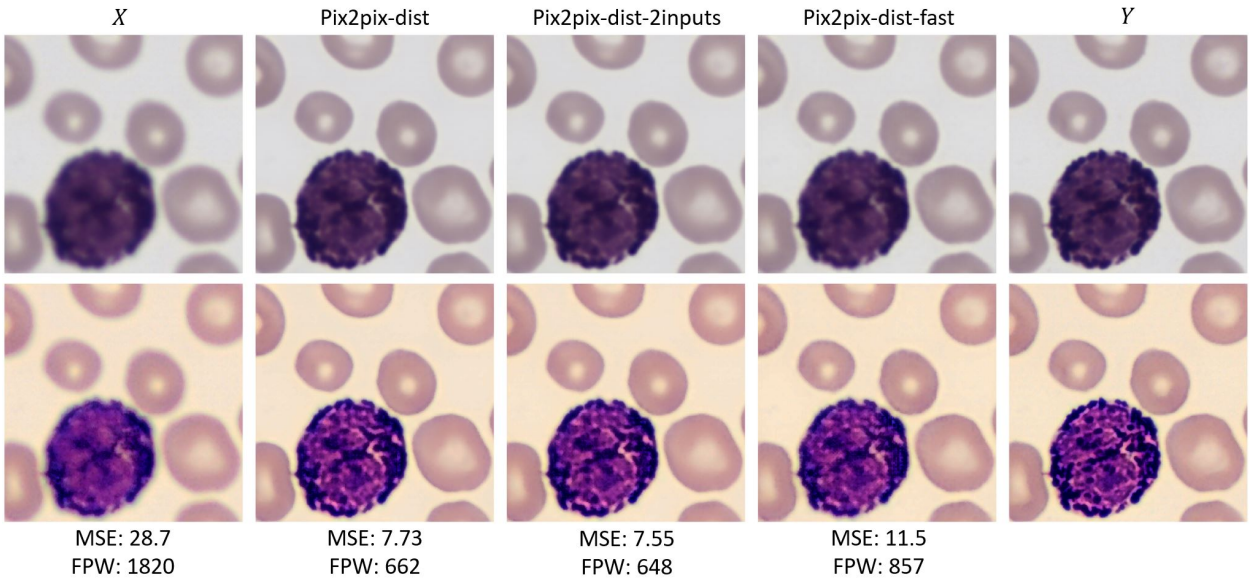


Figure 6.10: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

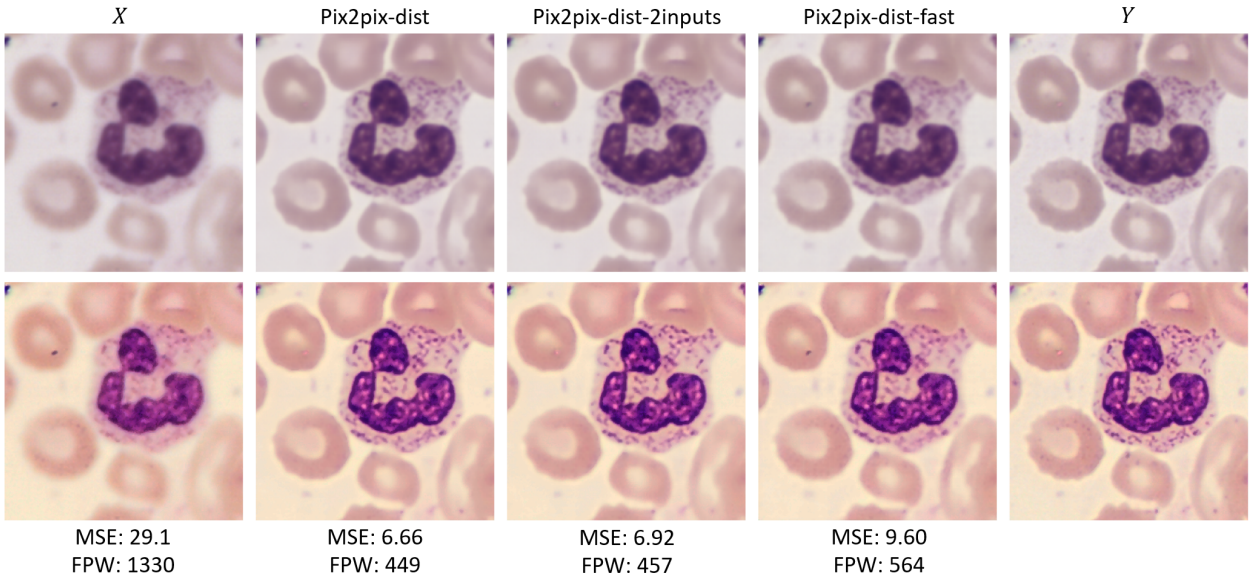


Figure 6.11: Results of transformations of a cell image taken at 1.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after.The MSE and FPW scores are written under respectively image.

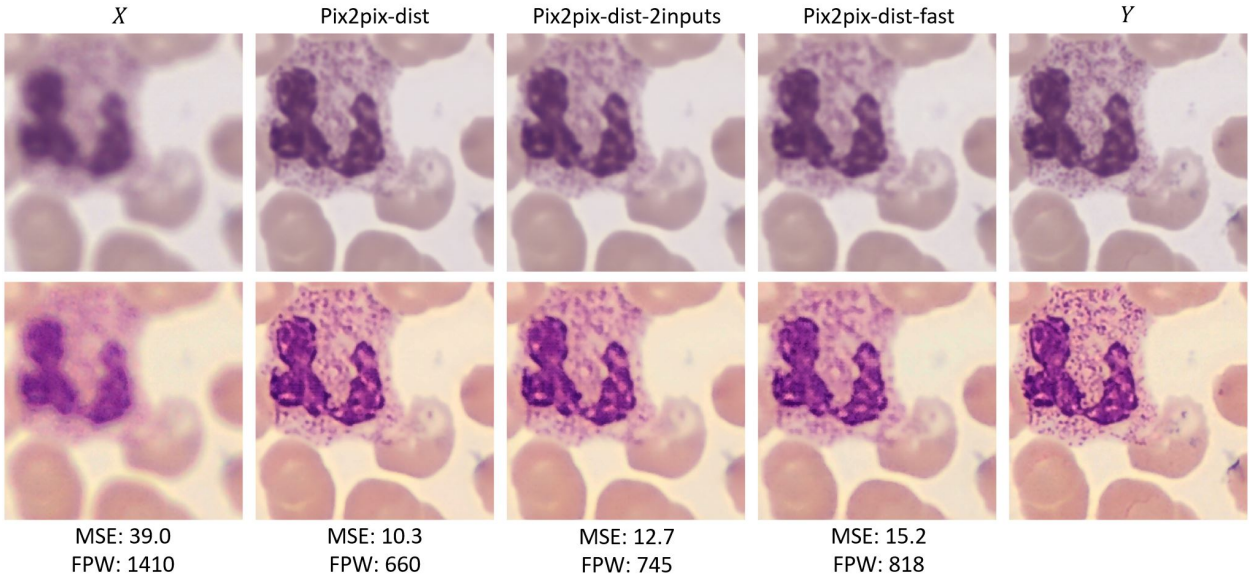


Figure 6.12: Results of transformations of a cell image taken at -1.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

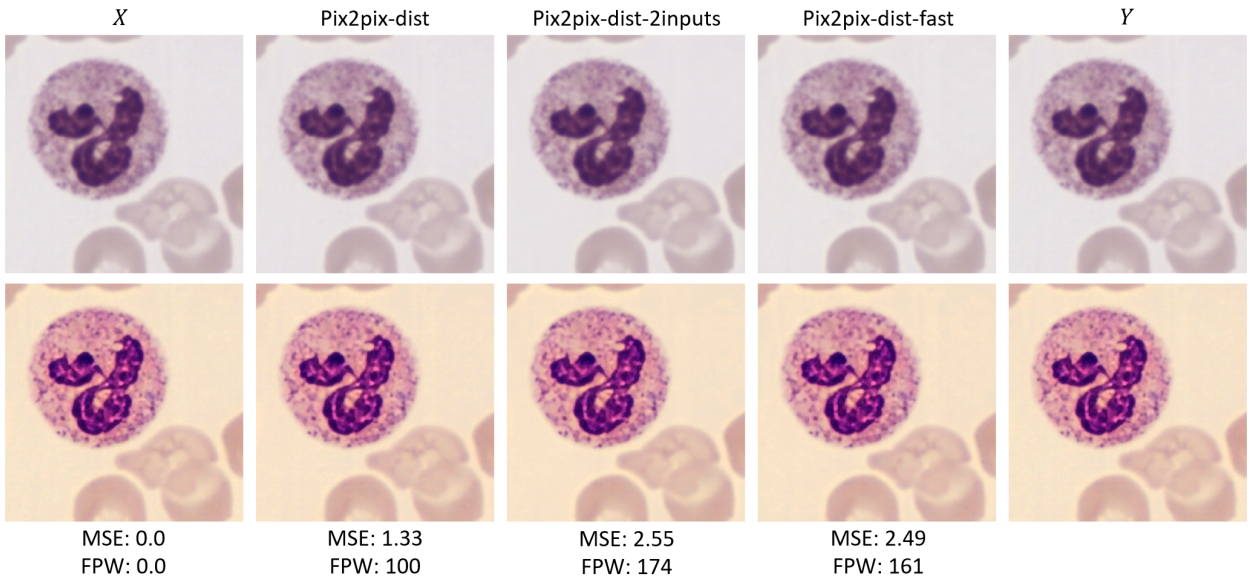


Figure 6.13: Results of transformations of a cell image taken at 0.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

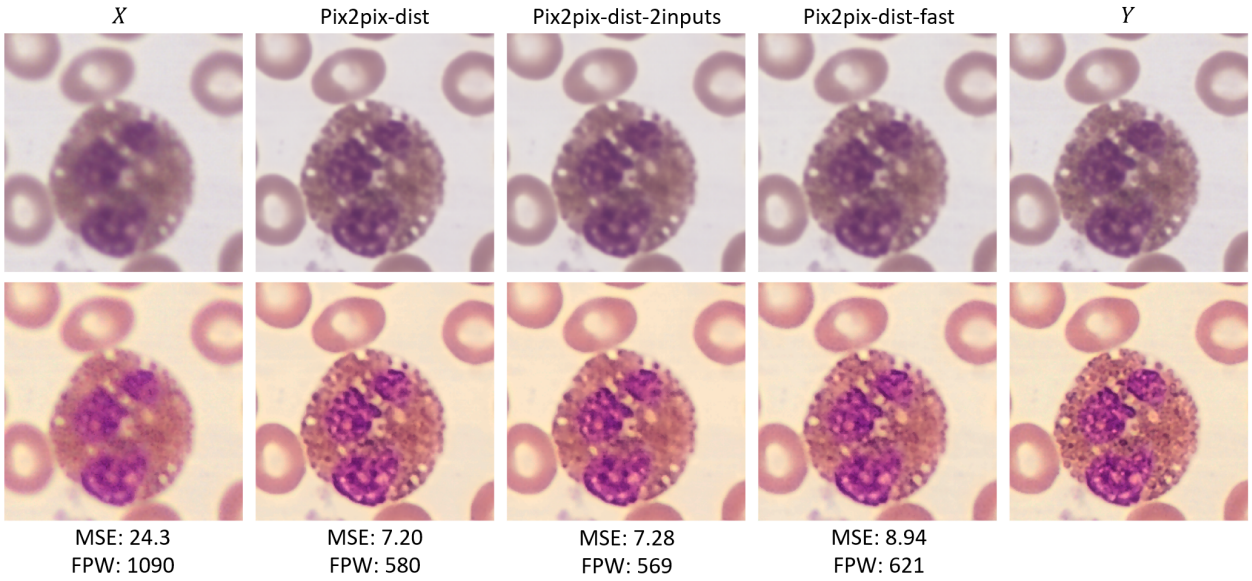


Figure 6.14: Results of transformations of a cell image taken at -0.6 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

6.7 Unbalanced networks in pix2pix

The objective of the project was to investigate whether a GAN could be used to sharpen unfocused images. The choice to use a GAN was mainly based on the fact that this type of network has been used for similar tasks before and so should have some fitting properties for the problem. In the end we wanted to assess the value of using a GAN rather than a simple CNN. We did this by using the generator of the original pix2pix as a stand alone CNN and comparing it to the GAN. By leaving out the discriminator we reduced the network to being a CNN without any adversarial behaviour. This means that it was no longer trained using a adversarial loss, but only a L_1 loss. The results of this can be seen in Table 6.17. Here one can see that the CNN even outperforms the GAN in most of the metrics.

Table 6.17: Evaluation scores for the original pix2pix GAN and the pix2pix generator as a stand alone CNN. The evaluation is made after 150 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix GAN	7.791	0.997769	39.77	**0.9370**	448.0
Pix2pix CNN	**6.961**	**0.997943**	**40.27**	0.8969	**425.0**

To find out why this was the case we studied the training loss for pix2pix. This seemed to indicate that the generator and discriminator were unbalanced, as the discriminator loss reached zero very quickly. This could prevent the discriminator network from helping the generator to learn, as the generator must sometimes succeed in order to find out how to fool the discriminator.

Due to that the loss of the discriminator went to zero very fast, our suspicion is that the discriminator was "too good". Though we want the discriminator to perform well, the risk of a too good discriminator is that the adversarial behaviour collapses and the generator is left only being guided by its other losses. This is a common problem with GANs and is described in [9]. To fix this, we tested a number of changes to make the networks more balanced. The changes are listed and described below.

- Soft target labels for the discriminator
- Smaller discriminator network
- Training ratio 1:2
- Smaller learning rate for training of discriminator
- Train separate on real and fake samples

The first change was based on [4], where it is suggested that the discriminator in a GAN should not be trained with hard labels, i.e. 0 or 1. Instead the labels should be soft, meaning that they are randomly sampled from an interval. We choose the intervals $[0, 0.1]$ for false and $[0.9, 1]$ for true. This is to prevent the discriminator from forcing the classification to be to certain, and instead output soft probabilities that are more easily adapted during the training.

A way to make the discriminator less complex is to remove some of its layers. In the pix2pix terminology, this would result in a patchGAN with smaller patches. By removing the last two convolutional layers of the discriminator, we got a 16x16 patchGAN instead of the 70x70 patchGAN used in the original pix2pix. The new discriminator is shown in Figure 6.15.

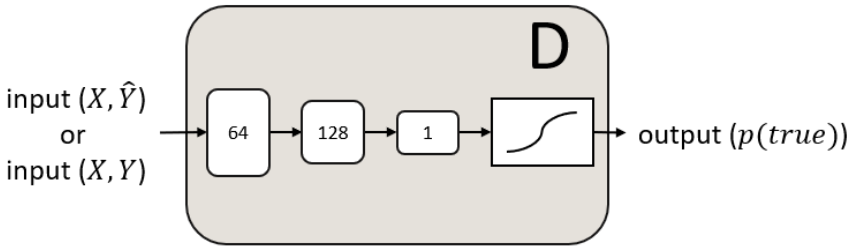


Figure 6.15: The pix2pix discriminator with fewer layers.

If the discriminator is too good, a way of combating this would be to tone down the training of it. This would allow the generator to catch up in the case that it is not good enough. We tried doing this in two different attempts. First, we changed the training ratio to 1:2, which means that we let the generator train twice as often as the discriminator. In a second attempt, we tried lowering the learning rate for the discriminator to be a tenth of the learning rate of the generator ($2 \cdot 10^{-4}$ vs $2 \cdot 10^{-3}$). Both these changes let the generator get more of an upper hand in the training, and could stop the discriminator from outperforming the generator.

During training the discriminator is passed both real and generated samples in order for the network to learn to distinguish between them. Our way of doing this was to, for each batch of (X, Y) that is sent through the generator training, construct a batch of both the ground truth and corresponding generated samples, $([X, X], [Y, \hat{Y}])$. However, according to [2], the real and fake samples should not be passed to the network in the same batch, but rather in separate mini batches.

After the training of each of the pix2pix networks with changed discriminators, we evaluated them. The results are presented in Table 6.18 along with the original pix2pix and the standalone generator CNN. To assess the performance of the discriminators we study their outputs given blurry images, as well as generated and real sharp images, the results are given in Table 6.19. The results are discussed in section 7.5.

Table 6.18: Evaluation scores for pix2pix with the discriminator changes described in section 6.7. The evaluation is made on the test set after 150 000 iterations and is represented by the mean score of all images and focus levels. The best score for each metric is marked in bold.

	MSE	SSIM	PSNR	FR	FPW
Pix2pix	7.791	0.997769	39.77	**0.9370**	448.0
CNN	6.961	0.997943	40.27	0.8969	**425.0**
Soft labels	6.792	0.997893	40.52	0.8783	438.6
Smaller D	7.993	0.997043	39.71	0.8450	459.6
Training ratio 1:2	6.709	0.997916	40.55	0.9020	430.3
Lower learning rate	**6.653**	**0.997964**	**40.60**	0.8843	427.5
Separate batches	17.66	0.992601	36.49	0.6593	794.9

Table 6.19: Discriminator outputs given X, $\hat{Y}$ and Y for the different changes in the discriminator. The evaluation is made on the test set after 150 000 iterations.

	$D(X)$	$D(\hat{Y})$	$D(Y)$
Pix2pix	0.111	0.0984	0.189
Soft labels	0.573	0.0478	0.594
Smaller D	0.999	0.893	0.999
Training ratio 1:2	0.990	0.000135	0.996
Lower learning rate	1.0	1.0	1.0
Separate batches	0.214	0.211	0.214

Chapter 7

Discussion and conclusion

7.1 Quality of results

The aim of the thesis was to transform images from blurry to sharp, and to determine how far away from focus an image can be captured and still be sharpened with the network. As seen in Chapter 6, the networks succeed in sharpening the images to a certain degree. What is left to determine is whether the results are good enough to be considered properly sharpened, and how this varies with the distance from focus.

From studying the data set, it can be noted that the images captured at focus levels -0.2 μm to 0.4 μm are all very similar. This can for example be seen in Figure 2.3 in Chapter 2. Because of this, all images at these focus levels were considered to be sharp. Furthermore, we decided to set these images as the threshold for what is sharp when doing our final evaluation. In other words, if we can get the generator to produce images that are as good as these images, the transformation to sharpness has succeeded. To assess the results of the transformations we compare the metric scores calculated for the transformed images $\hat{Y}$ to those of the input images X of the corresponding stack. Through this comparison we can get an idea of how much the generator has changed the images. The goal would be that the evaluation scores of any $\hat{Y}$ would be as good as those of the X images taken at focus levels -0.2, 0.0, 0.2 and 0.4 μm of the same stack.

In Tables A.1-A.6 in Appendix A, the generated images are compared to the input images for the test set with respect to the MSE and FPW metric. The rows of the table contains the probability of the score for the generated image $\hat{Y}$, at a certain focus level, being less than or equal to the score of the input images X at the different focus levels in that stack. The goal is to only have non-zero values in columns -0.2, 0.0, 0.2 and 0.4, as this would mean that the generated images are always sharpened properly. However, since the input image X with focus level 0.0 is the same as the ground truth, the MSE and FPW scores are always 0 for X at this level. This means that the generated image would need to be exactly the same as the target image for the probability of the center column to be non-zero, which does not happen.

One thing to note in these tables is that the probabilities in the first three columns are always close to zero. This means that the generated images are almost always better than the images captured at distances -1.4 to -1.0. The rightmost columns, however, are non-zero. The higher probabilities here has to do with how the focus changes when moving the camera in different directions. As there is an asymmetry in the metric curves (such as those that can be seen in Figure 6.8), there will be a higher probability that the sharpened images resemble the X captured in the interval 0.2

to 1.2 than the corresponding interval on the other side of the optimal focus. This asymmetry can also be seen to affect the success rate. When moving 0.8 μm closer to the slide, the success rate of the transformation is 95% for pix2pix-dist, while moving the same distance away from the slide lowers the success rate to 61%.

The success rate for the three networks, based on MSE and FPW, for different intervals can be seen in Table 7.1. The columns are intervals for the distance from focus the images may be captured, i.e. $-1.2 : 1.2$ means the interval from -1.2 to 1.2 μm from focus. The success rates are calculated for the three final networks and are based on the MSE and FPW scores. The reason to that pix2pix-dist-2inputs has missing values for the interval $-1.2 : 1.2$ is because it has two input images and the best of the input images always lays somewhere in the interval $-1.2 : 1.0$. The table shows that the best performing network for each of the intervals is the pix2pix-dist network, which can successfully transform 93.3% respectively 87.3%, of all images captured between focus levels -0.8 and 0.8 μm, with respect to MSE and FPW. It is closely followed by the pix2pix-dist-2inputs with a success rate of 92.4% for MSE and 86.4% for the FPW in this interval. The pix2pix-dist-fast network gives the poorest result and could properly sharpen 78.9% according to the MSE and 77.2% according to FPW for images from the same interval. This is to be expected as its small network size sets limits on the transformation. We can see that the fast network performs well in the smallest interval, however. This interval only contains images generated from sharp inputs, and does not call for an advanced transformation.

Table 7.1: The success rates (%) for the three final networks based on the MSE and FPW, given for different intervals, based on the focus level being closest. The numbers are calculated from the tables in Appendix A.

		$-1.2 : 1.2$	$-1.0 : 1.0$	$-0.8 : 0.8$	$-0.6 : 0.6$	$-0.4 : 0.4$	$-0.2 : 0.2$
Pix2pix-dist	MSE	79.1	87.0	93.3	97.6	99.7	100
	FPW	73.8	80.9	87.3	92.9	97.4	99.5
Pix2pix-dist-2inputs	MSE	–	85.5	92.4	97.6	99.7	99.9
	FPW	–	79.2	86.4	92.3	97.2	99.3
Pix2pix-dist-fast	MSE	58.2	67.8	78.9	90.2	97.5	100
	FPW	59.6	68.0	77.2	86.6	94.7	99.0

We chose to asses the results using MSE and FPW, due to that MSE being widely used and established respectively that FPW is based on a measure specific for CellaVision. SSIM and PSNR always correspond to the MSE in the tables, and so they strengthen the choice of MSE. The MSE compares the whole image pixel by pixel and is therefore suitable for an image generation problem, while the FPW compares the details that are detected with the focus measure. This means that they compare different aspects between the images, which is probably the reason for that the success rates based on MSE and FPW differ a bit. To say exactly which metric that is the best, one would need an expert to look at many different images of generated cell images, to determine what best describes a focused image.

As can be noticed, the mean of all metric scores implied that pix2pix-dist-2inputs generates the best images, but when visually assessing the results pix2pix-dist seemed best. Also, the success rates imply that pix2pix-dist is the best. The difference in result is a bit strange and is hard to explain. It could depend on that pix2pix-dist-2inputs in general has a higher standard deviation for the metrics and therefore has better result on some and worse result on some. It is also a randomness in pix2pix-dist-2inputs that could affect, since it takes in two blurred images at different

focus levels which means that the scores in the metrics could change depending on the outcome. The reason for that pix2pix-dist-2inputs does not produce better results even though having more information from two images, could be that it has no way of telling which of the inputs is the sharpest. This might lead to that it collects blurriness from the blurrier of the images which could worsen the result.

7.2 Model selection

The choice of the three networks pix2pix, PAN and deblurGAN was based on that they had performed well in similar transformation and because they had different architectures, losses and training methods. The deblurGAN was not developed to fix focus blur, but rather motion blur, which could be one of the reasons for it not producing focused results. This network tries to estimate the PSF for each image, which may be harder when working with focus blur. PAN on the contrary, produces pretty good results but has some artifacts. One idea is that those artifacts depend on the batch normalization and may therefore have disappeared after removing the batch normalization. Another idea is that it could depend on the perceptual loss it uses and that using different loss weights may have removed them. Both deblurGAN and PAN might also have performed better if they were trained until fully converged.

A large aspect of the development of the final network was the choice of losses. As a loss determines what the network should aim to accomplish, this was a vital part of the design. The original pix2pix network has two losses, the binary cross entropy used to train the generator and the discriminator against each other, as well as the pixelwise L_1 loss used only in training the generator. The idea of the L_1 loss is to encourage the generated image to resemble the target image when compared pixelwise. One could think that this loss by itself should be sufficient when creating a network whose aim is to translate between two images. However, as described in [15], the L_1 loss tends to produce blurry results on image generation problem. Since our translation problem is centered around deblurring images, this loss may not be sufficient. It can, however, in many cases accurately capture the low frequencies in the image.

The addition of losses to the pix2pix network was limited to testing three different types: Focus loss, VGG loss and distance loss. The two perceptual losses, i.e. VGG and distance loss, were tested by extracting features from different layers, which is something that could be done even more, as some of the layers were not tested. The losses could be further investigated by combining the different losses, or by doing feature extraction from more than one layer and combining them as a sum in the perceptual losses. One could also use completely different losses, that have not been explored in this project. For example one could define a new perceptual loss on the classification network used in CellaVision's systems. As this network extracts features that are important when classifying cells, it could help the generator to preserve relevant information. This could be an issue with the VGG loss, as this is based on a network that is trained on a vast variety of images, and so the features it extracts could be irrelevant for this problem.

When it comes to pix2pix-dist-2inputs, the two inputs were aligned before passed to the network. When the system captures the images, it is not always certain due to mechanical issues. This implies that you either have to create a network that could handle inputs that are not perfectly aligned or that you have to align the images before you send them to the network, which would take more time. This is something that should be taken into consideration when choosing the method, as it could affect either the time or performance of the network.

7.3 Potential risks with the method

The process of choosing our optimal network was based on a series of tests of different architectures, losses and hyper parameters. Because of the time limitations on the project, all networks could not be trained until fully converged. This could possibly mean that the one that was best in a certain test would not be the best after convergence. The fact that the initial state of every network is randomly assigned could also affect which network is the best. One solution for making the results more certain would be to train the same network from start to finish a number of times and then averaging the results. These issues could also be minimized by initializing all networks with the same network weights and letting them train until convergence, if time permitted.

In order to choose the best network at any step of our tests we used the evaluation metrics described in Chapter 5. These were chosen as numerical ways of assessing the quality of the generated images, which was necessary in order to efficiently compare and find the best alternative among the networks. Due to the results of the different networks being very similar, the metrics also helped in finding the best alternative when it was difficult doing this by simply looking at the images. Using the metrics could come with some problems however. Due to the fact that the focus of images is essentially a subjective property, a numerical and a visual assessment might not always coincide. Therefore it would have been desirable to let an expert visually assess the result to ensure that the network chosen is the best, but it takes resources and it is a very time consuming process.

Using a machine learning approach to focus a blurred cell image instead of mechanically finding the correct focus comes with some risks. As could be seen in the images, sometimes important details in the cells are lost which could lead to that the biomedical scientist makes another assessment of the blood. This is very dangerous since it could lead to incorrect conclusions of human diseases. It would therefore be desirable to know how much important information that is lost in the image when moving the objective away from the optimal focus in order to say how far away one can go. Another aspect is that using a machine learning method means that it could come up with details in the image that does not exist, which could be just as bad as losing information. This is however something that has not been identified in the images we have seen for the test set.

7.4 Faster network

Since CellaVision's systems do not have a GPU, one of the requirements on the network was to make the generator transform images on a CPU in 70 ms or less. In section 6.5 we tried to make the generator faster by reducing the number of filters, the number of blocks and the kernel size. One could proceeded in another way when making the generator smaller. For example by investigating which steps in the network that where time consuming and removing those one at a time. We could also have investigated if there where any filters or connections with weights close to zero, since those may not contribute to the result and therefore could be removed. Another thought is to continue reducing the number of filters, the number of blocks and the kernel size even more, but from watching the result, one can conclude that this probably would have resulted in worse performance.

The result from making the generator faster was that we made it possible to transform a 360×360 image on a CPU in around 70 ms and around 15 ms on a simple GPU, while the fastest time for a 1920×1200 image on a CPU was around 1 s and 0.2 s on a simple GPU. This means that if it should be applied to CellaVision's systems today, it would be most reasonable to only focus the WBC images digitally and not the scan images. By doing this, the system only needs to use the mechanical auto focus once per analysis to focus a scan image. When proceeding to capture the

WBC images, it only needs to stop very fast to take one or two images without applying the auto focus again, to then send these images through the network.

It could be an idea for CellaVision to upgrade the systems with a GPU, especially if they want to use our method. Since we saw that it is possible to affect the speed of the transformation remarkably, this is truly an advantage and a simple way to improve a system. This is not simply the case when it comes to mechanically focusing the images, as making this faster is often harder than just adding one component to the system. There are many more powerful GPUs available today than the one used for timing in this project. As big data and deep learning is a constantly growing field, the need for better GPUs is always increasing. By upgrading the systems with a more powerful GPU, the transformation could be done even faster. This might lead to that the best of our networks could be used in the future, or that the auto focus could be replaced with the network for the larger scan image as well. A drawback with a GPU is the extra effect it uses and the physical size of it, which could affect the system in other ways.

7.5 The effectiveness of GAN

In our project we have primarily used GANs to solve our problem. However, when comparing the pix2pix with and without the discriminator, the generator as a stand alone CNN produces better results. This would indicate that the discriminator does not contribute to the network in a positive way. There could be several reasons for this. As mentioned earlier, it could mean that the discriminator and generator networks are unbalanced, resulting in a poor collaboration between the two.

This was investigated through a number of changes to the pix2pix discriminator. It seemed here that some of the changes could produce better results than the original network. Looking at the results in Table 6.18, we see that the discriminator with lower learning rate is the networks that most outperforms the original pix2pix and CNN. However, the discriminator does not seem to work for this network. According to Table 6.19, the discriminator always outputs 1, no matter the input. So even though this discriminator change improves the performance of the generator, the discriminator does not seem to be an active part of the training. When it comes to the other discriminator changes, the networks with training ratio 1:2 and soft labels have the best performing discriminators, as well as quite good evaluation scores. This might imply that these changes improve the adversarial behaviour, something that could be taken into consideration in future work.

The fact that the CNN was better than the GAN could also mean that the GAN concept is not a good fit for this type of problem. GANs are often used to make style transfers, where translations are made between two different types of images (e.g. a photo and a painting by a certain artist). In these cases a simple pixelwise comparison is not sufficient, as the thing that is learned is the overall type of image which is more general than this. It could be that the issue of sharpening unfocused images is a simple enough transformation, so that there is no need for the adversarial behaviour of the GAN. In our transformation, all information is essentially in the input image. The task of the network is therefore not to transfer information from the target, but rather to learn from the target how the information in the image should be rearranged.

Just because CNN gives better results, it does not necessarily mean that GAN is a worse alternative than the CNN. It could be a consequence of the randomness of deep learning. It is however clear that the adversarial part of the GAN does not contribute remarkably to the results. Combined with the long training time that comes with having two networks (generator and discriminator), the value of using a GAN instead of a CNN seems low for this problem.

7.6 Quality of data

One important factor that plays a big role when training a network is the quality of the data. As could be seen in section 6.2, the result improved when having more data from different systems and when using the augmentation. Even though the coloration in the augmentation is not always realistic, it probably makes the generator concentrate on focusing the images and not to learn different colors. In terms of the amount of data, more data would probably have improved the result even more, which is the general case for all deep learning methods. It is more a question about time and resources.

Another aspect of the quality of data is the diversity of the images. The cell images used contain white blood cells of many cell classes, which vary in appearance. Since the frequencies of occurrence differs between the types of cells, some classes will be over represented in the training data. This could result in that the network has problems with transforming the more uncommon cell types. A way to prevent this is to make sure that each of the cell class occurs in fairly equal amount, but this implies that either more amount of data would be needed or that part of the data used needed to be discarded.

Calling back to the focus curves in section 2.1.1, we saw that the data that was collected using CellaVision's systems was not always properly focused. This led to that the optimal focus needed to be located after collecting the images. In 60.8% of the stacks we collected, the optimal focus needed to be shifted in some way. The optimal focus in 4.5% of all stacks was not even in the interval of sharpness. This means that even today, the system is not perfect and sometimes disturbances or lacks in mechanics can cause the system to fail in capturing focused images. Which is something to take into account when evaluating and drawing conclusions about our method.

7.7 Conclusions

To conclude, it is possible to transform an unfocused image to a sharp one using GANs. Our fastest network, pix2pix-dist-fast, met the time requirements of transforming a 360×360 image in 70 ms on a CPU. It could properly sharpen around 78% of images captured at focus levels -0.8 to 0.8 μm. Our best network was pix2pix-dist, which can successfully transform around 90% of all images captured between focus levels -0.8 and 0.8 μm. Regarding how much one can deviate from the optimal focus position for the transformations to work, it heavily depends on the direction of the camera movement. When moving 0.8 μm closer to the slide, the success rate of the transformation is 95%, while moving the same distance away from the slide lowers the success rate to 61%. To be able to conclude how far away one can go from the optimal focus, one needs to set a limit for how much errors that is allowed, which we leave for CellaVision to decide.

7.8 Future work

Although the results were satisfactory, there is room for further development of the method. The networks we started from, i.e. pix2pix, PAN and deblurGAN, were not the only previously presented networks used for image-to-image transformations. There are a wide variety of architectures and network losses that could be investigated and, considering the large difference between the three networks we tested, it could be valuable to look into more of these.

When developing the fast network we did a series of tests where the network size was systematically reduced, with no particular respect paid to the effect of the different parts of the network.

This could be done differently, by example removing parts whose contributions were small in relation to their time cost. There are many functions in Python that could help with this as well. Another way of speeding up the network is to implement it in a low level language.

Since we saw that the discriminator did not contribute to the final result that much and that there were changes in how the GAN was trained that improved the results, one could further investigate these changes. One could also, instead of using GANs, develop a CNN with other architecture and losses, since we saw that a CNN generated satisfactory result.

Bibliography

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, arXiv:1701.07875v3, 2017.

[2] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu. How to Train a GAN? Tips and tricks to make GANs work. https://github.com/soumith/ganhacks, 2016.

[3] L. Dean. Blood Groups and Red Cell Antigens: Chapter 1, blood and the cells it contains. Bethesda (MD): National Center for Biotechnology Information (US), 2005.

[4] U. Desai. Keep Calm and train a GAN. Pitfalls and Tips on training Generative Adversarial Networks. https://medium.com/@utk.is.here/keep-calm-and-train-a-gan-pitfalls-and-tips-on-training-generative-adversarial-networks-edd529764aa9, 2018.

[5] X. fen Wan and X. L. Yi Yang. Point Spread Function Estimation For Noisy Out-offocus Blur Image Restoration. IEEE, 2010.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. *ArXiv e-prints*, arXiv:1406.2661v1, 2014.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GAN. *ArXiv e-prints*, arXiv:1704.00028v3, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *ArXiv e-prints*, arXiv:1512.03385v1, 2015.

[9] J. Hui. GAN — Why it is so hard to train Generative Adversarial Networks. Medium, 2018.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks, v3. *ArXiv e-prints*, arXiv:1611.07004v3, 2018.

[11] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ArXiv e-prints, 2016.

[12] J. Jönsson and E. Sjöstrand. Cell Image Transformation Using Deep Learning. LTH, 2018.

[13] A. Karpathy. Convolutional neural networks (cnns / convnets). *Convolutional Neural Networks for Visual Recognition*.

[14] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks, v4. *ArXiv e-prints*, arXiv:1711.07064v4, 2018.

[15] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. ArXiv e-prints, 2016.

[16] D. H. Marimont and B. A. Wandell. Matching color images: The effects of axial chromatic aberration. Journal of the Optical Society of America A. 11 (12): 3113, 1994.

[17] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *ArXiv e-prints*, arXiv:1411.1784v1, 2014.

[18] M. Nasse and J. Woehl. Realistic modeling of the illumination point spread function in confocal scanning optical microscopy. Journal of the Optical Society of America A Vol. 27, 2010.

[19] K. Scaman and A. Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *ArXiv e-prints*, arXiv:1805.10965v1, 2018.

[20] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints*, arXiv:1409.1556v6, 2015.

[21] S.-H. Tsang. Review: VGGNet—1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014. Medium, 2018.

[22] C. Wang, C. Xu, C. Wang, and D. Tao. Perceptual Adversarial Networks for Image-to-Image Transformation, v2. *ArXiv e-prints*, arXiv:1706.09138v2, 2017.

[23] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. ArXiv e-prints, 2018.

[24] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, 2, 2003.

Appendix A

Tables for evaluation

Table A.1: Evaluation of the pix2pix-dist network trained for 900 000 iterations. The table shows the probability (%) of the MSE score of a generated image $\hat{Y}$, of a certain focus level, being lower than the score of the different images X in the corresponding stack.

		X													
	FL (μm)	-1.4	-1.2	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2	**SR** (%)
$\hat{Y}$	-1.4	0	0	0	1.7	17.9	15.5	0.1	0	0.1	9.5	27.8	22.1	5.0	0.3	**9.7**
	-1.2	0	0	0	0.4	11.7	17.1	2.2	0	0.4	20.1	29.3	16.9	1.9	0	**22.7**
	-1.0	0	0	0	0.1	5.9	15.3	5.1	0	2.2	34.4	23.8	12.5	0.7	0	**41.7**
	-0.8	0	0	0	0	1.3	10.7	12.4	0	8.1	40.7	20.1	6.6	0.1	0	**61.2**
	-0.6	0	0	0	0	0	3.6	30.9	0	26.3	27.6	10.0	1.4	0.1	0	**84.8**
	-0.4	0	0	0	0	0	0	44.7	0	41.8	12.0	1.4	0.1	0	0	**98.5**
	-0.2	0	0	0	0	0	0	43.1	0	54.9	2.1	0	0	0	0	**100**
	0.0	0	0	0	0	0	0	42.9	0	56.8	0.3	0	0	0	0	**100**
	0.2	0	0	0	0	0	0	40.0	0	59.7	0.3	0	0	0	0	**100**
	0.4	0	0	0	0	0	0	39.5	0	56.1	4.4	0	0	0	0	**100**
	0.6	0	0	0	0	0	0.3	31.1	0	49.7	18.8	0.1	0	0	0	**99.6**
	0.8	0	0	0	0	0.6	2.3	20.1	0	29.7	45.4	1.7	0.3	0	0	**95.2**
	1.0	0	0	0	0.1	2.5	8.3	14.0	0	9.6	52.7	12.4	0.4	0	0	**76.3**
	1.2	0	0	0	0.4	5.9	15.3	9.1	0	1.7	37.6	26.4	3.6	0.1	0	**48.4**
	Total (%)	**0**	**0**	**0**	**0.2**	**3.3**	**6.3**	**23.9**	**0**	**28.4**	**21.8**	**10.9**	**4.6**	**0.6**	**0**	**74.1**

Table A.2: Evaluation of the pix2pix-dist network trained for 900 000 iterations. The table shows the probability (%) of the FPW score of a generated image $\hat{Y}$, of a certain focus level, being lower than the score of the different images X in the corresponding stack.

		X													
	FL (μm)	-1.4	-1.2	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2	**SR** (%)
$\hat{Y}$	-1.4	0	0.1	0	2.8	18.4	12.2	0.1	0	0	10.5	30.3	20.2	5.1	0.3	**10.6**
	-1.2	0	0	0.1	1.0	12.8	15.0	1.2	0	0	20.2	33.1	15.4	1.1	0	**21.4**
	-1.0	0	0	0	0.4	10.3	16.5	1.9	0	0.3	29.8	29.4	10.9	0.4	0	**32.0**
	-0.8	0	0	0	0	3.9	16.1	2.9	0	1.4	41.5	26.4	7.7	0.1	0	**45.8**
	-0.6	0	0	0	0	0.4	13.3	6.7	0	7.8	51.2	16.8	3.7	0	0	**65.7**
	-0.4	0	0	0	0	0	2.8	24.2	0	22.0	42.1	8.1	0.7	0.1	0	**88.3**
	-0.2	0	0	0	0	0	0.4	51.8	0	32.3	14.7	0.7	0.1	0	0	**98.8**
	0.0	0	0	0	0	0	0	45.7	0	53.4	0.9	0	0	0	0	**100**
	0.2	0	0	0	0	0	0.1	29.2	0	68.7	1.9	0.1	0	0	0	**99.8**
	0.4	0	0	0	0	0	0	29.2	0	48.3	22.4	0	0.1	0	0	**99.9**
	0.6	0	0	0	0	0.1	1.0	25.3	0	24.3	48.4	0.7	0.1	0	0	**98.0**
	0.8	0	0	0	0	0.6	4.8	18.0	0	12.7	58.6	5.2	0	0.1	0	**89.3**
	1.0	0	0	0	0	2.8	10.2	11.3	0	5.2	55.8	14.3	0.3	0.1	0	**72.3**
	1.2	0	0	0	0.3	6.7	13.1	7.3	0	1.4	44.0	25.3	1.8	0	0.1	**48.4**
	Total (%)	**0**	**0**	**0**	**0.3**	**4.0**	**7.5**	**18.2**	**0**	**19.8**	**31.6**	**13.6**	**4.5**	**0.5**	**0**	**69.6**

Table A.3: Evaluation of the pix2pix-dist-2inputs network trained for 1 000 000 iterations. The table shows the probability (%) of the MSE score of a generated image $\hat{Y}$, of a certain focus level, being lower than the score of the different images X in the corresponding stack.

		X														
	FL (μm)	-1.4	-1.2	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2	**SR** (%)
$\hat{Y}$	-1.4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**-**
	-1.2	0	0	0	1.2	14.5	18.6	1.2	0	0	17.5	29.3	15.4	2.4	0	**18.7**
	-1.0	0	0	0	0	7.0	15.2	4.5	0	2.4	30.9	24.1	14.9	1.1	0	**37.8**
	-0.8	0	0	0	0	1.9	12.6	9.2	0	8.5	39.9	20.0	7.7	0.2	0	**57.6**
	-0.6	0	0	0	0	0	3.5	26.8	0	24.5	33.9	9.2	2.1	0	0	**85.2**
	-0.4	0	0	0	0	0	0	43.1	0	41.8	13.7	1.3	0.1	0	0	**98.6**
	-0.2	0	0	0	0	0	0	43.2	0	54.9	1.7	0.1	0.1	0	0	**99.8**
	0.0	0	0	0	0	0	0	41.7	0	58.0	0.4	0	0	0	0	**100**
	0.2	0	0	0	0	0	0	42.1	0	57.3	0.6	0	0	0	0	**100**
	0.4	0	0	0	0	0	0	35.9	0	56.8	7.3	0	0	0	0	**100**
	0.6	0	0	0	0	0	0.7	27.4	0	42.8	29.1	0	0	0	0	**99.3**
	0.8	0	0	0	0	1.2	3.9	18.4	0	20.2	52.4	3.4	0.5	0	0	**91.0**
	1.0	0	0	0	0	3.1	10.2	14.5	0	8.0	49.1	14.8	0.3	0	0	**71.6**
	1.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**-**
	Total (%)	**0**	**0**	**0**	**0**	**1.4**	**4.0**	**28.8**	**0**	**36.5**	**20.0**	**6.4**	**2.7**	**0.2**	**0**	**80.0**

Table A.4: Evaluation of the pix2pix-dist-2inputs network trained for 1 000 000 iterations. The table shows the probability (%) of the FPW score of a generated image $\hat{Y}$, of a certain focus level, being lower than the score of the different images X in the corresponding stack.

		X													
	FL (μm)	-1.4	-1.2	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2	**SR** (%)
$\hat{Y}$	-1.4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**-**
	-1.2	0	0	0	0.3	15.7	14.6	0.3	0	0	20.4	30.3	16.8	1.7	0	**20.7**
	-1.0	0	0	0	0.5	9.4	16.8	2.3	0	0.5	29.4	28.4	12.5	0.3	0	**32.2**
	-0.8	0	0	0	0	4.2	16.0	2.6	0	1.5	42.5	24.5	8.6	0	0	**46.6**
	-0.6	0	0	0	0	0.5	12.6	6.8	0	10.0	48.8	18.8	2.6	0	0	**65.5**
	-0.4	0	0	0	0	0	3.8	24.9	0	25.1	38.3	7.4	0.4	0.1	0	**88.3**
	-0.2	0	0	0	0	0	0.9	50.1	0	33.4	14.7	0.8	0.2	0	0	**98.2**
	0.0	0	0	0	0	0	0	45.3	0	53.2	1.4	0	0	0	0	**99.9**
	0.2	0	0	0	0	0	0	37.3	0	58.9	3.7	0	0.1	0	0	**99.9**
	0.4	0	0	0	0	0	0.1	29.0	0	40.4	30.5	0	0	0	0	**99.9**
	0.6	0	0	0	0	0	3.3	20.9	0	21.7	51.9	1.8	0.5	0	0	**94.5**
	0.8	0	0	0	0	0.8	6.1	17.5	0	10.7	56.3	8.5	0	0	0	**84.5**
	1.0	0	0	0	0	4.6	13.6	11.1	0	3.4	46.7	19.2	1.2	0	0	**61.2**
	1.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**-**
	Total (%)	**0**	**0**	**0**	**0**	**1.9**	**5.9**	**24.5**	**0**	**27.1**	**28.6**	**9.1**	**2.7**	**0.1**	**0**	**74.3**

Table A.5: Evaluation of the pix2pix-dist-fast network trained for 600 000 iterations. The table shows the probability (%) of the MSE score of a generated image $\hat{Y}$, of a certain focus level, being lower than the score of the different images X in the corresponding stack.

		X													
	FL (μm)	-1.4	-1.2	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2	**SR** (%)
$\hat{Y}$	-1.4	0	0	0.3	12.0	27.6	3.7	0	0	0	0.7	9.4	25.0	17.3	4.0	**0.7**
	-1.2	0	0	0	5.5	25.3	10.2	0	0	0	2.3	20.8	24.9	9.9	1.1	**2.3**
	-1.0	0	0	0	1.4	18.7	16.4	0	0	0	10.5	26.0	21.6	5.4	0.1	**10.5**
	-0.8	0	0	0	0	7.7	20.1	0.7	0	0.6	25.9	26.3	16.0	2.9	0	**27.2**
	-0.6	0	0	0	0	0.4	14.9	5.2	0	5.8	45.0	19.9	8.4	0.4	0	**56.0**
	-0.4	0	0	0	0	0	1.2	33.6	0	25.0	29.2	9.6	1.4	0	0	**87.8**
	-0.2	0	0	0	0	0	0	47.7	0	43.2	8.9	0.1	0	0	0	**99.9**
	0.0	0	0	0	0	0	0	42.9	0	56.8	0.3	0	0	0	0	**100**
	0.2	0	0	0	0	0	0	30.8	0	64.4	4.8	0	0	0	0	**100**
	0.4	0	0	0	0	0	0	25.3	0	37.1	37.6	0	0	0	0	**100**
	0.6	0	0	0	0	1.5	5.0	15.7	0	9.1	63.0	5.7	0	0	0	**87.8**
	0.8	0	0	0	0.6	6.2	15.0	9.4	0	0.6	41.4	25.7	1.2	0	0	**51.4**
	1.0	0	0	0	1.8	11.6	15.0	5.5	0	0	19.9	39.2	6.9	0.1	0	**25.4**
	1.2	0	0	0.1	3.7	16.4	12.9	1.4	0	0	6.6	37.0	21.5	0.4	0	**8.0**
	Total (%)	**0**	**0**	**0**	**1.8**	**8.2**	**8.2**	**15.6**	**0**	**17.3**	**21.1**	**15.7**	**9.1**	**2.6**	**0.4**	**54.0**

Table A.6: Evaluation of the pix2pix-dist-fast network trained for 600 000 iterations. The table shows the probability (%) of the FPW score of a generated image $\hat{Y}$, of a certain focus level, being lower than the score of the different images X in the corresponding stack.

		X													
	FL (μm)	-1.4	-1.2	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2	**SR** (%)
$\hat{Y}$	-1.4	0	0.1	0	10.0	22.8	3.0	0	0	0	1.4	16.9	29.0	15.0	1.7	**1.4**
	-1.2	0	0	0.1	3.2	19.1	10.0	0.1	0	0	6.7	25.6	26.7	8.0	0.4	**6.8**
	-1.0	0	0	0	1.7	18.8	13.5	0.1	0	0	13.2	29.0	18.8	4.7	0.1	**13.3**
	-0.8	0	0	0	0	9.9	16.6	0.4	0	0.1	28.5	27.4	15.0	2.1	0	**29.0**
	-0.6	0	0	0	0	1.1	19.3	2.5	0	2.6	44.0	22.9	7.4	0.3	0	**49.1**
	-0.4	0	0	0	0	0	6.1	9.5	0	13.9	53.8	14.7	1.9	0.1	0	**77.2**
	-0.2	0	0	0	0	0	1.0	34.1	0	34.5	28.6	1.7	0.1	0	0	**97.2**
	0.0	0	0	0	0	0	0	45.3	0	53.4	1.2	0.1	0	0	0	**99.9**
	0.2	0	0	0	0	0	0.1	29.0	0	53.9	16.9	0	0.1	0	0	**99.8**
	0.4	0	0	0	0	0	0	23.1	0	21.3	55.2	0.3	0.1	0	0	**99.6**
	0.6	0	0	0	0	1.0	8.3	13.5	0	5.8	64.2	7.1	0.1	0	0	**83.5**
	0.8	0	0	0	0	4.7	13.5	7.4	0	1.8	50.3	21.6	0.7	0	0	**59.5**
	1.0	0	0	0	1.0	8.0	14.0	5.0	0	0	34.7	34.2	3.0	0.1	0	**39.7**
	1.2	0	0	0	2.3	13.6	14.6	1.8	0	0	18.2	38.7	10.5	0.4	0	**20.0**
	Total (%)	**0**	**0**	**0**	**1.3**	**7.0**	**8.6**	**12.3**	**0**	**13.4**	**29.8**	**17.2**	**8.1**	**2.2**	**0.1**	**55.5**

Appendix B

Predictions

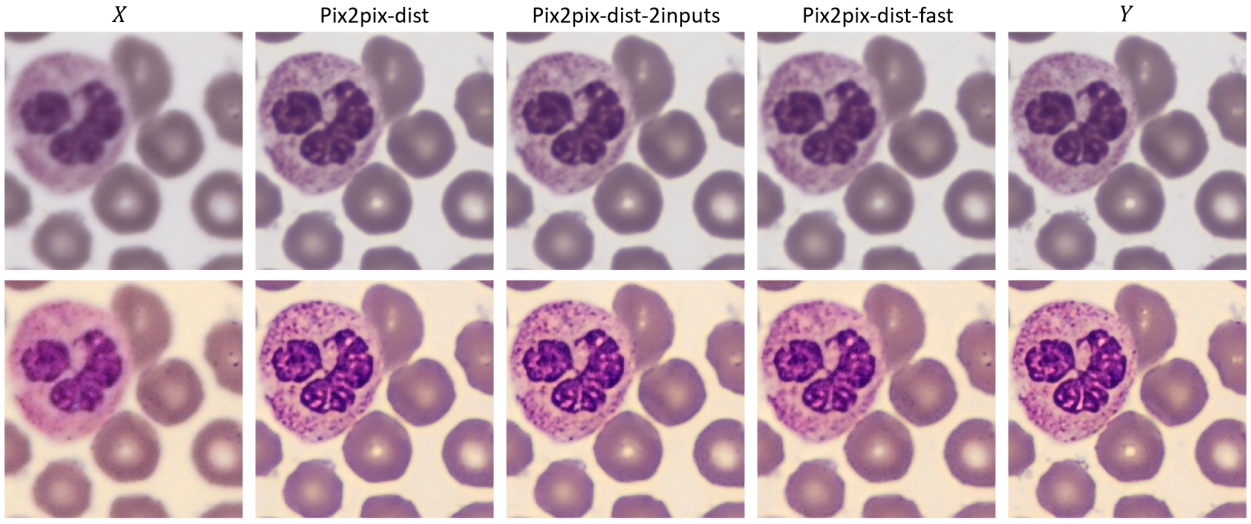


Figure B.1: Results of transformations of a cell image taken at 1.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

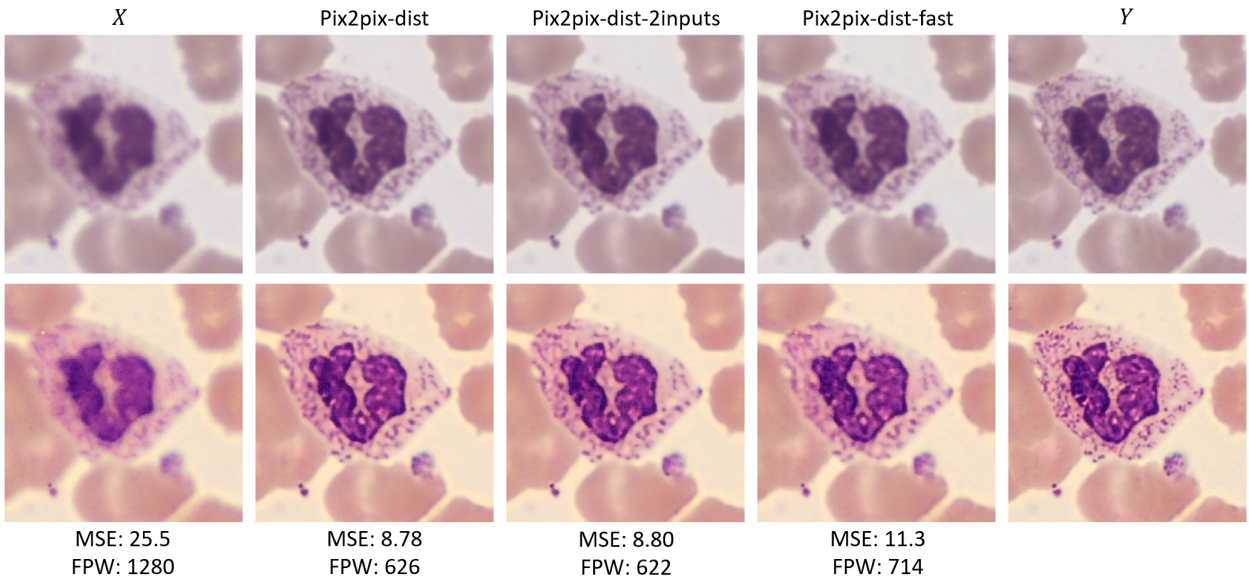


Figure B.2: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

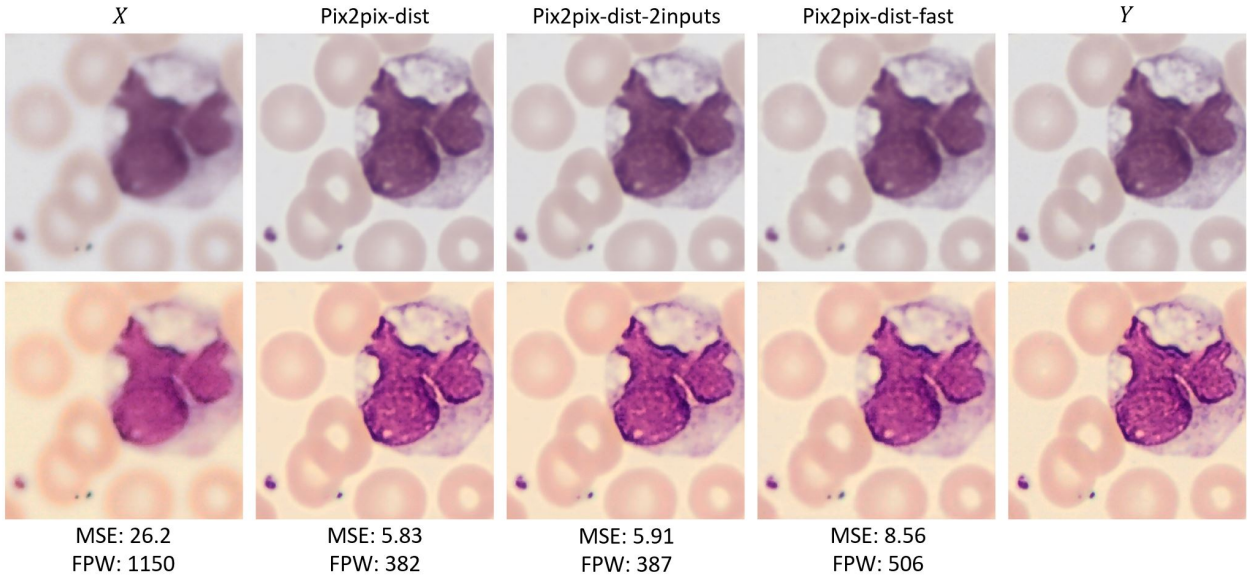


Figure B.3: Results of transformations of a cell image taken at 1.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

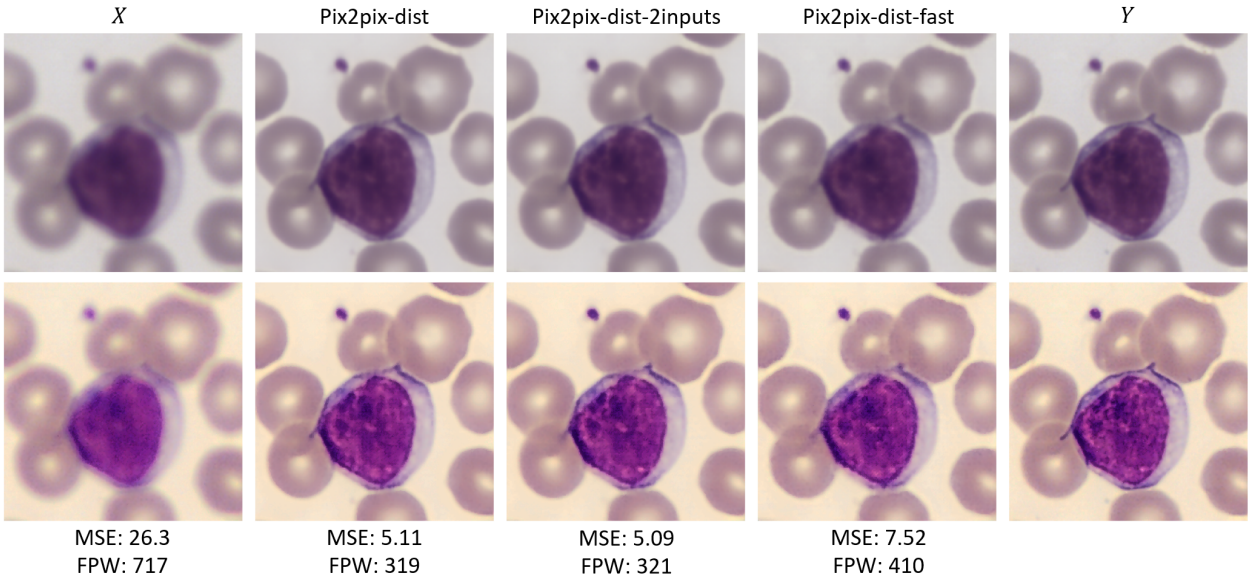


Figure B.4: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

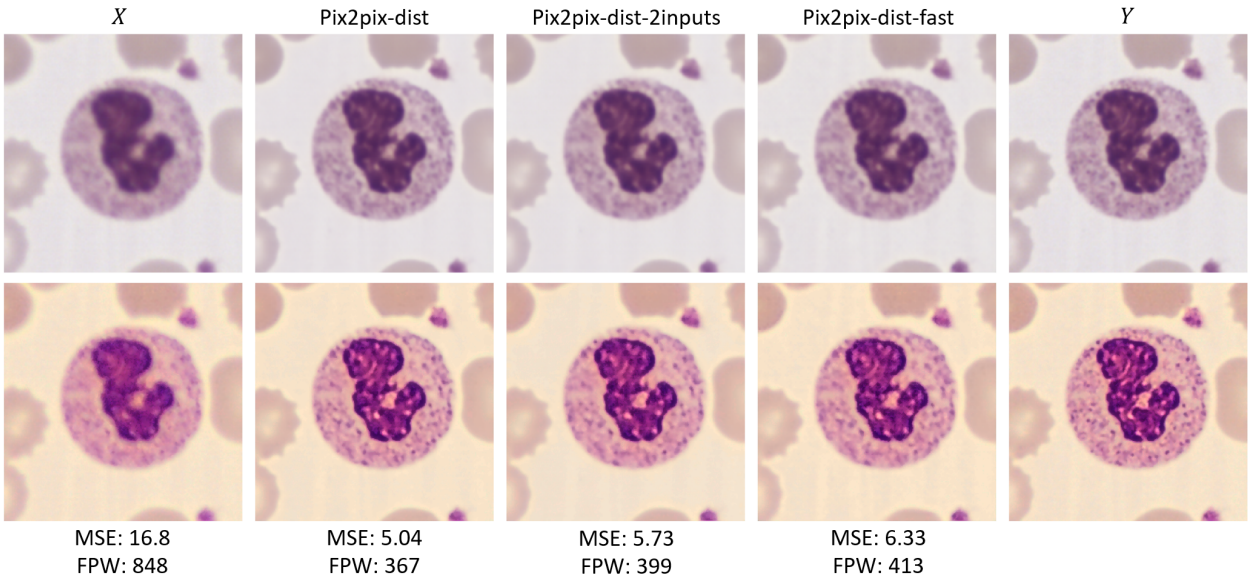


Figure B.5: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

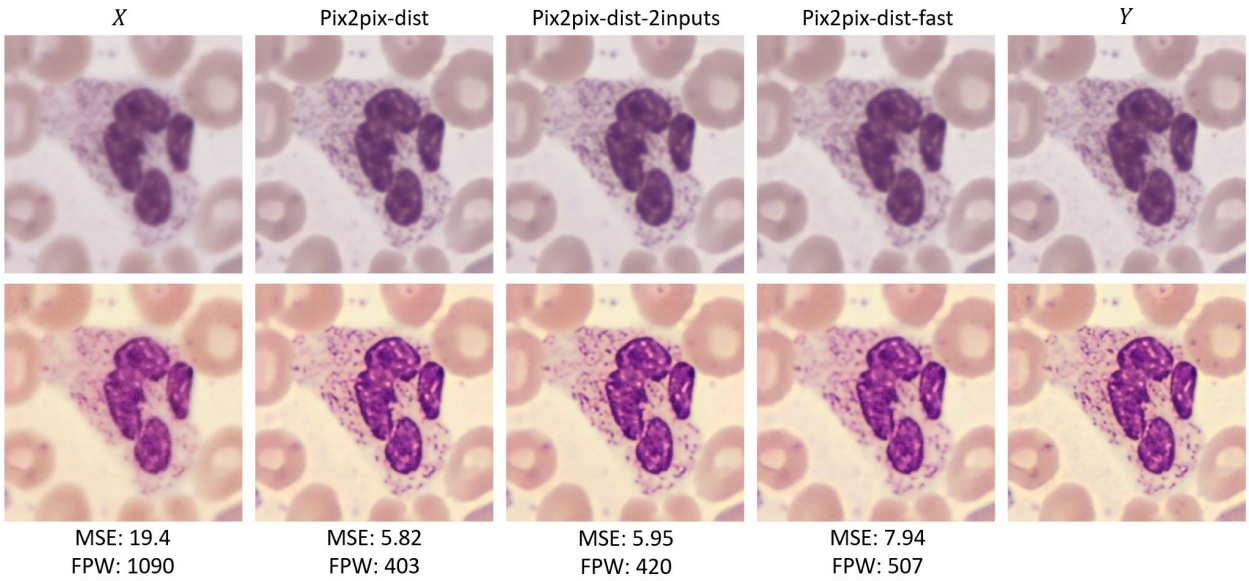


Figure B.6: Results of transformations of a cell image taken at 0.6 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

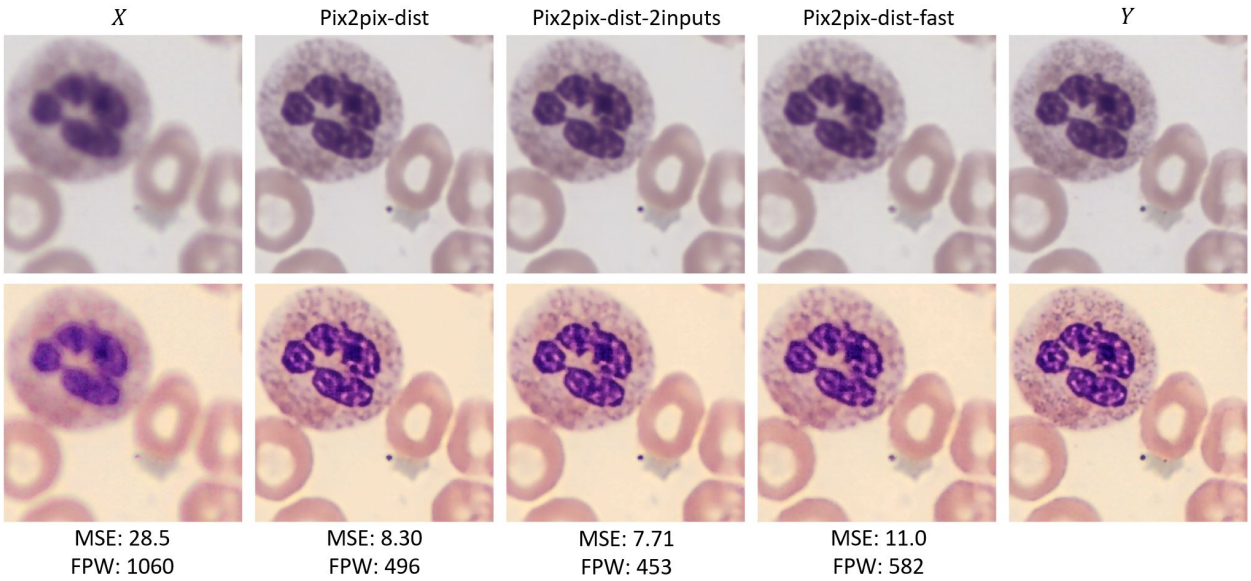


Figure B.7: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

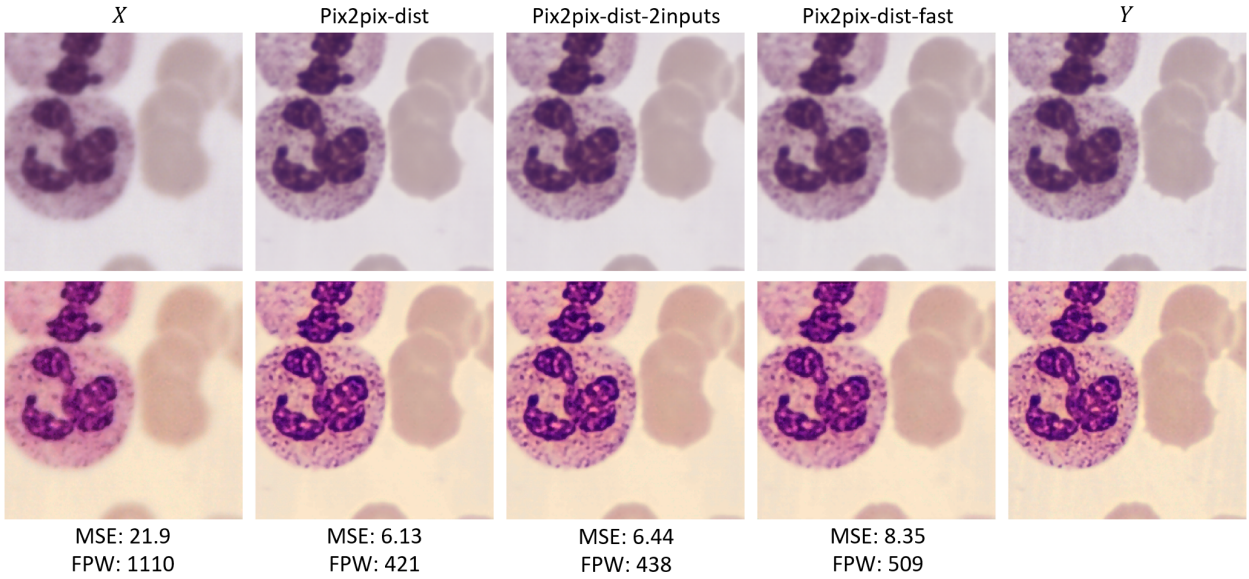


Figure B.8: Results of transformations of a cell image taken at 1.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

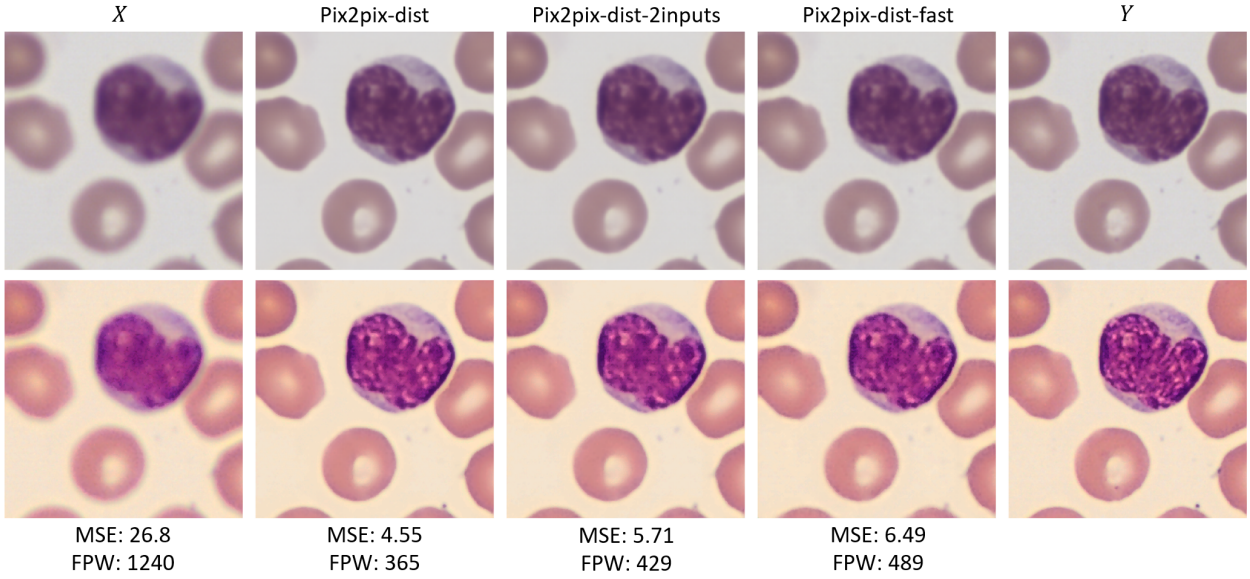


Figure B.9: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

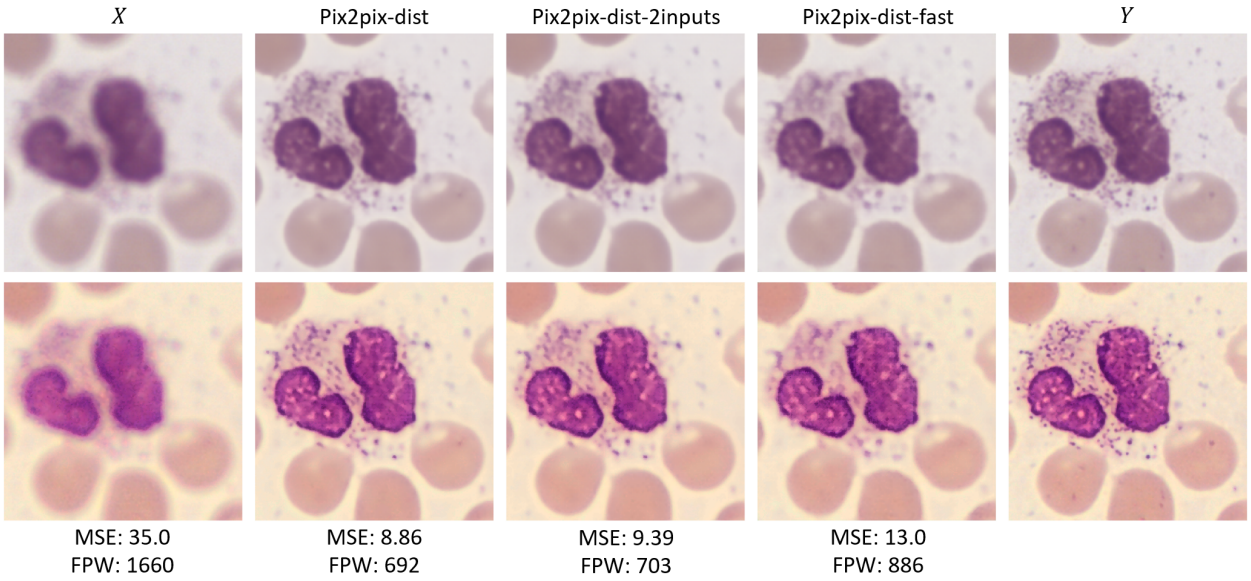


Figure B.10: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

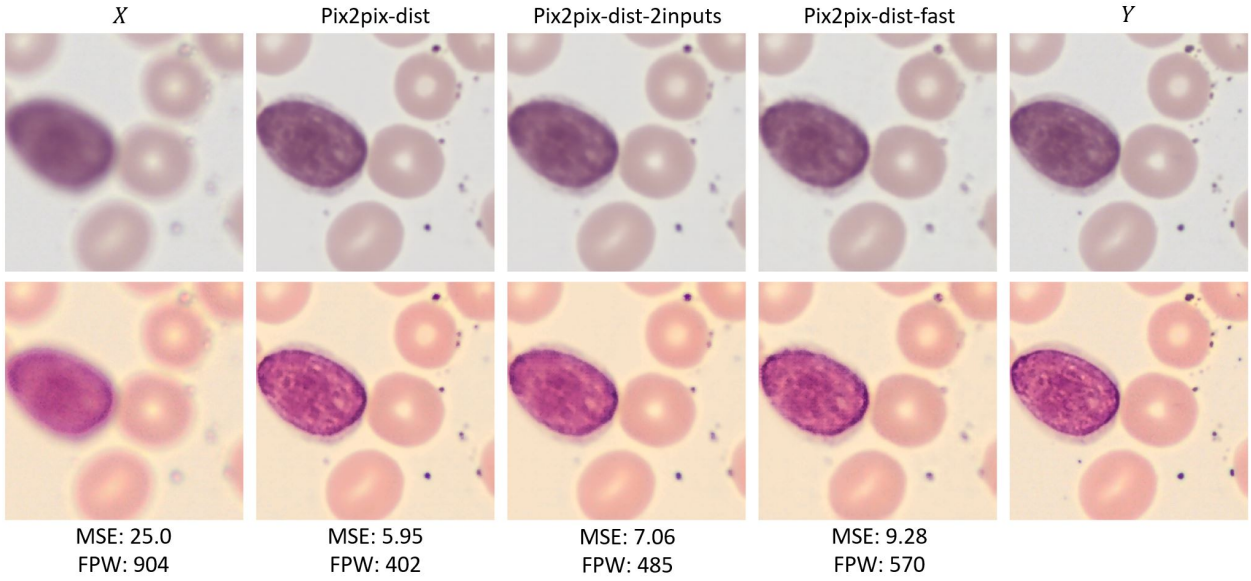


Figure B.11: Results of transformations of a cell image taken at -0.8 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

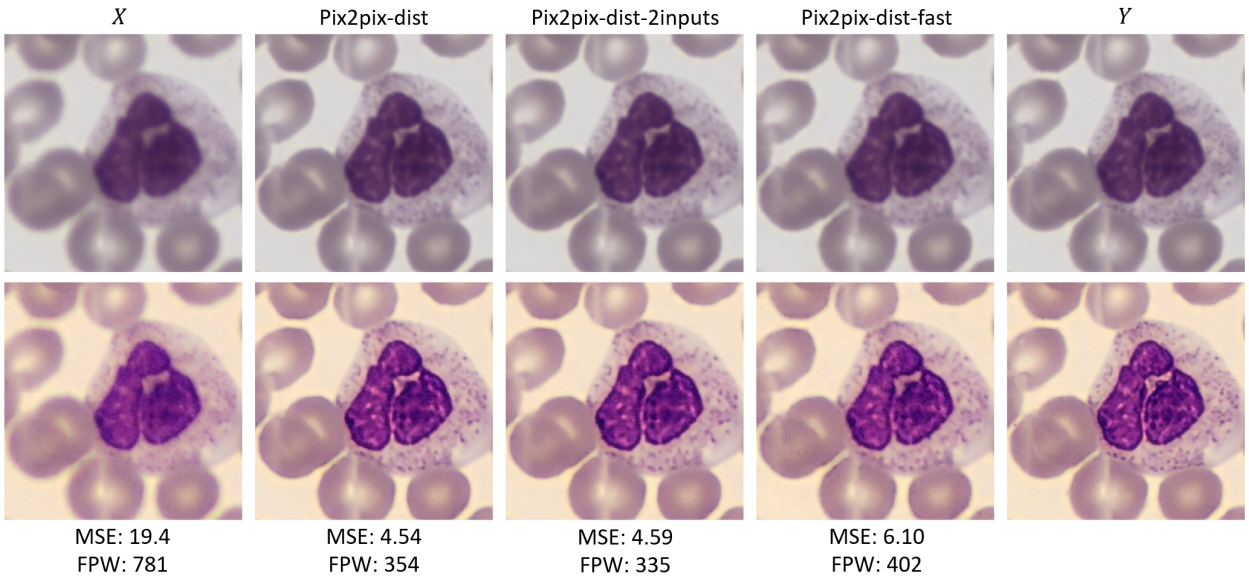


Figure B.12: Results of transformations of a cell image taken at -0.6 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.

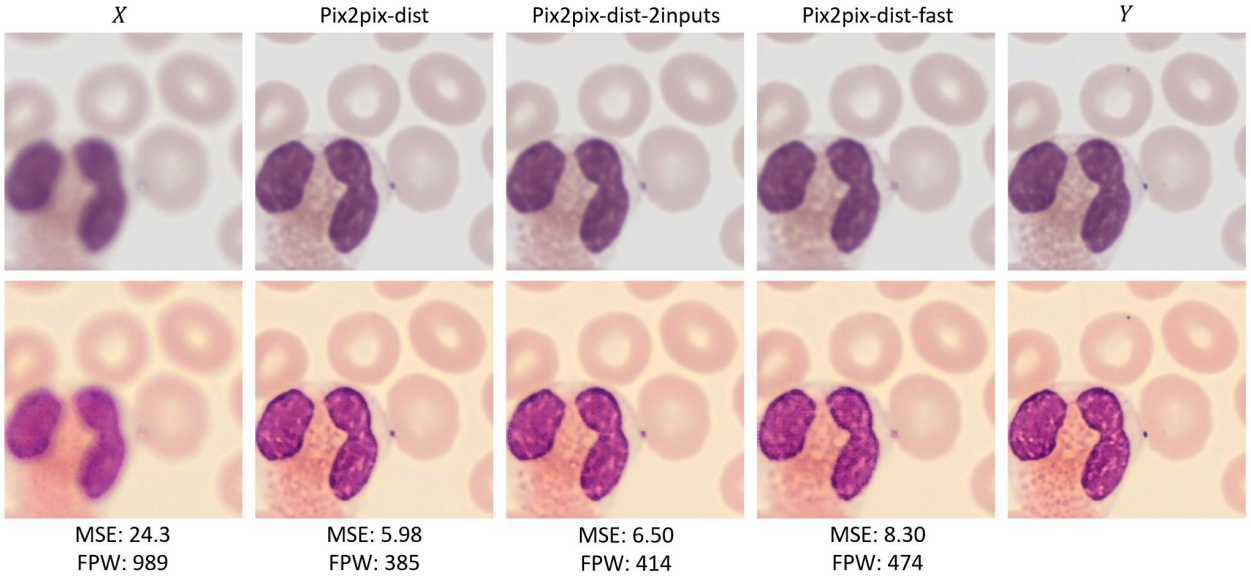


Figure B.13: Results of transformations of a cell image taken at -1.0 μm from focus before and after normalization. From the left: input image X, generated images $\hat{Y}_{dist}$, $\hat{Y}_{dist-2inputs}$, $\hat{Y}_{dist-fast}$ and the ground truth image Y. Top row is before normalization and bottom row is after. The MSE and FPW scores are written under respectively image.