Transfer Learning For Cell Image Reconstruction

Andrei Raureanu
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
a.raureanu@student.utwente.nl

ABSTRACT

Whole slide scanners are a useful tool for doing cell analysis in an efficient manner. A big problem with this is that scanning is not always perfect, which results in artifacts such as blur in some parts of the scan. This causes further problems for the specialists using the scanner, as they have to manually inspect the blurry areas in question and give an objective conclusion. To solve this issue, two Convolutional Autoencoders models are designed and implemented to reconstruct cell slide images. The performance of the models to remove blur from sections of cell slides will then be investigated. The robustness of the Autoencoders is also tested on cell images generated artificially that have had Gaussian blur applied to them. Both trained models successfully deblur cell images with minor performance decreases when the blur is caused by the camera lens focusing below the focal plane. The reconstructions of synthetic cells is also achievable with only a 15% performance decrease when deblurring cell images with high amounts of Gaussian blur applied to them.

Keywords

denoising, Autoencoder, neural network, cell imaging, machine learning, image reconstruction

1. INTRODUCTION

In the biomedical industry, whole slide scanners are one of the many tools to analyse cells for research purposes or disease diagnosis. The underlying process of scanners, cell imaging, consists of staining the cell dish with a fluorescent substance, taking pictures of different areas using a microscope with high magnification (20x - 40x) and then patching all of them together to create one high-resolution scan. The scan image can then be further used for analysis of the cells such as classification or counting.

However, this technique cannot deliver high-quality images all the time. It is estimated that 5% of the scans present artifacts [14]. One of the most common problems encountered in cell imaging is out-of-focus blur [5]. This is caused by the fact that cells are not on the same level of the Z-axis on the chamber slide, meaning that individual cells can be higher or lower from each other on the dish. This results in some scan patches being blurry due to incorrect focus of the lens. This also happens even if the microscope has an automated focus system. The blurry areas in the scan then need to be inspected manually which makes it tedious and time consuming.

Various algorithms and systems have been proposed to deal with this issue [9, 15, 12]. There are deep learning solutions that detect blurry areas and segment them [11] which is relevant as those areas are the most in need of deblur. This allows the microscope to re-scan just the blurry area specifically without doing a full scan. Unfortunately even rescanning a small section requires heavy time-loads.

In recent years, Autoencoders have gained a lot of popularity. This is due to their ability to reconstruct input containing artifacts into its clear counterpart. Because of this, the use of Autoencoders looks to be very suitable for the scope of this project. Another advantage of Autoencoders is not needing labeled data for training which also makes it ideal for unsupervised learning. It is also important to note that sometimes getting big quantities of labeled data for a specific model training can be difficult.

In this research, two convolutional Autoencoder are trained using two different methods to reconstruct blurry cell images into its non-blurry counterpart. The focus will be made on out-of-focus blur anomalies as these are the most common problem in cell imaging.

2. PROBLEM STATEMENT

Research has been done on ways to detect areas with blurry anomalies in cell images for re-scanning purposes [17]. Some examples even use deep learning for this [11]. Despite that, there have been very few attempts at "fixing" the out-of-focus image directly. This paper presents two systems that are trained to reconstruct out-of-focus cell images, thus de-blurring them. The two systems are also able to reconstruct images captured by other scanners other than the one they were trained on.

2.1 Research Questions

In retrospect of the problem statement, the following research questions are presented:

- **RQ1**: To what extent can an Autoencoder reconstruct out-of-focus cell images?
- **RQ2**: How robust is the Autoencoder for out-of-focus reconstruction when presented with cell images recorded with a different microscope?
- **RQ2.1**: To what extent can it reconstruct out-of-focus cell images recorded with a different microscope?

35th Twente Student Conference on IT July 2nd, 2021, Enschede, The Netherlands.

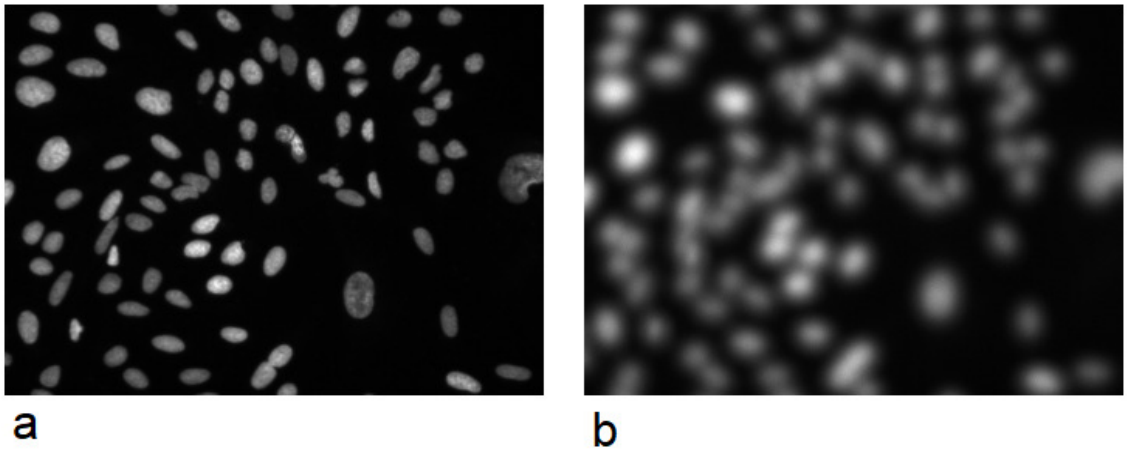

Figure 1. An example of a pair of two cell images resulted from Hoechst staining. a) shows an in-focus cell image while b) shows the same cell slide but this time out-of-focus due to incorrect microscope calibration.

3. RELATED WORK

There has been substantial research done in the detection of out-of-focus images in whole slide scanning. Senaras et al. [11] presented a deep learning solution to detect and segment blurry areas in digital whole slide images under the form of a convolutional neural network. In another paper, Yang et al. [17] argued that relative differences between two or more images in general lead to inconsistencies, which is why they created a deep neural network to predict an absolute measure of quality based on level of blur. What is different from Senaras' [11] model is that their training data was synthetically defocused to 11 absolute defocus levels.

Besides deep learning solutions, there have also been algorithms and systems proposed for detection. One such example of these is the system designed by Zerbe et al. [18]. They make use of the image analysis software ImageJ and Tenenbaum gradient to assess and classify image patch sharpness based on the degree of blur present. The full microscope image is split into image patches. Then, using batch processing through the use of multiple computing nodes, each image patch is classified. Finally, all the patches are aggregated in a sharpness map.

In a similar fashion, Shakhawat et al. [12] propose a method to find the origin of the artifact, meaning whether the distortion happened during the slide preparation stage or scanning, but with the help of a support vector machine to grade the quality of the region containing artifacts from the whole cell image scan.

There are records of Autoencoders being used for image processing. Gupta et al. [6] perform motion blur removal using Coupled Autoencoders. They use two Autoencoders, the first one being the source and the second being the target. The first one makes use of a blurry image, which is formed out of the clear image with a blur kernel applied to it, and the second Autoencoder uses the clear image sample. The coupling then learns the mapping of the blurry image to the clear representation. This is done by having the first Autoencoder learn the latent representation of the blurry image, which is the output of the encoder of the first Autoencoder. Afterwards, it is used as input for the decoder of the second Autoencoder which was trained on reconstructing clear images.

Shiva Shankar et al. [13] provides a technique for removing the blur from an image by sharpening it. For this they use convolutional layers inside the Autoencoder to downsample the training data, extract the relevant features from it and then convert the image to a sharper version of itself.

From what has been gathered so far, there is a solid foundation of artifact recognition and classification along with previous use cases of Autoencoders in image related tasks, but not enough work on using Autoencoders on cell images specifically for deblurring. In this paper two different Autoencoders are proposed for dealing with this problem.

4. METHODS

For this research, two Autoencoder architectures have been designed and implemented. They will be compared between each other using the metrics mentioned to see which approach works best. The first Autoencoder model is trained using the whole image from the training set, meaning no segmentation is used. The second Autoencoder however, is trained on a data set derived from the original Human U2OS cell data set, each image being segmented into 36 patches that correspond with an area of the original image. Each image patch is then used as training data for the second Autoencoder.

Moving forward into the paper, the first Autoencoder shall be referred to as "whole image Autoencoder" and the second Autoencoder design as "split image Autoencoder". The diagrams for both Autoencoders can be found in Appendix C1 and C2 respectively.

4.1 Whole image Autoencoder

The whole image Autoencoder starts with an Input layer of 128x128x1, which corresponds to the image shape of the training set. Afterwards, two pairs of Conv2D and MaxPooling2D layers are added, with a BatchNormalization layer inbetween them. The filter size of the first Conv2D layer is 32 with kernel size of (3, 3) while the second Conv2D layer has a filter size of 64 also with a kernel size of (3, 3). Both Conv2D layers have the same ReLu activation function with padding "same".

The second part of the whole image Autoencoder starts with a Conv2DTranspose layer with filter size of 64, kernel size of (3, 3), strides size of 2, activation ELU and padding "same". Next layer is a BatchNormalization layer, with the second to last layer on the decoder side being another Conv2DTranspose with the same parameters as the

previous one, with the exception of the filter size, which is now 32. Finally, a Conv2D layer with filter size of 1, kernel size of (3, 3) and activation "sigmoid" is used for the final step in the reconstruction of the image.

The reason for choosing to use Conv2DTranspose over a pair of Conv2D and UpSampling2D layers is due to the fact that UpSampling2D just scales up the image by using nearest neighbour or bilinear upsampling. On the other hand, Conv2DTranspose does both a convolution operation just like Conv2D but at the same time it learns what is the best upsampling for the job [10].

4.2 Split image Autoencoder

The second Autoencoder trained with split images starts off similarly with an Input layer of 128x128x1, which corresponds with the size of a single image patch from the whole full training cell image. On the encoder side we have three pairs of Conv2D and MaxPooling2D layers, all Conv2D layers having a filter size of 64 and kernel size of (3, 3) with the first one's activation function being "sigmoid" and the other two ReLu.

On the decoder side the opposite process of encoding is happening, with three pairs of Conv2D and UpSampling2D, the Conv2D layers having the same parameters as the encoder ones only in reverse. The last layer is the same as the whole image Autoencoder, a Conv2D layer with filter size of 1 and kernel size of (3, 3).

The decision to test out two different Autoencoders came from literature review and finding two papers to use as starting points, namely Gupta et al. [6] for the split image Autoencoder and Shiva Shankar et al. [13] for the whole image Autoencoder. Shiva Shankar uses a Convolutional Autoencoder that is trained with the whole image while Gupta only uses patches of image from the original picture. The final architecture for each Autoencoder was realized through a combination of empirical trials and inspiration from the two papers until a visually correct result was achieved for reconstructing cell images.

5. EXPERIMENT

5.1 Data

The data used for the research comes from the BBBC (Broad Bioimage Benchmark Collection) database [1]. This database offers a collection of microscopy image sets. For the training and testing data, the Human U2OS cells data set will be used as it contains pairs of out-of-focus and in-focus cell images. An example of such a pair of cell images can be seen in Figure 1. The data set was created by scanning cells at 34 different z-stacks, ranging from z-stack 0 to z-stack 33. Each z-stack consists of 1536 images, half of the cell slices being created through Hoechst staining and the other half through phalloidin staining, totalling at 52224 images in the whole data set.

The z-stack 16 collection of cell images is considered ground truth for in-focus images while the others are out-of-focus. The blurring is produced naturally through microscope defocus between stacks making it important for the Autoencoder training as artificial blurring might end us as bias for the model. Furthermore, the image pairs are aligned, meaning a cell in the in-focus image is in the same position in the out-of-focus image.

The second data set used in the research is also taken from the Broad Bioimage Benchmark Collection database [3]. It consists of 19200 cell images that have been generated with a simulator and then an artificial blur kernel was applied to it. Half of the images are cell images while the other half are nuclei stains. Thus, we will be working only with the cell image part which consists of 9600 images. This data set will be used to mimic the behaviour of microscopes with different settings or lenses. The trained Autoencoder will then be tested on random images from the data set for cross-data-set performance analysis.

In order to prepare the data, NumPy [2] is used for matrix operations and storing and loading of processed data, OpenCV is used for image operations and loading the images from the data set. For image visualization, matplotlib is used. Each image is resized to 128x128 from their original size of 696x520 using bicubic interpolation and then min-max normalization is applied. Furthermore, the values of each image are normalized to be between the [0, 1] range.

As mentioned previously, the Human U2OS cells data set contains images of cells obtained from both Hoechst staining and phalloidin staining. For the purpose of this research, only the images obtained through Hoechst staining shall be used. As a result, the new size of the usable data set consists of 26112 images and 768 images per z-stack level. From each z-stack level, 80% of the images are going to be used for training, while the remaining 20% will be used for testing the capabilities of the Autoencoder model.

5.2 Metrics

The metrics that will be used to see how the two models perform will be Mean Square Error (MSE), Root Mean Square Error (RMSE), Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [16].

5.2.1 *Mean Squared Error*

The Mean Squared Error is used to measure the quality of the model. It is an indicator of how well the prediction made by the model matches the real data. A MSE score of 0 indicates that the two images are the same. The formula for MSE is as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

where n represents the size of the data set, y_i is the real data and $f(x_i)$ the output of the model.

5.2.2 *Root Mean Squared Error*

The second metric, Root Mean Squared Error, is just the square root of the Mean Squared Error. It can be interpreted as the distance between the real data and the prediction of the Autoencoder.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{n}}$$

5.2.3 *Peak Signal to Noise Ratio*

The third metric, Peak Signal to Noise Ratio, represents the ratio between the maximum possible power of a signal, in this case the maximum pixel value and the power of the noise in the reconstruction. Higher values of PSNR indicate bigger signal power than noise power.Despite it being used in this project for image quality assessment, it should be noted that it had faced criticism before due to the values of the metric having a weak correlation with subjective quality scores [8].

$$\text{PSNR} = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right)$$

where MAX_f represents the maximum signal value from the ground truth image and *MSE* is the Mean Squared Error of the reconstruction.

Table 1. Average metrics for both Autoencoders calculated from the testing set of the Human U2OS data set.

Autoencoder	MSE	RMSE	PSNR	SSIM
Whole image	0.002	0.315	30.277	0.876
Split image	0.003	0.524	26.905	0.802

5.2.4 *Structural Similarity Index Measure*

The last metric, Structural Similarity Index Measure, is used to measure the similarity between two given images. The similarity score is calculated by inspecting three features of the images: luminance, contrast and structure. The final score is a value between [0,1], with 0 meaning the two images are very different from each other while 1 shows that the two images are very similar.

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

In the formula above, μ_x and mu_y represent the averages of x and y respectively, C_1 and C_2 are two variables to stabilize the division with weak denominator, σ_x^2 and σ_y^2 are variance of x and y, thus σ_{xy} is the covariance of x and y.

5.3 Training

To build the two Autoencoders, Tensorflow [4] and Keras are used. Both Autoencoder models were trained using the Adam optimizer [7] with a learning rate of 0.001. Besides the learning rate decay implemented in the Adam optimizer, a custom step decay learning rate scheduler is implemented that halves it by half every 15 epochs. Each model is trained for a maximum of 50 epochs, with a batch size of 16 and a learning rate decay of 1e-3.

The loss function used for both the whole image and split image Autoencoder is the Mean Absolute Error also known as L1. The reason for using this over Mean Squared Error, also known as L2, is that Mean Squared Error is more susceptible to outliers than Mean Absolute Error is.

Another reason is that Mean Squared Error gets stuck in local minimums while Mean Absolute Error can reach better minimums in the same amount of training. This also has an impact on the quality of the result, as Mean Squared Error leaves in artifacts that Mean Absolute Error easily removes [19].

6. RESULTS

The metrics discussed above have been implemented from the skimage library for the project. Below in Table 1 the average MSE, RMSE, PSNR and SSIM is shown resulted from the real cells testing data set. The values in the table have been calculated by taking the average of each metric from 4608 testing images. The values overall are high for each metric indicating that the reconstructions are similar to their ground truth counterparts.

Next, the average for the same metrics is shown in Table 2 for the synthetic cell images data set made up of 9000 images. The same procedure has been applied here as with the testing set of the Human U2OS data set. Again, the high values indicate that reconstruction is successful for synthetic cell images as well.

For each z-stack level, the average MSE, RMSE, PSNR and SSIM were calculated. The tables containing all these values can be checked in Appendix A. Z-stack level 16 is not present as that is the ground truth. What is important to show however, is the SSIM values across all the z-stack levels. Figure 2 shows the average SSIM value for all the

Table 2. Average metrics for both Autoencoders calculated from the synthetic cell images data set.

Autoencoder	MSE	RMSE	PSNR	SSIM
Whole image	0.004	0.295	26.441	0.893
Split image	0.002	0.244	28.319	0.816

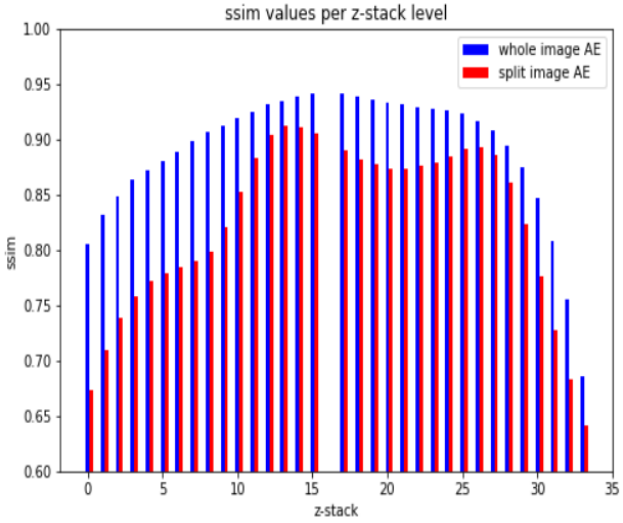

Figure 2. A graph visualization of the average SSIM values per z-stack level for both Autoencoders. Observe how the performance falls off as it starts reconstructing cell images above z-stack 16.

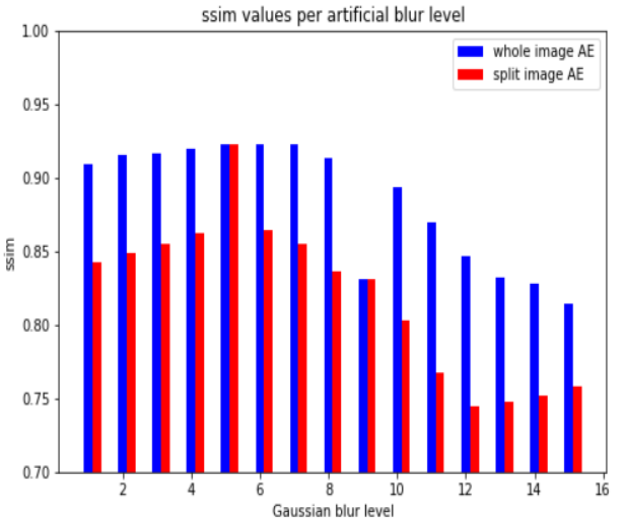

Figure 3. A graph visualization of the average SSIM values per artificial blur level for both Autoencoders.

z-stack values from 0 to 33 with the exception of z-stack level 16. As it can be seen, from z-stack level 25 the SSIM value drops off considerably.

Similarly, the synthetic cell images data set has 16 levels of blur applied to it with increasing blurriness, with level 0 being the ground truth. The average SSIM was calculated for each level of blur and plotted onto a graph for better visualization.

Figure 3 gives a representation of the performance of both

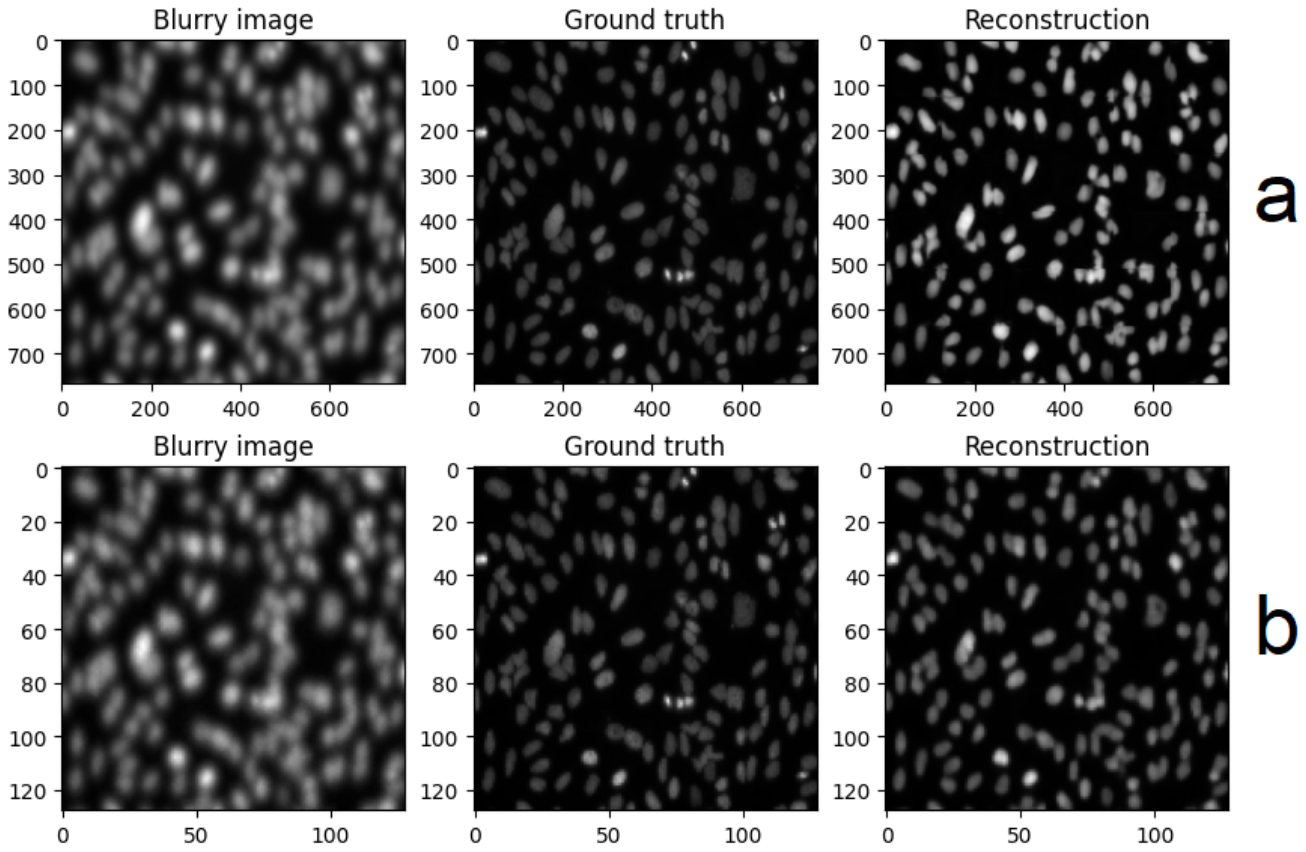

Figure 4. The same cell image being reconstructed and deblurred. a) represents the split image Autoencoder while b) represents the whole image Autoencoder.

the whole image and split image Autoencoders on synthetic cell images. For the full list of values, please refer to Appendix B.

6.1 Discussion

As seen in Figure 4, the deblurring process is done successfully when using both the whole image and split image Autoencoders. This means it is possible to use Autoencoders for cell image reconstruction of out-of-focus images, thus answering the first research question. For the reconstruction of a synthetic cell image, please refer to Appendix D. However, as Figure 2 shows, there is a steep decrease in performance from z-stack level 25 onward. The reason for that is actually in the kind of blur that occurs whether the camera lens focuses above the ground truth or below it. Figure 5 displays how natural out-of-focus blur can change from one extreme to the other while Figure 6 shows a reconstruction of a cell image from the z-stack level 33. Besides this, the split image Autoencoders seems to have learned how to also increase the brightness of cells when deblurring them as an added bonus.

When looking at Table 1 and Table 2 which showcase the average metrics for the real cells testing data set and the synthetic cell images, it can be seen that the two Autoencoder models perform very similarly with the whole image Autoencoder performing just a bit better than split image Autoencoder. This can be attributed to the fact that the split image Autoencoder has a "bordering" effect on some reconstructions due to the stitching process. This introduces unwanted distortions which, to the human eye, are not a problem but it does affect the SSIM score.

One problem that both Autoencoders seem to have is the inability to reconstruct large cells properly. Instead of reconstructing it fully, the Autoencoders thinks that there are actually a bunch of small cells next to each other. The reason for this is due to the small amount of images that contain big cells in the first data set.

The synthetic cell image data set was used to answer **RQ2** and **RQ2.1** respectively. From Table 2 it can be seen from the high PSNR and SSIM values that both Autoencoders are robust to shapes of cells with a blur type they've never seen before. The whole image Autoencoder performs just a bit better than the other model, similarly to the difference in performance calculated from the Human U2OS cells data set. Figure 3 displays the decrease in performance as the intensity of the artificial blur applied increases. While this may seem like a big decline just like in Figure 2 it is only a decrease of 15% in performance in 15 levels of blur. What is interesting to see is that the scores are higher than the real cells data set. A big difference between the two data sets used for the research is that the synthetic images are all uniform even in the type of blur applied to them which is very similar to the blur found in z-stack levels 0 to 15. The shape of an artificial cell is generally only round while the natural cells are both elliptical and round with varying cell sizes.

7. CONCLUSION

In this paper, two Autoencoder models are designed for the purpose of reconstructing and deblurring cell images that have been captured by microscopes with incorrect

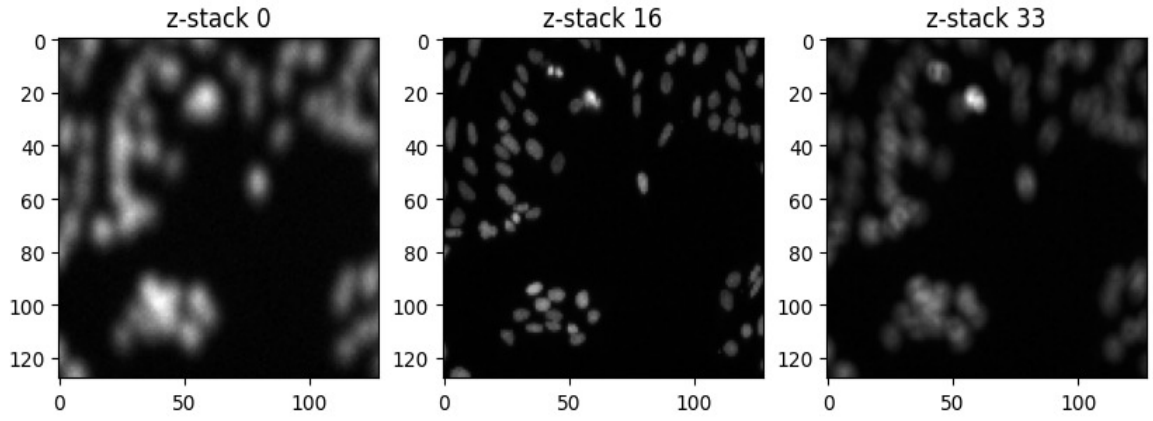

Figure 5. The same cell image at three different z-stack levels: 0, 16 and 33.

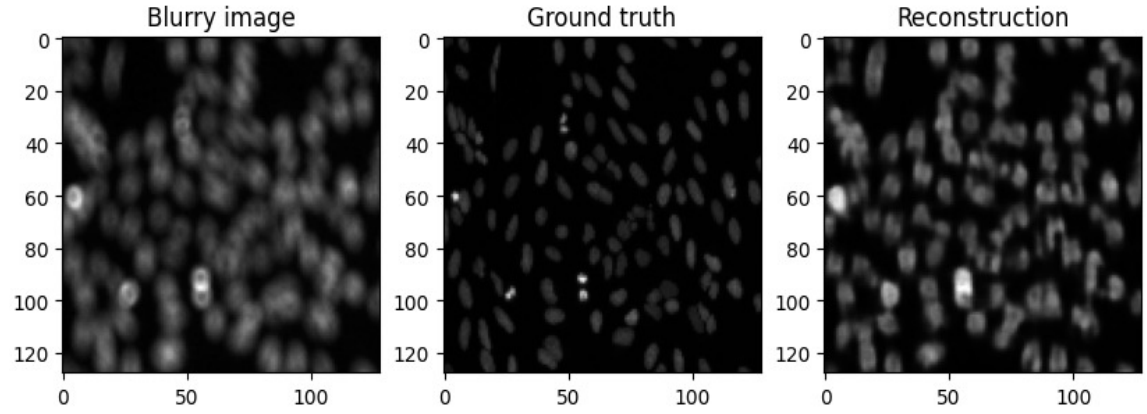

Figure 6. An example of a reconstruction of a cell image from z-stack 33 made by the whole image Autoencoder.

calibration. Both are implemented and tested against each other using two data sets, one formed out of image pairs of real cells both in and out of focus, and the second data set consisting of image pairs of artificially generated cells with Gaussian blur applied to them at various intensity levels. The first Autoencoder is trained with whole images while the second Autoencoder is trained with patches of image from the original data set.

After just 50 epochs of training, both Autoencoders successfully perform their task of deblurring cell images affected by out-of-focus blur, both natural and artificial blur. When both Autoencoders are compared between each other using the 4 metrics, MSE, RMSE, PSNR and SSIM, both score similary with the whole image Autoencoder performing just a bit better than the split image Autoencoder. The same result also applied when the synthetic images data set is used for performance analysis.

For a more in-depth look, each metric was calculated per level of blur for both data sets resulting in Figure 2 and Figure 3. For Figure 2, it shows how the performance of reconstructing cell images that are calibrated above the ground truth focal plane than if it was below the focal plane as z-stack levels 25 to 33 indicate. When the same analysis was performed on the synthetic data set, the performance just went down 15% due to the artificial blur that does not change as drastically as it does with real blur.

7.1 Recommendations

For future work, it would be interesting to see how well the reconstructions fare compared to the ground truth counterparts when subjected to cell counting or classification. Besides this, the two Autoencoder models can be improved further such as increasing the input size of the whole image Autoencoder for more detail or adding batch normalization to the split image Autoencoder. Regularization and dropout might also help with improving the reconstructions of both models.

Besides the 4 metrics used in this research, another good metric used very frequently in image assessment is the Fréchet Inception Distance (FID). The use of this metric is very popular for images generated by General Adversarial Networks (GANs) and it could also be used for this research as well even if Autoencoders differ quite a bit from GANs. In the case of using Fréchet Inception Distance as a performance metric, MS-SSIM can then be used in combination with L1 as the loss function for training the Autoencoders as suggested by Zhao et al. [19].

Finally, since the reconstructions made are scored high by SSIM, these images can be used as input in cell classification and counting systems for further testing of how usable these reconstructions are in real world use cases.

8. ACKNOWLEDGEMENTS

I would like to thank my supervisors, Nicola Strisciuglio and Shunxin Wang, for their guidance throughout the whole research period, for providing useful feedback and helping me with writing this research paper. I would also like to thank Mauk Muller for the support and advice he

has given me throughout these six weeks with familiarizing myself with Tensorflow and Keras.

9. REFERENCES

[1] Human u2os cells (out of focus). `https://bbbc.broadinstitute.org/BBBC006`. [Online].

[2] Numpy. `http://www.numpy.org/`. Software available from numpy.org.

[3] Synthetic cells. `https://bbbc.broadinstitute.org/BBBC005`. [Online].

[4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, pages 265–283, 2016.

[5] M. Bray, A. N. Fraser, T. P. Hasaka, and A. E. Carpenter. Workflow and metrics for image quality control in large-scale high-content screens. *Journal of Biomolecular Screening*, 17(2):266–274, 2012.

[6] K. Gupta, B. Bhowmick, and A. Majumdar. Motion blur removal via coupled autoencoder. In *Proceedings - International Conference on Image Processing, ICIP*, volume 2017-September, pages 480–484, 2018.

[7] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[8] J. Korhonen and J. You. Peak signal-to-noise ratio revisited: Is simple beautiful? In *2012 4th International Workshop on Quality of Multimedia Experience, QoMEX 2012*, pages 37–38, 2012.

[9] X. M. Lopez, E. D'Andrea, P. Barbot, A. . Bridoux, S. Rorive, I. Salmon, O. Debeir, and C. Decaestecker. An automated blur detection method for histological whole slide imaging. *PLoS ONE*, 8(12), 2013.

[10] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[11] C. Senaras, M. Khalid Khan Niazi, G. Lozanski, and M. N. Gurcan. Deepfocus: Detection of out-of-focus regions in whole slide digital images using deep learning. *PLoS ONE*, 13(10), 2018.

[12] H. M. Shakhawat, T. Nakamura, F. Kimura, Y. Yagi, and M. Yamaguchi. Automatic quality evaluation of whole slide images for the practical use of whole slide imaging scanner. *ITE Transactions on Media Technology and Applications*, 8(4):252–268, 2020.

[13] R. Shiva Shankar, G. Mahesh, K. V. S. S. Murthy, and J. Rajanikanth. A novel approach for sharpening blur image using convolutional neural networks. *Journal of Critical Reviews*, 7(7):139–148, 2020.

[14] N. Stathonikos, T. Q. Nguyen, C. P. Spoto, M. A. M. Verdaasdonk, and P. J. van Diest. Being fully digital: perspective of a dutch academic pathology laboratory. *Histopathology*, 75(5):621–635, 2019.

[15] S. Walkowski and J. Szymas. Quality evaluation of virtual slides using methods based on comparing common image areas. *Diagnostic Pathology*, 6(SUPPL. 1), 2011.

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

[17] S. J. Yang, M. Berndl, D. M. Ando, M. Barch, A. Narayanaswamy, E. Christiansen, S. Hoyer, C. Roat, J. Hung, C. T. Rueden, A. Shankar, S. Finkbeiner, and P. Nelson. Assessing microscope image focus quality with deep learning. *BMC Bioinformatics*, 19(1), 2018.

[18] N. Zerbe, P. Hufnagl, and K. Schlüns. Distributed computing in image analysis using open source frameworks and application to image sharpness assessment of histological whole slide images. *Diagnostic Pathology*, 6(SUPPL. 1), 2011.

[19] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.

APPENDIX

A. AVERAGE METRICS PER EACH Z-STACK LEVEL

A.1 Average metrics for z-stacks 0 to 15

Table 3. Metrics for each focal plane resulted from both Autoencoders for z-stacks 0 to 15.

Autoencoder	z-stack	MSE	RMSE	PSNR	SSIM
Whole Image	0	0.003	0.409	26.848	0.805
Split Image	0	0.007	0.819	22.017	0.673
Whole Image	1	0.002	0.379	27.523	0.831
Split Image	1	0.007	0.787	22.432	0.709
Whole Image	2	0.002	0.355	28.094	0.849
Split Image	2	0.006	0.757	22.804	0.739
Whole Image	3	0.002	0.336	28.596	0.863
Split Image	3	0.006	0.730	23.125	0.758
Whole Image	4	0.001	0.321	29.026	0.872
Split Image	4	0.006	0.698	23.479	0.772
Whole Image	5	0.001	0.307	29.431	0.880
Split Image	5	0.005	0.665	23.870	0.779
Whole Image	6	0.001	0.291	29.920	0.889
Split Image	6	0.005	0.627	24.366	0.785
Whole Image	7	0.001	0.275	30.437	0.898
Split Image	7	0.004	0.588	24.891	0.790
Whole Image	8	0.001	0.259	30.968	0.906
Split Image	8	0.003	0.541	25.637	0.799
Whole Image	9	0.001	0.245	31.479	0.912
Split Image	9	0.003	0.477	26.760	0.821
Whole Image	10	0.001	0.231	32.030	0.919
Split Image	10	0.002	0.403	28.285	0.853
Whole Image	11	0.001	0.214	32.664	0.925
Split Image	11	0.001	0.331	29.941	0.883
Whole Image	12	<0.001	0.200	33.240	0.931
Split Image	12	0.001	0.275	31.385	0.904
Whole Image	13	<0.001	0.187	33.819	0.935
Split Image	13	0.001	0.236	32.460	0.912
Whole Image	14	<0.001	0.175	34.391	0.938
Split Image	14	0.001	0.215	33.068	0.911
Whole Image	15	<0.001	0.165	34.934	0.941
Split Image	15	<0.001	0.208	33.262	0.905

A.2 Average metrics for z-stacks 17 to 33

Table 4. Metrics for each focal plane resulted from both Autoencoders for z-stacks 17 to 33.

Autoencoder	z-stack	MSE	RMSE	PSNR	SSIM
Whole Image	17	<0.001	0.165	35.005	0.942
Split Image	17	0.001	0.231	32.301	0.890
Whole Image	18	<0.001	0.175	34.520	0.939
Split Image	18	0.001	0.246	31.747	0.881
Whole Image	19	<0.001	0.184	34.033	0.936
Split Image	19	0.001	0.256	31.394	0.877
Whole Image	20	<0.001	0.192	33.669	0.933
Split Image	20	0.001	0.261	31.242	0.874
Whole Image	21	0.001	0.196	33.494	0.932
Split Image	21	0.001	0.261	31.270	0.874
Whole Image	22	0.001	0.200	33.342	0.929
Split Image	22	0.001	0.257	31.431	0.876
Whole Image	23	0.001	0.202	33.246	0.928
Split Image	23	0.001	0.251	31.696	0.879
Whole Image	24	0.001	0.206	33.102	0.926
Split Image	24	0.001	0.245	31.984	0.885
Whole Image	25	0.001	0.216	32.715	0.923
Split Image	25	0.001	0.241	32.213	0.891
Whole Image	26	0.001	0.235	31.930	0.917
Split Image	26	0.001	0.247	32.097	0.893
Whole Image	27	0.001	0.264	30.904	0.908
Split Image	27	0.001	0.278	31.181	0.886
Whole Image	28	0.002	0.298	29.793	0.894
Split Image	28	0.001	0.333	29.684	0.861
Whole Image	29	0.002	0.338	28.616	0.875
Split Image	29	0.002	0.407	27.848	0.823
Whole Image	30	0.002	0.386	27.411	0.847
Split Image	30	0.003	0.494	26.096	0.776
Whole Image	31	0.002	0.444	26.192	0.808
Split Image	31	0.004	0.587	24.563	0.727
Whole Image	32	0.003	0.510	24.991	0.755
Split Image	32	0.005	0.681	23.277	0.683
Whole Image	33	0.004	0.583	23.861	0.686
Split Image	33	0.007	0.774	22.189	0.642

B. AVERAGE METRICS PER EACH ARTIFICIAL BLUR LEVEL

Table 5. Metrics for each artificial blur level resulted from both Autoencoders.
Gauss. B. L. = Gaussian Blur Level

Autoencoder	Gauss. B. L.	MSE	RMSE	PSNR	SSIM
Whole Image	1	0.002	0.236	27.898	0.909
Split Image	1	0.001	0.192	29.877	0.842
Whole Image	2	0.002	0.219	28.560	0.915
Split Image	2	0.001	0.169	30.938	0.849
Whole Image	3	0.002	0.217	28.613	0.916
Split Image	3	0.001	0.164	31.224	0.855
Whole Image	4	0.002	0.207	29.051	0.920
Split Image	4	0.001	0.152	31.857	0.862
Whole Image	5	0.001	0.195	29.565	0.923
Split Image	5	0.001	0.142	29.565	0.923
Whole Image	6	0.002	0.208	28.959	0.923
Split Image	6	0.001	0.159	41.469	0.864
Whole Image	7	0.002	0.224	28.324	0.923
Split Image	7	0.001	0.178	30.528	0.855
Whole Image	8	0.003	0.267	26.824	0.913
Split Image	8	0.002	0.220	28.684	0.836
Whole Image	9	0.003	0.279	26.441	0.831
Split Image	9	0.002	0.222	28.578	0.831
Whole Image	10	0.004	0.325	25.123	0.894
Split Image	10	0.003	0.267	26.977	0.803
Whole Image	11	0.006	0.363	24.149	0.870
Split Image	11	0.003	0.301	25.948	0.768
Whole Image	12	0.006	0.399	23.335	0.847
Split Image	12	0.004	0.330	25.151	0.745
Whole Image	13	0.008	0.424	22.823	0.832
Split Image	13	0.004	0.340	24.902	0.748
Whole Image	14	0.008	0.434	22.620	0.828
Split Image	14	0.005	0.340	24.889	0.752
Whole Image	15	0.009	0.453	22.251	0.814
Split Image	15	0.005	0.333	25.079	0.758

C. AUTOENCODER ARCHITECTURES

C.1 Whole image Autoencoder

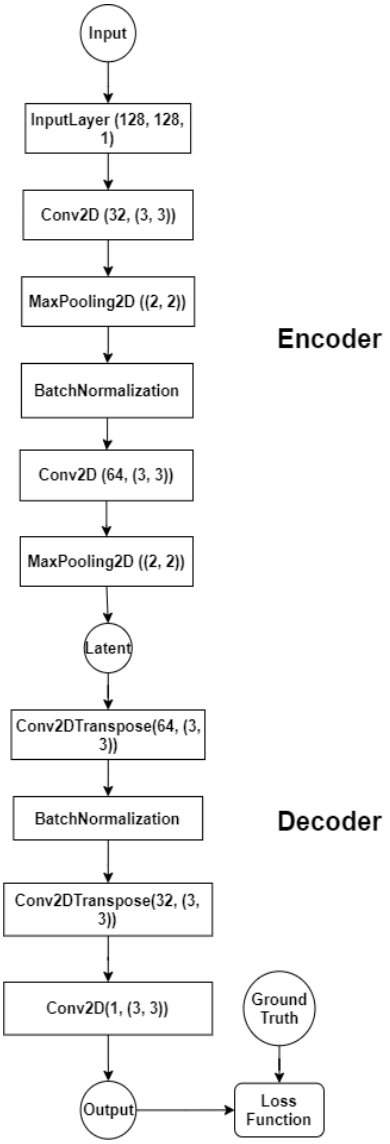

Figure 7. Whole image Autoencoder architecture.

C.2 Split image Autoencoder

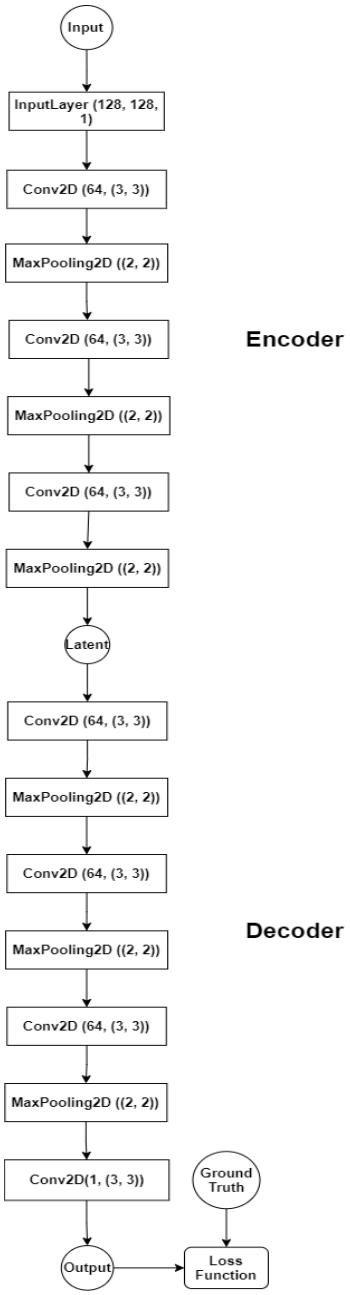

Figure 8. Split image Autoencoder architecture.

D. SYNTHETIC CELL RECONSTRUCTION EXAMPLE

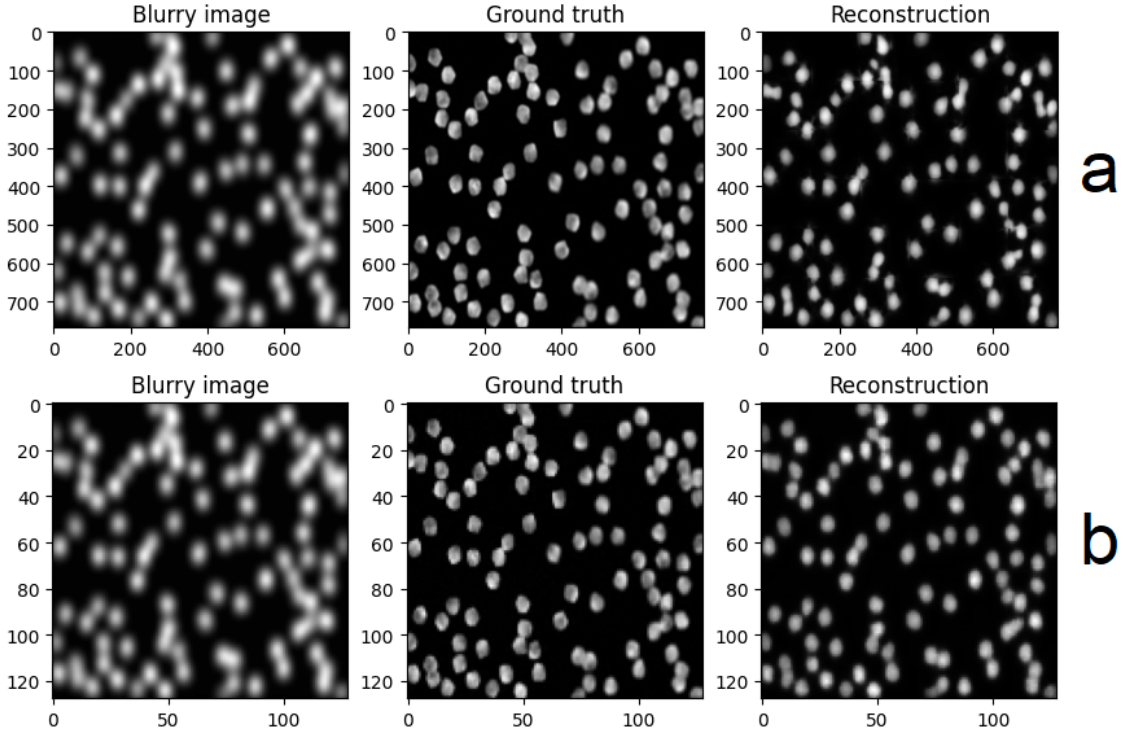

Figure 9. The same synthetic cell image being reconstructed and deblurred. a) represents the split image Autoencoder while b) represents the whole image Autoencoder.