UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Anhelina Lohvina

Table2Cell: generating realistic nuclei images from numeric properties for data compression

Master's Thesis (30 ECTS)

Supervisor: Dmytro Fishman, PhD

Tartu 2023

Table2Cell: generating realistic nuclei images from numeric properties for data compression

Abstract: Microscopy image analysis is the process of extracting quantitative information from images obtained from microscopes. It involves techniques and methods from computer vision, image processing and machine learning to identify and extract numerical features from images of biological samples such as cells. Modern software is capable of extracting a great number of these properties from images of cells. Provided that there are hundreds of cells per one image and thousands of images per experiment, the amount of data extracted becomes a significant computational burden. In this work, we address the problem of feature selection using image generation with neural networks. Here we show that by generating cell images from different sets of numeric characteristics and assessing the resulting image quality we can decide which input parameters are essential and which can be discarded, helping us to perform feature selection. We propose a novel Table2Cell model that can generate high-quality nuclei images from vectors of features. Our results demonstrate that the generated images have a high degree of similarity to real images, and that the Table2Cell model is responsive to variations in its input parameters. This study not only addresses the issue of feature selection, but also has broader implications for the field of image generation. We believe that the results of our research provide valuable insights for further research and development of this technology.

Keywords: deep learning, computer vision, neural networks, feature selection, conditional generative adversarial networks, fluorescent microscopy, image generation

CERCS: P176 - Artificial intelligence; T111 - Imaging, image processing; B110 - Bioinformatics, medical informatics, biomathematics, biometrics

Table2Cell: numbrilistest omadustest realistlike rakutuuma piltide genereerimine andmemahtude kahandamise jaoks

Lühikokkuvõte: Mikroskoopiapiltide analüüsi käigus eraldatakse mikroskoobiga kujutatud piltidelt kvantitatiivset informatsiooni. Selle käigus kasutatakse pilditöötluse, masinnägemise ja masinõppe meetodeid et leida ja eraldada bioloogiliste proovide (näiteks rakkude) piltidelt numbrilisi mõõdikuid. Kaasaegne tarkvara suudab rakupiltidelt suurel hulgal erinevaid omadusi välja leida. Sel viisil eraldatud andmemaht muutub kiiresti arvestatavaks arvutuslikuks koormuseks, sest ühel pildil on sadu rakke ja ühe eksperimendiga on seotud tuhandeid pilte. Magistritöös pakutakse genereerivate närvivõrkude abil välja lahendus oluliste omaduste valimise probleemile. Numbrilistest mõõdikutest rakupilte genereerides, ja loodud piltide kvaliteeti hinnates, saab otsustada, millised mõõdikud on olulised ja millised saab kõrvale jätta. Töös kirjeldatakse uudset mudelit nimega Table2Cell, mis suudab rakuomaduste vektoritest genereerida kõrge kvaliteediga pilte rakutuumadest. Table2Cell on tundlik muutustele sisendvektoris ja mudeliga genereeritud pildid on väga sarnased päris rakutuuma piltidele. Peale rakuomaduste valiku ülesande lahendamisele on teadustööl ka laiem mõju piltide genereerimise valdkonnale. Töö autorid usuvad, et saadud tulemused annavad väärtuslikku teavet selle tehnoloogia edasiseks uurimiseks ja arendamiseks.

Võtmesõnad: sügav õppimine, tehisnägemine, tehisnärvivõrgud, omaduste valik, generatiivne võistlev võrgustik, fluorestsentsmikroskoopia, pildi genereerimine

CERCS: P176 - Tehisintellekt; T111 - Pilditehnika; B110 - Bioinformaatika, meditsiiniinformaatika, biomatemaatika, biomeetrika

Contents

1 Glossary

A short vocabulary of biomedical words used in this thesis.

- **Cell** — the basic structural and functional unit of all living organisms.
- **Cell line** — a population of cells derived from a single cell and grown in a laboratory setting for biological and drug research.
- **Fluorescence microscopy** — a form of microscopy, in which certain cell structures are marked with molecules that can emit light, which is captured by the microscope [35].
- **Homeostasis** — is any self-regulating process by which an organism tends to maintain stability while adjusting to conditions that are best for its survival. If homeostasis is successful, life continues; if it's unsuccessful, it results in a disaster or death of the organism [8].
- **Nucleus (*pl. nuclei*)** — a cell organelle present in most cells of organisms that contains the cell's genome and regulates its activities.

2 Introduction

Image generation models have a great potential in generating synthetic images that can be used to augment or enhance existing microscopy datasets, increase their size, and improve the quality of the images [21]. In particular, these models can be used to generate synthetic images that correspond to specific conditions or features such as a certain cell type, shape, size, and other cell-related features [23].

Image generation plays an important role in many areas of biology, and the development of deep learning methods has greatly advanced this field. To our knowledge, the following methods have been developed to transform tabular data into images using neural networks. Sharma et al. developed DeepInsight [38] which projects feature vectors onto a 2-D space using t-SNE20 [42]. Ma and Zhang developed OmicsMapNet [27] to convert gene expression data of cancer patients into 2-D images for the prediction of tumor grade using CNNs. OmicsMapNet uses extracted functional annotations of genes to construct images via TreeMap [40], so that genes with similar molecular functions are closely located in the image. Bazgir et al. developed REFINED (REpresentation of Features as Images with NEighborhood Dependencies) [6], which uses the Bayesian multidimensional scaling as a global distortion minimizer to project the features onto a 2-D space.

Image generation will serve as a basis for our feature selection solution.

2.1 Problem

Feature extraction from microscopy images is a critical step in various applications of microscopy image analysis, including cell biology, cancer research, drug discovery, and tissue engineering [31]. By providing quantitative and objective measurements of cellular and molecular features it is possible to compare cells, classify cell types, and track cell behavior over time. This information can be valuable for a wide range of applications, including disease diagnosis, drug development and understanding the effects of various treatments [11].

Modern computer vision techniques are capable of extracting vast number of features from cell images [14]. These techniques involve preprocessing of images to remove noise and enhance features, followed by segmentation to separate cells from the background [7]. Once individual cells have been identified, features such as shape, size, texture, brightness, etc. can be extracted using various algorithms.

Selecting the most useful features that were extracted from cells is an essential step for conducting quantitative analysis of microscopy images. First, feature selection can reduce the dimensionality of the data, making it more manageable for analysis and modeling. Second, including irrelevant or redundant features can lead to overfitting and decrease the generalization performance of the any machine learning model, which may be further used. Finally, identifying the most important features can provide insights into

the underlying biology of the system being studied and potentially lead to the discovery of novel biomarkers or targets for drug development.

2.2 Motivation

Our motivation to do this research comes from the perspective of the microscopy image analysis industry. Most of the modern feature selection techniques are biased, since they are performed without considering the underlying biological mechanisms, and hence, important features may be overlooked or irrelevant features may be included. In out research, we investigate the possibility of combining feature extraction technique with image generation for a better feature selection.

2.3 Contribution

In this work, we explore a method of using vector of features for image generation for a further feature selection. We present a solution that is able to generate high-quality nuclei images with a high degree of similarity to real images from different sets of numeric characteristics. We also investigate the responsiveness of the model to variations in input parameters (features) and the ability to generate high-quality nuclei using a smaller number of features, performing data compression. Overall, this work can serve as a baseline for further research and development that will bring this technology into the various pipelines.

2.4 Outline

Background gives a general overview of microscopy and describes neural network methods for image generation.

Data and methods characterizes the datasets used for networks training and performing experiments, explains proposed architectures and algorithms and describes quality metrics.

Experiments and results describes the conducted experiments and reports the results on the dataset.

Conclusion shortly summarizes the contribution and key results.

References contains a list of used literature sources.

Appendix includes supplementary tables and a public license.

3 Background

This chapter will provide a comprehensive background on microscopy, emphasizing the significance of studying nuclei and discussing the principle of fluorescence microscopy. We will also describe generative adversarial networks and autoencoders including their architectures, training process, and applications in various fields.

3.1 Microscopy background

Microscopy is a technique that is used to study functions and structure of biological entities at a microscopic level. Advanced microscopy techniques [4], [13], [34] have revolutionized the field of biology, enabling researchers to explore the intricate details of cells as basic units of life. Through studying cells, biologists gain insights into the ways in which organisms develop, grow, and reproduce.

One of the key components of a cell is the nucleus, which contains the genetic material of the cell in the form of DNA and is located within cytoplasm, a jelly-like substance. There are other structures and organelles within the cell, such as the mitochondria, ribosomes, endoplasmic reticulum, etc. that maintain the cell's overall function. However the nucleus is the most essential organelle and therefore, its study is crucial in microbiology, as it helps us to understand the fundamental mechanisms of gene expression, DNA replication, cell division, and other vital processes in cells [46] A deeper understanding of the nucleus can also help us to better comprehend the mechanisms of genetic disorders, such as cancer, and develop effective treatments. Additionally, the nucleus plays a critical role in regulating cellular responses to external stimuli [32] and maintaining cellular homeostasis [41], which has significant implications in areas such as immunology [28] and neuroscience [16]. Thus, studying the nucleus is crucial for advancing our understanding of the basic principles of life and the development of innovative therapeutic approaches [29].

Cells can be studied using various microscopy techniques that allow scientists to visualize and analyze their structures and functions, study the location, movement, and interactions of specific structures within cells. Our research is focused on nuclei images, captured by a fluorescence microscopy, so we will elaborate on it more.

Fluorescence microscopy [36] involves the use of fluorescent dyes or proteins are specifically targeted to certain structures of ineterest in cells. When exposed to specific wavelengths of light, these dyes or proteins emit light of a different wavelength, which can be detected and visualized using a fluorescent microscope. The image produced by fluorescent microscopy can be further enhanced and analyzed using image processing software. This can include adjusting the brightness and contrast of the image, deconvolution to remove blur caused by out-of-focus light, and three-dimensional reconstruction of the sample.

3.2 Approaches to image generation

This section provides an overview of some of the most popular approaches to image generation, including generative models based on autoencoders and generative adversarial networks. We will also discuss the most common challenges related to autoencoders and our task.

3.2.1 Autoencoders

Prior to initiating the development of a high-quality solution that may consume significant resources, we opted to experiment with a simple model. Our objective was to demonstrate the feasibility of generating an image from a vector of features, even if the resultant image lacked quality. Hence, we identified autoencoders as an appropriate baseline model.

Autoencoders [5] are artificial neural networks in which output is expected to be an exact copy of the input. Autoencoders are capable of learning a compressed representation of the input data without any supervision. These representations are called latent representations. These representations are produced by the network layer, called bottleneck, between two autoencoder parts: encoder and decoder.

An encoder aims to extract/capture meaningful features from the input data to latent representation and thus compresses it from a high-dimensional space to a low-dimensional space. The latent representation of the input data is simply a compressed feature vector, which contains all the important information needed to represent the original data. A decoder tries to restore the input from the compressed feature vector.

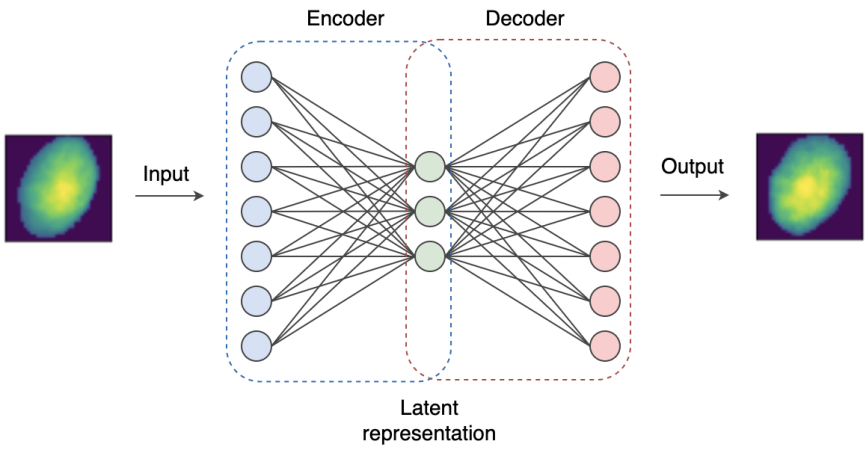


Figure 1. The model of a basic autoencoder architecture. The encoder extracts meaningful features from the input, forming its latent representation (a bottleneck). These compressed features are used by the decoder to restores the input.

Both an encoder and a decoder can have more than one layer. In this case the full network is called a deep autoencoder [15]. Adding more layers increases the depth of each component and thus expands the capacity of an autoencoder allowing it to learn the data features interaction on a non-linear level. If an encoder is powerful enough it can find more complex features representing the input data. Similarly, a sufficiently deep decoder can learn how to interpret these compressed data representations better.

Since the aforementioned compressed feature vectors have a much lower dimensionality than the input data, autoencoders can be used for dimensionality reduction.

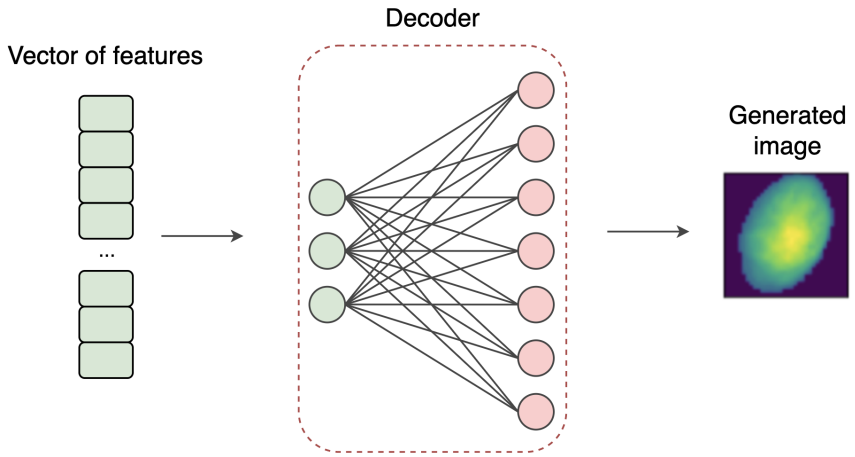


Figure 2. A simple decoder architecture that is used for nuclei generation from a vector of features: the decoder model receives a vector of features as an input and outputs a generated image.

The generative capacities of an autoencoder are very limited [45] due to usage of l_2 loss (an MSE function), which is based on the assumption that the data follows a Gaussian distribution [24], [25]:

$$l_2 = \frac{1}{k}\sum_{i=1}^{k}(\hat{x}_i - x_i)^2 \tag{1}$$

L_2 loss penalizes high-frequency details such as edges and textures which are often lost during the reconstruction process. This is because l_2 loss only measures the overall difference between the input and output, rather than preserving fine details. Also l_2 loss assigns equal weight to all pixels in the image, including those that contain noise or outliers. As a result, the autoencoder may learn to generate images that are an average of all possible inputs, which can lead to blurry outputs.

There is a lot of evidence [10], [26], [12] of blurry image generation using variational autoencoders (VAE) [24], which gained a big popularity over the past years. The main

advantage of a VAE its capability of learning smooth latent representations of the input data: instead of outputting a single value to describe each feature from a bottleneck like in regular autoencoder, VAE encoder is capable of capturing the probability distribution for this feature. Thus by changing the feature vector distributions one can make the decoder generate new data that looks very similar to the training data. Figure 3 shows an example of blurry blood cells image reconstructions generated by VAE [26].

To overcome the limitation of l_2 loss some improvements were made [10] to VAE to generate high quality images such as improving the decoder capacity by increasing the network depth and employing residual blocks and skip connection. However as it can be seen from Figure 4, the effect of low detalisation from l_2 loss still remains in different modifications of variational autoencoders.

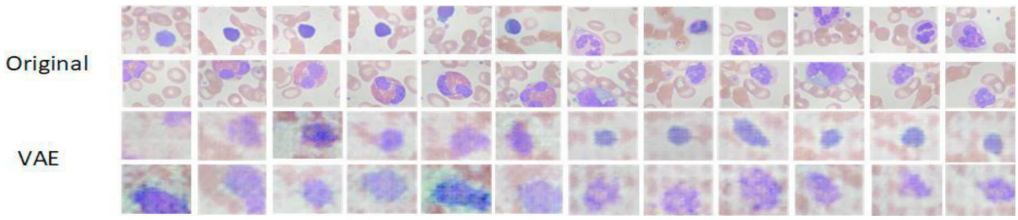


Figure 3. An example of blurry blood cells image reconstruction by a VAE [26]. It can be seen that VAE generates blurry images that lack texture details.

For the task of generating nuclei images from tabular data, there is no need to employ both components of an autoencoder architecture. Specifically, the encoder component may be unnecessary, as the tabular vector of features itself already serves as input to the decoder. This tabular data describes the images to be generated, and acts as an encoding.

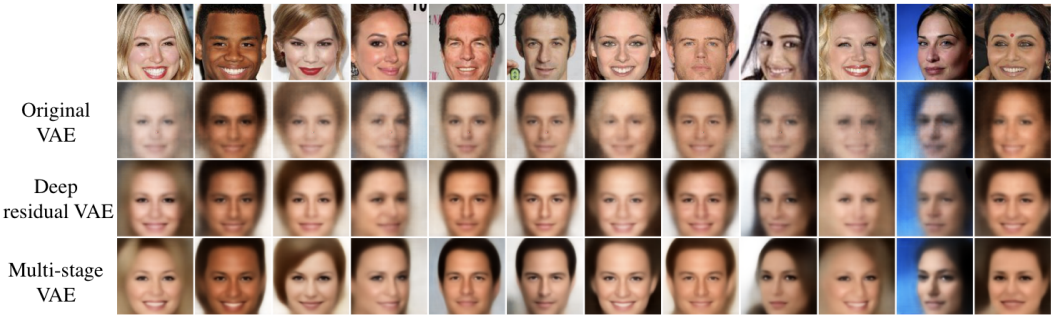


Figure 4. An example [10] of blurry face reconstruction by different variational autoencoders: original VAE, deep residual VAE and multi-stage VAE. It is evident that even the most advanced multi-stage VAE fails to generate a high quality texture for the given images.

Thus, during the current research we have decided to explore a simple decoder struc-

ture as a baseline model presented in Figure 2. Its detailed architecture is discussed in 4.2.1. After a few experiments with the decoder architecture we had an aforementioned problem of blurry image generation. We decided to stop exploration of autoencoders topic, since we found a more promising existing solution - generative adversarial networks, described in the next section.

3.2.2 Generative Adversarial Networks

Generative adversarial networks (GANs) were proposed in 2014 by Ian Goodfellow et al. [17]. A GAN is an unsupervised model that works in an adversarial process, in which two models are simultaneously trained: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake [17].

As shown in Figure 5 a GAN is composed of two neural networks:

- Generator

  A neural network that takes a sample from random distribution, e.g. Gaussian, as input and outputs data, in our case, an image. Random inputs act as the latent representations of the image to be generated. The generated instances become negative training samples for the discriminator.

- Discriminator

  A neural network that randomly samples negative and positive samples (real images) from the training data and attempts to determine if they are real or generated.

The goal of the generator is to create output images of such a high quality that discriminator is not able to understand which of its two inputs is real and which is generated. On the other hand, the discriminator aims to learn to perfectly classify two input images so that the generator cannot fool it.

Both the generator and the discriminator are trained simultaneously through backpropagation algorithm: the discriminator's classification provides a signal that the generator uses to update its weights. In more detail, the discriminator is trained to maximize the probability of assigning the correct label to both training samples and samples from a generator [17].

Let $D(\mathbf{x})$ be the discriminator's estimate of the probability that real data instance $\mathbf{x}$ is real, $G(\mathbf{z})$ be the generator's output when given noise $\mathbf{z}$ and $D(G(\mathbf{z}))$ be the discriminator's estimate of the probability that a fake instance is real. We train D to maximize the probability of assigning the correct label to both training samples and samples from G [18] :

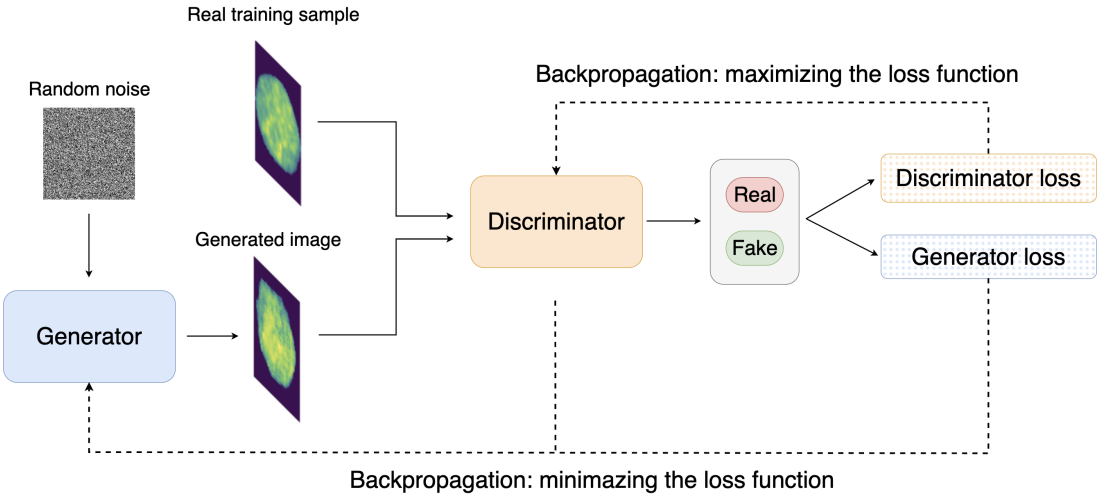


Figure 5. A standard GAN architecture. Generator receives random noise as an input and outputs a generated image. Both a generated image and a ground truth sample create an input to a discriminator, which predicts whether its input image is generated or real. The weights are updated though the backpropagation algorithm

$$\max \rightarrow \log D(\mathbf{x}) + \log(1 - D(G(\mathbf{z}))). \tag{2}$$

G is trained to minimize the inverse probability predicted by the discriminator for generated images. It encourages the generator to output samples that have a low probability of being fake:

$$\min \rightarrow \log(1 - D(G(\mathbf{z})). \tag{3}$$

We train G and D simultaneously, as if they are following the two-player min-max game with value function $V(G, D)$ [30] :

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{4}$$

where $p_{\text{data}}(\mathbf{x})$ and $p_{\mathbf{z}}(\mathbf{z})$ are the probability distributions of the real data and the random noise, respectively.

Our objective is to generate nuclei images based on multiple conditions such as size, shape, brightness, texture, and other relevant features. These conditions are described in a vector of features, which should act as input to the generator. However, a regular GAN is capable of generating images using randomly sampled input, i.e. unconditionally. Section 3.2.3 describes the modifications made to a regular GAN so it can generate images given a vector of features.

3.2.3 Conditional Generative Adversarial Networks

In section 3.2.2 we have described the limitations of a regular GAN in using conditional features for image generation. To overcome this problem Conditional GANs (CGANs) [30] was introduced. By integrating additional input information in the form of a set of conditional variables (we call it a label, for simplicity) CGANs can generate images that meet specific requirements, such as shape, texture, or brightness. In the context of our task of generating nuclei images based on various conditions, CGANs are a natural choice as they allow us to control the appearance of the generated nuclei by providing the desired conditions in the form of a vector of features.

In a CGAN, illustrated inn Figure 6, its generator takes two inputs: the random noise vector and the conditioning information. The conditioning information can be any type of auxiliary information that is relevant to the image generation task. We use a vector of features as a conditional label.

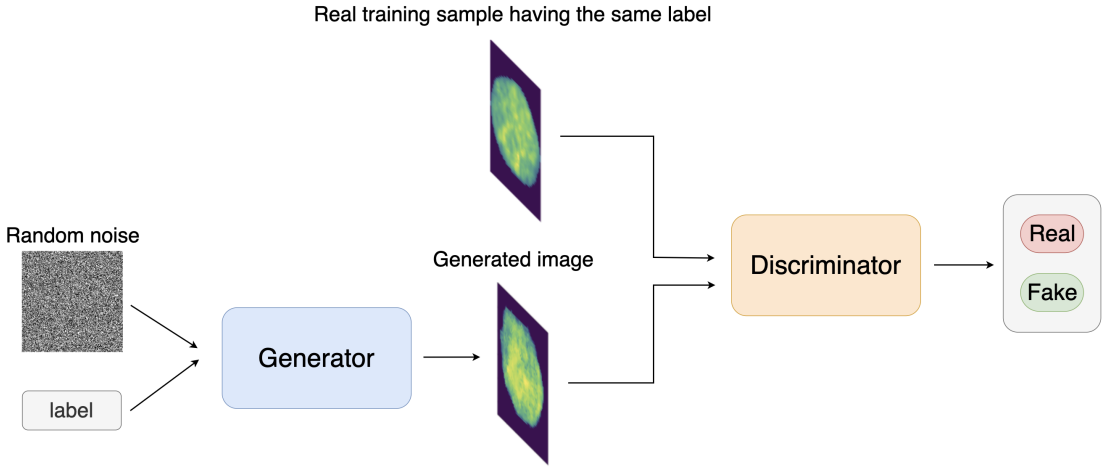


Figure 6. A conditional GAN architecture. Here, besides random noise the generator receives a label. This label contains conditions of data to be generated. In our case the label is going to be a vector of features

Similarly to a standard GAN, the discriminator in CGAN uses two inputs: a real image, which corresponds to an input set of conditional vector of features, and a generated one. The modified loss function used in training CGANs operates similarly to the traditional GAN loss function in a min-max optimization game. However, the primary distinction between them is that instead of using a standard probability distribution, both the generator and discriminator employ a conditional probability distribution:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))]. \quad (5)$$

4 Data and Methods

In this section, we describe the microscopy image and features dataset used in our experiments and the preprocessing steps applied to them. Then, we discuss methods used for nuclei generation from a vector of features. Finally, we define the metrics used for evaluation of the generated images.

4.1 Dataset

Since we are using a neural network model for image generation from numerical features we need a dataset to train this model as well as to evaluate its performance.

PerkinElmer, Inc. provided the dataset for research of an ongoing project. This dataset contains nuclei images (Figure 7(a)), annotated nuclei masks (Figure 7(c)) and numerical features for each nuclei (Figure 7(b)). Nuclei images contain cells of seven different cell lines, hence we further denote the dataset as *7 cell lines (7CL)*.

7CL dataset consists of 3024 microscope images of 1080×1080 pixels. These images are presented in a fluorescent modality meaning that it contains cell nuclei stained with a Hoechst 33342 dye [9], [37]. We will further refer to image of 1080×1080 pixels as *an initial image*. The density of cell nuclei is different for every initial image, moreover some initial images have overlapping or touching cells.

There are 151 numerical features corresponding to every nuclei present on the 1080×1080 image. These features include area, perimeter of a nucleus, its brightness and contrast, etc. The full list of features can be found in Table 4.

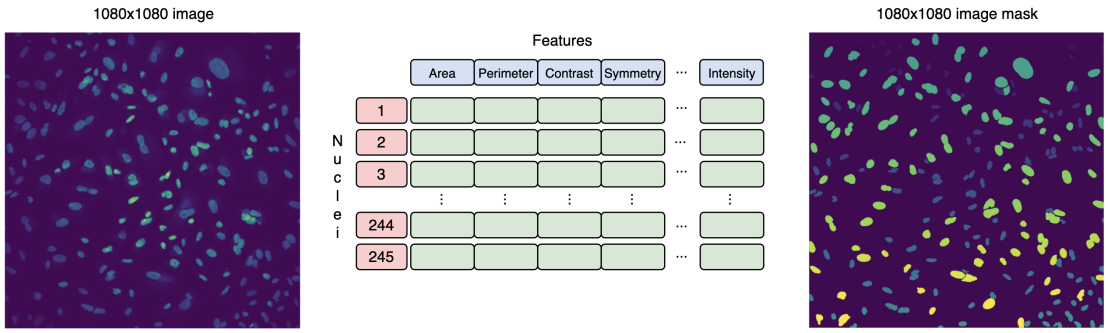


Figure 7. An example of (a) an initial image 1080×1080 containing 245 nuclei with (b) the corresponding features table and (c) nuclei segmented masks

We have used a subset of the 7CL dataset, resulting in a split of the original dataset into training (500 images), validation (50 images), and testing (50 images) sets.

4.1.1 Exploratory Data Analysis

During an exploratory data analysis stage we noted the following observations about the dataset:

First, each nucleus has a different size starting from only 88 px^2 and ending up with above 5500 px^2 of area as illustrated in Figure 8b. The figure shows a zoomed region of the right end of histogram, where the outliers can be clearly seen. We exclude them from a dataset later. Also we investigated that 99,5% of nuclei in the dataset can fit into the 56 $\times$ 56 bounding box.

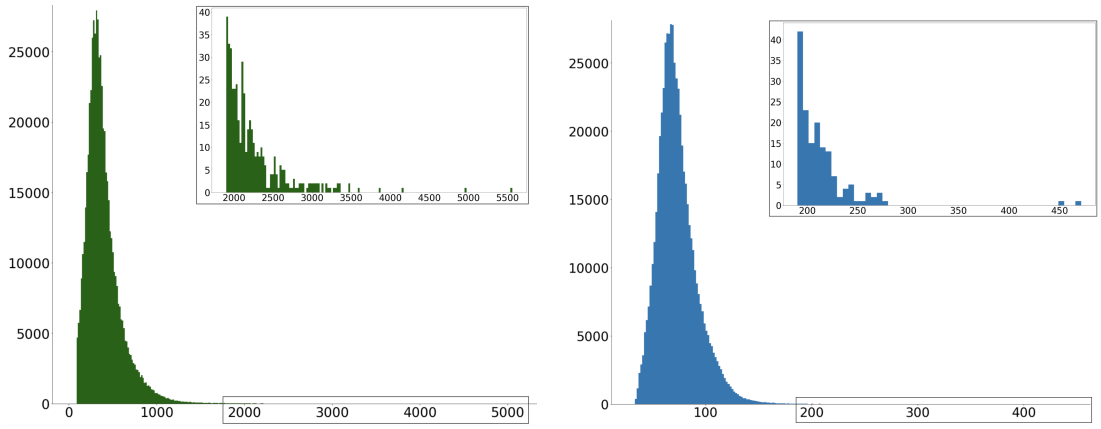


Figure 8. The distribution of (a) area and (b) perimeter values across the training dataset. A zoomed region of the right end of histogram shows the outliers.

Secondly, in the context of generating nuclei from a vector of features, it is essential to identify which input features are relevant for the generation task. It is important to note that not all input features contribute equally to the generation of high-quality images. For instance, features related to the location of a nucleus within the physical plate are irrelevant for individual nucleus generation, and their exclusion does not impact the quality of the generated images. Consequently, we exclude features such as 'Row', 'Column', 'Field', 'Plane', 'Timepoint', 'Object Number', 'X', 'Y', 'Position X [µm]' and 'Position Y [µm]' from further analysis, as they are arguably irrelevant for the task of generating realistically looking nuclei.

Next, to ensure the successful insertion of the generated nuclei back into the 1080×1080 original image, it is necessary to consider the location of each nucleus in this image. Four features that describe the coordinate boundaries of the rectangular bounding boxes around each nucleus are 'Bounding Box X1', 'Bounding Box Y1', 'Bounding Box X2', and 'Bounding Box Y2'. While these features may provide information on the length-to-width ratio of a nucleus, there are 10 other features that describe the width and length of a nucleus. Therefore, the bounding box features are not included in the training

process, but will be kept separately for further post-processing steps described in 5.3.

Finally, based on the results of the correlation analysis, it was found that some of the features used to describe nuclei were highly correlated, specifically, features such as median and maximum pixel intensity, axis x- and y-projections, and certain symmetry features. In total, 29 of the features were found to be particularly correlated.

4.1.2 Data prepossessing

Data preprocessing is an essential step in any machine learning pipeline, and it plays a crucial role in the success of the entire project. The primary purpose of data preprocessing is to clean, transform and prepare raw data so that it can be used effectively by machine learning algorithms.

In the context of image generation from a vector of features using CGANs and deep convolutional decoder model, data preprocessing is crucial because it helps to ensure that the input data is in the correct format, contains all the necessary features, and is free from errors or inconsistencies that could negatively impact the performance of the model.

This chapter will provide a detailed account of the data preprocessing steps taken and the reasons behind them. There are two separate prepossessing pipelines for images and features.

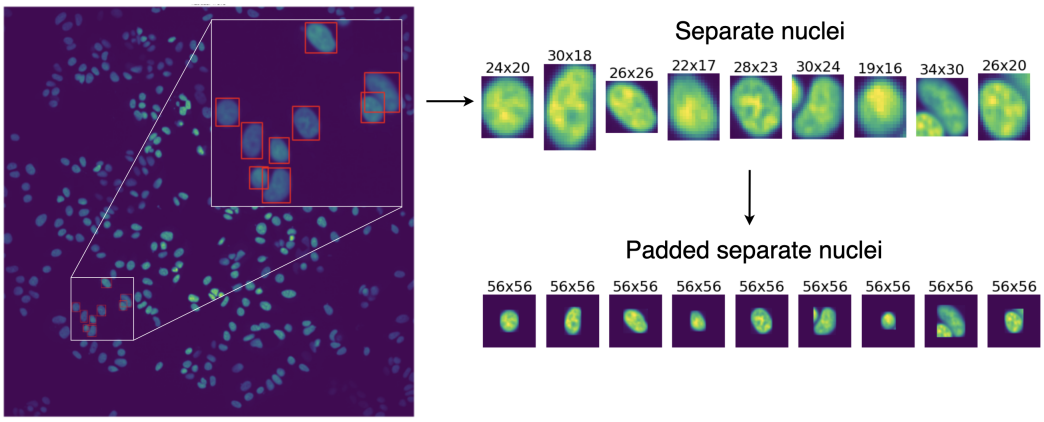


Figure 9. An example of a fluorescent image (left) with bounding boxes cutting algorithm applied to every nuclei (right)

Prepossessing of images

1. We apply a custom function for cutting images into separate nuclei using the bounding box coordinates as demonstrated in Figure 9. We end up with having

193447, 19864, 19589 separate (individual) nuclei of different sizes in training, validation and test sets, respectively.

2. In this study, a decision was made to use a fixed input size of 56×56 to avoid the extra work involved in making the neural network model be compatible with different input sizes. A small fraction of the images in the dataset (0.5%) did not fit into the 56×56 bounding box and were therefore considered outliers. To maintain consistency and improve the model's ability to capture typical patterns in the data, these images were excluded from the training, validation, and test sets along with their corresponding vectors of features. After removing the outliers, the training, validation, and test sets comprised 190988, 19459, and 19381 separate nuclei images, respectively.

3. To have a fixed image size the images are padded with zero pixels to top, bottom, left and right, so that their final size is 56×56 and the separate nucleus image is placed right in the center of the padded one. From now on we will refer to this step as a *padding* procedure. Figure 10 illustrates an example of how padding is applied to an individual nucleus image of size 21×22 pixels. It is worth mentioning that there is also a possibility of reshaping each image to 56×56. However, reshaping the images to a fixed size can result in loss of information or distorted images. On the other hand, by applying padding we can preserve the original aspect ratio of the images and ensure that all features related to the edges, object boundaries or texture patterns remain the same in the image.

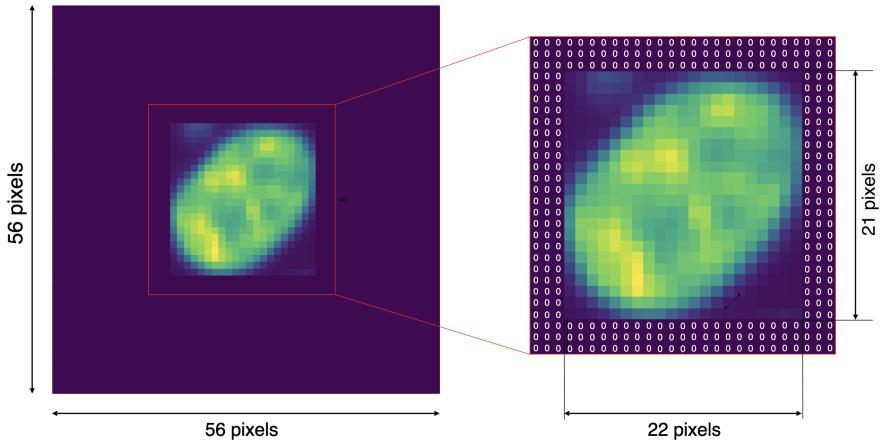


Figure 10. An example of how a separate nuclei of a size 21×22 is padded to 56×56 image. The nucleus image is placed at the center of 56×56 image and filled with zero-value pixels to the top, bottom, right and left.

4. As a standard image preprosessing step we also normalize pixel values. The purpose of normalization is to reduce the effect of differences in brightness or contrast between images on a model performance.

   In this context, we normalized all padded separate nuclei images to a [0; 1] range, so that the model can learn to recognize patterns and features based on their relative values, rather than absolute. Normalization does not introduce any new information or distortions into the image, so it is still a valid representation of the ground truth.

   In case of RGB images (where each pixel channel value is between 0 and 255) the pixel values of each image are usually divided by 255. However the images from a 7CL dataset have non-standard pixel values ranging from 0 to 10456. Hence each pixel in every padded image was divided by 10456.

After all the aforementioned preprocessing steps are applied the resulting images are considered to be the ground truth (the target) for our model.

Prepossessing of features

There are 151 features given as numeric numbers in a .csv file. The goal of this prepossessing pipeline is to create vectors of features for training and testing a neural network model.

Let's denote $N-$ a number of nuclei in a set to be normalized, which is also a number of the corresponding vectors of features.

1. As described in 4.1.1 there are non relevant and highly correlated features, which are removed. As a result, the number of relevant features available for model building was reduced to 108.

2. Feature values that are undefined will be set to zeroes.

3. Vectors of features are normalized vertically. Let's stack all N vectors of features on the top of each other. They will create a table of features as in Figure 11(a). It can be seen that each feature will include N values: 1 from each vector of features. By vectical normalization we denote normalization across each of 108 features. Normalizing the features means scaling them so that they have a similar range of [0; 1], which can help prevent one feature from dominating over others. Normalization for each feature is formulated as:

$$f_{norm} = \frac{f - min(f)}{max(f) - min(f)} \tag{6}$$

   where $min(f)$ and $max(f)$ are the minimum and maximum value of a selected feature, respectively.

As we can see from the Figure 11, the distribution of data did not change, however, all features now are ranged between 0 and 1.

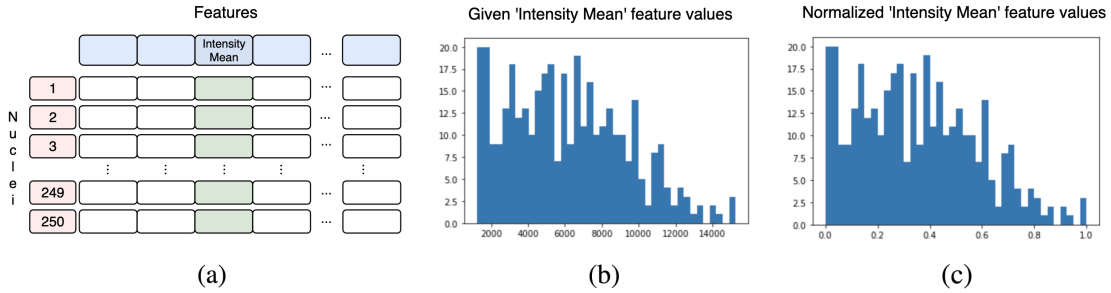


Figure 11. An example of how 250 vectors of features are vectically normalized. Each feature (shown green in (a)) is taken separately as a vector of 250 values and normalized according to formula (6). The distribution of values remains unchanged (b, c).

4. Vectors of features are normalized horizontally. Here by horizontal normalization we denote standardization of a vector, which is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance). If the vector of features is not normally distributed, some areas of the image space may be overrepresented while others may be underrepresented. This can result in generated images that are biased towards certain characteristics and may not accurately represent the full range of possible images.

   Given that the vector of features does not follow a Gaussian distribution and instead follows a random distribution (Figure 12b), we employ the StandardScaler() [3] function, a standard preprocessing technique included in the Scikit-learn library. The function facilitates the standardization of the distribution of features by removing the mean value and scaling the data to unit variance, resulting in the transformed feature vector being closer to a Gaussian distribution, as shown in Figure 12c.

5. (For CGAN-based model only): To fulfill the requirement of generating diverse and realistic images using CGANs, each vector of features was added as a condition to a 800 vector of random values sampled from a Gaussian distribution, generated independently for each vector of features. Consequently, we obtained a vector of size 908, that was used as input for a Table2Cell model (4.2.2), comprising of the original 108 features and 800 random values.

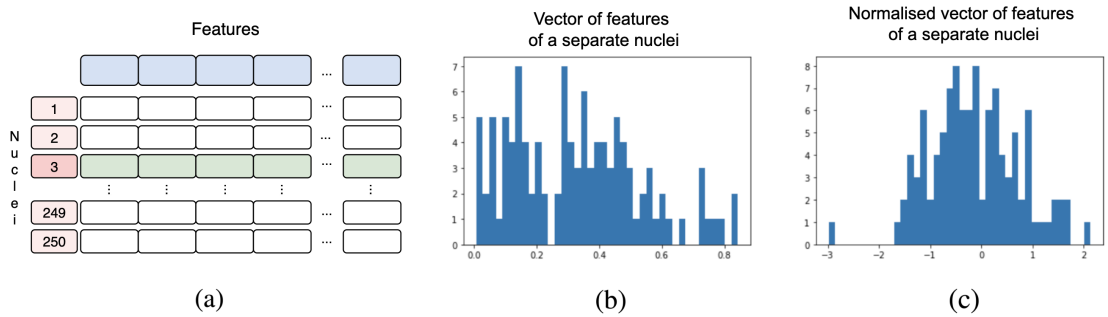


Figure 12. An example of how 250 vectors of features are horizontally normalized. Each vector of features (shown in green in (a)) is standardized by StandardScaler() function. Distribution of feature values changes to Gaussian (b).

4.2 Methods

Here we describe a detailed architectures of models that we have built to generate nuclei images from numerical features. First, we show our deep convolutional decoder approach and explain why it was insufficient. Next, we describe the CGANs-based model, which we call Table2Cell.

4.2.1 Deep Convolutional Decoder

A decoder neural network described in section 3.2.1 takes a compressed representation of data (a vector of features) and outputs an image. Since we use a decoder for image generation purpose it consists of multiple transposed convolutional layers, which gradually increase the spatial resolution of the input vector while reducing the number of channels.

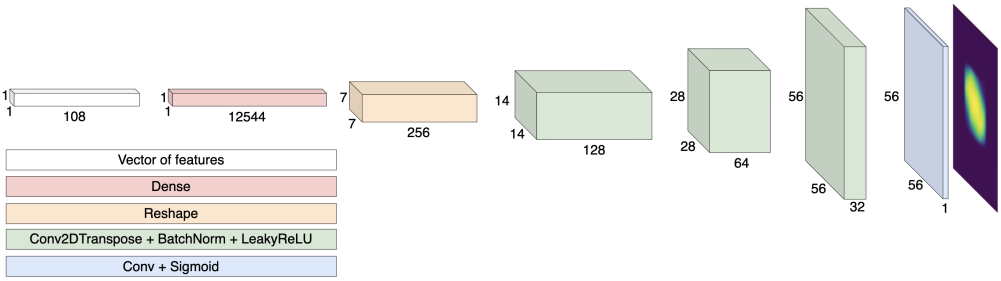


Figure 13. An example of a deep convolutional decoder architecture. A vector of 108 features is passed through dense and convolutional layers from left to right. The last network layer outputs a generated image of a single nucleus. The image size is fixed to 56x56 pixels.

The first network that we have build is shown on Figure 13. During the data exploration stage 4.1.1 we discovered that there are 108 weakly correlated features that can be used for image generation. Therefore, an input vector of 108 features is used as input to a fully connected dense layer with 12544 neurons. The output of the dense layer is then reshaped into a 3D tensor with the shape *(width, height, number of channels)*= (7, 7, 256). The reason we chose the width and height to be 7 is because later this tensor is passed thorough a series of convolutional layers, which increase the width and height by a factor of 2. We decided to add 3 upsampling layers, which in total increase the size of our tensor by a factor of 8 in both the width and height dimensions. To get an image of size $56 \times 56 \times 1$ as a final network output there has to be a tensor of width and height equal to 56/8 = 7. The (7, 7, 256) tensor is passed through the deconvolutional layer, which is a transposed convolutional layer, that increases the resolution of the input tensor by a factor of a stride. We used a kernel size of 4×4, a stride of 2 and a large number of filters, 256. Next, a batch normalization layer was added to stabilize the training process by normalizing the output of the previous layer. Then a LeakyReLu activation function is applied to the output. We repeat the previous three layers 3 times, with the number of filters decreasing and the spatial resolution increasing in each layer. The final convolutional layer maps the output tensor to the image space and uses a kernel size of 4×4, a stride of 2. The output tensor has shape (56, 56, 1). A sigmoid activation function is also applied to the output of the previous layer to ensure that the pixel values are within the [0, 1] range. The final output of the model is a generated image of size $56 \times 56 \times 1$, but we reshape it to 56×56 for convenience.

We trained the deep convolutional decoder model until convergence using binary crossentropy loss and an Adam optimizer with 1×10^{-2} learning rate and batch size 64.

The performance results of deep convolutional decoder model are avaliable in section 5.1. Overall, the images generated were blurry and lacked detailed texture. In section 3.2.1 we have explained that the problem of blurry image generation is common to encoder/decoder models due to MSE loss. Hence we left the deep convolutional decoder model as a baseline and decided to explore another method disscussed in section 4.2.2.

4.2.2 Table2Cell

A Conditional Generative Adversarial Network (CGAN) described in section 3.2.3 maps a source data distribution to a target distribution by optimizing a generator and a discriminator. In our case the source is a vector of features and the target is an image. We have used two different deep convolutional neural networks in both the generator and the discriminator. The generator is based on transpose convolutions and the discriminator is based on regular convolutions. A detailed architecture of both parts is presented in Figure 14.

The generator model depicted in Figure 14a accepts an input composed of two components: a vector of features and random noise sampled from a uniform distribution.

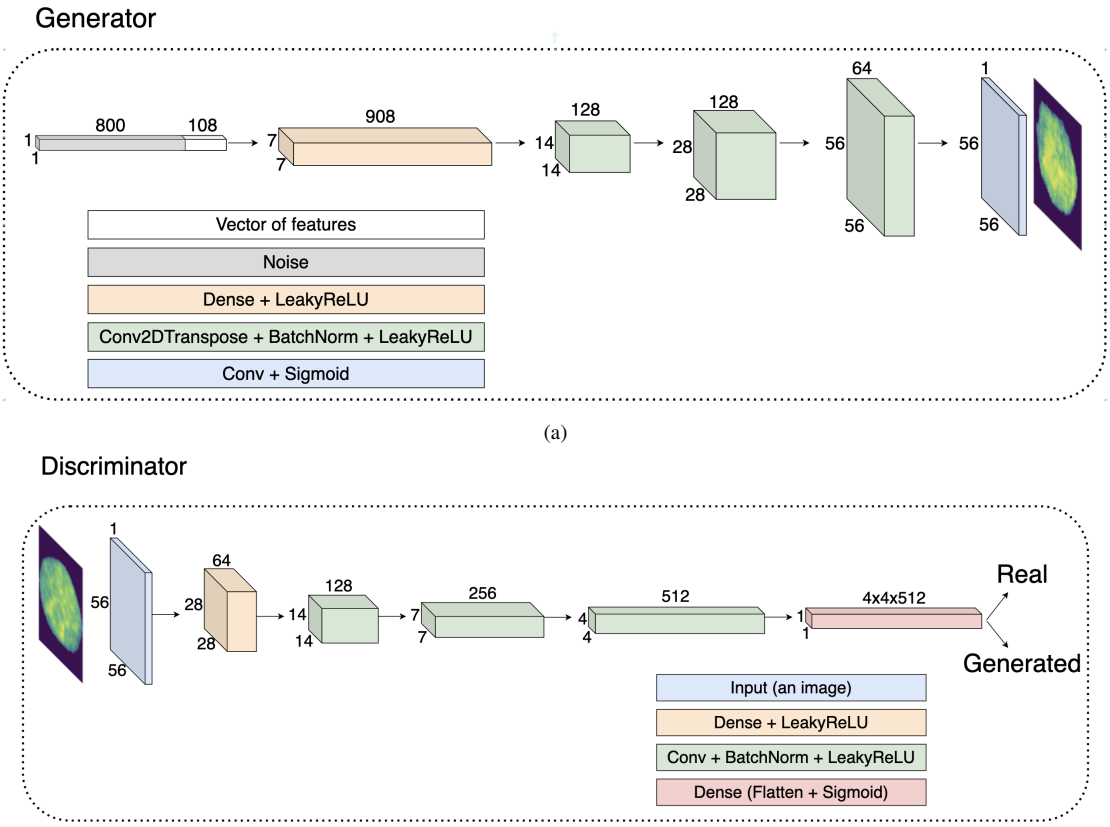


Figure 14. An architecture of a generator (a) and a discriminator (b) parts of a CGAN.

The size of the noise input plays a crucial role in determining the ability of a CGAN to model the data distribution. Prior studies, such as those conducted by Mirza at el. [30] and Radford at el. [33], used a noise input size of 100 in their GAN models. However, choosing this size in our model resulted in generated samples being insufficiently diverse and complex, so to improve their quality we have opted to employ a larger noise input vector of size 800. We recognize that the selection of the noise vector size may not be optimal and could benefit from further experimentation. However, due to time constraints, we were unable to thoroughly explore and test the various options. A vector of 108 conditional features was concatenated to a random noise vector of size 800 in an input layer of generator. Then a fully connected dense layer is added to map the input vector of size 908 to a higher-dimensional representation followed by a LeakyReLU function. The received image of size $7 \times 7 \times 908$ is then upsampled three times to reach a $56 \times 56 \times 1$ output size. The model uses batch normalisation and LeakyReLU activation in each of the upsampling layers. The output layer uses a sigmoid activation function.

As demonstrated inFigure 14b, the discriminator model takes as input one $56 \times 56 \times 1$

image and outputs a binary prediction as to whether the image is real (class=1) or generated (class=0). Similarly to the generator, it is implemented as a convolutional neural network. In each convolutional downsampling layer, the number of filters doubles while the image shape shrinks by a factor of two. Every block has a batch normalization and LeakyReLU activation function. The final layer includes a sigmoid activation function. All convolutional layers of both generator and discriminator have kernel size of 4×4 and a stride of 2.

Next, our CGAN model was defined as a combination of both the generator model and the discriminator model.

We trained the model for 150 epochs using a binary crossentropy loss and an Adam optimizer with 3×10^{-4} learning rate and 0.5 momentum. Batch size was set to 64.

We call this model Table2Cell, since it uses vectors (that can be stacked on the top of each other, forming a table) of features to generate nuclei images, which are the main components in cells.

4.3 Metrics to assess the quality of generated images

We assess the prediction of the nuclei generation model based on specific metrics, which evaluate how similar generated images are to ground truth images. Selection of metrics plays a pivotal role in evaluating the performance of deep learning models, because they help to determine how well the model generalizes to the new dataset.

We chose three score metrics for image generation evaluation: mean squared error (MSE), structural similarity index measure (SSIM) and a cosine similarity (CS) score. The first two are focused on measuring similarity between a generated image and a ground truth image. The last one evaluates the similarity between generated and ground truth vectors of features. By considering these metrics, we can comprehensively evaluate the performance of our model and obtain valuable insights into its ability to accurately reproduce image of nucleus from the given vector of features.

Mean squared error is a very popular way to quantify the performance of any regression model, including the model that generates pixel intensity values.

Let N be the number of pixels in the image, f_i be the pixel value of an image generated by a model, y_i be the pixel value of a ground truth image. As demonstrated in Equation (7), first we calculate a pixel-wise difference between a generated and a ground truth image. This can be achieved by subtracting each pixel value of one image from its corresponding pixel value in the other image: $f_i - y_i$. The resulting image has pixel values representing the difference between the two images. Next, the square of each pixel value in the difference image is calculated. This is done to ensure that all pixel differences are positive, and to give more weight to larger differences. Finally, the average of all the squared pixel differences in the difference image is calculated.

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(f_i - y_i)^2 \tag{7}$$

Given that the ground truth and generated nuclei images were both initially padded to fit within a 56×56 bounding box, there were zero pixels that did not contain any useful information about the nuclei. In order to avoid any impact that these extraneous zero values may have on the final evaluation score, we implemented a procedure to remove the redundant pixels from the border of the images. We refer to this process as an unpadding procedure, and elaborate on it further in 5.3. Figure 15 illustrates the aforementioned process.

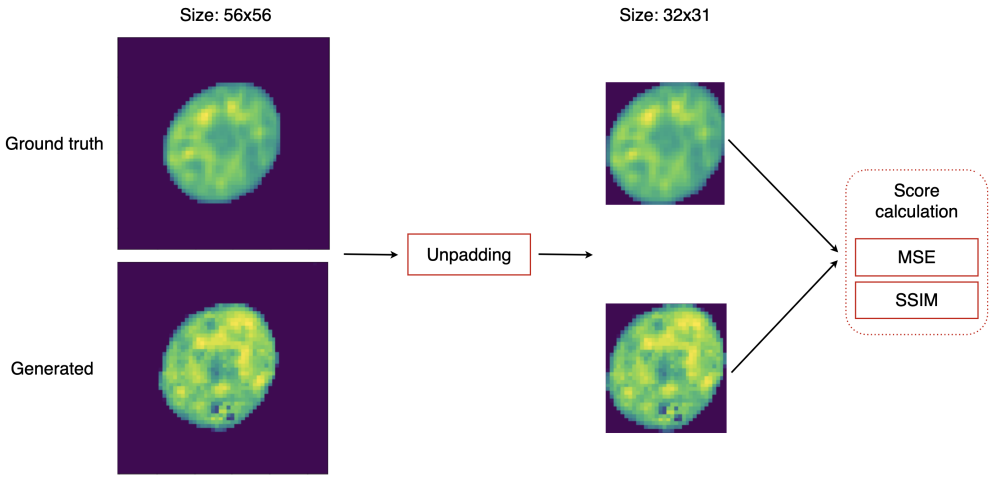


Figure 15. Evaluating model performance using MSE and SSIM. First, redundant pixels are removed from the borders of the ground truth as well as generated image of the nuclei (left). For MSE score calculation, the average squared differences of corresponding pixel intensities of images are measured, and the final score is computed by taking the mean of all MSE scores. The SSIM score can be calculated using a formula that measures the similarity of structural information between two images by comparing luminance, contrast, and structural terms.

In the context of image generation, a perfect model should ideally achieve an MSE of 0, implying that the generated image is identical to its corresponding ground truth. However, it is worth to point out that the MSE score can theoretically tend towards infinity, as it is dependent on the scale of the data. In our case, given that the pixel values of both training and test images are normalized within the range of 0 and 1, the highest achievable MSE score for any model is limited to 1.

Structural similarity index measure is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality. SSIM

index was proposed in 2004 by Wang et al. [43]. The minimum possible SSIM value is -1, which indicates the two given images are very different. An SSIM score of 1 means perfect structural similarity, as is expected out of identical images.

Similarly to calculating MSE, the unpadding procedure is used to get rid of padded zero pixels as shown in Figure 15.

The key difference between MSE and SSIM is that the former calculates the mean square error between each pixel for the two images we are comparing, whereas the latter does the opposite and looks for similarities within pixels; i.e. if the pixels in the two images line up and or have similar pixel density values.

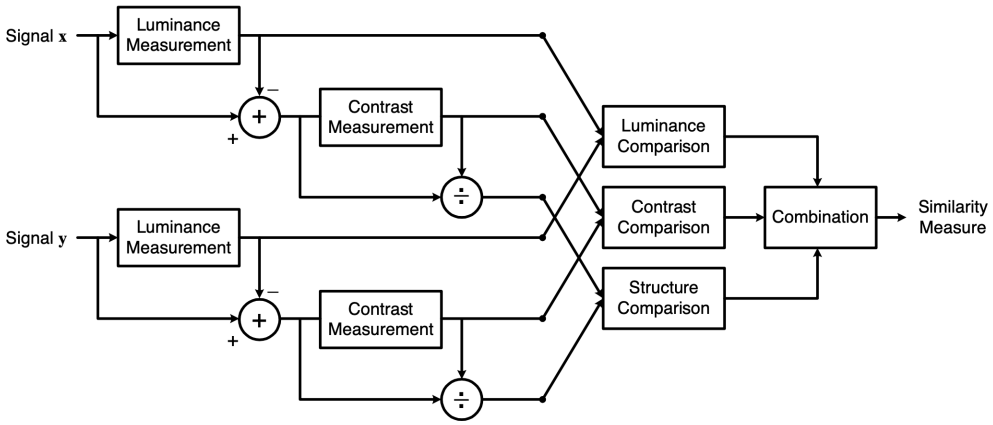


Figure 16. Diagram of the structural similarity (SSIM) measurement system [43]. The system separates the task of SSIM measurement into three comparisons: luminance, contrast and structure.

The diagram of the SSIM quality assessment system is shown in Figure 16. Assuming that and are two nonnegative image signals **x** and **y** that have been aligned, we can use a SSIM measure as a quantitative assessment of the quality of the second signal relative to the first one. The system divides the similarity measurement task into three comparisons: luminance(l), contrast(c), and structure(s) [43], which are defined as follows:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{8}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{9}$$

$$s(x,y) = \frac{2\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{10}$$

where x and y are two image signals being compared, μ_x and μ_y are the mean values of x and y respectively, σ_x^2 and σ_y^2 are the variances of x and y respectively, σ_{xy} is the

covariance between x and y, C_1, C_2 and C_3 are constants added for numerical stability in denominators.

Finally, (8), (9) and (10) are combined resulting the SSIM index between signals **x** and **y**:

$$\text{SSIM}(x, y) = (l(x, y))^{\alpha}(c(x, y))^{\beta}(s(x, y))^{\gamma} \tag{11}$$

where $\alpha, \beta, \gamma > 0$ are weights, used to adjust the relative importance of the three components. Usually these weights are set to 1 [43].

(a) Original image (b) Increased pixel intensity (c) Salt and pepper noise

Figure 17. An example [20] of two images (b, c) with different corruptions, having the same MSE of 0.4 with respect to the original image (a). SSIM captures more significant variation: 0.841 and 0.105 for (b) and (c), respectively.

There are studies [43], [39], [44] that explore the limitations of MSE and propose alternative metrics. There studies mention that MSE does not correlate well with perceived image quality. This is because MSE does not take into account the human visual system's sensitivity to contrast and structural changes in the image, which can be more important than changes in individual pixel values. Additionally, MSE treats all pixels equally, while in reality some regions of an image may be more important than others for accurate visual perception.

Figure 17 demonstrates an instance where two images, with different corruptions, have the same MSE with respect to the original image (Figure 17a). Despite significant visual dissimilarities between the two modified images, with one having all pixel intensities increased by 0.2 times (Figure 17b), and the other having half of its pixels' intensities increased and the other half decreased by the same amount (Figure 17c), both have an identical MSE of 0.4 comparing to the original image. Consequently, MSE was unable to capture the perceived similarity of these images. In contrast, SSIM captures more significant variation: 0.841 and 0.105 for Figure 17b and Figure 17c, respectively. It is evident that the modified image in Figure 17b is rated much closer to the original image in terms of perceived similarity, which is consistent with its visual appearance.

Therefore we included SSIM in addition to MSE to be the quality metric in our study.

Cosine similarity score is a measure of similarity between two sequences of numbers. It is measured by the cosine of the angle between two vectors **x** and **y**, and determines whether these vectors are pointing in roughly the same direction. The output can be a nonnegative similarity score between 0 and 1: 1 if the two images are completely similar to each other, otherwise 0.

$$\text{similarity(x,y)} = \cos(\theta) = \frac{x \cdot y}{|x| \cdot |y|} \tag{12}$$

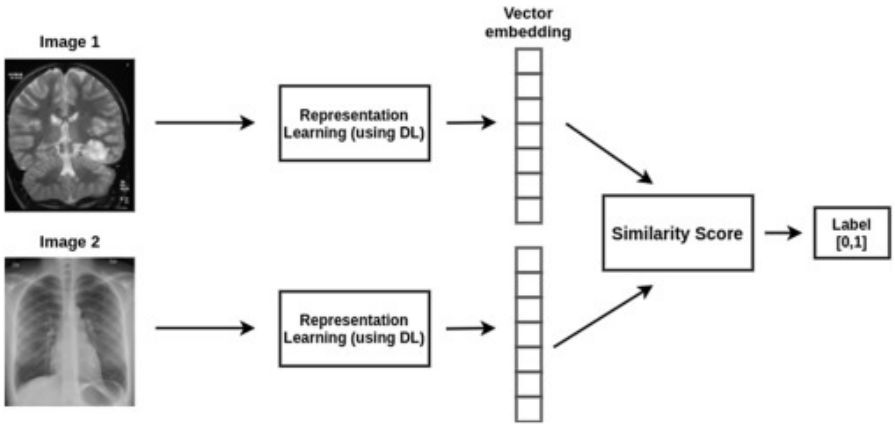


Figure 18. Calculating the similarity score. DL, Deep learning. Vector embeddings are extracted from both images using a pre-trained representaion learning model. The similarity score is calculated from these embeddings [19].

However, it cannot be used directly on raw images to assess their similarity, because the images have high dimensionality and a lot of noise. By extracting features, we can reduce the dimensionality and represent the image in a more meaningful way [22].

In this study, we adopted the approach proposed by Gupta et al. [19] for calculating cosine similarity between images, shown in Figure 18. This method involves embedding images into vector representations using a pre-trained deep learning architecture. These embeddings are then passed to a similarity metric learning function.

In our case, we did not need to learn the representation of image features, as they were already given. To compare the feature vectors of generated nuclei images with the original feature vectors, we first remove all zero-padded pixels from the generated image using an unpadding procedure, explained in 5.3. Then we insert these individual images back into the original 1080×1080 image using a custom function described in 5.3. Next, PerkinElmer software extracts 151 features from each nucleus found in 1080×1080 image. The selected extracted 108 features are then compared with the original vector of 108 corresponding features using CS metric. This process is illustrated in Figure 19.

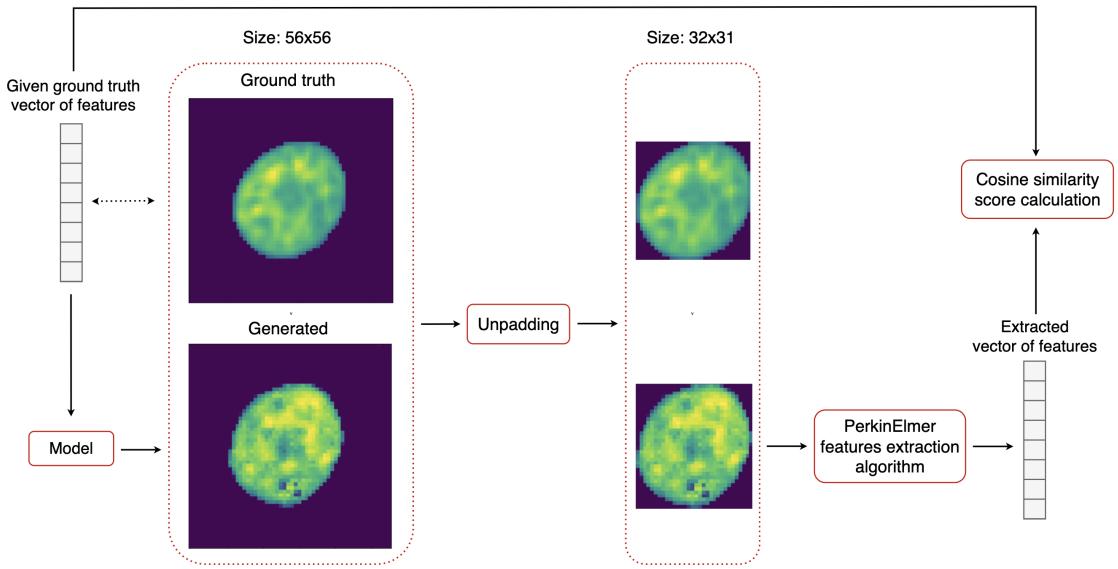


Figure 19. The process of calculating the similarity score between a generated and ground truth image. First, the generated image is unpadded to be used by a PerkinElmer software which extracts the feature vector from it. These features are then compared to the given vector of features using Formula (12)

5 Experiments and results

In this section, we report and analyse results for deep convolutional decoder and Table2Cell model for a task of image generation from a vector of features. We also show how the quality of generated images can be improved by additional preprocessing and postprocessing steps. Additionally we performed experiments to test Table2Cell model ability to generate high-quality nuclei with fewer input features and its responsiveness to parameter variations.

5.1 Deep Convolutional Decoder and Table2Cell generation results

We have evaluated the performance of nuclei image generation from a vector of features of a deep convolutional decoder and Table2Cell models on 7CL testing set and reported the MSE, SSIM and cosine similarity metrics in Table 1. We have also visualized the predicted nuclei images vs. ground truth nuclei images for both models in Figure 20.

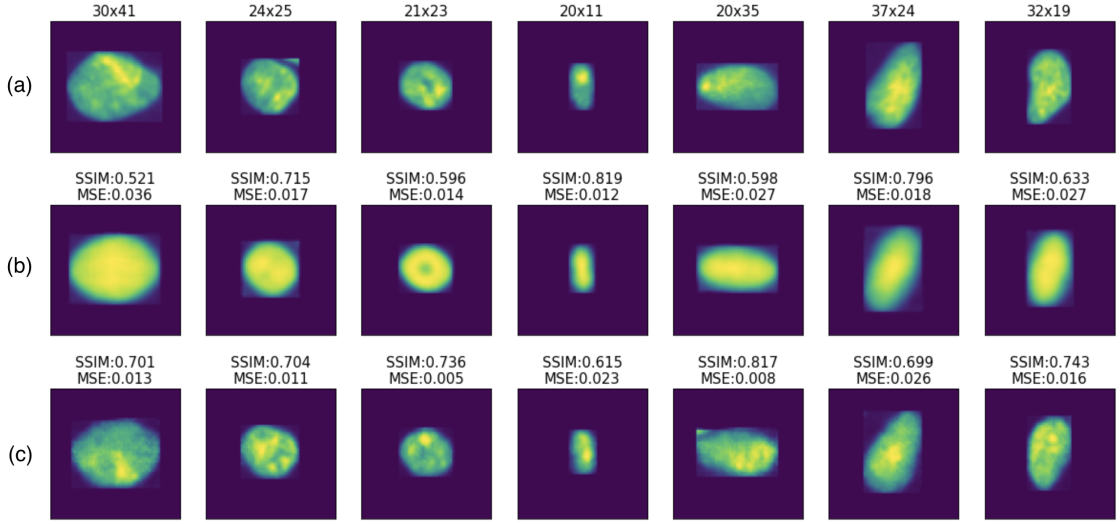
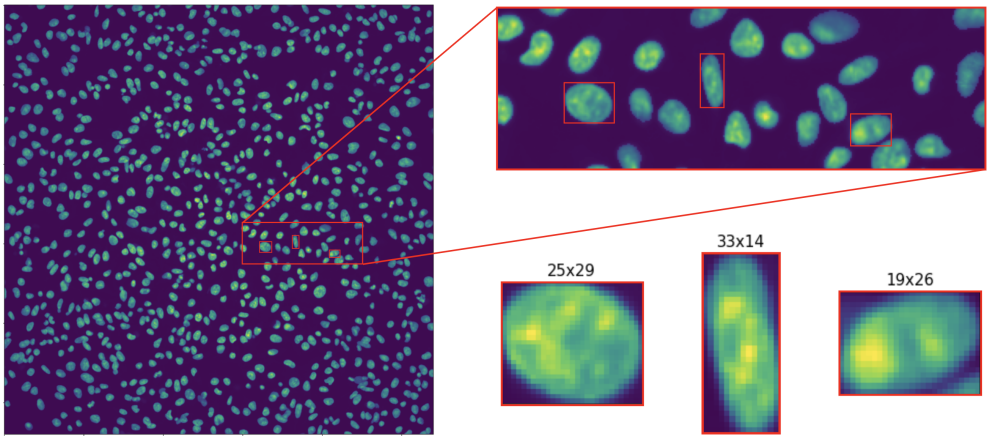
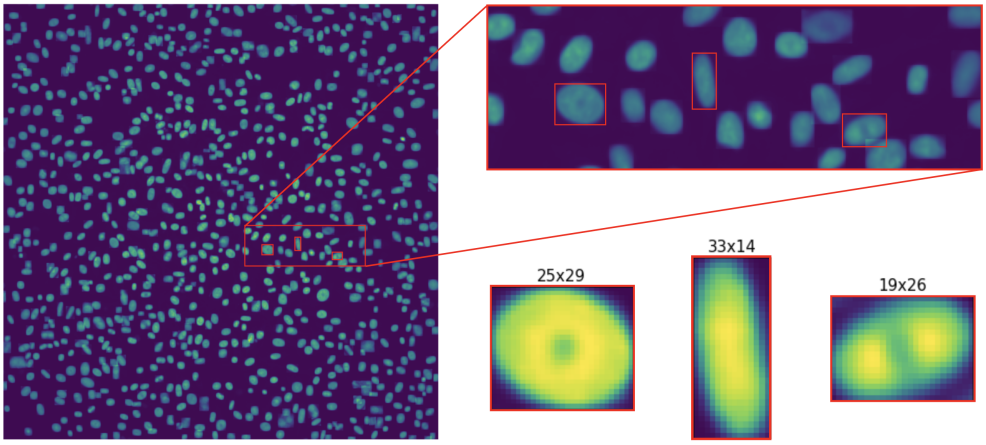


Figure 20. Examples of deep convolutional decoder (b) and Table2Cell model(c) nuclei image generation predictions compared to the ground truth (a).

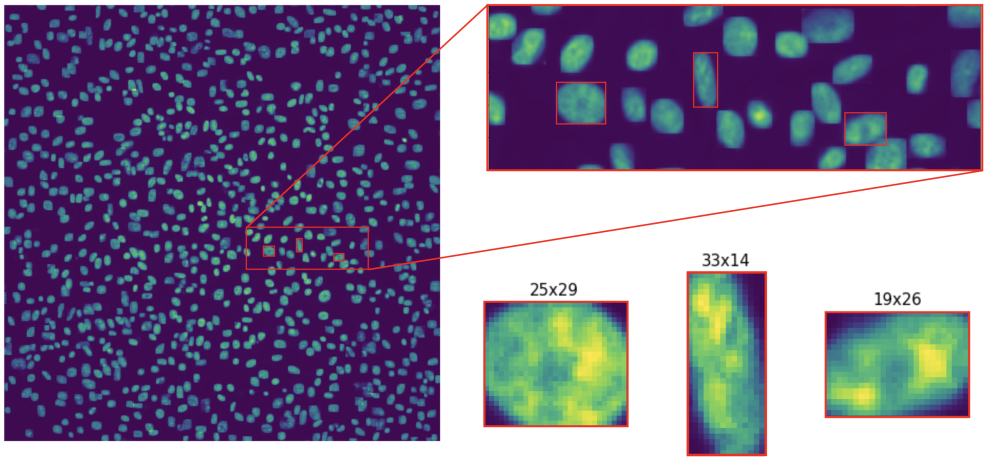
Figure 20 shows that a deep convolutional decoder model succeeded to generate images of the correct shape and angle, however, iproduced blurry images, which lack texture details similar to decoder-based approaches [10], [26]. In comparison, Table2Cell managed to generate texture in images, although it does not perfectly match the texture of the corresponding real images. Numeric performance results in Table 1 also indicate that Table2Cell outperformed a deep convolutional decoder, especially in terms of the cosine similarity score.



(a) Initial image with normalized individual nuclei (ground truth)



(b) Images generated by a Deep convolutional decoder



(c) Images generated by a Table2Cell model

Figure 21. Examples of Deep convolutional decoder (b) and Table2Cell predictions (c) in comparison to ground truth (a). Each example shows 3 unpadded generated nuclei images, taken from a zoomed region of a 1080×1080 image. The 1080×1080 image was received by inserting the unpadded predictions back to the intial image.

Table 1. Nuclei image generation results on PerkinElmer 7CL Fluorescent dataset

Model	MSE	SSIM	CS
Deep convolutional decoder	0.0378	0.6300	0.501
Table2Cell	0.0191	0.7208	0.923

5.2 Towards improving cells image generation

In section 4.1.2 we have described the preprocessing techniques applied to dataset images. The key steps include cutting each initial 1080×1080 image into separate nuclei images, then padding them with zero value pixels to 56×56 size and doing normalization after this.

The PerkinElmer software follows a specific method to extract features from the 1080×1080 image of nuclei. It first segments all nuclei in the image and then calculates 151 properties for each of the segmented nucleus regions. The resulting properties in the given vector of 151 features describe only the nucleus itself, without being influenced by the background or neighboring nuclei. This led us to the thought that there are still image preprocessing improvements to consider, since our training dataset includes pictures with noise and neighboring nuclei.

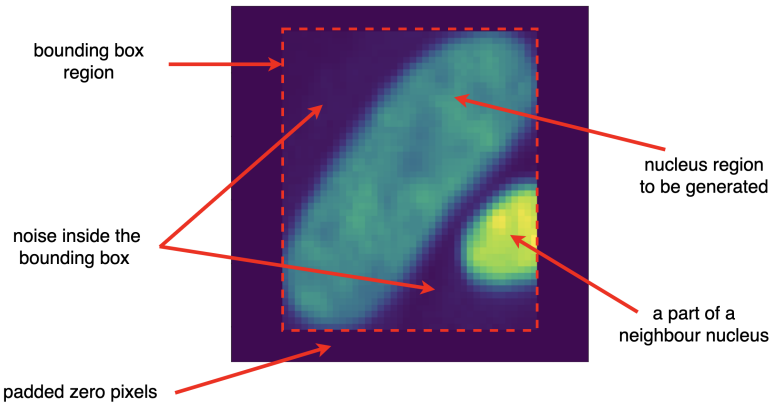


Figure 22. An image sample from a training set. Besides nucleus region to be generated there is also some background noise present and parts of neighbour nuclei may be visible as well.

During the preprocessing of images explained in section 4.1.2 we did not take into consideration that some information in the training image is not accounted for in the corresponding feature vector. Figure 22 depicts all possible parts of images utilized for

the Table2Cell model's training and testing. The figure illustrates that each individual nucleus image has some background noise surrounding the nucleus region within the suppositive bounding box. Moreover, there is a possibility for neighboring nuclei to be visible due to high nuclei density in certain areas of the original image. Therefore, using ground truth nuclei as on Figure 22 will result in Table2Cell model hallucinating some surrounding cells and noise, which is undesirable due to mostly stochastic nature, which is not even represented in the input feature vector.

To overcome the aforementioned issue, we introduced an additional optional preprocessing step in our approach, namely the background filtering algorithm, illustrated in Figure 23. This step may not be mandatory if there are features that effectively capture the background noise and the presence of neighboring nuclei within the bounding box of a given nucleus region, including their spatial coordinates, shape, and other relevant characteristics.

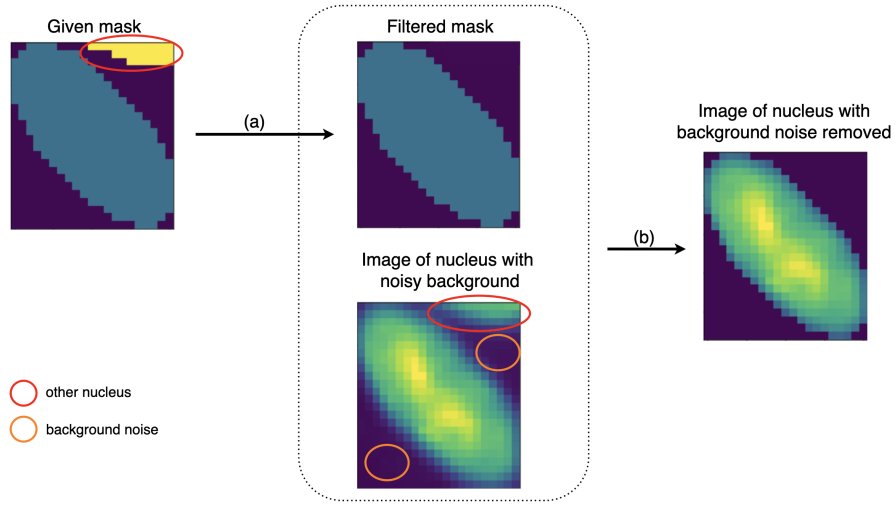


Figure 23. The procedure of noise filtering for each individual nucleus image. Noise may include background as well as parts of other nuclei. The first step involves the removal of the segmented regions of neighboring nuclei from the nucleus mask (a). Then the filtered mask is applied to a noisy ground truth individual nucleus image (b). The output of the preprocessing is an image with background pixels set to 0 and only one nucleus visible.

The background filtering algorithm uses 1080×1080 pixels segmented masks of nuclei (Figure 7c). This method involves the division of the large image masks in a manner that aligns with the approach used for cutting fluorescent images into individual nuclei images, as described in section 4.1.2. The segmented regions of neighbouring nuclei are removed from the masks and resultant cleaned masks are applied to the non-padded individual nuclei images. The output images have the background pixels set to 0, with a single, isolated nucleus clearly visible, as depicted in the right image in Figure 23.

Table 2. Nuclei image generation results on PerkinElmer 7CL Fluorescent dataset with background filtering algorithm usage.

Model	MSE	SSIM	CS
Deep convolutional decoder	0.0297	0.6770	0.520
Table2Cell	0.0154	0.7516	0.981

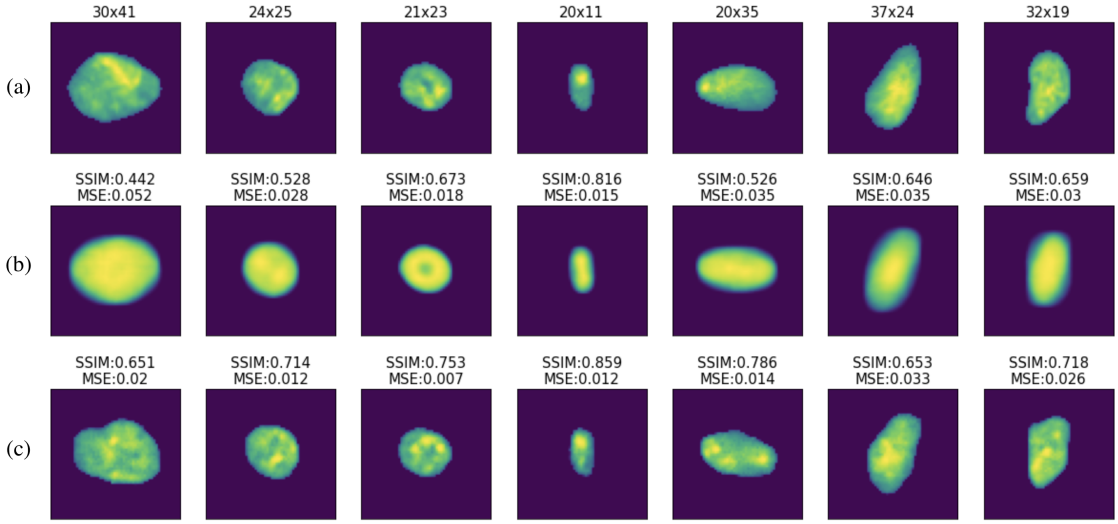


Figure 24. Examples of deep convolutional decoder (b) and Table2Cell model (c) nuclei image generation predictions compared to the ground truth (a) using nuclei images with cleaned background during training and testing. SSIM and MSE score were calculated on unpadded images.

The visual results in Figure 24 show the improvement of generated images in terms of the absense of background for both models. Cosine similarity score presented in Table 2 demonstrates the improvement of a Table2Cell model generate nuclei with the appropriate features, however there was not much improvement for a deep convolutional decoder.

5.3 Postprocessing

The postprocessing of the generated images involves two steps: an inverse padding procedure and pasting the generated nuclei back into the original 1080×1080 image.

Following the generation of all 56×56 images, an inverse padding procedure is performed to eliminate the zero-padded pixels in each nucleus image added during preprocessing. From now on we call this process an *unpadding procedure* for simplic-

ity. First, the bounding box is positioned precisely in the center of a 56 × 56 image. Subsequently, all pixels located outside the bounding box region are removed. The resulting image has the same dimensions as the bounding box and individual image of a nucleus. An example of the unpadding procedure applied to a generated image to restore its original shape is depicted in Figure 25.

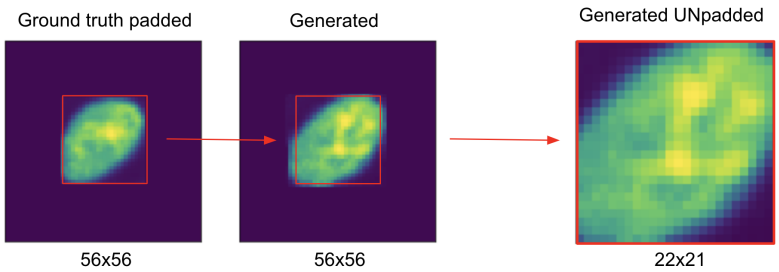
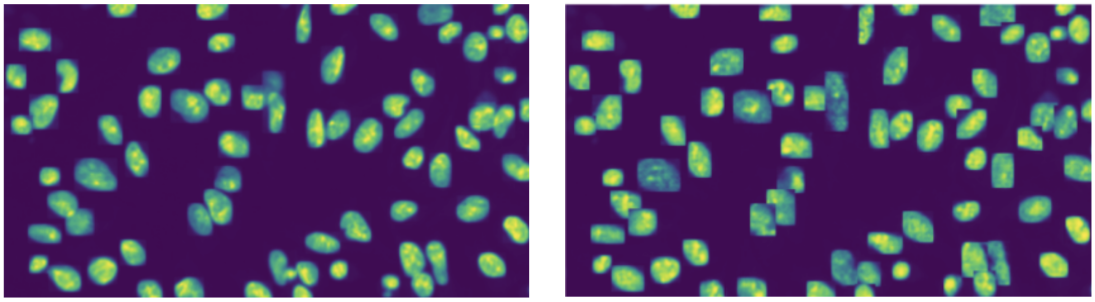


Figure 25. An illustration of an unpadding procedure. The features, that describe the bounding box (shown in red) coordinates are used to remove zero padded pixels and make the shape of the generated image the same as ground truth image.

One of the primary objectives of our study is to accurately reproduce the original 1080×1080 fluorescent image, shown in Figure 7a. To achieve this goal, it is necessary to insert the unpadded generated nuclei images to their precise positions on the initial image. This is done using 4 features that describe the bounding box coordinates of each nucleus on the 1080×1080 fluorescent image (features 9-12 in Table 4). Notably, the unpadded generated image maintains the same dimensions as the corresponding individual nuclei image, and therefore, we substitute the pixel values within the bounding box of the initial nucleus image with those of the generated one. Figure 21 provides a visual representation of the results obtained from this process.

However, it is evident that the nuclei in the generated images (Figure 21c, Figure 26b) appear with unnaturally sharp edges, in contrast to the oval shapes of the nuclei in the initial image (Figure 21a, Figure 26a). The cause of this issue can be traced back to the unpadding algorithm: occasionally the generated nuclei are larger than their actual size, rendering them incompatible with their respective bounding boxes, hence the edges are cut from the sides. The right image in Figure 25 shows an example of when that some of the edge components of the generated nucleus extend beyond the bounding box, leading to their removal and the creation of edges that look incomplete within the resulting image.

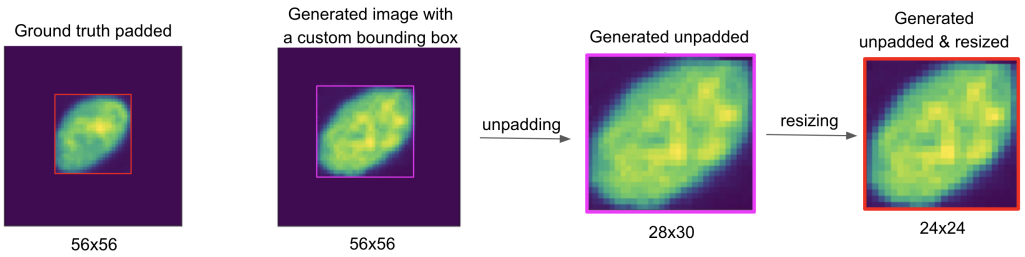
To overcome the aforementioned issue we modified the unpadding algorithm. Specifically, we have created a function to identify a personalized (custom) bounding box for each generated image instead of relying on the given bounding box coordinates features to unpad the 56 × 56 generated image. The algorithm finds the upper, lower, right and left boundaries of the generated nucleus and outputs the new bounding box region, specific



(a) Real normalized nuclei images (b) Nuclei generated by Table2Cell

Figure 26. An example of a part of a 1080×1080 fluorescent image (individual nuclei images are normalized). Image (b) illustrated the problem if cut edges of nucleus after a basic unpadding procedure. The pixel values were intensified to see more details.

to each generated nucleus image. The size of this custom bounding box may appear to be different from the one, described in the vector of features. However, it is crucial that the unpadded generated and initial individual nuclei images are of the same size. Thus, we resize the unpadded image to match the dimensions of the initial bounding box. An illustration of this procedure is shown in Figure 27, where the initial bounding box is shown in red, and the custom bounding box of a generated image is shown in purple.



(a) Ground truth nuclei (b) Generated nuclei

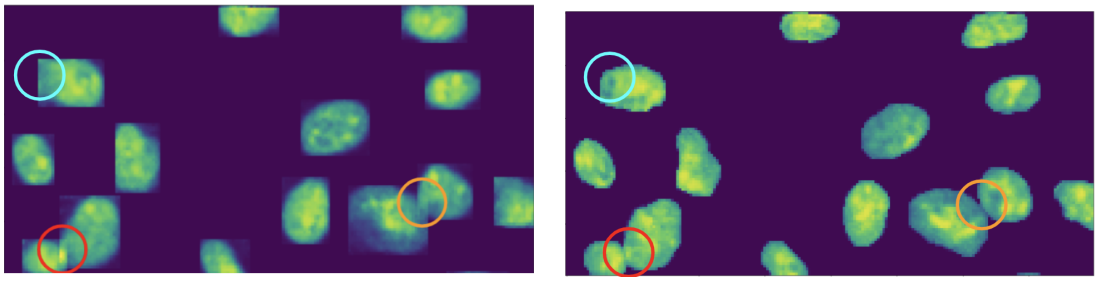
Figure 27. An example of how modified unpadding procedure is applied to a generated image. Instead of using the ground truth bounding box (shown in red) we find a custom bounding box for a generated image and apply this box to remove zero-padded pixels outside of it. The unpadded image is then resized to the shape of the ground truth bounding box described in the correspoding features.

This modification allows us to preserve the oval shape of the generated nuclei without the need to cut its edges, as well as results in images that have the correct image size as described in the bounding box features (Figure 28).

Overall, the effect of modified preprocessing with background cleaning and postprocessing steps impact on generated images is shows in Figure 28.

Table 3. Nuclei image generation results on PerkinElmer 7CL Fluorescent dataset with background filtering algorithm usage and modified unpadding postprocessing

Model	MSE	SSIM	CS
Deep convolutional decoder	0.0277	0.6821	0.535
Table2Cell	0.0131	0.7940	0.991



(a) Regular unpadding (b) Modified unpadding

Figure 28. An example of a part of a 1080×1080 fluorescent image (individual nuclei images are normalized). Image (a) illustrats the problem if cut edges of nucleus after a basic unpadding procedure as well as the presence of background. Image (b) shows the results after a modified unpadding procedure was applied to zero-background generated images. The pixel values were intensified to see more details.

5.4 Investigating the Responsiveness of Table2Cell Model to Variations in Input Parameters

To evaluate the responsiveness of the Table2Cell model to variations in its input parameters, we conducted several experiments involving the artificial modification of a single feature within the input vector of features. Our first objective was to observe changes in the generated image resulting from alterations to a given feature. We also wanted to be sure that changes in one feature do not majorly affect the other features of a generated nucleus.

We selected area feature for modification as it can be easily calculated and evaluated by us. It should be noted that the majority of features are not completely independent of each other, e.g. increasing an area of a nucleus one should expect for its area to increase as well. Therefore, we anticipated that changes to one feature might result in incidental changes to closely related features.

Figure 29 demonstrates the idea behind a single feature artificial modification. Data exploration analysis showed that the majority of nuclei' area lies in the range of [100; 3100] pixels^2. However, in theory the maximum area value can be $56 \times 56 = 3136$ (perfectly squared-shape nuclei). Hence we interpolated the area values from 100 to 3100 with the step of 100 pixels^2. The rest of the features remained the same. Next we

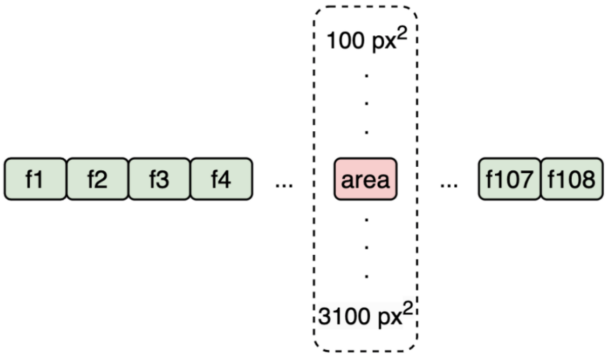


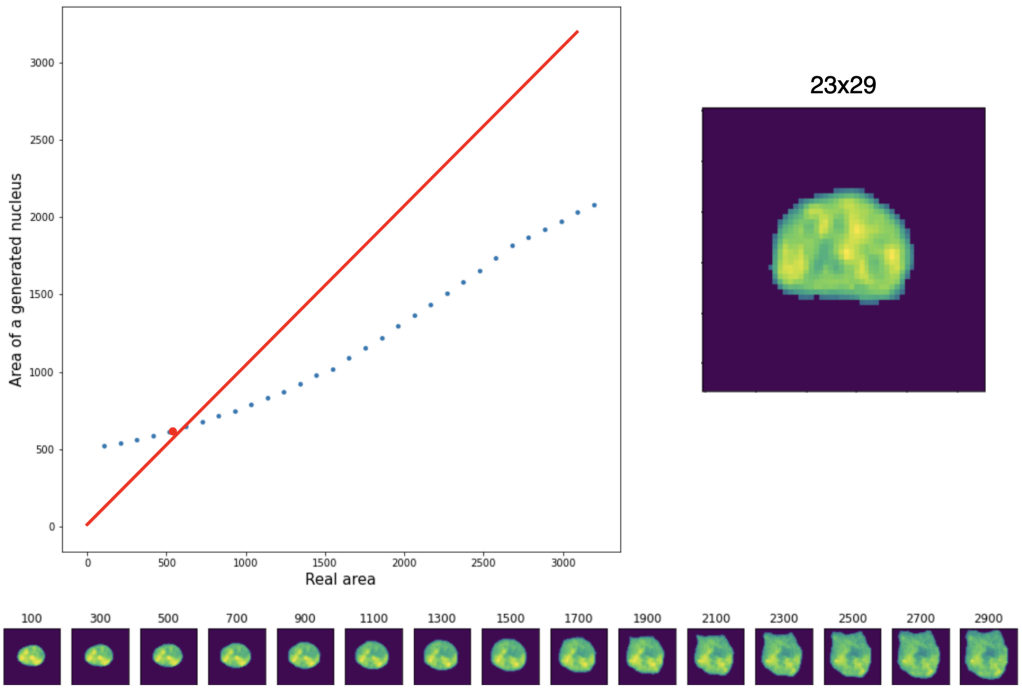
Figure 29. An example of an artificial modification of a single feature (area) within the input vector of features. We examined the effects of varying area, from a a chosen minimum value of 100 px^2 to a maximum value of 3100.

generated nuclei from the artificially modified vectors of features.

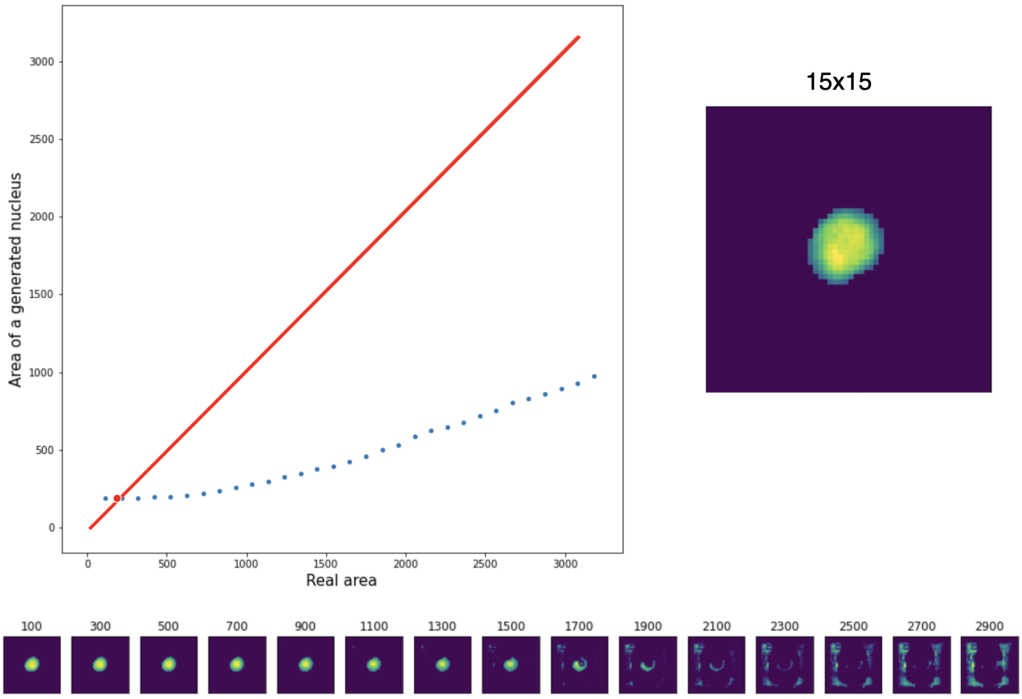
The results of the experiment are demonstrated in Figure 30. The graph shows the relationship between the real area, which was set by us (we call it *real* area in the graph), and the area measured from a generated nucleus: red line shows an ideal case when area of a generated nucleus is the same as the area we were anchoring it with; blue points present the results of experiments. Figure 30a clearly demonstrates the model's ability to respond to variation within a single feature, as evidenced by the gradual increase of the blue points. However, the model struggles to accurately predict the value of the area feature, as indicated by the deviation of the blue points from the red line. We hypothesised that this deviation is caused by other size-related features, such as perimeter and axis length, which remained the same.

In some cases, when a nucleus is very small (Figure 30b), the model is not able to generate a nucleus starting from a certain threshold area value. The likely reason for this is that we only change the area value and when it becomes quite high it contradicts a low perimeter value.

It was important to test a hypothesis that increasing the value of other non-size-related features does not result in an increase in image size (area). In other words, we wanted to see if the model can generate a nucleus of a big size because of a big non-size-related feature value. For this experiment, we chose 3 non-size-related features that weakly correlate with size-related ones, such as 'Garbor Max 2px', 'SER Edge 2px' and 'Symmetry 02 SER-Edge'. We have gradually increased their values while area values remained the same.



(a) An example of a nucleus of a median size



(b) An example of a nucleus of a small size

Figure 30. The results of an experiment, which tests the Table2Cell model responsiveness to variations in its input parameters. The graph shows the relationship between the real area and the area measured from a generated nucleus. The bottom images were generated from the artificially modified area with a step of 200 pixels^2.

Figure 31 shows an example of artificial change of each non-size-related feature. In more detail, it shows, that the area did not change because of the increased value of other non-size related features. It proved that the Table2Cell model is able to generate the correct predictions feature-wise depending on changes in the vector of features.

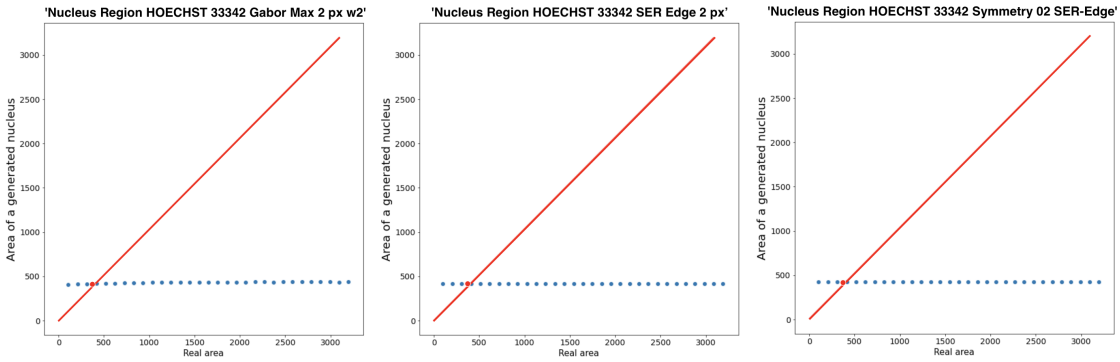


Figure 31. The results of 3 experiments, in which only a single non-size related feature was changed. The area of a nucleus did not change because of the increased value of any of a non-size related feature.

5.5 Evaluating the model performance by estimating the original features

Section 4.3 mentions that it is possible to do a reverse process to image generation and extract the features from a generated image. The section also explains how a cosine similarity index can be calculated between a real nucleus image and a generated one. Here we will elaborate more on this procedure and present another way of using CS index for understanding the results.

We measure the cosine similarity score in two ways:

1. Cosine similarity between the initial vector of features and a vector of features extracted from a generated nucleus image. In both cases only the selected 108 features are compared. Figure 32 demonstrates the process of cosine similarity index calculation between two vectors of features.

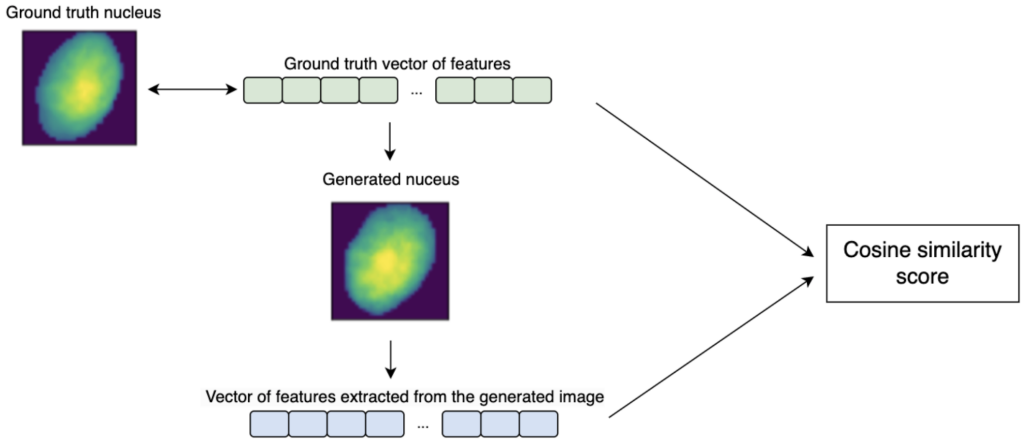


Figure 32. The process of cosine similarity index calculation between the initial vector of features and a vector of features extracted from a generated nucleus image. In both cases only the selected 108 features are compared.

We have calculated the cosine similarity between 500 randomly selected generated nuclei from an 1080×1080 image presented in Figure 21c and their corresponding real nuclei images. The most of the similarity scores of generated image by Table2Cell were close to 1, with the least similar nuclei having a CS score of 0.94, indicating that the feature values of the generated images were highly similar to the real feature values. However, when the CS value was lower, it was difficult to identify which poorly generated feature led to the low CS score. Therefore, to gain a better understanding of the results, we computed the CS score using the following approach:

2. Calculating cosine similarity for a single feature across all nuclei.

To do that we stack all vectors of features as a table shown in Figure 33 for both real nuclei image and generated images. Now instead of selecting 2 rows (vectors of 108 features) from each table for CS calculation, we select a column, which corresponds to a certain feature. At the end 108 CS scores were calculated.

We divided all features into feature classes, such as direction, morphology, intensity, symmetry, etc. and calculated the mean CS score within each feature class and visualized the results in Figure 34.

Based on the analysis presented in the Figure 34, it can be concluded that the model was found to have a better understanding of size-related and morphology features as compared to textural features. However, it is currently unclear why this phenomenon has occurred, as we did not do a further investigation to find the underlying reasons.

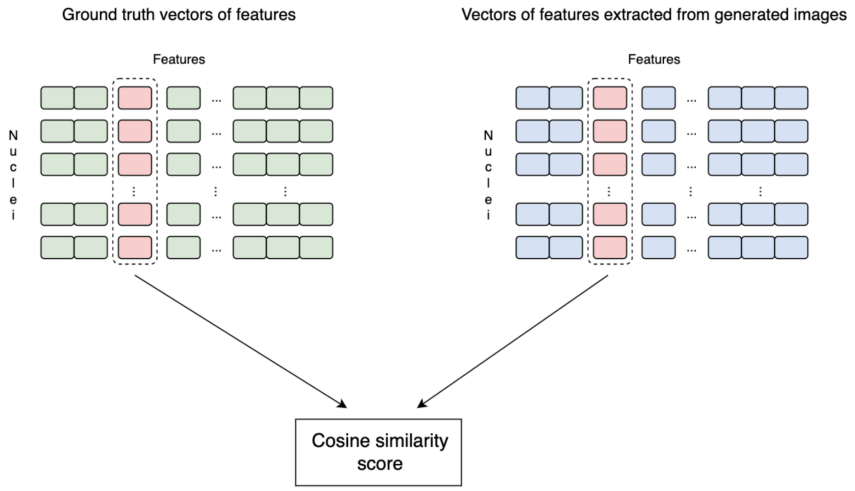


Figure 33. The process of cosine similarity index calculation for a single feature across a nuclei set: real feature values are taken from a real vector of features and a generated feature value is taken from a vector of features extracted from a generated nucleus image.

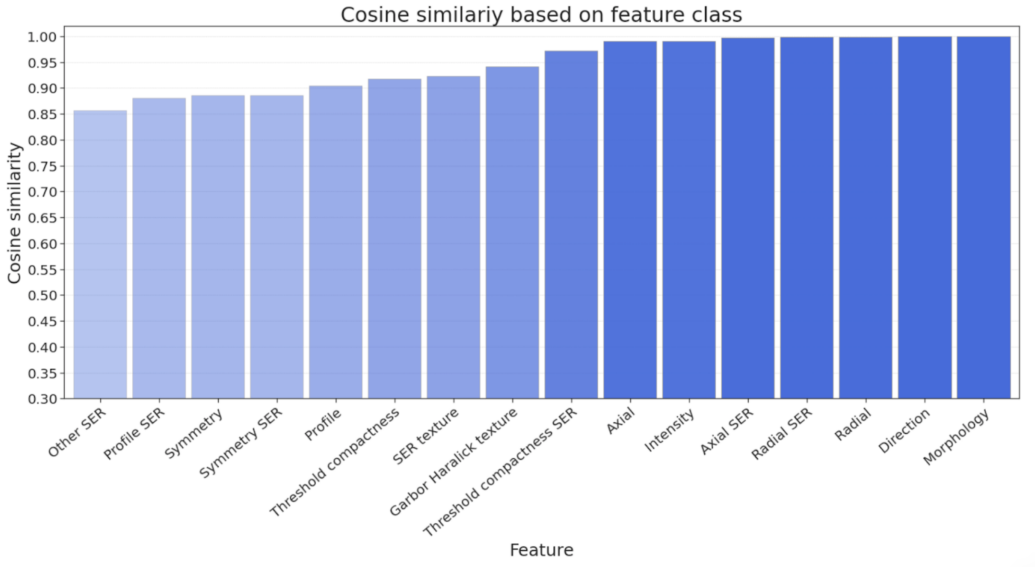


Figure 34. The results for cosine similarity index for groups of features calculated on test set by a Table2Cell model. The model has better understanding of size-related and morphology features as compared to textural features.

5.6 Feature reduction for high quality nuclei generation

One of the goals of this research is to investigate feature selection for data compression of vector of features, with the objective of generating high-quality cell images. The 108 features were sorted from least to most correlated, and each feature was removed one-by-one to train the Table2Cell model on 107, 106, ... , 1 feature. Hence we obtained 108 models which were able to generate images from any number of features, starting from 1 (which is clearly not useful) and ending with the maximum available number. The goal was to determine the number of features that could be removed while maintaining image quality, as if all 108 features were used.

The results of this experiment are presented in Figure 35. The figure shows 5 different generated images of nuclei using 1, 10, 20, ..., 90, 100 and 108 features. It can be seen than from a certain number of features the image quality does not change.

To understand the exact number of features to be removed with no effect on similarity with real image we have calculated SSIM and MSE scores for all test images. Figure 36 demonstrates the results for 11 selected numbers of features for a better vizualization in this thesis. We have also separated the results for big (bounding box is bigger than 40×40) and small images (bounding box is smaller than 15×15) for a more meaningful insights.

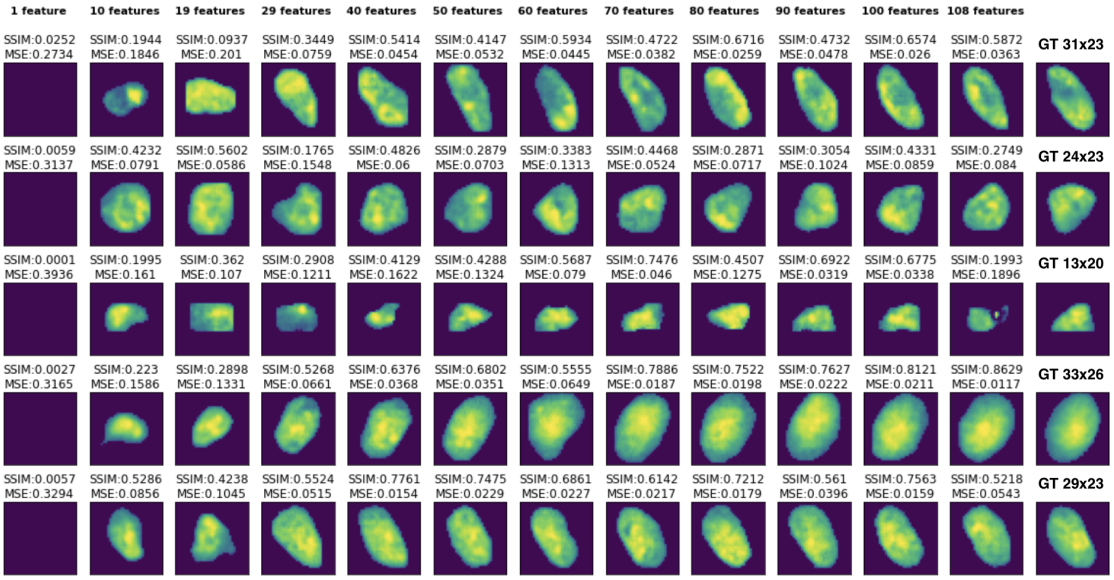
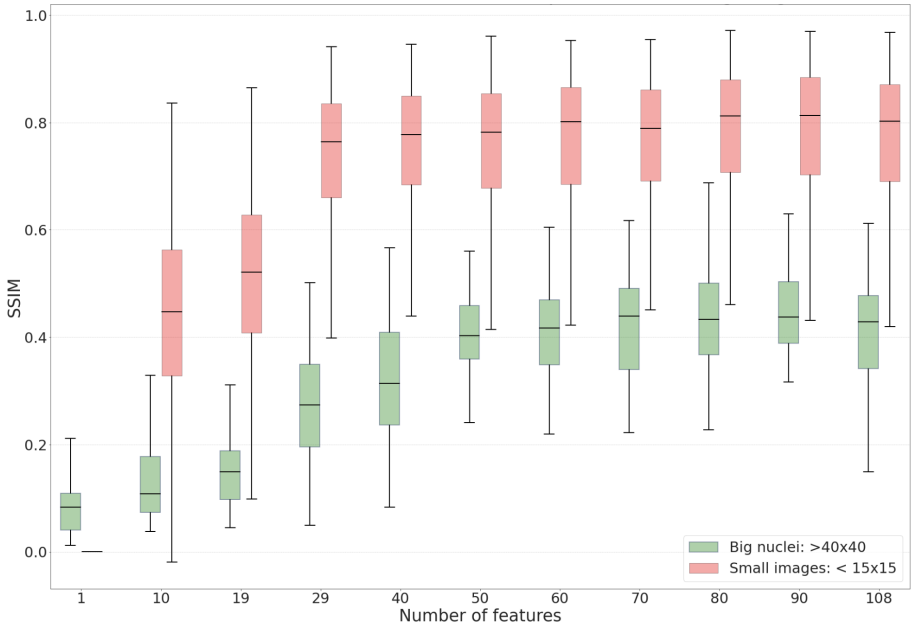


Figure 35. Examples of nulcei images generation using a different amount of features (from left to right) by a Table2Cell model. SSIM and MSE metrics were calculated on the unpadded images. The rightmost images show the groud truth along with the size of an actual nucleus bounding box.

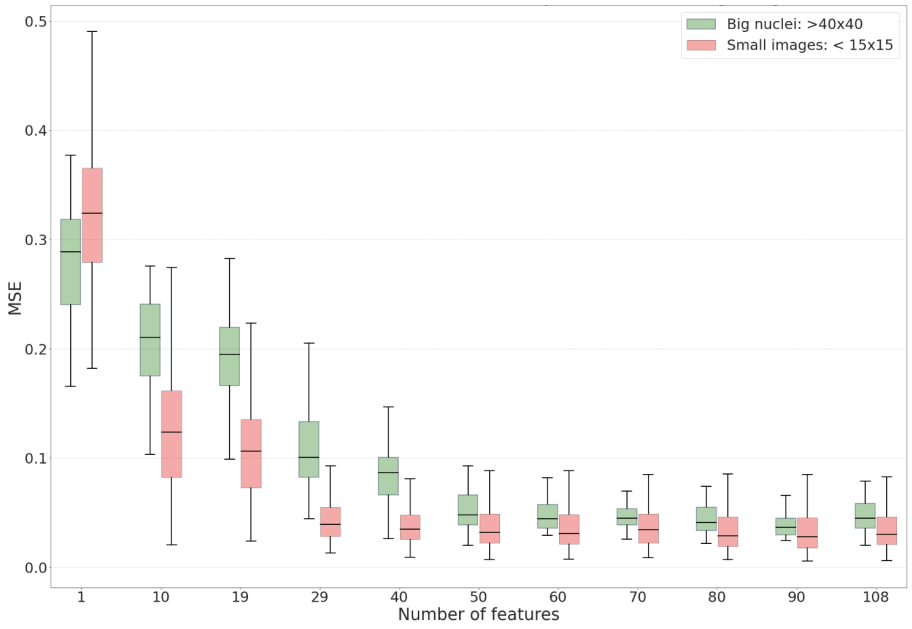
The present analysis involved integrating the observations obtained from Figure 36a

and Figure 36b, revealing a noticeable difference in the SSIM score between the distinct categories of small and large nuclei. There may be two reasons for this. First, MSE is not a very accurate metric for measuring images, hence it may not have captured intricate details added with a new feature included in generation. Another reason is that most likely, smaller nuclei have less complex textural characteristics when compared to larger nuclei.

From this analysis we conclude that Table2Cell needs 63 features (58%) to generate images that are visually similar to the real nuclei images and have a high similarity score in terms of their vectors of features.



(a)



(b)

Figure 36. Table2Cell performance using a different number of features. Images of big and small nuclei size were evaluated separately.

6 Conclusion

The advancement of deep learning methods in computer vision has significantly improved the analysis of microscopy images. Biologists can extract many features from images of cells, including their characteristics. The feature selection process can reduce the feature data's dimensionality, making it more manageable for analysis and modelling. In the case of nuclei image analysis, selecting the most valuable features can be challenging as different features may have varying levels of importance for different tasks.

We have proposed a method that can be used to analyze a selected subset of features based on generating nuclei images that accurately represent the variations in their features. By comparing the generated images to the corresponding real images, one can examine whether a specific feature contributed to generating high-quality nuclei images. To achieve this, we have designed and trained a Table2Cell model that takes a vector of features as input and generates an image of a corresponding nucleus. Our experimental results demonstrate that the proposed Table2Cell model can generate high-quality nuclei images that capture variations in size, shape, and other morphological features. We have also shown that the model can learn to generate nuclei images corresponding to specific feature values and generalize well to new feature combinations. We have conducted extensive experiments to evaluate the model's performance and analyze its strengths and weaknesses. Finally, we showed that it is possible to generate high-detailed images by using only 58% of features, provided to us for this research.

There are many ways to further develop the proposed solution, for example, by designing a more complex neural network architecture. In addition, the main advantage of Table2Cell is the ability to work with any number of features; therefore, more experiments with a vast number of features may be conducted. We conclude that the present solution will serve as a baseline for further research and development that will bring this technology into the various pipelines.

7 Acknowledgements

This work was funded by PerkinElmer, Inc. and the Estonian Ministry of Foreign Affairs Development Cooperation and Humanitarian Aid funds. The computational resources were provided by the High Performance Computing (HPC) Center of the University of Tartu. I want to thank Kaupo Palo for his idea to start this project and his support of the Bioiformatics Computer Vision Lab. I am also beyond grateful to my supervisor Dmytro Fishman for his patience, motivation, brave ideas and endless care. Finally, I would like to acknowledge my research group colleagues - Denys Kaliuhnyi, Mikhail Papkov, Olavi Ollikainen, Mohammed Ali, Leopold Parts for their support.

References

[1] Grammarly. https://www.grammarly.com/.

[2] Openai. https://openai.com/blog/chatgpt.

[3] Scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[4] Khaled Alrfou, Amir Kordijazi, and Tian Zhao. Computer vision methods for the microstructural analysis of materials: The state-of-the-art and future perspectives, 2022.

[5] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2020.

[6] Omid Bazgir, Ruibo Zhang, Saugato Rahman Dhruba, Raziur Rahman, Souparno Ghosh, and Ranadip Pal. Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks. *Nature Communications*, 11(1), September 2020.

[7] Anil K. Bharodiya. Feature extraction methods for CT-scan images using image processing. In *Computed-Tomography (CT) Scan*. IntechOpen, May 2022.

[8] Britannica. Dictionary. https://www.britannica.com/science/homeostasis.

[9] Jonas Bucevičius, Gražvydas Lukinavičius, and Rūta Gerasimaitė. The use of hoechst dyes for DNA staining and beyond. *Chemosensors*, 6(2):18, April 2018.

[10] Lei Cai, Hongyang Gao, and Shuiwang Ji. Multi-stage variational auto-encoders for coarse-to-fine image generation, 2017.

[11] Juan C. Caicedo, Jonathan Roth, Allen Goodman, Tim Becker, Kyle W. Karhohs, Matthieu Broisin, Csaba Molnar, Claire McQuin, Shantanu Singh, Fabian J. Theis, and Anne E. Carpenter. Evaluation of deep learning strategies for nucleus segmentation in fluorescence images. *Cytometry Part A*, 95(9):952–965, July 2019.

[12] Francesco Paolo Casale, Adrian V Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian process prior variational autoencoders, 2018.

[13] Jaehyun Cho, Luke L. Hsiung, Robert E. Rudd, and Sylvie Aubry. A novel mechanism for the formation of dislocation cell patterns in bcc metal, 2023.

[14] Chiranji Lal Chowdhary and D.P. Acharjya. Segmentation and feature extraction in medical imaging: A systematic review. *Procedia Computer Science*, 167:26–36, 2020.

[15] Romain Cosentino, Randall Balestriero, Richard Baraniuk, and Behnaam Aazhang. Deep autoencoders: From understanding to generalization guarantees. 2020.

[16] David S. Goldstein and Irwin J. Kopin. Homeostatic systems, biocybernetics, and autonomic neuroscience. *Autonomic Neuroscience*, 208:15–28, December 2017.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[18] Developers Google. Gan. loss functions. `https://developers.google.com/machine-learning/gan/loss?hl`.

[19] Vagisha Gupta, Shelly Sachdeva, and Neha Dohare. Deep similarity learning for disease prediction. In *Trends in Deep Learning Methodologies*, pages 183–206. Elsevier, 2021.

[20] https://juliaimages.org/. Structural similarity index. `https://juliaimages.org/v0.22/democards/examples/image_quality_and_benchmarks/structural_similarity_index/`.

[21] He Huang, Philip S. Yu, and Changhu Wang. An introduction to image synthesis with generative adversarial nets, 2018.

[22] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, August 2009.

[23] Yifan Jiang, Han Chen, Murray Loew, and Hanseok Ko. Covid-19 ct image synthesis with a conditional generative adversarial network, 2020.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[25] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. 2019.

[26] Kaikai Liu, Renjun Shuai, Li Ma, and ZeXu. Cells image generation method based on VAE-SGAN. *Procedia Computer Science*, 183:589–595, 2021.

[27] Shiyong Ma and Zhen Zhang. Omicsmapnet: Transforming omics data to take advantage of deep convolutional neural network for discovery, 2018.

[28] Rafael Elias Marques, Pedro Elias Marques, Rodrigo Guabiraba, and Mauro Martins Teixeira. Exploring the homeostatic and sensory roles of the immune system. *Frontiers in Immunology*, 7, March 2016.

[29] Rafael Elias Marques, Pedro Elias Marques, Rodrigo Guabiraba, and Mauro Martins Teixeira. Exploring the homeostatic and sensory roles of the immune system. *Frontiers in Immunology*, 7, March 2016.

[30] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[31] Erick Moen, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. Deep learning for cellular image analysis. *Nature Methods*, 16(12):1233–1246, May 2019.

[32] Fabrizio A. Pennacchio, Paulina Nastały, Alessandro Poli, and Paolo Maiuri. Tailoring cellular function: The contribution of the nucleus in mechanotransduction. *Frontiers in Bioengineering and Biotechnology*, 8, January 2021.

[33] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[34] Shankarachary Ragi, Md Hafizur Rahman, Jamison Duckworth, Kalimuthu Jawaharraj, Parvathi Chundi, and Venkataramana Gadhamshetty. Artificial intelligence-driven image analysis of bacterial cells and biofilms, 2021.

[35] Michael J. Sanderson, Ian Smith, Ian Parker, and Martin D. Bootman. Fluorescence microscopy. *Cold Spring Harbor Protocols*, 2014(10):pdb.top071795, October 2014.

[36] Michael J. Sanderson, Ian Smith, Ian Parker, and Martin D. Bootman. Fluorescence microscopy. *Cold Spring Harbor Protocols*, 2014(10):pdb.top071795, October 2014.

[37] Gail M. Seigel and Lorrie M. Campbell. High-throughput microtiter assay for hoechst 33342 dye uptake. *Cytotechnology*, 45(3):155–160, July 2004.

[38] Alok Sharma, Edwin Vans, Daichi Shigemizu, Keith A. Boroevich, and Tatsuhiko Tsunoda. DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific Reports*, 9(1), August 2019.

[39] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, 2006.

[40] Ben Shneiderman. Tree visualization with tree-maps. *ACM Transactions on Graphics*, 11(1):92–99, January 1992.

[41] Roman Barbara Soledad, Steenbergen Charles, and Das Samarjit. The secret messages between mitochondria and nucleus in muscle cell biology. *Archives of Biochemistry and Biophysics*, 666:52–62, May 2019.

[42] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9(nov):2579–2605, 2008. Pagination: 27.

[43] 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, April 2004.

[44] Liwen Yao, Jun Zhang, Jun Liu, Liangru Zhu, Xiangwu Ding, Di Chen, Huiling Wu, Zihua Lu, Wei Zhou, Lihui Zhang, Bo Xu, Shan Hu, Biqing Zheng, Yanning Yang, and Honggang Yu. A deep learning-based system for bile duct annotation and station recognition in linear endoscopic ultrasound. *EBioMedicine*, 65:103238, March 2021.

[45] Weili Zeng. How to construct energy for images? denoising autoencoder can be energy based model, 2023.

[46] Monika Zwerger, Chin Yee Ho, and Jan Lammerding. Nuclear mechanics in disease. *Annual Review of Biomedical Engineering*, 13(1):397–428, aug 2011.

Appendix

I. Usage of ChatGPT language model in academic writing

In the course of our research, we employed a natural language processing model, namely ChatGPT [2], to facilitate the academic writing process for this thesis. Specifically, ChatGPT was used to generate novel ideas and suggest alternative phrasings for sentences and paragraphs. Additionally, it was employed to answer our inquiries related to the thesis, however the responses provided by the model were verified against academic literature. Similar to the functionality of Grammarly [1], ChatGPT served to proofread and identify errors in grammar and syntax.

II. Supplementary table

Table 4. A full list of given features

1.	Row
2.	Column
3.	Field
4.	Plane
5.	Timepoint
6.	Object Number
7.	X
8.	Y
9.	Bounding Box X1
10.	Bounding Box Y1
11.	Bounding Box X2
12.	Bounding Box Y2
13.	Position X [µm]
14.	Position Y [µm]
15.	Intensity Nucleus Region HOECHST 33342 Mean
16.	Intensity Nucleus Region HOECHST 33342 StdDev
17.	Intensity Nucleus Region HOECHST 33342 Median
18.	Intensity Nucleus Region HOECHST 33342 Maximum
19.	Intensity Nucleus Region HOECHST 33342 Minimum
20.	Intensity Nucleus Region HOECHST 33342 Sum
21.	Intensity Nucleus Region HOECHST 33342 CV [%]
22.	Intensity Nucleus Region HOECHST 33342 Quantile 90%

23.	Intensity Nucleus Region HOECHST 33342 Contrast
24.	Nucleus Region Area [px²]
25.	Nucleus Region Roundness
26.	Nucleus Region Perimeter [px]
27.	Nucleus Region Width [px]
28.	Nucleus Region Length [px]
29.	Nucleus Region Ratio Width to Length
30.	Nucleus Region Length of Ellipse Longest Axis
31.	Nucleus Region Length of Ellipse Shortest Axis
32.	Nucleus Region Ellipse Short/Long Axis Ratio
33.	Nucleus Region Ellipse Longest Axis x-Projection
34.	Nucleus Region Ellipse Longest Axis y-Projection
35.	Nucleus Region Ellipse Shortest Axis x-Projection
36.	Nucleus Region Ellipse Shortest Axis y-Projection
37.	Nucleus Region HOECHST 33342 Length of Ellipse Longest Axis
38.	Nucleus Region HOECHST 33342 Length of Ellipse Shortest Axis
39.	Nucleus Region HOECHST 33342 Ellipse Short/Long Axis Ratio
40.	Nucleus Region HOECHST 33342 Ellipse Longest Axis x-Projection
41.	Nucleus Region HOECHST 33342 Ellipse Longest Axis y-Projection
42.	Nucleus Region HOECHST 33342 Ellipse Shortest Axis x-Projection
43.	Nucleus Region HOECHST 33342 Ellipse Shortest Axis y-Projection
44.	Nucleus Region HOECHST 33342 Symmetry 02
45.	Nucleus Region HOECHST 33342 Symmetry 03
46.	Nucleus Region HOECHST 33342 Symmetry 04
47.	Nucleus Region HOECHST 33342 Symmetry 05
48.	Nucleus Region HOECHST 33342 Symmetry 12
49.	Nucleus Region HOECHST 33342 Symmetry 13
50.	Nucleus Region HOECHST 33342 Symmetry 14
51.	Nucleus Region HOECHST 33342 Symmetry 15
52.	Nucleus Region HOECHST 33342 Threshold Compactness 30%
53.	Nucleus Region HOECHST 33342 Threshold Compactness 40%
54.	Nucleus Region HOECHST 33342 Threshold Compactness 50%
55.	Nucleus Region HOECHST 33342 Threshold Compactness 60%
56.	Nucleus Region HOECHST 33342 Axial Small Length
57.	Nucleus Region HOECHST 33342 Axial Length Ratio
58.	Nucleus Region HOECHST 33342 Radial Mean
59.	Nucleus Region HOECHST 33342 Radial Relative Deviation
60.	Nucleus Region HOECHST 33342 Profile 1/2

61.	Nucleus Region HOECHST 33342 Profile 2/2
62.	Nucleus Region HOECHST 33342 Symmetry 02 SER-Dark
63.	Nucleus Region HOECHST 33342 Symmetry 03 SER-Dark
64.	Nucleus Region HOECHST 33342 Symmetry 04 SER-Dark
65.	Nucleus Region HOECHST 33342 Symmetry 05 SER-Dark
66.	Nucleus Region HOECHST 33342 Symmetry 12 SER-Dark
67.	Nucleus Region HOECHST 33342 Symmetry 13 SER-Dark
68.	Nucleus Region HOECHST 33342 Symmetry 14 SER-Dark
69.	Nucleus Region HOECHST 33342 Symmetry 15 SER-Dark
70.	Nucleus Region HOECHST 33342 Threshold Compactness 30% SER-Dark
71.	Nucleus Region HOECHST 33342 Threshold Compactness 40% SER-Dark
72.	Nucleus Region HOECHST 33342 Threshold Compactness 50% SER-Dark
73.	Nucleus Region HOECHST 33342 Threshold Compactness 60% SER-Dark
74.	Nucleus Region HOECHST 33342 Axial Small Length SER-Dark
75.	Nucleus Region HOECHST 33342 Axial Length Ratio SER-Dark
76.	Nucleus Region HOECHST 33342 Radial Mean SER-Dark
77.	Nucleus Region HOECHST 33342 Radial Relative Deviation SER-Dark
78.	Nucleus Region HOECHST 33342 Radial Mean Ratio SER-Dark
79.	Nucleus Region HOECHST 33342 Profile 1/2 SER-Dark
80.	Nucleus Region HOECHST 33342 Profile 2/2 SER-Dark
81.	Nucleus Region HOECHST 33342 Symmetry 02 SER-Ridge
82.	Nucleus Region HOECHST 33342 Symmetry 03 SER-Ridge
83.	Nucleus Region HOECHST 33342 Symmetry 04 SER-Ridge
84.	Nucleus Region HOECHST 33342 Symmetry 05 SER-Ridge
85.	Nucleus Region HOECHST 33342 Symmetry 12 SER-Ridge
86.	Nucleus Region HOECHST 33342 Symmetry 13 SER-Ridge
87.	Nucleus Region HOECHST 33342 Symmetry 14 SER-Ridge
88.	Nucleus Region HOECHST 33342 Symmetry 15 SER-Ridge
89.	Nucleus Region HOECHST 33342 Threshold Compactness 30% SER-Ridge
90.	Nucleus Region HOECHST 33342 Threshold Compactness 40% SER-Ridge
91.	Nucleus Region HOECHST 33342 Threshold Compactness 50% SER-Ridge

92.	Nucleus Region HOECHST 33342 Threshold Compactness 60% SER-Ridge
93.	Nucleus Region HOECHST 33342 Axial Small Length SER-Ridge
94.	Nucleus Region HOECHST 33342 Axial Length Ratio SER-Ridge
95.	Nucleus Region HOECHST 33342 Radial Mean SER-Ridge
96.	Nucleus Region HOECHST 33342 Radial Relative Deviation SER-Ridge
97.	Nucleus Region HOECHST 33342 Radial Mean Ratio SER-Ridge
98.	Nucleus Region HOECHST 33342 Profile 1/2 SER-Ridge
99.	Nucleus Region HOECHST 33342 Profile 2/2 SER-Ridge
100.	Nucleus Region HOECHST 33342 Symmetry 02 SER-Spot
101.	Nucleus Region HOECHST 33342 Symmetry 03 SER-Spot
102.	Nucleus Region HOECHST 33342 Symmetry 04 SER-Spot
103.	Nucleus Region HOECHST 33342 Symmetry 05 SER-Spot
104.	Nucleus Region HOECHST 33342 Symmetry 12 SER-Spot
105.	Nucleus Region HOECHST 33342 Symmetry 13 SER-Spot
106.	Nucleus Region HOECHST 33342 Symmetry 14 SER-Spot
107.	Nucleus Region HOECHST 33342 Symmetry 15 SER-Spot
108.	Nucleus Region HOECHST 33342 Threshold Compactness 30% SER-Spot
109.	Nucleus Region HOECHST 33342 Threshold Compactness 40% SER-Spot
110.	Nucleus Region HOECHST 33342 Threshold Compactness 50% SER-Spot
111.	Nucleus Region HOECHST 33342 Threshold Compactness 60% SER-Spot
112.	Nucleus Region HOECHST 33342 Axial Small Length SER-Spot
113.	Nucleus Region HOECHST 33342 Axial Length Ratio SER-Spot
114.	Nucleus Region HOECHST 33342 Radial Mean SER-Spot
115.	Nucleus Region HOECHST 33342 Radial Relative Deviation SER-Spot
116.	Nucleus Region HOECHST 33342 Radial Mean Ratio SER-Spot
117.	Nucleus Region HOECHST 33342 Profile 1/2 SER-Spot
118.	Nucleus Region HOECHST 33342 Profile 2/2 SER-Spot
119.	Nucleus Region HOECHST 33342 Symmetry 02 SER-Edge
120.	Nucleus Region HOECHST 33342 Symmetry 03 SER-Edge
121.	Nucleus Region HOECHST 33342 Symmetry 04 SER-Edge
122.	Nucleus Region HOECHST 33342 Symmetry 05 SER-Edge
123.	Nucleus Region HOECHST 33342 Symmetry 12 SER-Edge
124.	Nucleus Region HOECHST 33342 Symmetry 13 SER-Edge
125.	Nucleus Region HOECHST 33342 Symmetry 14 SER-Edge
126.	Nucleus Region HOECHST 33342 Symmetry 15 SER-Edge
127.	Nucleus Region HOECHST 33342 Threshold Compactness 30% SER-Edge

128.	Nucleus Region HOECHST 33342 Threshold Compactness 40% SER-Edge
129.	Nucleus Region HOECHST 33342 Threshold Compactness 50% SER-Edge
130.	Nucleus Region HOECHST 33342 Threshold Compactness 60% SER-Edge
131.	Nucleus Region HOECHST 33342 Axial Small Length SER-Edge
132.	Nucleus Region HOECHST 33342 Axial Length Ratio SER-Edge
133.	Nucleus Region HOECHST 33342 Radial Mean SER-Edge
134.	Nucleus Region HOECHST 33342 Radial Relative Deviation SER-Edge
135.	Nucleus Region HOECHST 33342 Radial Mean Ratio SER-Edge
136.	Nucleus Region HOECHST 33342 Profile 1/2 SER-Edge
137.	Nucleus Region HOECHST 33342 Profile 2/2 SER-Edge
138.	Nucleus Region HOECHST 33342 SER Spot 2 px
139.	Nucleus Region HOECHST 33342 SER Hole 2 px
140.	Nucleus Region HOECHST 33342 SER Edge 2 px
141.	Nucleus Region HOECHST 33342 SER Ridge 2 px
142.	Nucleus Region HOECHST 33342 SER Valley 2 px
143.	Nucleus Region HOECHST 33342 SER Saddle 2 px
144.	Nucleus Region HOECHST 33342 SER Bright 2 px
145.	Nucleus Region HOECHST 33342 SER Dark 2 px
146.	Nucleus Region HOECHST 33342 Haralick Correlation 2 px
147.	Nucleus Region HOECHST 33342 Haralick Contrast 2 px
148.	Nucleus Region HOECHST 33342 Haralick Sum Variance 2 px
149.	Nucleus Region HOECHST 33342 Haralick Homogeneity 2 px
150.	Nucleus Region HOECHST 33342 Gabor Min 2 px w2
151.	Nucleus Region HOECHST 33342 Gabor Max 2 px w2

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Anhelina Lohvina**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Table2Cell: generating realistic nuclei images from numeric properties for data compression,
   *(*title of thesis)

   supervised by Dmytro Fishman, PhD.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Anhelina Lohvina
09/05/2023