Journal of Theoretical and Applied Information Technology
15th October 2023. Vol.101. No 19
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IDENTIFICATION OF CELL MEMBRANES IN 2D IMAGES USING COMPUTER VISION

PAUL D. SARAVIA-VELASQUEZ[1], ROXANA FLORES-QUISPE[2], YUBER VELAZCO-PAREDES[3]

[1,2,3] Universidad Nacional de San Agustín de Arequipa

E-mail: [1]psaravia@unsa.edu.pe, [2]rfloresqu@unsa.edu.pe, [3]yvelazco@unsa.edu.pe

ABSTRACT

Today in many types of research the cellular structure has been studied to identify diseases or to do clinical diagnostics; of fungi and epithelial cells, because the increase in invasive fungal infections in recent years, especially in immunocompromised patients, has prompted the search for new antifungal agents with greater efficacy. Where the fungal cell membrane is enriched with various lipids belonging to the class of glycerophospholipids, sphingolipids, and sterols. In addition, the precise roles of membrane lipids in the organization of these membrane domains in epithelial cells, where they are polarized and maintain apical and basolateral membranes, are largely unknown. These epithelial cells have morphologically distinct membrane structures with specific functions. This research presents a significant contribution to achieving the precise identification of cell membranes in 2D images based on Inception-type CNN architecture specifically designed for this research context, allowing effective detection of cell membranes. In addition, Canny, Thresholding, and Gaussian noise filters were implemented to improve the edge detection extracting relevant information from the object of interest and reducing unnecessary elements in the image. In addition, other segmentation techniques and size adjustments were used to improve the variability present in the images. Finally, the experimental results show the best performance with an accuracy rate of 86.6% using the Adagrad optimizer, which proved its efficiency in the search for membranes in 2D images. Our proposal, based on CNN, image processing, and adjustment techniques, offers a robust approach to this task, strengthening research in this field and opening opportunities for future improvements.

Keywords: *Computer Vision, CNN, Cell Membranes*

1. INTRODUCTION

Recent studies indicate that the cell membrane interacts with its attached cytoskeleton, which is an important regulator of the cell function, exerting and responding to external forces. This relationship is possible to investigate by looking for connections between cell membrane elastic properties, especially surface tension and bending modulus, and cell function. Those properties are measured by pulling tethers from the cell membrane with optical tweezers [1].

Likewise, the rapid dynamics of membrane–cytoskeleton adhesion are better explained by many weak bonds between membrane lipids and proteins than by a few strong bonds between membrane glycoproteins and cytoskeletal proteins [2].

In addition, the structural integrity of cellular membranes is paramount for the proper function of cells and organelles. Specific molecules must be allowed to cross the membrane at specific times, such as metabolites or ions during nerve cell propagation. Other molecules must be barred, creating concentration gradients essential as a source of energy [3].

On the other hand, the primary goal for eliminating visual inspection was to speed up the tedious processes of finding cells in mitosis, arranging the chromosome images into karyograms, and measuring and classifying features in the chromosome images for the purpose of detecting mutations and defects in the genome. Also, these studies present successful results due to the goal of a computer vision program is to learn iteratively, as opposed to linearly, about the image content. Then the complexity of the learning algorithm required will of course depend on the complexity of the analysis task or the variability between scenes [4].

For these reasons, this paper proposes a new method to identify the number of cell membranes, based on a Convolutional Neural Network – CNN, where each image is processed using a Canny filter, Thresholding, and Gaussian noise to eliminate the noise from images obtaining the cell membrane. In addition, in this research, different methods of segmentation and analysis of the structure cell, nuclei cell, and cell membranes were reviewed. The following are the main contributions of this proposed model.

- Pre-processing procedures are being used for the different nuclei cell photos, including image resizing.
- The relevant information about nuclei number, size, shape, and edges of cell membranes was obtained.
- Three different optimizers such as Adam, Adagrad, Adadelta, and image preprocessing techniques were used to get the best performer, also the architecture of the CNN was modified according to our problem.

The rest of this paper is organized as follows. In section 2, the state of the art is presented. The proposed method is presented in section 3. Experiments and results are presented in section 4. Finally, section 5 shows the conclusions of this paper.

2. RELATED WORKS

In recent years many types of research have been developed to analyze the internal structures and nuclei of fluorescent cells obtaining successful results. Some of them are shown below.

In that way, Siwei Yang et al. [5] have proposed an adaptive step length optimization scheme using a multiresolution scheme, where image segmentation is performed from the cell nucleus channel to smooth the resulting images using a Gaussian filter. In this study the main problem with cell nuclei images was that the intensity structure of different nuclei differs very much; due to an intensity-based registration scheme cannot be used directly. They have had successful results using confocal multichannel 3-D images from different FISH experiments with HeLa cells.

Likewise, Lorenz K.F et al. [6] proposed a method to segment multiphoton fluorescent microscopy images through the combination of segmentation and registration methods. The motivation for the development of this technique was the desire to analyze two types of data sets of multiphoton fluorescent microscopy images. The experimental results have demonstrated the efficacy of segmenting objects, in time series data sets as well as data sets comprised of images acquired at increasing tissue depths.

Similarly, Ben Appleton, et al. [7] proposed an image stitching method based on dynamic programming and they presented an application for automated slide acquisition for Virtual Microscopy (VM), which has several benefits over traditional light microscopy like browsed remotely. In addition, specimens do not degrade over time; and slides cannot be broken or lost. One of the central problems of automated VMs is the image acquisition stage. Slides are simply too large to be acquired as a single image, so instead, it is necessary to capture many fields of view (FOVs) before combining these into a single slide image. The results show that the proposed method produces visually superior images by reducing the number and extent of stitching artifacts.

Also, Junkang Zhang et al. [8] have proposed a deep detector for cells based on the framework of Faster R-CNN, and based on this, a Circle Scanning Algorithm (CSA) for the redetection of adhesion cells. Also, they consider that the detection and identification of cells is an important basis for further analysis of cell properties. In addition, in this work, there was enough data of separate individual cells to be used to train a deep model, but in the cases of adhesion cells lacking enough corresponding training samples, it was difficult to detect various individual cells successfully from the adhesion area. The results show that the proposed method can detect and identify all separate individual cells in an image and that the hybrid method by combining Faster R-CNN with the proposed CSA can effectively detect and identify the adhesion cells under the conditions of the limited samples of adhesion cells.

In addition, Yao Xue and Nilanjan Ray [9] proposed a convolutional neural network cell detection method that uses encoding of the output pixel space. Where the output space is the sparsely labeled pixel locations indicating cell centers. The essential idea of this method was that detectors are trained as a classifier in the image pixel space and the locations of target cells are labeled. In addition, the authors made substantial experiments on several mainstream datasets and challenging cell detection contests, where the proposed CNN and the CS framework (CNNCS) achieved competitive results (the highest or at least top-3 in terms of F1-score) compared to the state-of-the-art methods in the cell detection task.

Likewise, Sundaresh Ram et al. [10] have proposed a 3D convolutional neural network to simultaneously segment and detect cell nuclei in confocal microscopy images. The results show that the proposed method provides significantly improved detection and segmentation accuracy compared to existing algorithms.

Sato Masaya et al. [11] have proposed a semantic segmentation method of cell membrane and nucleus by improving pix2pix. In addition, this work consists of a generator and a discriminator. The information in the discriminator is used in the generator. To teach the generator how the discriminator classifies real or fake, the feature maps in the discriminator are concatenated with the encoder part in the generator. The accuracy of the cell membrane is much improved in comparison with standard pix2pix.

On the other hand, Dimauro, G et al. [12] have implemented a CNN based on three blocks. Defining seven categories to be classified, giving a label to each category, consisting of 3 main layer blocks, each containing the following sequence: convolution, activation, max-pooling, and dropout. The results obtained were classified according to the characteristics specified. Sensitivity, specificity, and accuracy were calculated for each cell type.

Also, Jun Du et al. [13] have proposed a region detection and classification method based on multi-semantic labels combined with morphological information analysis. The result showed that the mean average precision (mAP) was 66.98% and the accuracy was 91.61% by learning the depth and superficial features of cervical exfoliated cells with ResNet-101 as the backbone.

Similarly, Long, M. et al. [14] have proposed a cervical image target detection and segmentation model based on multi-scale feature fusion. Mask R-CNN's backbone network-feature Pyramid (FPN) which uses a top-down cross-layer connection. The results show that the improved model can effectively improve the accuracy, recall, and segmentation accuracy of epithelial cells, white blood cells, and fungi.

Ma, B. et al. [15] have proposed a module to generate fixed-sized feature maps of nuclei, which allows new information about nuclei will be used for classification. In addition, the cell images are captured in a fixed position, so the cell size is an important feature to determine whether a cell is abnormal. The results show that the MACD R-CNN can effectively improve the performance of abnormal cell detection.

So, as can be seen, several methods, several methods have been developed for the identification of cells in microscopic images, which classify cells by the intensity of their nuclei [5] [10], and other methods segment the cells using spatial filtering to verify which cells have a significantly higher intensity than the others [6] [8] and others locate the position of a cell by means of the magnification of fields of view [7]. Also, other methods have been proposed to achieve semantic segmentation for cell membranes and nuclei [11], in other cases new models of CNN have been developed to classify different kinds of cells [12]. In addition, some researchers have proposed methods for the detection, classification, and segmentation of regions to generate appropriate information for kinds of cells [13][14][15].

Nevertheless, our paper provides a more specific approach to identifying cell membranes, where the stage of image preprocessing was developed using the Canny, Thresholding, and Gaussian noise filters to classify cells by their number of nuclei in each image. In addition, the intensity of each nucleus will not affect the results we are looking for, since with the filters the border of each cell will be obtained. After obtaining the processed images, these are entered into the CNN to be classified depending on the number of nuclei, also different optimizers will be used to validate and compare the results.

3. PROPOSED METHOD

Figure 1 shows the representation of the proposed method to identify cell membranes in 2D images based on Computer Vision, which has six stages.

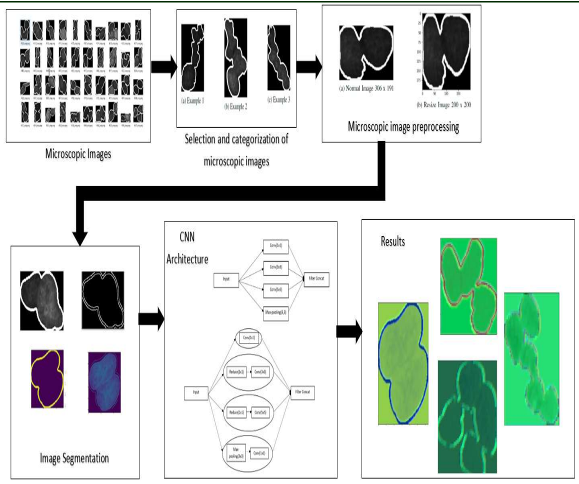
Microscopic Images
(a) Example 1
(b) Example 2
(c) Example 3
Selection and categorization of microscopic images
(a) Normal Image 306 x 191
(b) Resize Image 200 x 200
Microscopic image preprocessing
Image Segmentation
CNN Architecture
Results

Figure 1: Flowchart of the proposed method to identify cell membranes in 2D images based on computer vision

3.1 Microscopic Image

The test image dataset used in our experiments has been selected from the KAGGLE image datasets, which were used for the verification of cell membranes. This dataset has 597 images and some of them are shown in Figure 2.

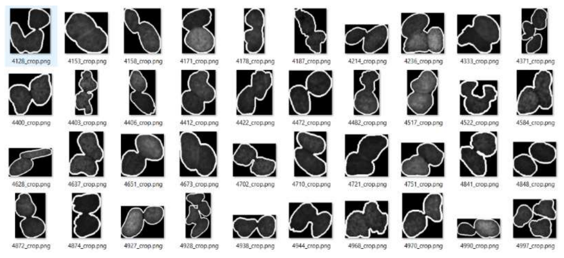

Figure 2: Microscopic Images from the KAGGLE website with different microscopic images of cells.

3.2 Selection and Categorization of microscopic image

The microscopic images in the database have been categorized into four groups considering the number of nuclei. Also, in each group, the data for training and the data for testing have been separated, which are shown in Table 1. In addition, as the research proposed by Vrigazova [16] the 70/30 proportion was used to split the dataset into training and test sets obtaining satisfactory results. However, depending on the characteristics of the data and their preliminary transformations, this proportion can differ from dataset to dataset.

Table 1: Categorization Of Microscopic Images

	Dataset		
Category	Total images	Images for training	Images for testing
2-cells	473	331	142
3-cells	95	66	29
4-cells	24	16	8
5-cells	5	4	1

In some cases, it was very difficult to identify the number of nuclei due to the complexity of the images or when the borders enclose an area that may be identified as a cell or could be only a part of it. Some of these examples are shown in Figure 3.

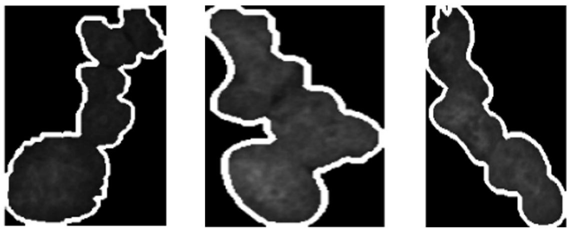

Figure 3: Complex examples to identify the category.

These images belong to the most complex examples to identify the category of each image because it is not possible to find easily the number of nuclei.

For that reason, to find the number of nuclei, the next specifications have been considered:

- The size of the cell image.
- The edge of the cell membrane to identify the place of closure or proximity of another nucleus.
- Segmenting the image by assigning a grayscale value to each pixel based on its brightness and ignoring other color information.

3.3 Microscopic image preprocessing

In the Dataset, each image has a different size, so it was difficult to process them. Therefore, it was necessary to find the best range, which would be the closest size where the input images do not lose their unique features and it was also important to consider that not all images had the same number of cells.

For that reason, all the images have been resized to the dimensions 150x150, 200x150, 200x200, 250x200, and 250x250 in order not to lose information, and also, the size of the deviation between images has not allowed modifying the results.

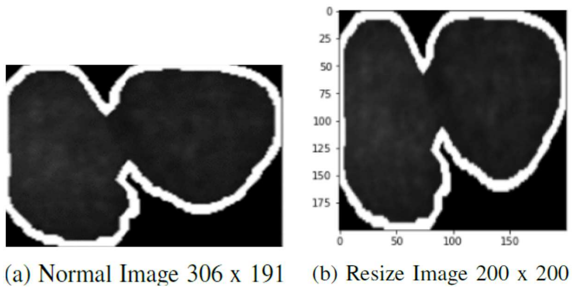

Figure 4: Results In Resizing Images Of Cells With Two Nuclei.

3.4 Image Segmentation

Image segmentation is one of the most critical tasks in automatic image analysis. It consists of subdividing an image into its constituent parts and extracting these parts of interest (objects). The actual segmentation results obtained by applying a segmentation algorithm, sometimes preceded by preprocessing and/or followed by post-processing processes, are compared with the references by counting their differences [25].

Then some techniques such as Gaussian filters are used to smooth the image to diminish the influence of noise. As the main body of the image segmentation system, the image segmentation algorithm determines the result of image segmentation [26].

Gaussian filtering has been intensively studied in image processing and computer vision. Using a Gaussian filter for noise suppression, the noise is smoothed out, and at the same time, the signal is also distorted. The use of Gaussian filters as pre-processing for edge detection will also give rise to edge position displacement, edges vanishing, and phantom edges [17]. The two-dimensional digital Gaussian filter is shown in Equation 1.

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp - \frac{x^2+y^2}{2\sigma^2} \quad (1)$$

where σ^2 is the variance of the Gaussian filter, and the size of the filter kernel 1 ($-1 \le x, y \le 1$) is often determined by omitting values lower than five percent of the maximum value of the kernel. The one-dimensional Gaussian filter is shown in Equation 2.

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp - \frac{x^2}{2\sigma^2} \quad (2)$$

It is also called electronic noise because it arises mostly in electronic amplifiers. The standard model of such a noise is additive, Gaussian, independent at each pixel and independent of the signal intensity, caused primarily by thermal noise. In color cameras where more amplification is used in the blue color channel than in the green or red one, there can be more noise in the blue channel [18].

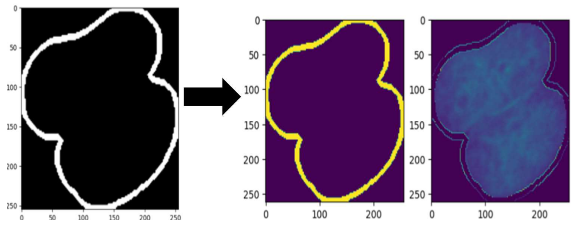

Figure 5: 5(a) Unprocessed Image, 5(B) Images With Filters Applied.

3.5 CNN Architecture

The principal purpose was to build an Inception CNN architecture in order to represent the proximity of the dispersion of pixels' images using Inception Blocks which will have convolutions in order to get the information from high-level layers of CNNs.

In the beginning, the input parameters have high values which would be reduced in each convolution to project more limited information.

So if the convolutional input has a 3x3 or 5x5 kernel with padding = 2 and stride = 2, these will be reduced by the number of layers. In general, a kernel with 3x3, 5x5, or 7x7 is related to the size of the region, where it would be moved. Padding means, the number of zeros around the input image which will be filled and stride means the number of pixels which are moved by the kernel. For that reason, the filters need to be greater than the kernel. In the ReLU layer will be changed the negative values by zeros and the positive values won't be changed.

A problem with the CNN structure is the use of convolutional layers with a 3x3 kernel since it is very difficult to handle and it depends on the input size of the images. This suggests that If the input information is very large, a filter should be used to reduce that input information allowing a more efficient flow. Then in this case the "Inception block" using a filter to compress the information would allow to have a more acceptable quality depending on the number of convolutional layers.

The CNN structure is shown in Figure 6 [19] [27], which has "Inception blocks" and different convolutional layers with a predetermined size to generate the most optimal result, also ReLU, max-pooling, and Linear will be used as filtering layers, and mainly the convolutional layers at the beginning are large enough to enter all the information in a filter for the purpose of optimization and this process will be repeated to generate a correlation between the first and the last layers, also at each step the information have been joined and compressed to reach an optimal result.

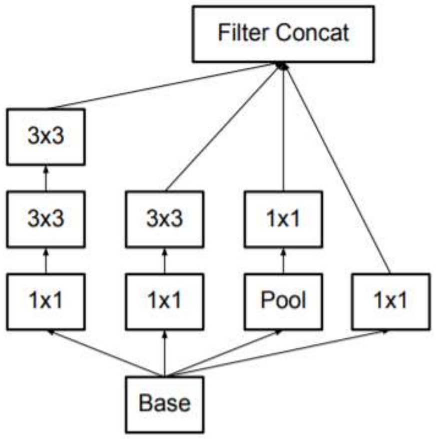

Figure 6: Inception Module Of Two 3x3 Convulsions [19]

Now the CNN structure is shown with the "Inception blocks" and the different convolutional layers with a predetermined size to generate the most optimal result, also ReLU, max-pooling, and Linear will be used as filtering layers.

In the beginning, the "Inception block" will depend on the input size generated by the first convolutional layers. If the input layer is too large, the information entered will be compressed either with 3x3 or 5x5 layers.

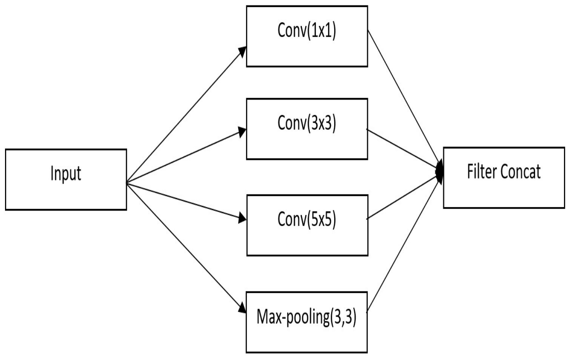

Figure 7: Inception Block Without Reductions.

Figure 7 shows the first Inception block, which contains a set of convolution layers that generate an increase in the number of outputs, but this will not affect its filtering time since the number of inputs and outputs will be divided with the reduction layers.

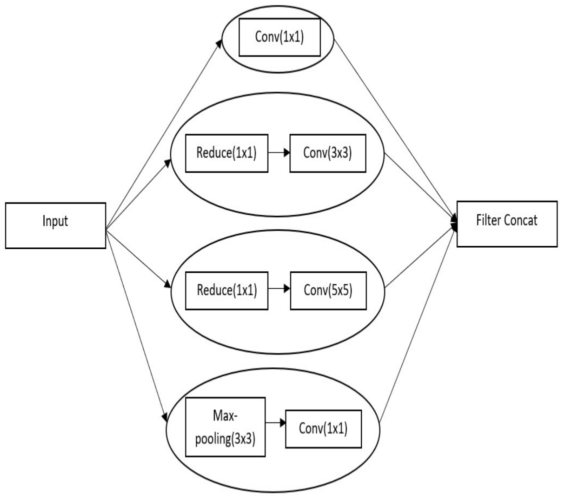

Figure 8: Inception Block With Reduction Per Layer.

Figure 8 shows the input of the reduction layers, which contain the information of Figure 6, compressing all the information of the input and output channels, where the 1x1 convolutions will allow us to calculate the reductions of the 3x3 and 5x5 layers. Then these Inception blocks will be used when the computational cost is high, thus avoiding a collapse of the information and compressing it to connect it with the following convolutional layers of the network.

Now is important to describe the connection line for Inception blocks with each CNN layer, which is shown in Table 2, where the first column is the type of layer (maxpooling, ReLU, Inception blocks, AveragePooling, Dropout, Linear and Softmax), the second column represents the Kernel/Stride and the third column represents the output size for each layer, which is necessary because if the size is unequal after processing the output size of the other layer, then the assigned value will not be suitable, giving a CNN Connection continuity error.

Table 2: CNN Architecture

Type of layer	Kernel/Stride	Output Size
conv1	7x7/2	128x128x64
ReLU1	--	--
maxpool1	3x3/2	64x64x64
conv2	3x3/1	64x64x192

ReLU2	--	--
maxpool2	3x3/2	32x32x192
Inception(6a)	Figure 7	32x32x256
Inception(6b)	Figure 7	32x32x480
maxpool3	3x3/2	16x16x480
Inception(7a)	Figure 8	16x16x512
Inception(7b)	Figure 8	16x16x512
AdaptiveAvgPool2d	6x4	6x4x512
Dropout(40%)	--	6x4x512
Linear	--	512x6x4x10

3.5.1 Convolution Layers

During the convolution operation, the filter slides over the input image and the filter is slipped, applying a point-to-point multiplication between the image pixel values and the kernel values at each position. This operation is repeated for each location in the input image, generating a feature map that highlights the features.

An example is shown in Figure 9 where the input is a 32*32-pixel grayscale image that will be processed with a 3x3 size filter called kernel, that will scroll over the image. Then the values will be multiplied by the pixel values and will be accumulated into a single value in the output.

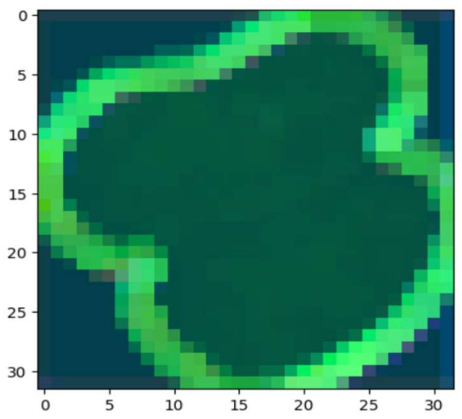

Figure 9: Example of the application of a 3x3 convolutional layer on an image of size 32 x 32 pixels.

3.5.2 ReLU Layer

The ReLU layer is defined in the equation

$$f(x) = max(0,x) \qquad (3)$$

Where $\boldsymbol{x}$ is the input value and $\boldsymbol{f(x)}$ is the output value. The ReLU layer takes an input and sets any negative values to zero, keeping positive values unchanged. This function has the advantage of being simple and easy to compute, but it also introduces nonlinearity into the network. Nonlinearity is important in neural networks because many real-world problems have complex, nonlinear relationships between variables. The ReLU layer allows the network to capture these non-linear relationships and can learn to represent more complex patterns in the input data.

3.5.3 Max-pooling

Max-pooling is a pooling technique that takes a region of the input image and selects the maximum value within that region as the output value. This is an iterative process over the whole input image, creating a reduced version of the original feature map. The result is an image with fewer pixels and reduced spatial dimensions. An example of max pooling is shown in Figure 10 using a kernel = 3x3 and a strike = 2 over the processed image in order to generate another smaller image with the maximum values per the number of regions of the image.

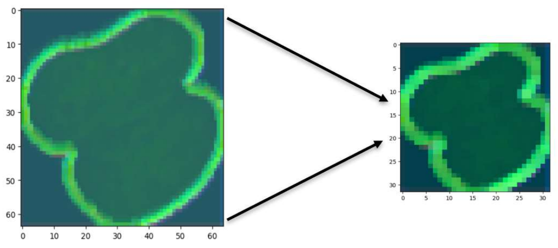

Figure 10: Example of using a max-pooling= 3x3 and a stride=2 on an image of 60x60 pixels.

3.5.4 Fully Connected Layer

In a fully connected layer, each neuron is connected to every neuron in the previous layer. The lineal combination is calculated for each input then the activated Nonlineal ReLU, sigmoid, or tanh functions are applied. For that reason, this layer has several trained parameters that permit the CNN to capture the complex relations between input characteristics and the output labels.

3.5.5 Optimizers

Optimizers are algorithms used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses. There are different kinds of optimizers, but the most useful are mentioned after that: Adam (Adaptive moment estimation), Adadelta, and AdaGrad (Adaptative Gradient Algorithm).

- Adam (Adaptive moment estimation): This is a stochastic gradient descent method that is based on the adaptive estimation of first-order and second-order moments. The method is straightforward to implement, computationally efficient, has little memory requirements, is invariant to diagonal re-scaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters [20]. Algorithm 1, describes this optimizer.

Algorithm 1 *Adam*, our proposed algotihm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\propto = 0.001$,$\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t

```
Require: $\propto$: Stepsize
Require: $\beta_1$, $\beta_2 \in [0,1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$(Initialize $1^{st}$ moment vector)
  $v_0 \leftarrow 0$(Initialize $2^{nd}$ moment vector)
  $t \leftarrow 0$(Initialize timestep)
  while $\theta_t$ not converged do
      $t \leftarrow t + 1$
      $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Gets gradients w.r.t stochastic objetive at timestep $t$)
      $m_t \leftarrow \beta_1 \cdot m_{t-1}$+(1-$\beta_1$)$\cdot$ $g_t$ (Update biased second raw moment estimate)
      $v_t \leftarrow \beta_2 \cdot v_{t-1}$+(1-$\beta_2$) $\cdot$ $g_t^2$
      $\widehat{m}_t \leftarrow m_t$/(1-$\beta_1^t$) (Compute bias-corrected first moment estimate)
      $\widehat{v}_t \leftarrow v_t$/(1-$\beta^t 2$) (Compute bias-corrected second raw moment estimate) $\theta_t \leftarrow_{t-1}$-$\propto \cdot \widehat{m}_t/(\sqrt{\widehat{v}_t}+\epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

- Adadelta: The method dynamically adapts over time using only first-order information and has minimal computational overhead beyond vanilla stochastic gradient descent [21]. Algorithm 2, describes the optimizer.

Algorithm 2 Computing ADADELTA update at time t

```
Require: Decay rate $\rho$, Constant $\epsilon$
Require: Initial parameter $x_1$
1: Initialize accumulation variables $E[g^2]_0 = 0$, $E[\triangle x^2]_0 = 0$
2: for $t$ = 1 : T do %% Loop over # of updates
3:     Compute Gradient: $g_t$
4:     Accumulate Gradient: $E[g^2]_t = \rho E[g^2]_{t-1}$+(1-$\rho$)$g_t^2$
5:     Compute Update: $\triangle x_t = -\frac{RMS[\triangle x]_{t-1}}{RMS[g]_t} g_t$
6:     Accumulate Updates: $E[\triangle x^2]_t = \rho E[\triangle x^2]_{t-1}$+(1-$rho)\triangle x_t^2$
7:     Apply Update: $x_{t+1} = x_t + \triangle x_t$
8: end for
```

- AdaGrad (Adaptative Gradient Algorithm): The basic idea of AdaGrad is to use different step sizes at every iteration. The step size is large at the beginning for rapid gradient descent. As the optimization process progresses, the step size is reduced for the gradient that has fallen significantly, and the larger step size is maintained for the gradient that has not decreased so much [22]. Algorithm 3, describes this optimizer.

Algorithm 3 AdaGrad algorithm

```
1: Input: The state cetor before AdaGrad algorithm
2: Output: The estimated state vector. Initialization:
3: $x_0 = [\epsilon_x \epsilon_y \epsilon_z \nabla_x \nabla_y \nabla_z]^T$, $k = 0$, $G_0 = 0$, $g_k = 0$
4: repeat:
5: Calculate $g_k$ :
6: $\sqrt{G_k + \epsilon}\eta^{-1}[F(x_k + \frac{\eta}{\sqrt{G_k+\epsilon}}) - F(x_k)]$
7: Update $G_k$ :
8: $G_k = G_{k-1} + gk$
9: Update $x_k$
10: $x_k = x_{k-1}$-$\frac{\eta}{\sqrt{G_k+\epsilon}}$ $g_k$
11: until $g_k <$ 1e-8
```

3.5.6 First and second convolutional layer

According to the structure's CNN, the size of each image is (255,255,3). However, the size of the batch of images could be different because the parameter depends on the purpose. Therefore, the size of the images will be changed after the first convolutional layer with the following parameters: kernel = 7, stride =2, and padding=3. For that reason, the size now is obtained by:

((255 + 2 * 3 - 7) / 2) + 1 = 128

And then is necessary to add max-pooling with a kernel=3, stride =2, and padding =1, having as a result.

((128 + 2 * 1 - 3) / 2) + 1 = 64

At the end of the first layer, the size of the image is 64x64 with 64-out filters. All this process is shown in Figure 11.

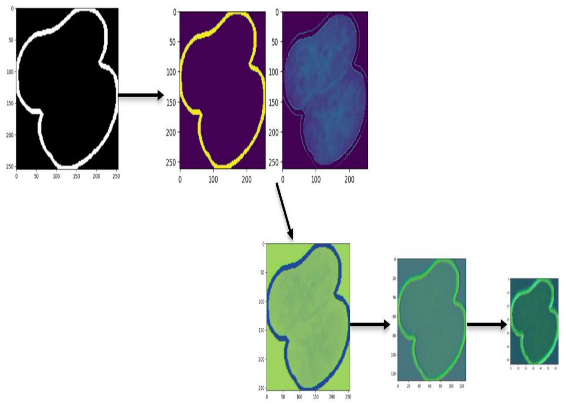

Figure 11: Results of the first convolutional layer.

The second layer has an image of (64,64,64), which is processed with these parameters: kernel=3, padding=1, having as a result an image of:

((64 + 2 * 1 - 3) / 1) + 1 = 64

Using a filter of 192; we have an image of 64x64 with 192 out filters.

And then is necessary to add max-pooling with a kernel=3, stride =2, and padding =1, having as a result.

((64 + 2 * 1 - 3) / 2) + 1 = 32

And an image of size 32x32 with 192 out filters. All this process is shown in Figure 12.

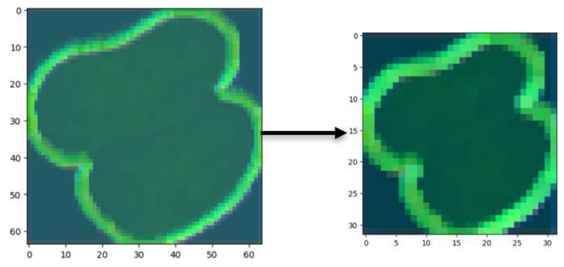

Figure 12: Results of the second convolutional layer.

3.5.7 Inception y Max-pooling block

In this case, the input image is 32x32 with 192 out filters then the inception block to reduce the input information. Using the following parameters kernel=3, stride=2 y padding=1 a max-pooling is generated, having as a result.

((32 + 2 * 1 - 3) / 1) + 1 = 16

At the end of this process, the size of the image is 16x16 with 480 out filters.

3.5.8 Inception and Average Pooling Block

The last layer the input image has a size of (16,16,480). The inception block is used to reduce the size of the information, and finally it is added an average pooling of size (6,4) to get an image of size 6x4 with 512 out filters.

4. EXPERIMENTS AND RESULTS

In the experiments carried out, the dataset mentioned in Section 3.1 was used to evaluate the proposed models where the pre-processing of the input microscopic image using filters such as the Gaussian noise filter to remove noise, the Canny filter, and the Threshold to verify the cell membranes were trained by our CNN after that, all these results were joined into a single image, which is shown in Figure 13.

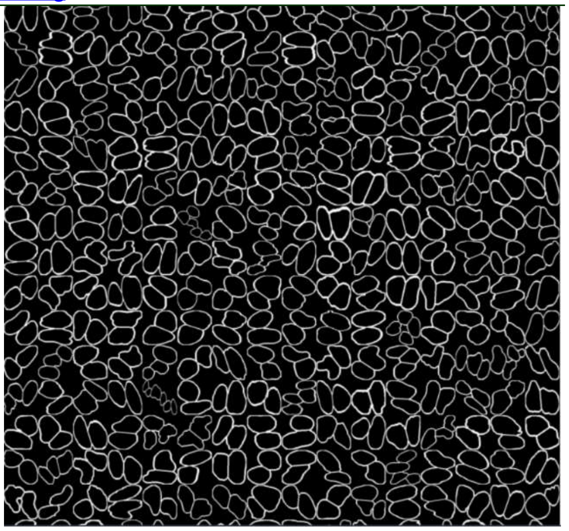

Figure 13: Set of images of a single batch of size 64

The experiments were performed using three optimizers: Adam, Adagrad, and Adadelta whose accuracy was compared with different numbers of epochs which is shown in Figure 14. It is necessary to know how the number of epochs influenced the optimizers for the recognition of the input data and how they adapted to the change of the parameters used, in this case, it was twenty epochs because if more epochs are used they would create an overfitting of the input data, varying its accuracy between more number of epochs.

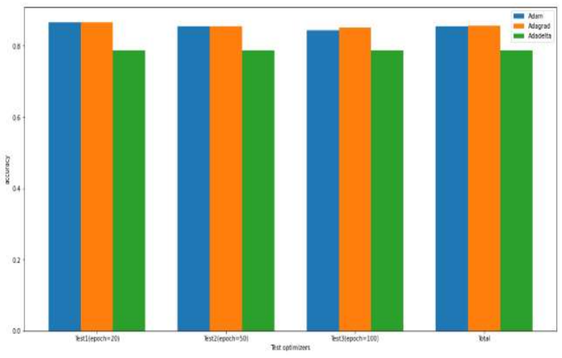

Figure 14: Comparison of the optimizers with the different number of epochs.

Figure 15 and Figure 16 show the confusion matrix where the predictions made by our CNN can be verified. There are 4 labels where you will find the total number of hits of the microscopic test images, this accuracy is based on the true positives divided by all the results (true positives and false positives). The results of this confusion matrix are shown in more detail in Table 3.

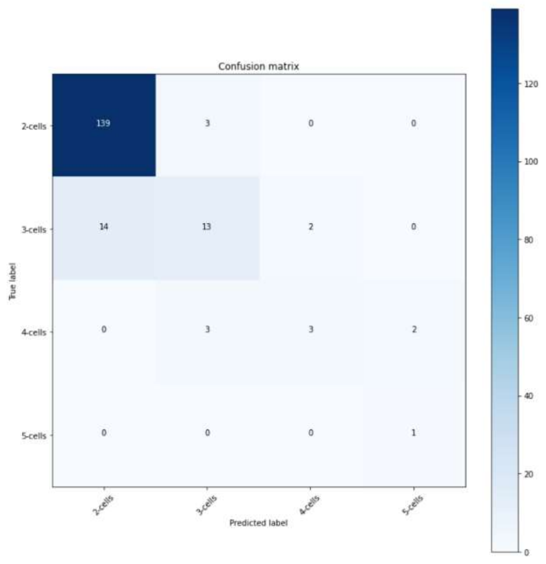

Figure 15: Confusion Matrix with normalize=False

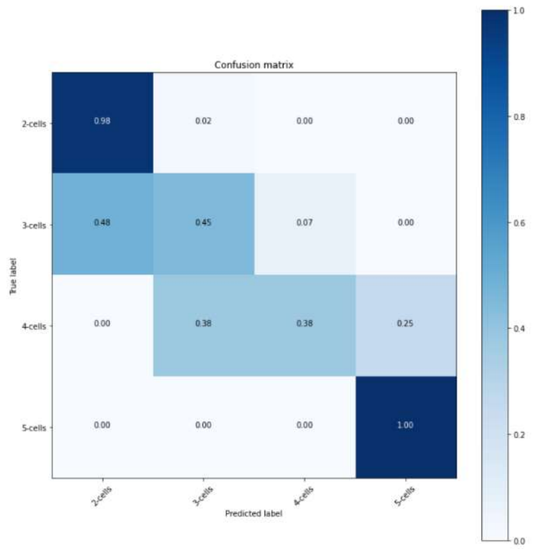

Figure 16: Confusion Matrix with normalize=True

Table 3: Results and percentages of each cell type by the number of nuclei using Adagrad optimizer.

Dataset-Algorithm-Adagrad			
Category	Nº. of correct cells	Nº. of incorrect cells	Percentages of correct classification rate
2-cells	139	3	98%
3-cells	13	16	45%
4-cells	3	5	38%
5-cells	1	0	100%
Total	156	24	86.6%

Table 3 shows the results obtained with the Adagrad optimizer, where the best accuracy was obtained in 2-cells and 5-cells, due to a given test and training batch size. In addition, the total number of correct cells was derived from the application of different types of filters, such as Canny and Thresholding, to extract information from the edges and to detect important objects, such as the separation of the cell membrane from its nucleus. In the precision of 3-cells and 4-cells, a low result was obtained due to the detection problem of the cell membrane, since when detecting the edges of a cell with several nuclei, its cell membrane can surround a nucleus of a dispersed way generating a separation between the number of nuclei that the cell has, developing an opening that can be confused with a nucleus.

Table 4: Comparative analysis of optimizer's accuracy.

Dataset				
Optimizers	Nº of epochs	Accuracy	Correct cells	Percentage deviated
Adam	100	85.5%	154	14.5%
Adagrad	100	86.6%	156	13.4%
Adadelta	100	78%	142	22%

Table 4 shows the results of all the optimizers used, using different numbers of epochs for training, the average of the precision obtained by the number of epochs used, the general number of correct cells obtained after training, and the deviation of information, which is the missing percentage not obtained in training. As a general result, the Adadelta optimizer reached 78% accuracy, which is the one at obtained the worst precision because it limits the gradients to a fixed size, The Adam optimizer reached 85.5% accuracy whose precision was similar to the Adagrad optimizer which reached 86.6% of accuracy, and this was the best precision for this type of training.

5. CONCLUSIONS

In this research, a CNN Inception has been proposed to identify the four types of cell membranes in 2D images based on the number of nuclei in each one. Furthermore, experimental results have shown an accuracy of 98% for the 2-cell category and 100% for the 5-cell category; the 3-cell and 4-cell categories had a large precision deviation in the results that negatively impacted the overall precision, which was 86.6%.

The effectiveness of the optimizers used Adam, Adagrad, and Adadelta was compared to analyze their performance, and the best result was obtained with the Adagrad optimizer, because of Adagrad has a parameter-specific learning rate, which is a ratio of how often a parameter is updated during training. Furthermore, as the optimization process progresses, the step size is reduced for the gradient that has decreased significantly and the larger step size is maintained for the gradient that has not decreased so much [22]. The results show that the proposed method works satisfactorily in the identification of cell membranes in 2D images, in simple and complex images, but if there are abnormalities in the shape of the cell membrane, this could generate an additional edge that joins another cell, causing confusion in the results.

REFERENCES:

[1] B. Pontes, Y. Ayala, A. C. C. Fonseca, L. F. Romao, R. F. Amaral, L. T. Salgado, F. R. Lima, M. Farina, N. B. Viana, V. Moura-Neto, et al., “Membrane elastic properties and cell function,” PloS one, vol. 8, no. 7, p. e67708, 2013.

[2] M. P. Sheetz, “Cell control by membrane–cytoskeleton adhesion,” Nature Reviews Molecular Cell Biology, vol. 2, no. 5, pp. 392–396, 2001.

[3] W. D. Bennett and D. P. Tieleman, “The importance of membrane defects lessons from simulations,” Accounts of chemical research, vol. 47, no. 8, pp. 2244–2251, 2014.

[4] G. Danuser, “Computer vision in cell biology,” Cell, vol. 147, no. 5, pp. 973–978, 2011.

[5] S. Yang, D. Kohler, K. Teller, T. Cremer, P. Le Baccon, E. Heard, R. Eils, and K. Rohr, “Nonrigid registration of 3-d multichannel microscopy images of cell nuclei,” IEEE Transactions on Image Processing, vol. 17, no. 4, pp. 493–499, 2008.

[6] K. S. Lorenz, F. Serrano, P. Salama, and E. J. Delp, “Segmentation and registration based analysis of microscopy images,” in 2009 16th IEEE International Conference on Image Processing (ICIP), 2009, pp. 4213–4216.

[7] B. Appleton, A. Bradley, and M. Wildermoth, “Towards optimal image stitching for virtual microscopy,” in Digital Image Computing: Techniques and Applications (DICTA’05), 2005, pp. 44–44.

[8] J. Zhang, H. Hu, S. Chen, Y. Huang, and Q. Guan, “Cancer cells detection in phase-contrast microscopy images based on faster r-cnn,” in 2016 9th International Symposium on Computational Intelligence and Design (ISCID), vol. 1, 2016, pp. 363–367.

[9] Y. Xue and N. Ray, “Cell detection in microscopy images with deep convolutional neural network and compressed sensing,” arXiv preprint arXiv:1708.03307, 2017.

[10] S. Ram, V. T. Nguyen, K. H. Limesand, and J. J. Rodriguez, “Combined detection and segmentation of cell nuclei in microscopy images using deep learning,” in 2020 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), 2020, pp. 26–29.

[11] Sato, M., Hotta, K., Imanishi, A., Matsuda, M., & Terai, "Segmentation of Cell Membrane and Nucleus by Improving Pix2pix.", In *Biosignals,*2018, pp. 216-220.

[12] Dimauro, G., Ciprandi, G., Deperte, F., Girardi, F., Ladisa, E., Latrofa, S., & Gelardi, M, “Nasal cytology with deep learning techniques”, *International Journal of Medical Informatics*, Vol. 122, 2019, pp 13-19.

[13] Jun Du, Li, X., & Li, Q, “Detection and classification of cervical exfoliated cells based on faster R-CNN”. In IEEE 11th International Conference on Advanced Infocomm Technology (ICAIT), 2019, pp. 52-57.

[14] Long, M., Liang, G., Zheng, Y., Li, Z., & Zhong, J., "Cervical cell TCT image detection and segmentation based on multi-scale feature fusion.", IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Vol. 5, 2021, pp. 192-196

[15] Ma, B., Zhang, J., Cao, F., & He, Y, "MACD R-CNN: an abnormal cell nucleus detection method.", IEEE Access, 2020, vol. 8, pp. 166658-166669

[16] B. Vrigazova, “The proportion for splitting data into training and test set for the bootstrap in classification problems,” Business Systems Research: International Journal of the Society for Advancing Innovation and Research in Economy, vol. 12, no. 1, pp. 228–242, 2021.

[17] G. Deng and L. Cahill, “An adaptive Gaussian filter for noise reduction and edge detection,” in 1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference. IEEE, 1993, pp. 1615–1619.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[18] L. ˇ Sroba, J. Grman and R. Ravas, "Impact of Gaussian noise and image filtering to detected corner points positions stability," Ph.D. dissertation, 2017.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," Ph.D. dissertation, 2016.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[21] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012. [Online]. Available: https://arxiv.org/abs/1212.5701

[22] Z. Wen, G. Yang, and Q. Cai, "An improved calibration method for the imu biases utilizing KF-based degrade algorithm," Sensors, vol. 21, no. 15, p. 5055, 2021

[23] Ikenouchi, J. (2018). Roles of membrane lipids in the organization of epithelial cells: Old and new problems. Tissue barriers, 6(2), 1-8.

[24] Sant, D. G., Tupe, S. G., Ramana, C. V., & Deshpande, M. V. (2016). Fungal cell membrane—promising drug target for antifungal therapy. Journal of Applied Microbiology, 121(6), 1498-1510.

[25] Zhang, Yu Jin. A survey on evaluation methods for image segmentation. Pattern recognition, 1996, vol. 29, no 8, p. 1335-1346

[26] Kang, W. X., Yang, Q. Q., & Liang, R. P. (2009, March). The comparative research on image segmentation algorithms. In 2009 First international workshop on education technology and computer science (Vol. 2, pp. 703-707). IEEE.

[27] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).,