Machine Learning-Based Web and Mobile Application for Counting Cells from Microscopy Images

Arda Bakıcı[1], Ali Zeyrek[2], and Doğa Çapanoğlu[3]

[1]arda.bakici@stu.bahcesehir.k12.tr
[2]ali.zeyrek@stu.bahcesehir.k12.tr
[3]Research Mentor

Abstract

Cell counting is a common method used in various fields, including pharmaceutical research and oncology, to determine the number of cells present in a sample. This process is typically carried out manually by researchers counting every cell individually with a microscope, which can be time-consuming and tedious. Automated cell counting equipment and software are available, but these options are often expensive, inaccessible, and not easy to use. In this project designed to meet this need of researchers, a dataset consisting of 792 photographs of trypan blue stained MCF7 breast cancer cells was created. Each cell was either labeled dead or live with bounding boxes. Then, machine learning models were trained to detect live and dead cells from images. Various models from the Scaled YOLOv4 family (YOLOv4-CSP, YOLOv4-P5, and YOLOv4-P6) with multiple variations of hyperparameters and options were trained and tested on the dataset to determine the optimal settings for maximum accuracy. After evaluating the performance of the models on the test dataset, the YOLOv4-P5 model with a batch size of 8, an epoch number of 193, an image size of 1024, transfer learning enabled, and multi-scaling disabled provided the best result of 0.865 precision, 0.961 sensitivity, 0.958 mAP.5 and 0.451 mAP.5:.95 scores. The speed of the developed model was found to be 0.06 seconds when processing a 1280x900 resolution photograph on a Tesla P100 GPU. To make the model accessible to researchers, a web application and a mobile application were also developed. These applications allow researchers to easily upload photographs and process them on the trained artificial intelligence model. The image processing is carried out on the web server, which means that the web application and mobile application can be accessed from any device with internet connection, without the need for any additional hardware. Both the web application and the mobile application are user-friendly and provide an accessible and cheap alternative to other cell counting methods while working nearly 300 times faster than manual counting.

Keywords: Machine learning, computer vision, mobile app, web app, cell counting

Contents

1 Objective

In cancer studies, molecular biology, and drug research, it is necessary to know the number of cells. One of the most popular methods for counting cells is staining. Stained cells are typically counted manually by researchers using a microscope. Manual counting is a time-consuming process. In addition, manual cell counting can cause discrepancies in the counts made by experts. It is important for the reproducibility of the results that the counts be standardized. There are alternative counting methods such as automatic counting machines and image processing software. However, automatic counting machines are expensive and not widely available. Image processing software requires the user to have prior knowledge and is not easy to use. A cheap, easy, accessible, and fast method is needed. The aim of this project is to develop a machine learning model that automates the processing of cell photographs stained with trypan blue on a website or mobile application and to provide this model on a web server.

2 Introduction

Testing the viability of cells is frequently used in studying the effectiveness and side effects of drugs as well as in oncological research [1]. The Trypan Blue Exclusion Test of Cell Viability is a method that allows the measurement of cell viability with a light microscope (CAS Registry Number: 72-57-1) [2]. Live cells have selective permeability of their membranes, so the dye cannot enter the cell cytoplasm, while dead cells have lost this property [3]. Therefore, trypan blue accumulates in the cytoplasm of dead cells, causing the cells to turn blue. As shown in 1, dead cells are dark while live cells are light in color.

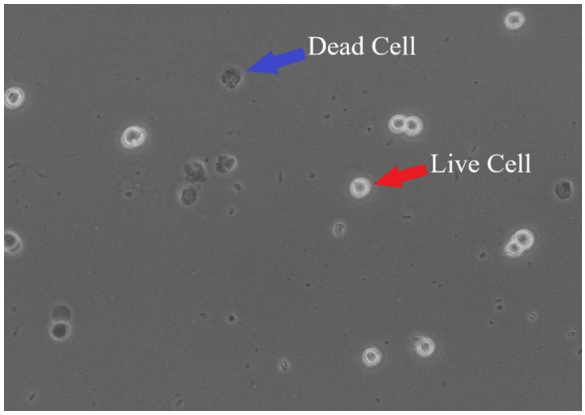


Figure 1: Stained live and dead cells for Trypan Blue Exclusion Test

The method is used to calculate the number of dead/live cells in a cell culture at a certain concentration. The number of cells should be in a range suitable for counting in a hemocytometer. The trypan blue/cell mixture is placed on the hemocytometer and focused on a hemocytometer light microscope slide as shown in 2 [4].

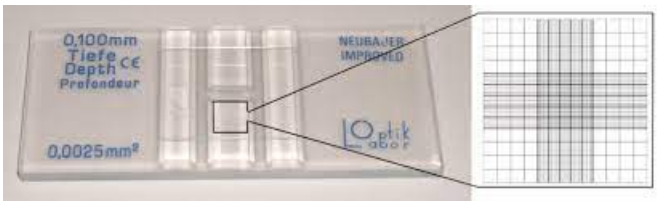


Figure 2: Hemocytometer

Stained (dead) and unstained (live) cells are counted separately in the hemocytometer [5]. For example, to obtain the total number of cells per milliliter, the total number of live and dead cells and the dilution factor are substituted to (1).

$$\text{Total Cells/ml} = \text{Total Counted Cells} \times \frac{\text{Dilution Factor}}{\text{Number of Squares}} \times 10000 \text{ cells/ml} \quad (1)$$

To calculate the percentage of viability, the number of live cells is divided by the total number of live and dead cells.

$$\%\text{Viability} = \frac{\text{Number of Live Cells}}{\text{Number of Total Cells}} \quad (2)$$

Cell counting using a microscope, which is performed by technicians, biologists, and specialists in clinical and research laboratories, is a difficult-to-standardize indirect method. The error margin may increase due to the evaluation based on cell membrane integrity [6].

There have been various studies on counting stained cells using image processing methods. Grishagin has created a plugin using his own algorithm for the ImageJ platform [7]. Aung and colleagues have developed an algorithm using image processing methods such as grayscale conversion, morphological operations, adaptive k-means clustering, and circle hough transform to count cells stained with trypan blue [8]. In addition to these studies, recent research has also been conducted using machine learning methods, which have replaced classical image processing methods. In his study, Ozkan first applied preprocessing to the photographs and then used a three-layer artificial neural network with one hidden layer and one output layer to count cells stained with trypan blue [9]. Artificial intelligence-based approaches aimed at automating cell counting have also begun to spread in industrial fields in recent times [10]. However, these artificial intelligence software can only work with the help of specialized equipment, so the applicability of these solutions is low. For laboratories using standardized equipment, resorting to such methods would render old equipment obsolete and therefore lead to unnecessary costs. A cheaper, user-friendly and accessible method is needed compared to existing methods.

The concept of machine learning, first proposed by Arthur Samuel in 1959, has revolutionized image processing technology with the development of convolutional neural networks by Krizhevsky and colleagues [11]. This eventually led to development of machine learning models capable of object detection. One of the early popular models was YOLOv1 published by Redmon and colleagues in 2016 [12]. Redmon and Farhadi continued to develop the model by publishing YOLOv2 in 2017 and YOLOv3 in 2018 [13] [14]. YOLOv4 was published by Bochkovskiy and colleagues in 2020 and Scaled YOLOv4 was published by the same group in 2021 [15] [16].

Scaled YOLOv4 model family consists of 3 model types: YOLOv4-Tiny, YOLOv4-CSP, YOLOv4-Large. These models differ in the number of layers and parameters. The YOLOv4-Tiny model is a lightweight model in terms of its required processing power and designed for use on phones and embedded systems. YOLOv4-CSP is heavier but more accurate than YOLOv4-Tiny and lighter in parameter size but less accurate than YOLOv4-Large. The YOLOv4-Large models are the heaviest and most accurate models in the Scaled YOLOv4 family. The YOLOv4-Large consists of three models: YOLOv4-P5, YOLOv4-P6, and YOLOv4-P7. YOLOv4-P5 is the lightest of these models, while YOLOv4-P7 is the heaviest. When creating the YOLOv4-Large models, the YOLOv4-P5 model was designed first and then the YOLOv4-P6 and YOLOv4-P7 models were created by adding additional layers on top of this model. 3 shows the architectures of the YOLOv4-Large models.

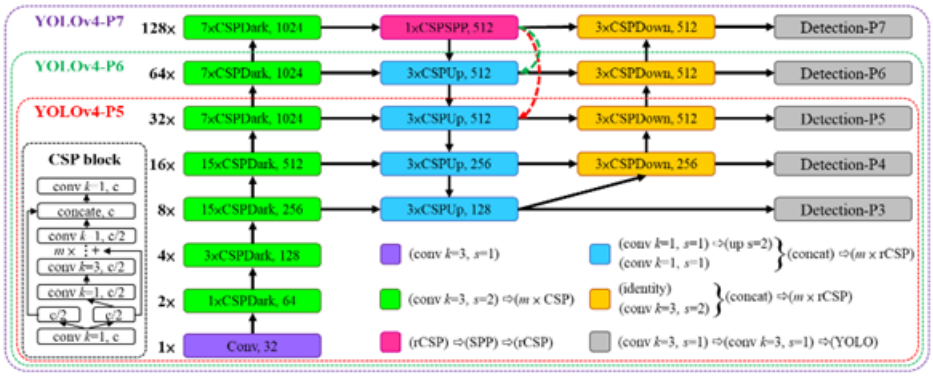


Figure 3: Architectures of YOLOv4-Large models [16]

GIoU (Generalized Intersection over Union), which is one of the loss functions used in training machine learning models, was developed by Rezatofighi and colleagues in 2019. GIoU is an improved version of the IoU (Intersection over Union) loss function, which is commonly used in problems such as object detection [17]. The IoU function is calculated by dividing the intersection of the model's estimate and the ground truth by the union of them. However, the IoU function returns zero if there is no intersection between the estimate and the ground truth and cannot return negative values. This prevents the IoU function from reflecting whether the estimate is close to or far from the reference. To solve this problem, Rezatofighi and colleagues proposed the GIoU function.

The objectness loss function is a loss function that determines whether the model's estimate is an object or not. The model can make multiple estimates for an object, and an objectness score is calculated for each estimate. While other loss functions are only calculated for the estimate with the highest objectness score, the objectness loss function is calculated for all estimates [14].

The classification loss function measures the accuracy of the class that the model predicts for the object. The model gives a probability for each possible class the object might belong to. The binary cross-entropy function is applied to all class probabilities and the results are summed [14].

Backpropagation is a method frequently used in neural networks. In short, the gradient of the error with respect to the weights is calculated, allowing the model to determine how much the weights should change [18]. Gradient descent is a method used to find the local minima of a differentiable function [19]. Stochastic gradient descent is similar to gradient descent, but it is calculated on random data rather than all data. Therefore, it is faster than classic gradient descent, but the loss values obtained during descent are less stable [20].

In image processing, bounding boxes are used in object annotation [21]. There are three types of information in this method: the coordinates of the box, the type of object, and the confidence of the prediction. Non-max suppression (NMS) prevents the same object from being marked by multiple bounding boxes at the same time, doing so by selecting the best estimate from the specific bounding boxes that represent the same object to ensure that the marking is done with higher accuracy. This selection is made by using the IoU method and comparing the confidence of the bounding boxes.

Precision and recall are two methods that help to find the accuracy of predictions and where the error is coming from [22]. Precision gives the percentage of how many of the model's predictions are correct. Recall gives the percentage of how many out of all objects were detected. The formulas for precision and recall are given in (3). In the formula, "TP" represents true positive predictions, "FP" represents false positive predictions, and "FN" represents false negative predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{3}$$

To calculate mAP (minimum average precision), precision values are marked for recall values that increase in increments of 0.1 from 0 to 1, creating a graph like 4 [23]. The precision values are summed for the 11 recall values and the average is taken, i.e., it is divided by 11. This value is repeated for all data types. The average of the AP values for all types gives the mAP value. The mAP@.5 is the value at an IoU threshold of 0.5, while mAP@.5:.95

is the average of the results obtained with IoU threshold values ranging from 0.5 to 0.95, increasing by 0.05 each time.

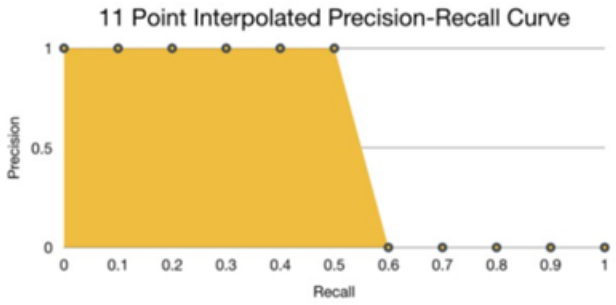


Figure 4: Precision-Recall Graph

Transfer learning is the process of storing data used to solve a problem and using it to solve a similar problem [24]. This speeds up and simplifies the solution of subsequent problems.

Grayscale is a color space consisting only of gray and its shades [25]. The value of each pixel is expressed with an 8-bit value between 0 and 256, so less information is needed for each pixel. The grayscale calculation formula is shown in (4).

$$0.299 \times \mathrm{R} + 0.587 \times \mathrm{G} + 0.144 \times \mathrm{B} \tag{4}$$

CVAT is a free, web-based photo and video annotation tool developed by Intel. It can be used for tasks such as object detection, image classification, and image segmentation [26].

3 Method

The project is a machine learning model that automates the processing of dyed cell photographs used in cell culture studies, and it can be accessed through a web site or mobile application.

In the previous tests of the project, trypan blue stained cell photographs were first converted to grayscale and then image processing methods such as non-local means denoising, CLAHE, Otsu thresholding, morphological operations, and watershed segmentation were applied to the photographs. As a result of these processes, 56.7% accuracy was obtained in the counting of dead cells and 85.5% accuracy in the counting of live cells. However, the accuracy and efficiency of this study were found to be low, and it was thought that the results

could be improved with different methods. It was seen that the classical image processing algorithms did not give good results in various lighting and contrast conditions, which might be the reason for the low accuracy values. For this reason, it was thought that machine learning, a new and popular image processing method, would give more accurate and faster results for this task.

The machine learning codes, and web server are written in the Python (Version 3.9.7) programming language. Pytorch (Version 1.10.2) library has been used for machine learning. The OpenCV (Version 4.5.5) image processing library has been used to read and process the photos. Numpy (Version 1.22.1) has been used to save the photos as an array and to perform operations on the arrays after the photos have been read. The machine learning model and web server have been written in the same language to make it easier to integrate the two codes. The web server has been written using the Flask (Version 2.0.2) library. The codes have been written using Visual Studio Code (Version 1.64). The user interface for the website has been written using the Bootstrap (Version 5.0) library. The mobile application for the project has been developed using the React Native (Version 0.67) library. The React Native library allows the application to run on both Android and iOS without the need for additional code. In addition to the React Native library, the react-navigation (Version 4.4.4) library has been used to provide the screens and navigation within the application, the react-native-vector-icons (Version 9.0) library has been used for additional icons in the application, the react-native-image-picker (Version 4.7.3) library has been used to add the ability to select photos from the phone camera and gallery, and the Redux (Version 4.1.2) library has been used to store universal variables in the application.

3.1 Dataset

In the study, the trypan blue stained MCF7 cancer cell photographs taken with the camera of the Olympus Inverted CKX41 microscope were used. The dataset consists of 704 photographs of 1280x900 and 88 photographs of 1280x960 resolution. Every cell in the 792 photographs were annotated as either living and dead with the help of two experts in biotechnology and molecular biology. Then, the annotations were combined, and a common decision was made in the conflicting ones. As a result of the counts, 5882 live and 1890 dead cells were counted among 792 photographs. The photographs were labelled using the bounding box method through the CVAT application. The interface of the CVAT application is shown in 5.

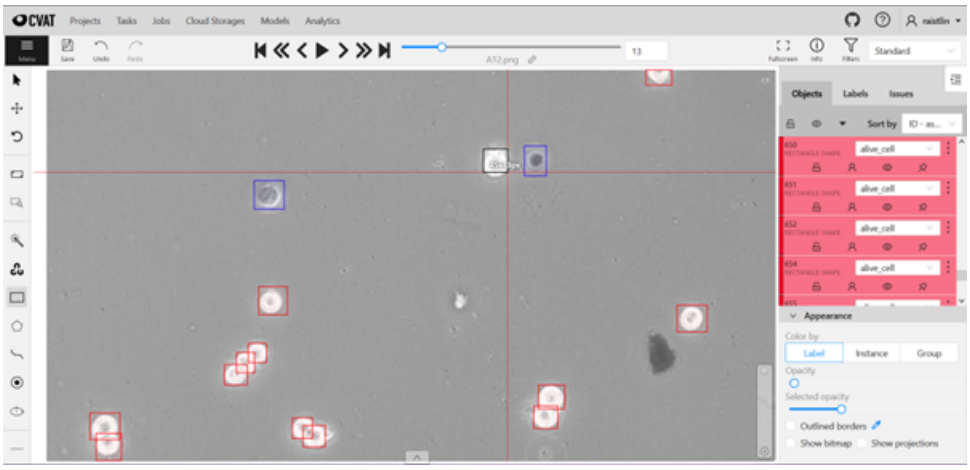
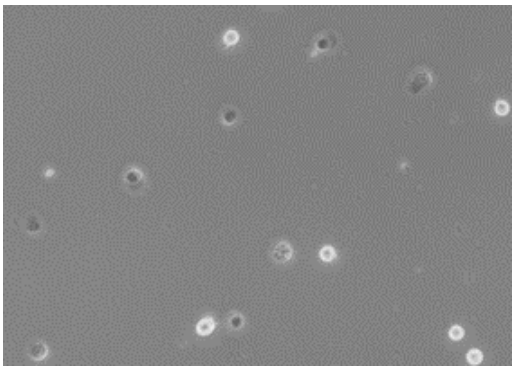
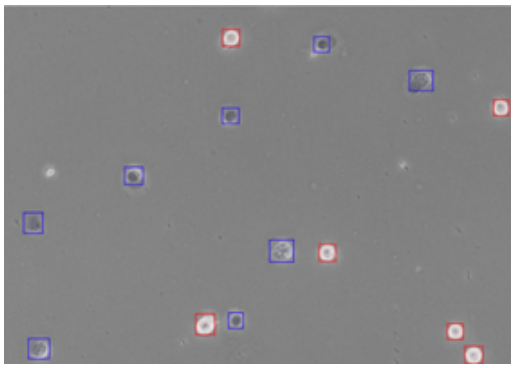


Figure 5: Interface of the CVAT application

The model has been trained with grayscale photos since most microscope cameras can only take photographs in grayscale. Training the model with grayscale photographs allows the model to process both grayscale photographs and colored photographs by converting them to grayscale. The dataset was split in a 75/25 ratio for training and testing the machine learning models. In this case, 592 photos were used for training and 200 photos were used for testing. An example of an annotated and unannotated photograph from the dataset is shown in 6.



(a) An unannotated photograph



(b) An annotated photograph

Figure 6

The Scaled YOLOv4 models chosen for use in this study require that the data to be in YOLO format. According to the YOLO format, a text file with a .txt extension and the same name as the image file is created in the same folder for each image file. Each text file contains the classes, coordinates, heights, and widths of the objects in the corresponding image file. A new line is opened for each object [27]. The contents of an example text file

written in YOLO format are shown in 7.

```
0 0.112012 0.258478 0.036508 0.042866
0 0.12749999999999986 0.282288 0.030312 0.0348
0 0.8948740000000001 0.254472 0.035594 0.041612
1 0.019921875 0.8019340000000001 0.03984375 0.057622
1 0.07783599999999993 0.141 0.041328 0.05433399999999999
1 0.010546875 0.706822 0.02109375 0.091156
```

Figure 7: An example of a text file formatted according to the YOLO format

According to the YOLO format, all values saved are normalized to a range between 0 and 1 by dividing them by the width and height of the photo. The first column in the text file indicates the class of the object, the second column indicates the x coordinate of the center of the bounding box surrounding the object, the third column indicates the y coordinate of the center of the bounding box, the fourth column indicates the width of the bounding box, and the fifth column indicates the height of the bounding box.

3.2 Machine Learning

It was planned to process the photos sent by users on a web server in this project, so a fast model was required to quickly respond to users and prevent traffic on the server. To this end, models that can perform real-time object detection were prioritized among the models studied. When models that perform real-time object detection were examined, it was seen that the YOLO model family was at the forefront in terms of speed and accuracy, and a decision was made to use models from the YOLO family. When the YOLO models were examined, it was decided to use the Scaled YOLOv4 models, which are one of the latest models.

After the dataset was created, which model to be used from the Scaled YOLOv4 models was decided. Tests were performed on the YOLOv4-CSP and YOLOv4-Large models with different photo sizes, transfer learning, epochs, and batch sizes to select the best model. As a result of these tests, YOLOv4-P5 was the best model to be used in this study. The model gave the best result with a photo size of 1024, epoch number of 193, transfer learning enabled, and a batch size of 8. The YOLOv4-Tiny model's performance was considered not accurate enough, and the YOLOv4-P7 model was considered too computation-intensive, so no tests were performed on these models. The Scaled YOLOv4 models used were written in Python programming language and using the Pytorch machine learning library by Yiu [28].

During model training, the photos are processed in groups of 8. After the photos are processed on the model, the model's predictions are compared with annotations. The function used in this comparison is called the loss function. In this study, the GIoU, objectness,

and classification loss functions were used. The goal of the machine learning model is to minimize the output of these functions. The loss values given by the three loss functions are added up during model training. This value is considered the total loss value of the model, which the model tries to minimize. To minimize this value, the backpropagation algorithm is applied to the model to find the gradient values. The SGD optimizer uses these gradient values to adjust the model's parameters in a way that minimizes the loss value.

Some of the models used the transfer learning technique during training. For transfer learning, the weights trained on the COCO 2017 dataset were used [28]. The model was trained on the training data set and its performance was evaluated on the test data set. To evaluate the performance of the model, the precision, recall, mAP@.5, and mAP@.5:.95 metrics were used. The values of these metrics range from 0 to 1. The model that performed the best on these metrics in the test data set was selected as the model to be used in the project.

The YOLO model family assumes that the images to be processed have a fixed pixel size and are square in shape. Images that do not conform to these dimensions are enlarged or reduced while maintaining their aspect ratio and black pixels are added to the empty parts. The rectangular training mode was used during model training [29]. With this mode, for non-square images, the aspect ratio of the photo is preserved by reducing or enlarging the photo until the long edge of the photo reaches the pixel size accepted by the model. The short edge is then filled with black pixels until it becomes a multiple of 32. In this way, it is not necessary to fill the short edge with black pixels until it reaches the pixel size processed by the model, and the processing speed of the model increases. The size of the photo that the model will process affects the performance of the model. Increasing the size of the photo increases the amount of space it occupies in the GPU memory during training. Therefore, the size of the photo was selected based on the amount of space it occupies in the GPU memory.

Data augmentation was applied to the dataset to prevent overfitting during model training. When the photos were given to the model, with a random probability, the photos were rotated horizontally or vertically and zoomed in. For training some models, a data augmentation method called multi-scale, which reduces the photo by a minimum of 50% and enlarges it by a maximum of 50%, was used during training.

The models were initially trained on the NVIDIA GTX 1660 TI Mobile 6 GB GPU, but it was found to be insufficient for model training and gave poor performance. Therefore, the models were trained on the Kaggle platform using a Tesla P100 NVIDIA 16 GB GPU. The CUDA library, which allows for the training of machine learning models on NVIDIA GPUs, was used during the training of the models.

3.3 Web Server

In the project, it was decided to process operations from the mobile application and website on a Flask-based web server. The operation scheme of the project is shown in 8.

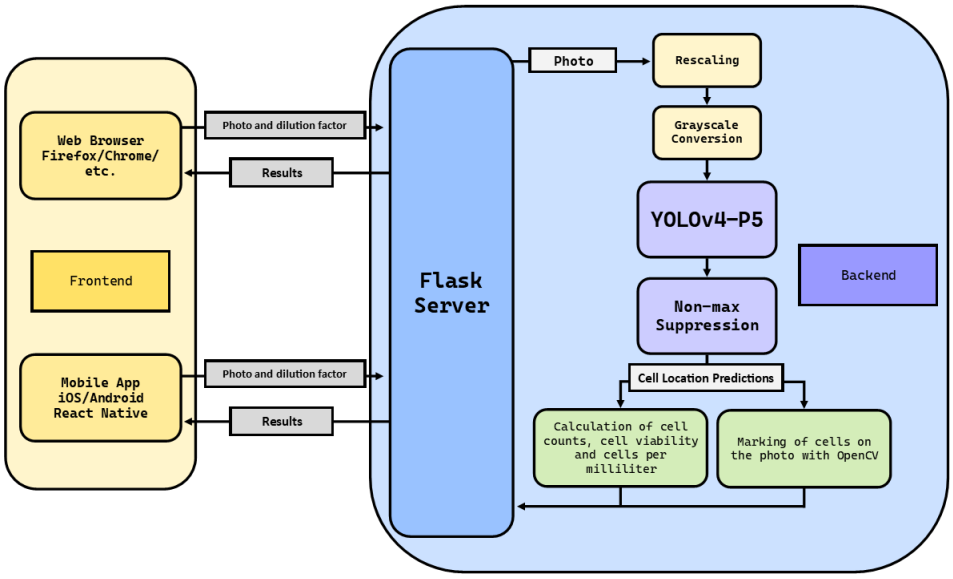


Figure 8: Operation Scheme of the Project

The workflow of the project is as follows: users upload photos and additional information through the mobile application or website, which are then sent to the server. The server receives the photo and converts it to grayscale, as the model was trained on grayscale photos. The photo is then fed to the YOLOv4-P5 model, which processes the photo and returns the potential locations of cells. Non-max suppression is applied to these potential locations to eliminate overlapping predictions. The OpenCV library's rectangle method is used to mark live cells with red rectangles and dead cells with blue rectangles on the photo. In addition to this marked photo, the number of cells and the cell viability calculated using (2), as well as the number of live and dead cells per milliliter calculated using (1) if the user entered the dilution factor, are returned to the user.

Users can access the web server from their web browsers and are redirected to the main page when they enter the website. In 9, the home page of the website is shown.

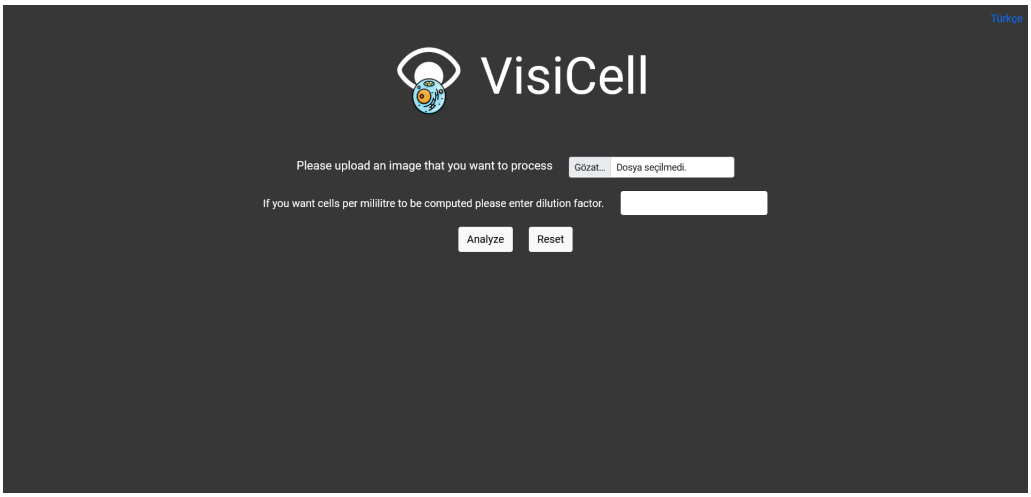


Figure 9: Home page of the website

Users can upload the photo they want to be processed by clicking the button next to the text ”Please Upload the Photo You Want to Process”. Below this, there is a text box where users can enter the dilution factor. If users want the system to calculate the number of live, dead, and total cells per milliliter, they can enter the dilution factor here. Users can reset the uploaded data by pressing the ”Reset” button, or they can send the photo and dilution factor information to the web server by pressing the ”Analyze” button. The photos are processed as shown in 8. After the photos are processed, users are redirected to the results page. 10 shows the results page. The results page shows users the photo with the cells marked and the number of live, dead, and total cells, cell viability and the amount of live, dead, and total cells per milliliter. Users can return to the home screen by pressing the logo at the top left of the website.

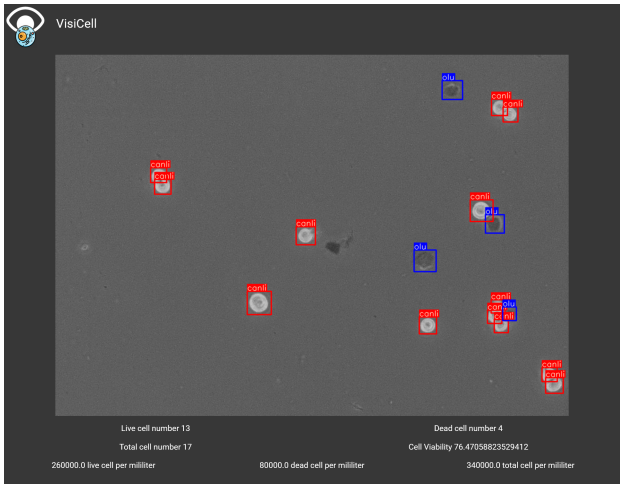
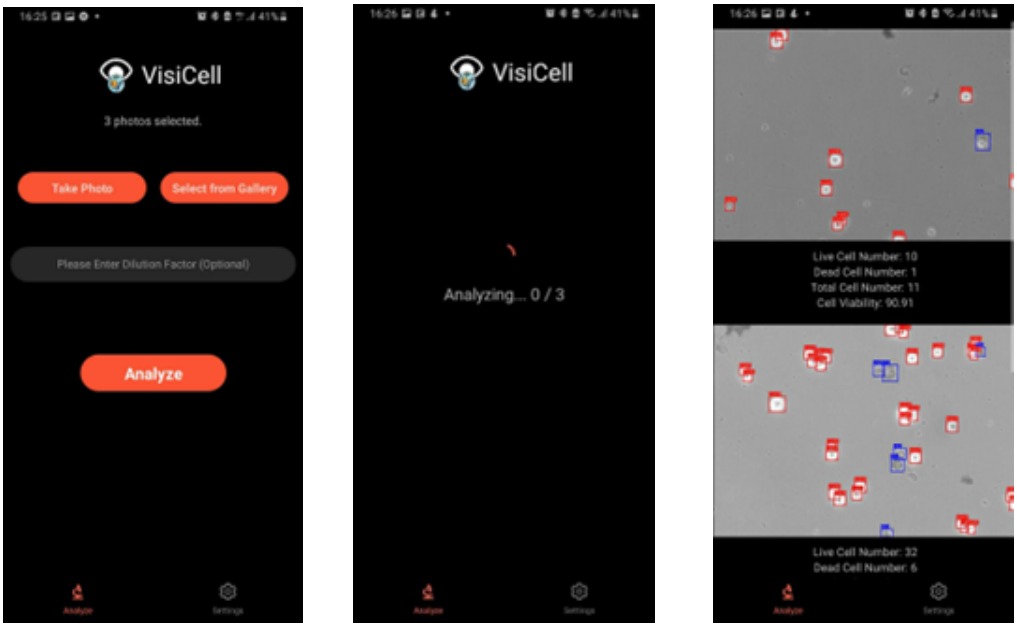


Figure 10: Results page of the website

3.4 Mobile Application

When users enter the application, they are directed to the home page. 11a shows the home page of the mobile application.



(a) Home page (b) Waiting screen (c) Results screen

Figure 11

On this page, there is a text box showing how many photos the user has selected, ”Take Photo” and ”Select from Gallery” buttons to select the photos to be processed, a text box to enter the dilution factor, and an ”Analyze” button to send the photos to the server. When users press the ”Take Photo” button, the device’s camera application opens. From here, users can take the photo they want to be processed using their phone’s camera application. When the ”Select from Gallery” button is pressed, the device’s gallery opens. Users can select one or more photos from the gallery. After selecting the photograph, users will be redirected back to the main screen. In the text box above, users can see how many photographs they have selected. If users wish, they can enter the dilution factor they calculated while staining their cells in the ”Enter Dilution Factor” text box. Entering data in this text box is optional and if no data is entered, the amount of live, dead, and total cells per milliliter will not be calculated. The dilution factor entered is taken the same for all selected images. When users press the ”Analyze” button, the images are sent to the server for processing. Meanwhile, the user is directed to the waiting screen. This screen is shown in 11b.

From this screen, users can see how many of their photographs have been processed. Photographs are added to a Form Data object to be sent to the server. This object is sent via a

POST request to the ”/api” address of the website. Each photograph is sent to the server separately and in parallel. This prevents the user from waiting when processing multiple photos. The processing of the scheme is the same as the website version of the app and shown in the 8. The processed photos and cell counts are returned to the mobile app from the web server in JSON format. The mobile app processes the cell data using the entered dilution factor value. After all photographs have been processed, the user is redirected to the result screen. The result screen is shown in 11c. On this screen, users can see the photos they have selected and their results. If more than one photo has been submitted, they can also view the results of other photos by scrolling down the screen. They can return to the main page by pressing the ”Analyze” button at the bottom of the page.

4 Work-Month Schedule

	Months					
Description of Work	**September**	**October**	**November**	**December**	**January**	**February**
Literature Review	X	X	X			
Collection and Marking of Photographs		X	X	X		
Investigation of the Models		X	X	X		
Training of Models				X	X	
Writing the codes for web server and mobile app and designing the user interface				X	X	
Testing the codes and bug fixing				X	X	
Data collection and analysis					X	X
Writing of the paper					X	X

5 Findings

In the first phase of the study, different models and configurations were tested while selecting the model to be used. GIoU, objectness, classification loss functions and precision, recall, mAP@.5 and mAP@.5:.95 metrics were used to compare the performance of the models. At the end of training, the performance of the models was evaluated on the test

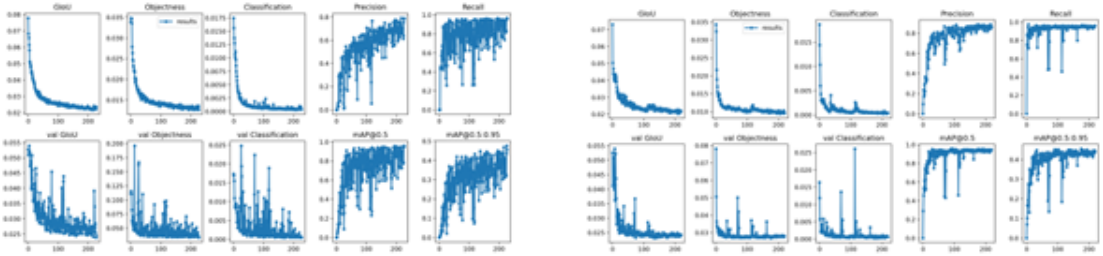
dataset and these values were tabulated. 12 shows the best performances of the models.

Model Number	Model Name	Batch Size	Epochs	Input Size	Mutli-scale	Transfer learning	Precision	Recall	mAP@.5	mAP@[.5:.95]
1	YOLOv4-CSP	4	211	1024	True	False	0.691	0.958	0.941	0.426
2	YOLOv4-CSP	8	151	800	True	False	0.6437	0.9553	0.9407	0.4355
3	YOLOv4-CSP	12	490	928	False	False	0.7996	0.9458	0.9484	0.4607
4	YOLOv4-P5	12	228	832	False	False	0.791	0.96	0.955	0.455
5	YOLOv4-P6	8	122	1056	False	False	0.753	0.9516	0.9429	0.437
6	YOLOv4-P7	8	193	1024	False	True	0.865	0.961	0.958	0.451
7	YOLOv4-P6	8	126	1024	False	True	0.8617	0.9706	0.9617	0.4583

Figure 12: Table of Model Results

When the data in the table is analyzed, it is observed that the models without the multi-scale data augmentation method perform better than the models with the multi-scale option turned on. This is thought to be due to the extra GPU space required by the multi-scale option during training. Turning on the multi-scaling option causes the batch size and the image size to be reduced, thus decreasing the performance of the model. It is possible that on a GPU with a larger memory, this data augmentation method may yield better results.

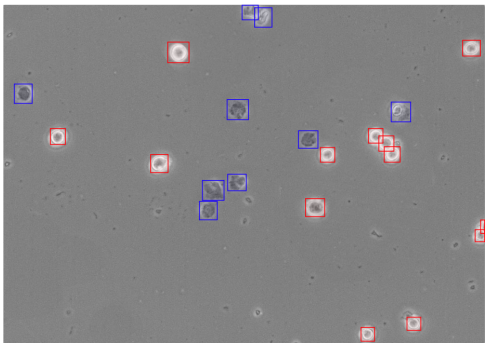
In all the trained models, it was found that precision was lower than recall. This indicates that most of the models had no problem recognizing the cells, but incorrectly predicted some impurities as cells. 13 compares the results of the model without transfer learning and the model with transfer learning.



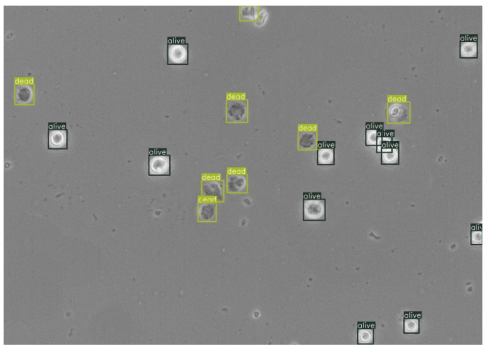
(a) Transfer learning not enabled model number 4 (b) Transfer learning enabled model number 6

Figure 13: Training graphs

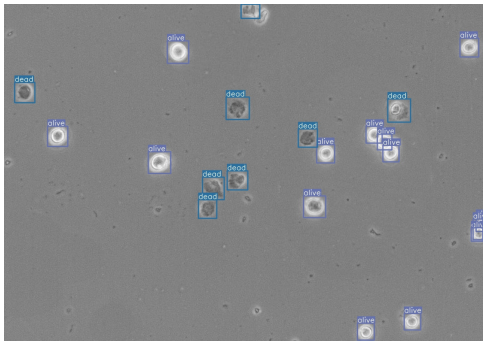
When the results are analyzed, it is seen that transfer learning enabled models learn faster and give better results. While the non-transfer learning models were being trained, there were sudden decreases between epochs in the results of the models. However, as the number of epochs trained increased, the frequency of these sudden drops started to decrease. In the models with transfer learning, although there were sometimes sudden drops between epochs, these drops were not as frequent and not as large as in the models without transfer learning. 14 shows the ground truth for a sample image and the predictions of all the models in the table for the sample image.



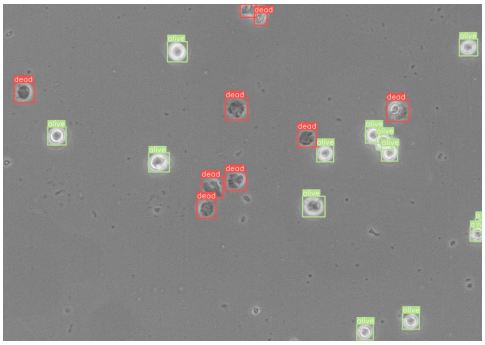
(a) Ground truth



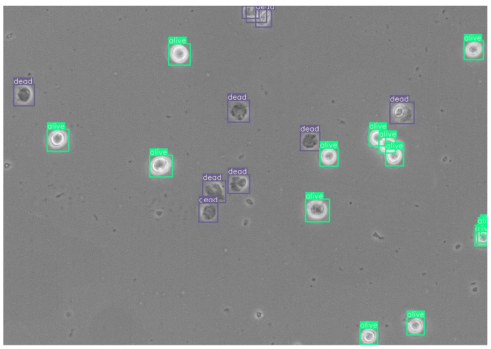
(b) Model 1



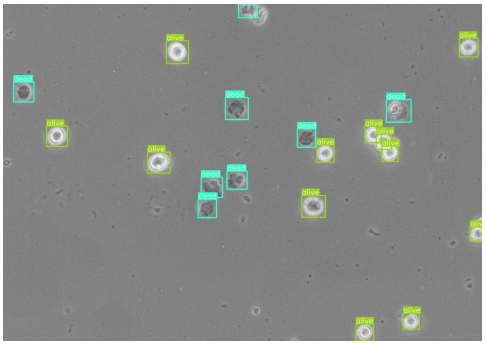
(c) Model 2



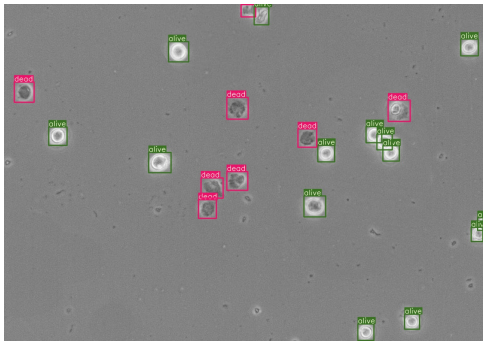
(d) Model 3



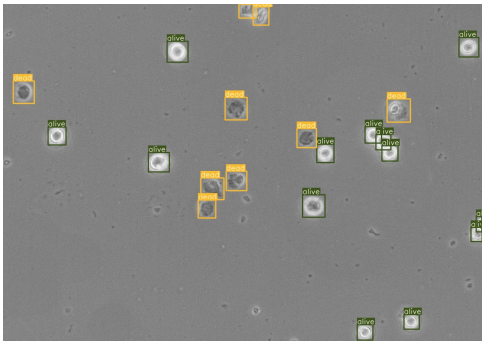
(e) Model 4



(f) Model 5



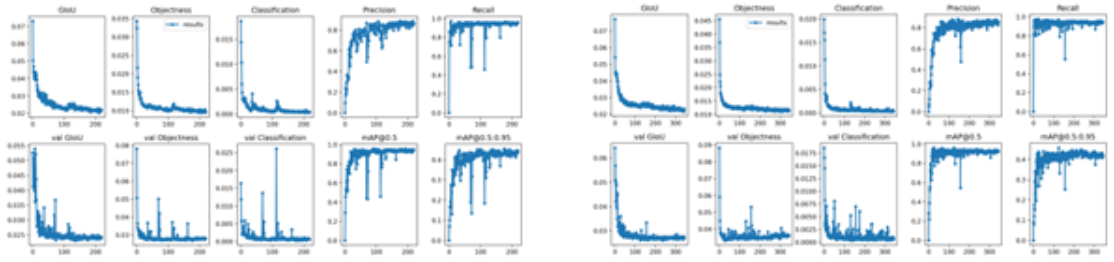
(g) Model 6



(h) Model 7

Figure 14

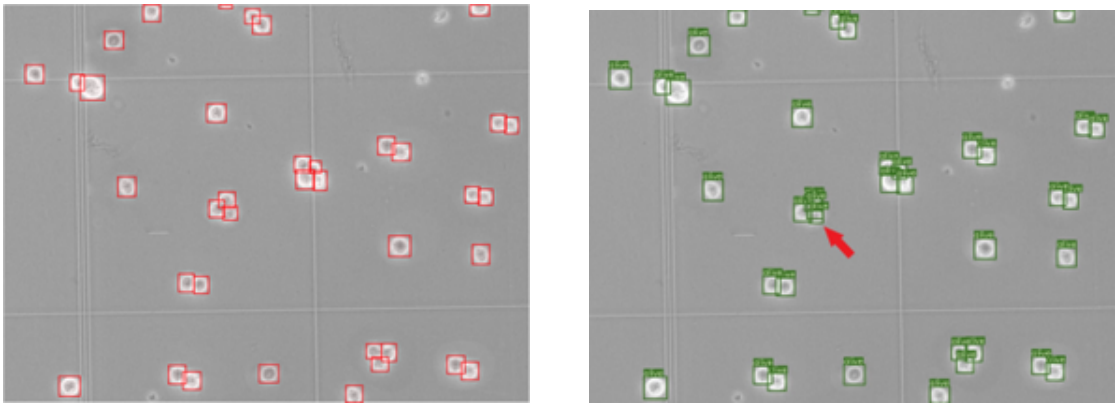
When all the results were analyzed, the best performing models were found to be model number 6 YOLOv4-P5 and model number 7 YOLOv4-P6 from the YOLOv4-Large models on which transfer learning was performed. The training graphs of these two models are shown in 15.



(a) Model 6 (b) Model 7

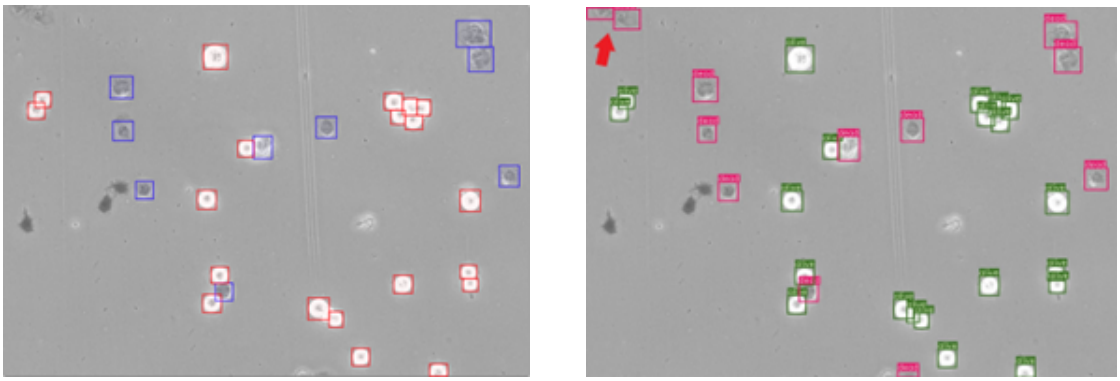
Figure 15: Training Graphs

Looking at the graphs and results of these models, it is seen that there is not much difference between them. Due to the small difference between the performance of the models, model number 6 YOLOv4-P5 was chosen as it works faster than model number 7 YOLOv4-P6. 16 and 17 show the ground truth and the prediction of the YOLOv4-P5 model in a sample image.



(a) Ground Truth (b) Prediction of Model 6

Figure 16



(a) Ground Truth (b) Prediction of Model 6

Figure 17

When the predictions of the model were analyzed, it was seen that it detected live and dead cells with high accuracy. However, the model had difficulty in detecting some clumped and dead cells. Although clumped cells are usually detected correctly by the model, they can sometimes be over or under counted. As can be seen in 16, the model correctly counted most of the clumped cells, but the group of cells marked with a red arrow was overcounted by one cell compared to the ground truth. Dead cells can be confused with blobs because they are not as bright as live cells and their contrast with the background is low. In 17, the red arrow marks a group of blobs that the model counted as cells.

The counting speed of the developed model was found to be 0.06 seconds for a 1280x900 resolution photo on the Tesla P100 GPU. Although the counting speed of a researcher varies according to the number of cells in the photo, it is approximately 20 seconds. The model developed in the study is 300 times faster than manual counting.

6 Conclusion and Discussion

A machine learning-based website and mobile application that detects live and dead cells on the photographs, thus automating cell counting and providing time and convenience has been developed. Apart from the developed work, various hardware and software solutions have been found aiming to automate cell counting. Some of this hardware are automatic cell counting machines. However, to use these machines, expensive and nonportable hardware is needed. Apart from automatic counting machines, there are image processing software such as ImageJ, ilastik, ImagePro, Metamorph, CellC and CellProfiler. However, image processing software requires knowledge of image processing and experience with the application to use this software. No application that enables automatic cell counting on mobile devices has been found in the literature. In the first phase of this study, classical image processing methods such as Otsu thresholding and watershed segmentation were tested for counting, but

higher accuracy was achieved using machine learning compared to standard image processing techniques. A web server was developed for users to easily access the trained machine learning model. A website and a mobile application were written to access the web server. To use the developed application, a device with internet access is sufficient and there is no need to download additional software. The image processing codes run on the server, not requiring any additional hardware. There is also no need for any prior knowledge or additional configuration to use the application. It is enough for users to upload their photos. The developed application was observed to be more accessible and user-friendly than alternative methods.

7 Suggestions

For the next phase of this study, it is aimed to expand the cell types that can be counted. It is planned to train new machine learning models with photos of cells stained with different dyes other than trypan blue. In addition, it is considered to train and incorporate different machine learning models for evaluating electrophoresis images of nucleic acids (DNA, RNA) and proteins, western blot images, and for quantitative analysis of bands in gels, into the web server and mobile application. The system can be developed to include more than one machine learning model and the photos can be pre-analyzed to find machine learning model in the system will give the best performance. Thus, all cell counting can be performed easily, cheaply and quickly on a single platform without the need for extra equipment.

References

[1] Teicher et al. *Anticancer Drug Development Guide: Preclinical Screening, Clinical Trials, and Approval*. Jan. 2004. ISBN: 978-1-4684-9841-7. DOI: 10.1007/978-1-59259-739-0.

[2] National Center for Biotechnology Information. *PubChem Compound Summary for CID 5904246, CID 5904246*. https://pubchem.ncbi.nlm.nih.gov/compound/5904246. 2022.

[3] Louis et al. "Cell Viability Analysis Using Trypan Blue: Manual and Automated Methods". In: *Methods in molecular biology (Clifton, N.J.)* 740 (Mar. 2011), pp. 7–12. DOI: 10.1007/978-1-61779-108-6_2.

[4] Sigma-Aldrich Protocol. *Cell Culture Protocol 6: Cell Counting Using a Hemocytometer*. https://www.sigmaaldrich.com/TR/en/technical-documents/technical-article/cell-culture-and-cell-culture-analysis/mammalian-cell-culture/cell-quantification.

[5] S. Zhang and J. R. Kuhn. *Cell isolation and culture*. https://medium.com/analytics-vidhya/non-max-suppression-nms-6623e6572536. 2018.

[6] Warren Strober. "Trypan blue exclusion test of cell viability." In: *Current protocols in immunology* Appendix 3 (2001), Appendix 3B.

[7] Grishagin and Ivan. "Automatic Cell Counting with ImageJ." In: *Analytical biochemistry* 473 (Dec. 2014). DOI: 10.1016/j.ab.2014.12.007.

[8] Aung et al. "Automatic Counting for Live and Dead Cells from Trypan Blue-Stained Images by Image Analysis Based on Adaptive K-Means Clustering". In: *Journal of Computer Science* 15 (Feb. 2019), pp. 302–312. DOI: 10.3844/jcssp.2019.302.312.

[9] A. Özkan. *An Alternative Image Processing Approach For The Viability Of Cells By Light Microscopy (Doctoral Dissertation)*. 2011.

[10] Thermo Fisher Scientific. *Countess™ 3 Automated Cell Counter User Manual*. https://www.thermofisher.com/tr/en/home/life-science/cell-analysis/cell-analysis-instruments/automated-cell-counters.html. 2022.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60 (2012), pp. 84 –90.

[12] Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: June 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[13] Redmon et al. “YOLO9000: Better, Faster, Stronger”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[14] Redmon et al. “YOLOv3: An Incremental Improvement”. In: (Apr. 2018).

[15] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. In: *ArXiv* abs/2004.10934 (2020).

[16] Wang et al. “Scaled-YOLOv4: Scaling Cross Stage Partial Network”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 13024–13033. DOI: 10.1109/CVPR46437.2021.01283.

[17] Rezatofighi et al. “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 658–666. DOI: 10.1109/CVPR.2019.00075.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Backpropagation and other differentiation algorithms*. 2016.

[19] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *ArXiv* abs/1609.04747 (2016).

[20] Bottou et al. “The Tradeoffs of Large Scale Learning.” In: vol. 20. Jan. 2007.

[21] V. S Subramanyam. *Non-Max Suppression (NMS)*. https://medium.com/analytics-vidhya/non-max-suppression-nms-6623e6572536. 2021.

[22] K. P. Shung. *Accuracy, Precision, Recall or F1?* https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. 2018.

[23] S. Yohanandan. *mAP (mean Average Precision) can confuse you!* https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2. 2020.

[24] George Karimpanal et al. “Self-organizing maps for storage and transfer of knowledge in reinforcement learning”. In: *Adaptive Behavior* 27 (Dec. 2018), p. 105971231881856. DOI: 10.1177/1059712318818568.

[25] R. Fisher et al. *Grayscale Images*. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm.

[26] K Wiggers. *Intel makes CVAT, a toolkit for data tagging*. https://venturebeat.com/2019/03/05/intel-open-sources-cvat-a-toolkit-for-data-labeling/. 2019.

[27] S. Pohkrel. *Image Data Labelling and Annotation - Everything you need to know*. https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1. 2020.

[28] W. K. Yui. *Scaled YOLOv4*. https://github.com/WongKinYiu/ScaledYOLOv4. 2020.

[29] G. Jocher. *Rectangular Inference*. https://github.com/ultralytics/yolov3/issues/232. 2019.