Application of AI-Enhanced Image Processing Methods for Educational Applied Physics Experiments

Eivin Laukhammer
Vilnius Kolegija, Faculty of Electronics and Informatics
J. Jasinskio st. 15, LT- 01111 Vilnius, Lithuania
E-mail: e.macerauskas@eif.viko.lt

Eugenijus Mačerauskas
Vilnius Kolegija, Faculty of Electronics and Informatics
J. Jasinskio st. 15, LT- 01111 Vilnius, Lithuania
E-mail: e.macerauskas@eif.viko.lt

Andžej Lučun
Vilnius Kolegija, Faculty of Electronics and Informatics
J. Jasinskio st. 15, LT- 01111 Vilnius, Lithuania
E-mail: e.macerauskas@eif.viko.lt

Antoni Kozič
l *Vilnius Kolegija, Faculty of Electronics and Informatics*
J. Jasinskio st. 15, LT- 01111 Vilnius, Lithuania
E-mail: e.macerauskas@eif.viko.lt

Romanas Tumasonis
Vilnius Kolegija, Faculty of Electronics and Informatics
J. Jasinskio st. 15, LT- 01111 Vilnius, Lithuania
E-mail: e.macerauskas@eif.viko.lt

Abstract. **This article examines the use of an AI-powered automated image analysis system. The system's purpose is to enhance the workflow of students during applied physics laboratory experiments, helping them analyze images and perform accurate microobject counting. On the software side, the system incorporates machine learning algorithms for visual processing applications using Python and its' extension libraries – CV2, Tensorflow, Keras, SkLearn etc.. The hardware consists of a camera and microprocessor, which, in conjunction with the image processing software, perform microobject recognition and counting in real-time. The goal is to automate applied physics laboratory experiments in which the counting of microobjects, be it organic or human-made, is usually done manually. During these applied physics laboratory experiments and with the aid of this system, students are exposed to a modern workflow, further preparing them for future work environments, teaching them about process automation, and further increasing their interest in micro-scale related science subjects. Automation using image processing technology combined with automatic data logging from images allows for fast and accurate micro-object counting.**

Keywords: Image Processing, Interdisciplinary Connections, Experiments Automation.

I. INTRODUCTION

Blood cell counting by laboratory utilizes a microscope. This conventional task is laborious and time consuming, and is largely dependent on the physician's skill. Fast and cost-effective production of blood cell count reports is of paramount importance in the health care industry. The traditional method of manual counting under the microscope yields inaccurate results and puts an intolerable amount of stress on the medical laboratory assistants [2]. Automated analysis of medical cell images has been gaining more importance in pharmacology and toxicology practice. Extraction of accurate quantitative data about the cell morphology is a critical task. An automated procedure for cell analysis is highly desirable since there may exist hundreds of images for each patient. In fact, one of the most challenging tasks is to extend the traditional approaches to segmentation and object classification.

In physics laboratories, students usually get their first touch with the microscope, which is a very important part of diagnostic laboratories. As mentioned previously microscopes are usually equipped with advanced software to diagnose, measure, or count specific particles found in patient fluids or tissues. But the first-year medical students are learning those techniques by hand. Working with a microscope using precise methods that are usually very time-consuming for the students as well as professors also that method is becoming outdated having in mind that applied sciences university's goal is to prepare students for future jobs as well as possible. The stained blood smears are typically viewed and identified with an upright brightfield microscope such as ZEISS AxioLab A1 [2].

Print ISSN 1691-5402
Online ISSN 2256-070X
https://doi.org/10.17770/etr2024vol2.8068

Precise medical drawings can still be found in use as reference images today (Figure 1).

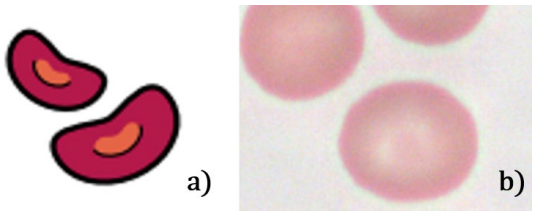

Fig 1. Model (a) + Real Erythrocyte (b) of blood cells

Maximum optical performance can only be achieved through the interaction of all the optical components, such as the lens, condenser, and eyepiece. A microscope camera with a high dynamic range precise image acquisition at the pixel level should be used to document or archive the results [3], [8].

The examination of blood under a microscope is a precise and equally complex procedure. The distinguishing features of the blood cells that are revealed must be clearly displayed in order to be classified correctly by the "eye of the trained observer" during daily routine activities [23], [25].

The traditional microscopic method based on the count of 100 cells has three types of errors: statistical error, distributional error owing to unequal distribution of cells in the smear, and error in identifying cells related to the subjective interpretation of the examiner. This method, therefore, suffers from imprecision, poor accuracy, and reduced clinical sensitivity [14], [15]. An example of an unknown object (Cancer) in blood is shown in Figure 3.

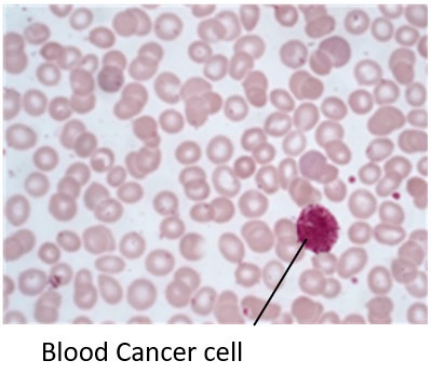

Fig 3. Blood Cancer cell detection example

Image processing applications are very important in medical diagnostics, and incorporating them in education makes the learning process faster and introduces medical students to technologies they will see later in life. With this technology, the motivation of students can be improved because they see the possible opportunities for the future. Figure 4.

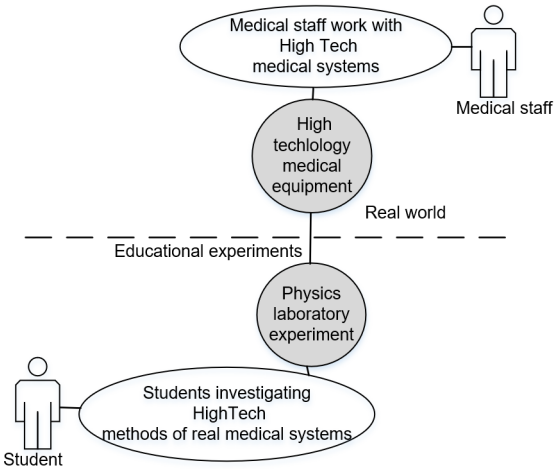

Fig 4. Introducing high-tech methods for the education process to improve student motivation

Due to fast improvements in hardware and software, physics laboratories can be provided with very precise measuring and counting software that uses video or photo analysis.

II. Realization of an Experimental Educational System

The modern information technology application to computer-based experiments is conducted on similar methods. With modern image processing technologies, Python programming, and OpenCV libraries, the system for automated physics laboratory experiments was created (Fig. 5).

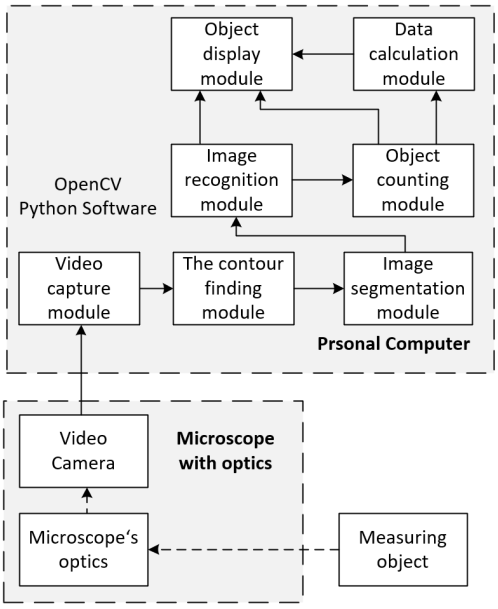

Fig 5. Block diagram for the system

The system consists of created software developed on a personal computer and a hardware measurement system consisting of a microscope and a video camera.

The research object is measured by the comparison method, where the dimensions of the measured object are compared with the dimensions of the reference object. The

resulting image enters the video camera through a microscope, and the camera converts it into a video signal. Several libraries can be utilized for image processing, classification, and counting tasks in Python to achieve different aspects of these tasks. In particular, the machine learning and deep learning libraries (TensorFlow, Keras, PyTorch) would be very useful for classification and counting. They provide algorithms to classify images after being trained on labeled data and perform object detection and counting. OpenCV also supports some machine learning algorithms and can be used with these libraries for such tasks [5], [8], [27].

Fungi classification

In image processing, the classification of fungi involves the use of algorithms and computational techniques to identify and categorize different types of fungi based on images. This process is significant in various fields, such as agriculture, food safety, environmental monitoring, and medical diagnosis. Image processing for fungi classification [6], [13], [17], [21].

Os: This is a standard Python library for interacting with the operating system. It provides functions for file and directory operations.

NumPy (np): NumPy is a fundamental package for numerical computing in Python. It supports large, multi-dimensional arrays and matrices and a collection of mathematical functions to operate on these arrays.

matplotlib.pyplot (plt): Matplotlib is a plotting library for Python. Pyplot is a module in Matplotlib that provides a MATLAB-like interface for creating and customizing plots.

seaborn (sns): Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

keras.applications.vgg16: Keras Applications provides pre-trained deep learning models for use in various applications. VGG16 is one such pre-trained convolutional neural network (CNN) model that is known for its simplicity and effectiveness.

keras.models: This module in Keras provides the basic building blocks for defining neural network models. It includes classes like Model for defining models and Sequential for creating models' layer by layer.

keras.layers: This module contains various layer implementations that can be used to build neural networks. We incorporated layers such as Flatten, Dense, Dropout, Conv2D, MaxPooling2D, and LeakyReLU.

keras.preprocessing.image: This module provides utilities for preprocessing image data and performing data augmentation. It includes functions for loading images from disk, resizing, and applying transformations.

keras.optimizers: This module contains implementations of various optimization algorithms that can be used to train neural networks. The Adam optimizer is an adaptive learning rate optimization algorithm that is widely used in training deep neural networks. It stands for Adaptive Moment Estimation and combines ideas from two other popular optimization algorithms: AdaGrad and RMSProp.

keras.callbacks: Keras callbacks are objects that can perform actions at various stages of training (e.g., at the start or end of an epoch). Examples include EarlyStopping, ModelCheckpoint, and ReduceLROnPlateau, which help control the training process and prevent overfitting.

sklearn.metrics: This module from scikit-learn provides functions for evaluating the performance of machine learning models.

keras.regularizers: Keras provides support for adding regularization to neural network layers. Regularization techniques like L2 regularization (weight decay) can help prevent overfitting by adding a penalty term to the loss function.

keras.backend (K): This module provides functions that operate on tensors, which are the basic data structures used in neural networks. It abstracts away the backend implementation (e.g., TensorFlow, Theano) and allows for writing code that is backend-agnostic.

cv2: OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. The cv2 module provides functions for image processing, manipulation, and computer vision tasks.

tensorflow (tf): TensorFlow is an open-source deep learning framework developed by Google. Keras can run on top of TensorFlow, providing a high-level interface for building and training neural networks. In our case TensorFlow is being used as the backend for Keras. Dataset [19], [21], [24].

License: For Fungi detection, the DeFungi Dataset was chosen [4], [10], [11]. It is being used under Attribution 4.0 International (CC BY 4.0 DEED) license, which allows us to use the dataset freely and to train our machine learning model based on this dataset.

The dataset consists of 5 classes:

- H1: Candida albicans
- H2: Aspergillus niger
- H3: Trichophyton rubrum
- H5: Trichophyton mentagrophytes
- H6: Epidermophyton floccosum

TABLE 1 DEFUNGI DATASET CLASS DISTRIBUTION

Class	Training set	Validation Set	Test Set
H1	1000	437	437
H2	1000	232	233
H3	1000	81	82
H5	1000	80	80
H6	1000	69	70
Total	**5000**	**899**	**902**

In order for the model to accurately predict a given class from an input image, the dataset needs to be diverse enough so that the model can train on all kinds of scenarios

and accurately predict a class given almost any circumstance: (Fig 6.).

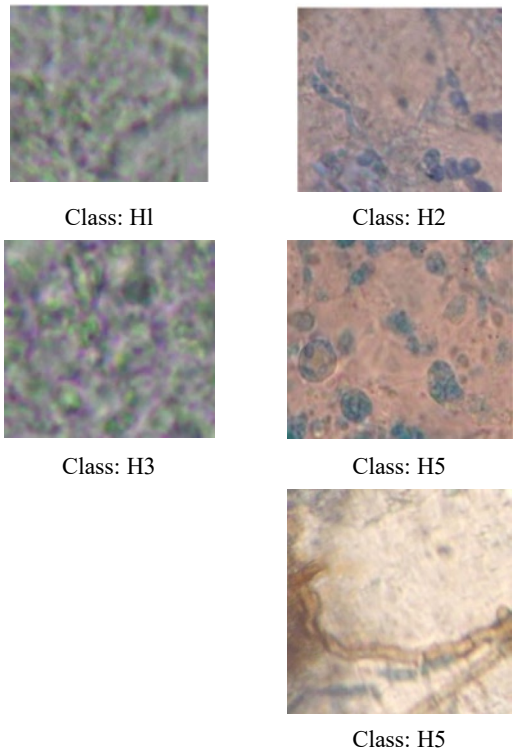

Fig 6. Example images from different class

Curating Challenging Examples: Including challenging examples in the dataset helps the model learn to handle complex scenarios and edge cases. These examples may include occluded objects, objects under different poses, cluttered backgrounds, variations in illumination, and other challenging conditions. By exposing the model to diverse and challenging examples during training, it becomes more resilient and capable of making accurate predictions in real-world scenarios.

Class Balancing Techniques: Addressing class imbalance is essential for ensuring that the model receives sufficient training examples for each class. Techniques such as oversampling minority classes, undersampling majority classes, or using synthetic data generation methods help balance the distribution of samples across classes. This ensures that the model does not become biased towards predicting the majority class.

Data Augmentation: Data augmentation involves applying a variety of transformations to existing images to create new, slightly modified versions of them. Common augmentation techniques include rotation, flipping, scaling, cropping, translation, brightness adjustment, and adding noise. Augmentation helps introduce variability into the dataset and enables the model to learn invariant features across different conditions.

Data augmentation techniques used:

Rescaling (Normalization): The rescale parameter is set to 1/255 for all data generators (train, validation, and test).

Rotation Range: Rotation range is set to 40 degrees. Rotation augmentation helps the model become more robust to variations in object orientations.

Width and Height Shift Range: Width and height shift ranges are set to 20% of the total width and height, respectively. This augmentation technique introduces translations of the image horizontally and vertically.

Shear Range: Shear range is set to 20%, and involves stretching or skewing the image along its x-axis or y-axis.

Zoom Range: Zoom range is set to 20%. Zoom augmentation allows the model to learn from images at different scales.

Horizontal Flip: Horizontal flip is enabled (horizontal_flip=True). Horizontal flipping mirrors the image horizontally, providing additional training examples while maintaining the same class label.

Fill Mode: The fill mode is set to 'reflect'. The fill mode determines the strategy used for filling in newly created pixels that may arise during augmentation, such as after rotation or shifting operations.

In summary, by employing these data augmentation techniques, the model is exposed to a more diverse set of training examples, which helps improve its generalization ability and robustness to variations in input data, ultimately enhancing its performance on unseen data.

Neural Network architecture.

Base Model (VGG16): VGG16 is a pre-trained convolutional neural network architecture that has been trained on the ImageNet dataset [18]. Additional parameter: include top=False parameter means that the fully connected layers (the classifier part) of the VGG16 model are not included [16]. This parameter allows for adding custom fully connected layers on top of the convolutional base. The input shape is set to (64, 64, 3), which is the shape of the dataset images [20], [22].

Neural Network architecture shown in Figure 7.

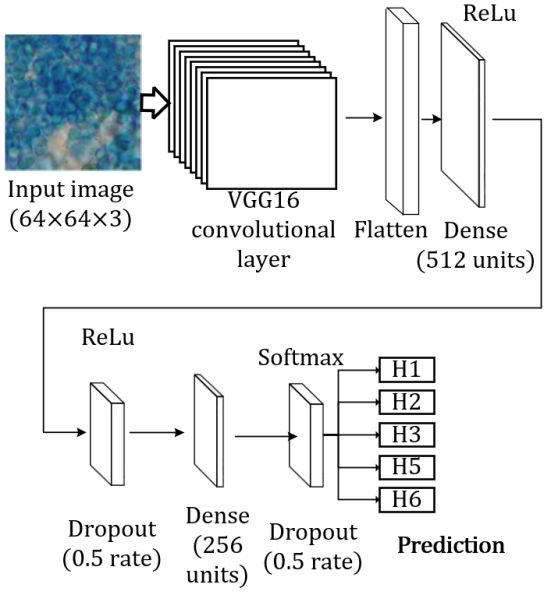

Fig 7. Neural Network architecture

Additional CNN Layers:

- After the base model, there's a Flatten layer to flatten the output of the base model before passing it to the fully connected layers.
- Two Dense layers are added with 512 and 256 units respectively. Each Dense layer is followed by a LeakyReLU activation function with an alpha of 0.1, which helps prevent the dying ReLU problem by allowing a small, positive gradient when the unit is not active.
- Dropout layers with a rate of 0.5 are added after each Dense layer to prevent overfitting by randomly dropping a proportion of the units during training.
- Finally, there's a Dense layer with 5 units (since the model is trained on 5 classes) and softmax activation function to output probabilities for each class.

Freezing Layers: All layers of the pre-trained VGG16 model are frozen, meaning their weights will not be updated during training. This is a common technique in transfer learning to prevent the pre-trained weights from being destroyed [26].

Optimizer: Adam optimizer is used with a learning rate of 0.0001 and a decay rate of 1e-6. The learning rate decay is applied to gradually reduce the learning rate during training, which can help the model converge better towards the end of training. Adam optimizer combines the benefits of AdaGrad and RMSProp optimizers. It maintains per-parameter learning rates and adapts them based on the moving averages of the first and second moments of the gradients.

Callbacks: EarlyStopping monitors the validation loss and stops training if there's no improvement after a certain number of epochs (patience). It helps prevent overfitting.

ReduceLROnPlateau: Reduces the learning rate when the validation loss has stopped improving, which can help the model to converge better.

ModelCheckpoint: Saves the best model based on validation loss.

Training

The model is trained using the mode.fit() method with the training and validation generators. The number of steps per epoch and validation steps are calculated based on the number of samples and batch size of the generators.

To summarize, this architecture is structured for transfer learning with VGG16 as the base model, and it utilizes techniques like dropout regularization, learning rate decay, and early stopping to prevent overfitting and improve convergence.

Model evaluation

Models' performance was evaluated by generating a classification report and confusion matrix.

TABLE 2 CLASSIFICATION REPORT

	precision	recall	f1-score	support
H1	0.49	0.54	0.52	437
H2	0.32	0.21	0.26	233
H3	0.09	0.10	0.09	82
H5	0.13	0.16	0.14	80
H6	0.10	0.10	0.10	70
accuracy			0.35	902
macro avg	0.23	0.22	0.22	902
weighted avg	0.35	0.35	0.35	902

From the classification report we can see that the H1 class was most effectively trained, producing the highest f1-score out of all evaluated classes. These results also show that the dataset has a class imbalance that the data augmentation techniques were not able to account for completely. The culprit is most likely that the training images were not that similar to the validation images, therefore resulting in a somewhat poor classification capability.

Train images examples for H1 shown in Figure 8.

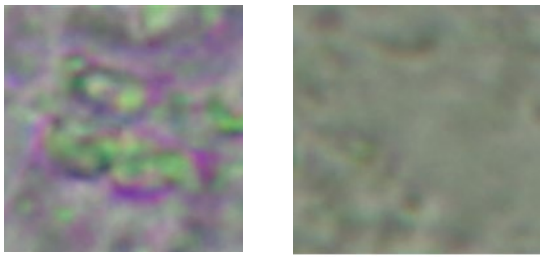

Fig. 8. H1 Train image examples

From the classification report we can see that the H1 class was most effectively trained, producing the highest f1-score out of all evaluated classes. These results also show, that the dataset has a class imbalance that the data augmentation techniques were not able to completely account for. The culprit is most likely that the training images were not that similar to the validation images, therefore resulting in a somewhat poor classification capability (Fig. 9).

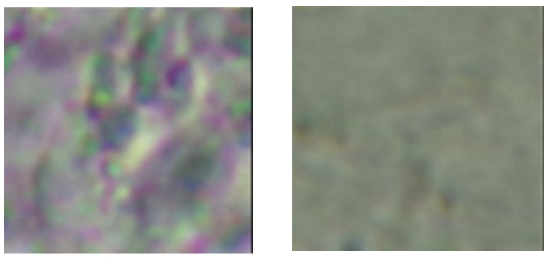

Fig. 9. H1 Validation image examples

The other issue with the dataset is it's image size. One image is 64 by 64 pixels, which is fairly small. An image upscaling algorithm could be used to increase the fidelity of the images, though the problem would be finding an algorithm that would properly work on the provided image type, since most upscales are trained on more common types of objects. The confusion matrix shown in Figure 10.

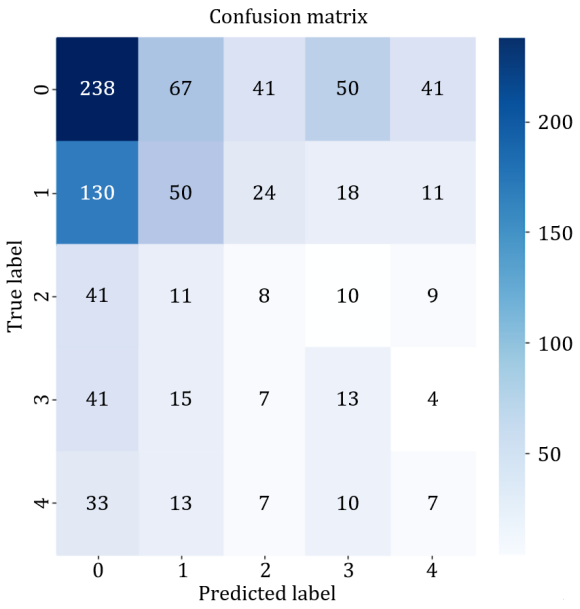

Fig 10. Confusion Matrix

Confusion matrix class labels – 0 is H1, 1 is H2, 2 is H3, 3 is H5, 4 is H6.

Some insights from the confusion matrix:

- The overall accuracy of the classification algorithm is 69%, which is calculated by dividing the sum of the diagonal cells by the total number of data points.
- The algorithm is most accurate at classifying H1, with an accuracy of 81%.
- The algorithm is least accurate at classifying H5, with an accuracy of 40%.
- The algorithm is more likely to misclassify H1 as H2 and H3 as H5.

The model accuracy vs epochs graph shows the Figure 11. Increasing accuracy of the model after a given number of epochs.

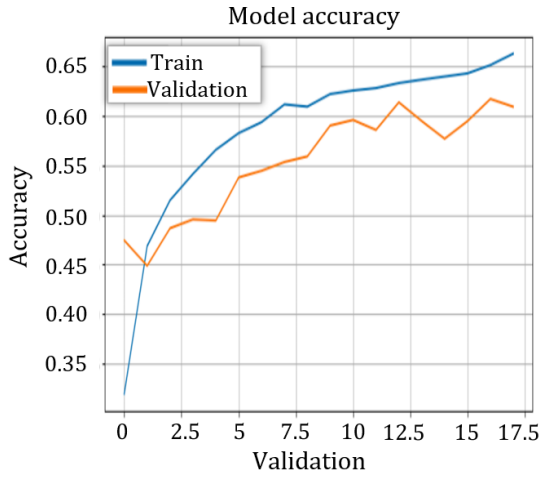

Fig 11. Model accuracy vs Epoch

The largest increase in accuracy occurs during the first 5 training epochs, after which the accuracy evens out. The blue line represents the training data and the orange line represents the validation data. Looking at the validation data line, we can yet again confirm the imbalance of the dataset, seeing that the model is not quite capable of easily increasing it's validation accuracy after each training epoch, as it doesn't always increase.

III. Results

Measurement example:

In applying artificial intelligence, a measuring system was utilized to conduct initial practical experiments in applied physics. The system counted cells with 93% accuracy compared to visually counted method in classic way. The example of computer screen during laboratory experiment, presented in Figure 12.

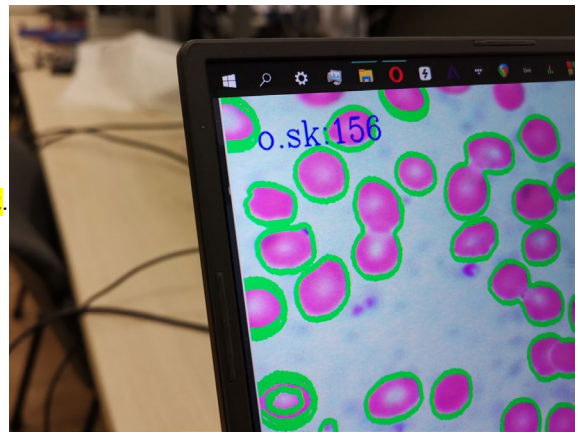

Fig 2.

Fig 12. Cells count = 156

One of the most important aspects, in terms of the application of automated image analysis based on dedicated algorithms, is the time required for image analysis. Traditional techniques for the quantitative determination are universal and efficient but, simultaneously, time-consuming (the classical manual method would take at least 50% longer).

The Figure 13 how students of different courses and different specialties can cooperate in an interdisciplinary way designing AI-based system and during applied physics experiments.

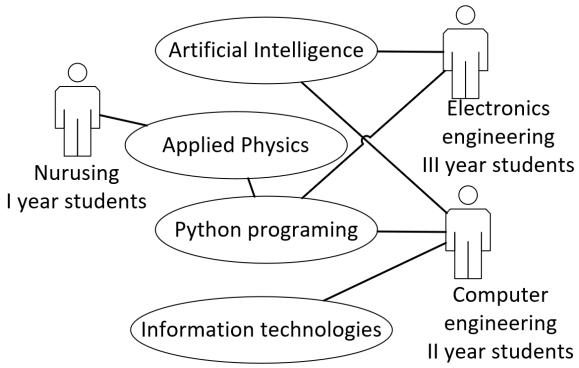

Fig 13. The interdisciplinary connections between different subjects.

The physics experiment of medical microscopy is designed for I year Nursing students who study Applied Physics. The III-year Electronics Engineering students participate in designing the physics experiment hardware and software. While developing the Applied physics

experiment system, they study subjects such as Artificial Intelligence and Python programming. Together with II-year Computer Engineering students, they create the software and hardware for the experiment. In this way, students of different specialties and different years not only learn the knowledge of their subject during their studies but also apply it to improve the study process on an interdisciplinary principle.

IV. CONCLUSION

The following conclusions can be made from the analysis of the application of the AI- based physics laboratory experiment developed by the students:

- AI-based Image processing applications for physics laboratory experiments facilitate the experimentation process and reduce the time spent on the analysis.
- During the first year, the student understands the practical benefits of the individual studies, and this option increases student motivation
- The automated system not only increased the interest of the students in experimental work, automation, and application of new technologies but also facilitated the professors' work by performing complex laboratory experiments that require a lot of precision work

AI-based image analysis methods can be successfully used for accurate, efficient and fast image analysis and widely applied in the learning process.

A comparison of traditional and automated image processing methods allows students to more easily understand the essence of modern research, to highlight the advantages and disadvantages of the methods.

ACKNOWLEDGMENT

This work has been supported by the Research Council of Lithuania (Project No. P-ST-23-168).

REFERENCES

[1] J. Aaron, T.L. Chrew, (2021), A guide to accurate reporting in digital image processing – can anyone reproduce your quantitative analysis? Journal of Cell Science, Volume 134, Issue 6. https://doi.org/10.1242/jcs.254151

[2] M. Buttarello and M. Plebani. Automated Blood Cell Counts: State of the Art. American Journal of Clinical Pathology, 130(1):104–116, 2008.

[3] I. Cinar, Y. S. Taspinar, Detection of Fungal Infections from Microscopic Fungal Images Using Deep Learning Techniques, proceedings of 11th International Conference on Advanced Technologies (ICAT'23), Istanbul-Turkiye, August 17-19, 2023. [DeFungi]

[4] Defungi Dataset https://ui.adsabs.harvard.edu/abs/2021arXiv210907322P/ [Accessed: Dec. 17, 2023]

[5] I. Goodfellow, Y. Bengio, A. Courville, & Y. Bengio, (2016). Deep learning (Vol. 1). MIT press Cambridge.

[6] A. Gulli, & S. Pal, (2017). Deep learning with Keras. Packt Publishing Ltd.

[7] R. Hayes, G. Pisano, D. Upton, and S. Wheelwright, Operations, Strategy, and Technology: Pursuing the competitive edge. Hoboken, NJ: Wiley, 2005.

[8] K. He, X. Zhang, S. Ren, & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[9] B. Herman and J. J. Lemasters (eds.), Optical Microscopy: Emerging Methods and Applications. Academic Press, New York, 1993, 441 pp.

[10] L. Howell et al. (2021), Multi-Object Detector YOLOv4-Tiny Enables High-Throughput Combinatorial and Spatially-Resolved Sorting of Cells in Microdroplets, Advanced Materials Technologies https://doi.org/10.1002/admt.202101053

[11] R. Lücking, S. D. Leavitt, & Hawksworth, D.L. Species in lichen-forming fungi: balancing between conceptual and practical considerations, and between phenotype and phylogenomics. Fungal Diversity 109, 99–154 (2021). https://doi.org/10.1007/s13225-021-00477-7

[12] B. Matsumoto Cell Biological Applications of Confocal Microscopy, Second Edition (Methods in Cell Biology) (Methods in Cell Biology, 70).

[13] L. Picek, M. Šulc, J. Matas, J. Heilmann-Clausen, T.S. Jeppesen, E. Lind. (2022), Automatic Fungi Recognition: Deep Learning Meets Mycology. Sensors. 2022; 22(2):633. https://doi.org/10.3390/s22020633

[14] J. Poomcokrak and C. Neatpisarnvanit. Red Blood Cell Extraction and Counting. In Proceedings of the 3rd International Symposium on Biomedical Engineering, 199–203, 2008.

[15] G. P. M. Priyanka, O. W. Senevirante, R. K. O. H Silva, and W. V. D. Soysa amd C. R. De Silva. An Extensible Computer Vision Application For Blood Cell Recognition And Analysis, 2006

[16] H. Qassim, A. Verma and D. Feinzimer, "Compressed residual-VGG16 CNN model for big data places image recognition," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2018, pp. 169-175, doi: 10.1109/CCWC.2018.8301729. [VGG16]

[17] A. Rohani, M. Mamarabadi, Free alignment classification of dikarya fungi using some machine learning methods. Neural Comput & Applic 31, 6995–7016 (2019). https://doi.org/10.1007/s00521-018-3539-5

[18] O. Ronneberger, P. Fischer, & T. Brox, (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241).

[19] S. A. Sanchez et al., 2020 IOP Conf. Ser.: Mater. Sci. Eng. 844 012024. doi 10.1088/1757-899X/844/1/012024 [tensorflow]

[20] T. Shi, N. Boutry, Y. Xu and T. Géraud, (2022) Local Intensity Order Transformation for Robust Curvilinear Object Segmentation, in IEEE Transactions on Image Processing, vol. 31, pp. 2557-2569, 2022, doi: 10.1109/TIP.2022.3155954.

[21] F. K. Shler, B. A. Sallow, (2022), Disease Diagnosis Systems Using Machine Learning and Deep learning Techniques Based on TensorFlow Toolkit: A review. AL-Rafidain Journal of Computer Sciences and Mathematics, 16 (1), 111-120. doi: 10.33899/csmj.2022.174415

[22] K. Simonyan, & A. Zisserman. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[23] Y. Skaf, R. Laubenbacher, "Topological data analysis in biomedicine: A review, Journal of Biomedical Informatics,Volume 130, 2022, 104082,mISSN 1532-0464, https://doi.org/10.1016/j.jbi.2022.104082.

[24] S. S. Sumit et al 2020 J. Phys.: Conf. Ser. 1529 042086, doi 10.1088/1742-6596/1529/4/042086

[25] O. Wu, F. Merchant, K. Castleman (2008). Microscope image processing. Academic Press is an imprint of Elsevier, UK, 2008. ISBN13 9780123725783

[26] H.Yang,, J. Ni, , J. Gao, et al. A novel method for peanut variety identification and classification by Improved VGG16. Sci Rep 11, 15756 (2021). https://doi.org/10.1038/s41598-021-95240-y [VGG16]

[27] A. Gulli, & S. Pal, (2017). Deep learning with Keras. Packt Publishing Ltd.