

# Image-based Cell Cycle Phase Classification using Machine-Learning Techniques

École Polytechnique Fédérale de Lausanne

*Supervisor:* Léo BÜRGY, Gönczy Lab

Elisa BILLARD  
Life Sciences Engineering Section

Souleyman BOUDOUE  
Computer Science Section

François DUMONCEL  
Data Science Section

**Abstract**—This study investigates the utilization of machine learning and deep learning methodologies for classifying cell nucleus images across the stages of the cell cycle, specifically G1, S, G2, and M phases. Confronting the constraints posed by a limited dataset, our methodology exhibits encouraging outcomes, as evidenced by an achieved accuracy rate of 70%. This was possible using a Convolutional Neural Network with few layers. In the future, the availability of a more substantial dataset would facilitate the development of an enhanced model, potentially yielding higher accuracy rates.

## I. INTRODUCTION

The monitoring of cell cycle phases in cellular imaging relies on the use of cell cycle markers. These markers typically localize to the nucleus and exhibit varying expression levels, based on the specific cell cycle phase. While these markers prove practical for numerous experiments, it would be best to avoid the need for them by developing an unbiased method for classifying cell cycle phases. Some attempts to avoid staining were based on measuring and quantifying the DNA content of the cells by fluorescence microscopy [1]. However, with this method the G2 and M phase were not differentiated and therefore is not sufficient for our need. In the context of research on centriole copy number, the information on the cell cycle state is very insightful. Centrioles are crucial in fundamental cellular processes, including signaling, motility, and division [2]. The copy number of centrioles is carefully regulated, since defects in their number and structure can cause diseases, including ciliopathies and cancers [3]. The data we have available are 493 fluorescence microscope images of cells stained for the nucleus and labeled G1, S, G2 or M. To address the challenge of classifying the images, we developed a deep-learning

pipeline using a convolutional neural network with an architecture inspired from a research conducting a similar task [4].

## II. PRE-PROCESSING

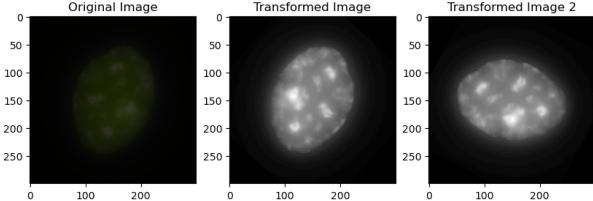
The original data set comprises a total of 493 images depicting cells in a singular phase of the cell cycle. These images have been obtained with DNA staining (DAPI), which therefore stains the nucleus. The dataset we were given is not balanced because the cells spend more time in G1 and S phase than in G2. This is even more drastic for the M phase as this is just the short time before the division. The images are distributed as follows:

Phase	Fraction	Percentage
G1	257/493	52,12%
S	166/493	33,67%
G2	49/493	9,94%
M	21/493	4,26%

**Table I:** Classes Distribution in Dataset

### A. Data augmentation

In order to mitigate over-fitting during training and ensure robust model performance, we implemented data augmentation techniques on our data set. Data augmentation is a technique used to artificially increase the size of a training data set by applying various transformations. The purpose is to introduce diversity and variability into the training set, helping the model generalize better to unseen data. Specifically, we applied random rotations, flips, Gaussian blur to augment the dataset. Because the cell color is artificial, we made the model staining agnostic, either by color jittering,



**Figure 1:** Example of rotation and gray-scaling transformations

or switching to gray scale (see Figures 1, 4). We did not use transformations that modified the images too much, like zooms or crops, as they are already quite hard to distinguish and would not represent any real biological potential transformation.

### B. Dataset balancing

The original dataset being very unbalanced, we decided to up-sample the under-represented classes using data augmentation. The reason is that models trained with unbalanced datasets tend to favor the majority class, leading to biased predictions and inaccurate metrics. This imbalance can hinder the model's ability to generalize and accurately predict minority classes. To be able to validate the effect of augmenting and balancing out data we trained our models on three different datasets:

- Dataset  $\mathcal{D}_1$ : raw dataset
- Dataset  $\mathcal{D}_2$ : augmented dataset
- Dataset  $\mathcal{D}_3$ : balanced augmented dataset (using upsampling)

## III. MACHINE LEARNING MODELS

To classify in the best way possible these images, we have tried several Machine Learning techniques, from the very simple Logistic Regression, to more complex Convolutional Neural Networks. The aim is to compare the performance of each of those techniques on our data.

### A. Multi-class Logistic Regression

Logistic regression is a statistical method often used for binary classification tasks, but it can be extended to handle multi-class classification problems. The idea is to construct a linear predictor function that constructs a score from a set of weights that are linearly combined

with the features (i.e pixel of pictures) of a given observation:

$$\text{score}(\mathbf{X}_i, k) = \boldsymbol{\beta}_k \cdot \mathbf{X}_i$$

where  $\mathbf{X}_i$  is the vector of explanatory variables describing observation  $i$ ,  $\boldsymbol{\beta}_k$  is a vector of weights (or regression coefficients) corresponding to the outcome  $k$  (here  $k \in \{0, 1, 2, 3\}$ ) and  $\text{score}(\mathbf{X}_i, k)$  is the score associated with assigning observation  $i$  to category  $k$ . Once we have learned the weights (e.g. using Gradient Descent), one can use the model to predict the probability that a fresh sample belong to each class

$$\mathbb{P}(Y_i = k) = \frac{e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}{1 + \sum_{j=0}^{K-1} e^{\boldsymbol{\beta}_j \cdot \mathbf{X}_i}} , \quad k < K = 4$$

The final prediction is simply

$$\hat{k}_{LR} = \arg \max_{k \in \{0, 1, 2, 3\}} \mathbb{P}(Y_i = k)$$

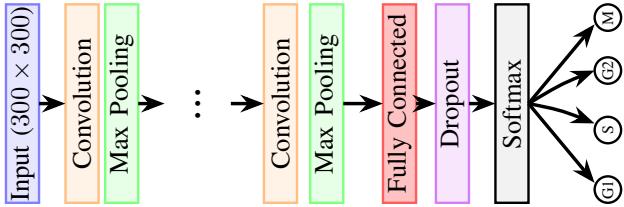
### B. Simple Neural Network

We then moved on to performing a simple Neural Network (NN) with a few layers to classify our data. A Neural Network consists of interconnected nodes organized into layers. NN are trained on data to learn patterns and relationships, adjusting the weights of connections to make accurate predictions or classifications. They are usually not used on image classification tasks as they require a flat input, meaning the image will be transformed into one flat vector, thereby losing the positional information.

### C. ResNet

In an attempt to learn better features for the images we tried training a Convolutional Neural Network with the successful ResNet architecture [5]. The key innovation of ResNet is the use of residual blocks, which contain skip connections that allow the network to skip one or more layers during training. These shortcuts enable the flow of information to bypass certain layers, making it easier to train deep networks. This normally leads to improved accuracy and performance in tasks such as image classification.

We experiment with different versions of ResNet by tuning the depth and block types (Residual and Non Residual). We observed that deeper networks had trouble generalizing well as the performance did not improve with more layers. Moreover, narrower models performed better without skip connections, whereas deeper networks seemed to benefit from having residual blocks.



**Figure 2:** Nagao CNN Architecture

#### D. Nagao’s Convolutional Neural Network

We also investigate the effectiveness of the architecture proposed by Nagao et al. [4], originally designed for the G2/not G2 binary classification task. The CNN they present in their work is comprised of an input convolutional layer, 2 to 50 stages of blocks comprised of a convolutional layer and a max-pooling layer each, finally ending in a fully connected classifier. They also propose to add a Dropout layer, and set the dropout rate to 0, 5.

We reuse this architecture and explore the design space, i.e. depth and width, we also take care to modify the last fully-connected layers to handle our 4-labels classification task.

#### E. Transfer learning

We also experimented with transfer learning. Specifically, we adopted Nagao’s model, trained it on the G2/Not G2 dataset reaching 93% accuracy on their binary classification task, and finally, finetuned the model on our own dataset. Although this approach did not yield optimal results, we incorporated the architecture of their model into the design of our own, which ultimately proved to be the most effective strategy.

#### F. Hyperparameter and Architecture Tuning

Using cross-validation, we tuned the hyperparameters of the different models we tested. An important hyperparameter is the learning rate, we found  $5 \times 10^{-4}$ . For the batch size we chose 16 as our dataset is limited in size. Then, for the ResNet class of CNN we observed that more residual blocks decrease the accuracy, due to the model failing to generalize with the few data we have, so we chose the smallest architecture ResNet8 and removed the skip-connections as narrower networks did not seem to benefit from residual blocks III. Overall, our best model was a modification of

the Nagao architecture, using 6 layers of convolution offering a good balance between accuracy and  $F_1$  score. (Figure 7).

## IV. RESULTS

We now inspect the outcomes derived from the application of our models to the datasets  $\mathcal{D}_1$ ,  $\mathcal{D}_2$  and  $\mathcal{D}_3$ . In Table II, we can see that the CNN with the Nagao

Model	Accuracy			Avg. Accuracy per Class				$F_1$
	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$G_1$	$S$	$G_2$	$M$	
LR	66%	61%	64%	78%	44.7%	4.6%	75%	0.53
NN	67%	62%	61%	80.7%	49.8%	16%	20%	0.33
ResNet 8	62.2%	69%	61%	83%	42.3%	16.5%	43%	0.60
Nagao 6	70%	68%	74%	78.5%	38.4%	53.6%	100%	0.71

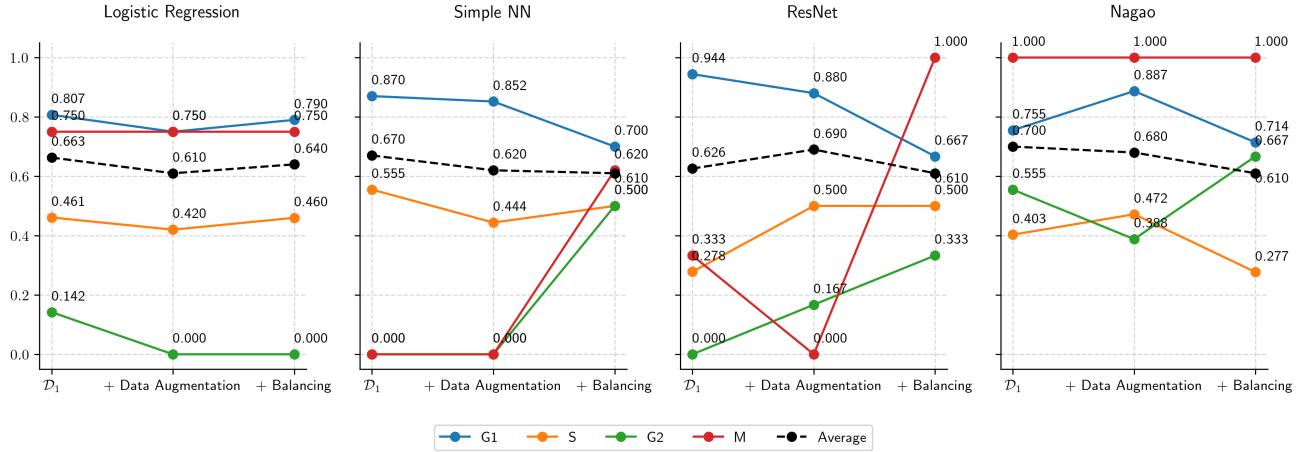
**Table II:** Model accuracy and  $F_1$  scores.

architecture yields the best results, with an  $F_1$  score of 0.71 coupled with an overall accuracy of 74%. We chose our final model based on the  $F_1$  score because it provides a single metric that balances both precision and recall, offering a comprehensive evaluation of a model’s performance across multiple classes. This balance is particularly valuable in our scenario where there is an uneven distribution of classes, preventing the dominance of performance metrics by the majority class  $G_2$ . Notably, this model exhibits a good accuracy in classifying class  $G_1$  (78.5%) and class  $M$  (100%). This is also presented in Figure 3 where we see that the Nagao model is the most balanced out of the four models we optimized. It is robust to data augmentation which means it will be better at generalizing than the others.

## V. CONTRIBUTION

Our deliverables consist of two principal elements:

- **Website.** An interactive, web-based interface has been designed to enable users to interact with our image classification models. This user-friendly platform is accessible [here](#). Users have the opportunity to choose from a range of pre-trained models and can upload an image to view the corresponding predictions. Additionally, all underlying code is open-sourced and can be accessed at the following location [here](#).



**Figure 3:** Overall Accuracy and Accuracy per Class for our models

- **CNNCC: A Python Package for Model-Based Image Inference [6].** The CNNCC (Convolutional Neural Network Cell Classification) package, can be installed via the Python Package Index (PyPI) using the `pip` command. It enables users to perform inference on new images utilizing our pre-trained models. Installation can be achieved through the following command:

```
$ pip install cnncc
```

Upon execution, the output includes the phase prediction by the model, along with the probability distribution across various phases. An illustrative example of the package's usage is as follows:

```
$ cnncc --image "<path/to/image>"
```

```
> Model: Nagao
> Predicted Phase: M
>
> Phase      | Probability
> -----
> G1         |   8.84 %
> S          |   0.38 %
> G2         |   0.12 %
> M          |  90.66 %
```

Additionally, the package offers flexibility to experiment with different pre-trained models by using the `--model` parameter, thereby allowing for a comprehensive analysis with various pre-trained models.

## VI. DISCUSSION

Classifying the cells according to the cell cycle phase is a difficult task due to the inherent similarity of images and the limited size of the dataset. Moreover, the continuous nature of the cell cycle causes some images to be hard to classify with high confidence. For example during the phase S, cells are duplicating their DNA and centrioles, which means the stained DNA observed is on a spectrum. Therefore, early S phase cells will be closer to the G1 class, whereas late stage S phase cells may be confused for the G2 class. Coupled with a low sample size, the class confusion was tough to manage, and trivial balancing may not be sufficient (Figure 6). With those difficulties in mind, we still managed to have an overall accuracy of 74% with the Nagao CNN architecture. However this might not be a satisfactory accuracy for the downstream research needs. It might be necessary to still combine this method with the eyes of an expert when the classification confidence is low.

## VII. CONCLUSION

Widely applicable, our Machine Learning cell cycle classifier is advantageous as it solely necessitates standard microscope equipment and routine cell preparations. This ease of application renders it suitable for a broad spectrum of studies involving nucleus stained cell images. However, the limited accuracy needs to be taken into account when using the method, and the resulting classification must be taken with precaution. In the future, retraining the model with more data would enable our method to work more accurately.

### VIII. ETHICAL RISK ASSESSMENT

Using the tool called the “Digital Ethics Canvas” [7] to guide us, we assess the ethical risks with 6 “ethical lenses” specific to the digital domain: beneficence, non-maleficence, privacy, fairness, sustainability and empowerment. First, the beneficiary of this project is the research lab of Prof. Gönczy and maybe in the future other cell division researchers. They will use our program to classify the cells they work on according to cell cycle phase. We see no risk of maleficence or privacy issues as the program only classifies public use cells, and does not contain any personal information on anyone. Then, to optimize the fairness of our method, we have made it open-source and therefore accessible by any researcher in need of such a method. To further improve the empowerment of the researchers, we have built a website that makes it very easy to use the method, only by uploading a photo of the cell to classify. Lastly, the method should not negatively impact our planet, apart from the training of our ML model that used a lot of computational power and therefore energy.

### IX. ACKNOWLEDGMENTS

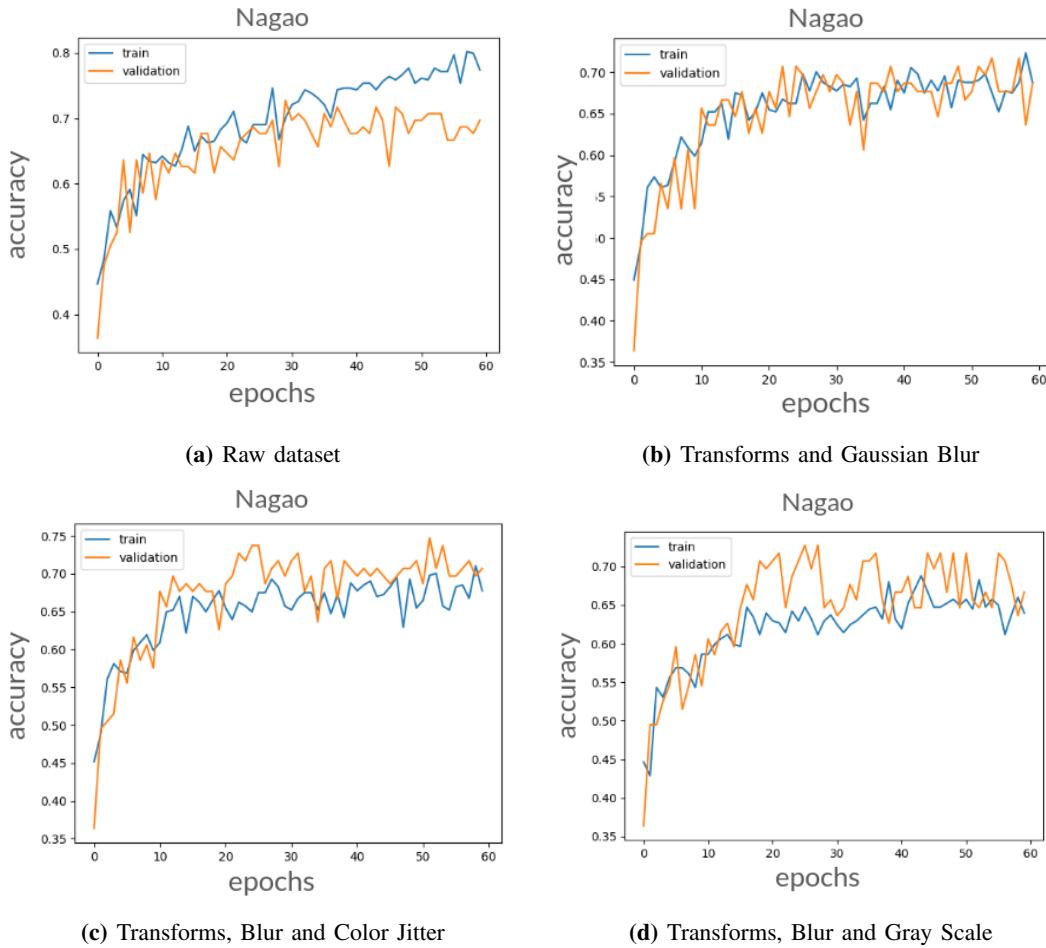
We would like to express our heartfelt gratitude to our supervisor Léo BÜRGY for his guidance and insights throughout this project. Working under his mentorship and applying our Machine Learning skills to a real scientific project has been an incredible learning experience. We are also deeply appreciative of Prof. Gönczy for his advices and for providing us with the chance to work in his lab.

### REFERENCES

- [1] V. Roukos, G. Pegoraro, T. Voss, and T. Misteli, “Cell cycle staging of individual cells by fluorescence microscopy,” *Nat Protoc.*, 2015.
- [2] H. A. Breslow DK, “Mechanism and regulation of centriole and cilium biogenesis,” *Annu Rev Biochem.*, 2019.
- [3] Ryniawec and Rogers, “Centrosome instability: when good centrosomes go bad,” *Cell. Mol. Life Sci.*, 2021.
- [4] Y. Nagao, M. Sakamoto, T. Chinen, Y. Okada, and D. Takao, “Robust classification of cell cycle phase and biological feature extraction by image-based deep learning,” *Molecular Biology of the Cell*, 2020.
- [5] H. Kaiming, Z. Xiangyu, R. Shaoqing, and al., “Deep residual learning for image recognition,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [6] F. Dumoncel, E. Billard, and S. Boudouh, 2023. [Online]. Available: <https://pypi.org/project/cnncc/>
- [7] C. Hardebolle, V. Macko, V. Ramachandran, A. Holzer, and P. Jermann, 2023. [Online]. Available: [https://docs.google.com/document/d/1lhGV\\_xJiuC98it5RB2TiitIBgu0pjUK8qq-8y-wbPw/edit#heading=h.3oqql7cc03h](https://docs.google.com/document/d/1lhGV_xJiuC98it5RB2TiitIBgu0pjUK8qq-8y-wbPw/edit#heading=h.3oqql7cc03h)

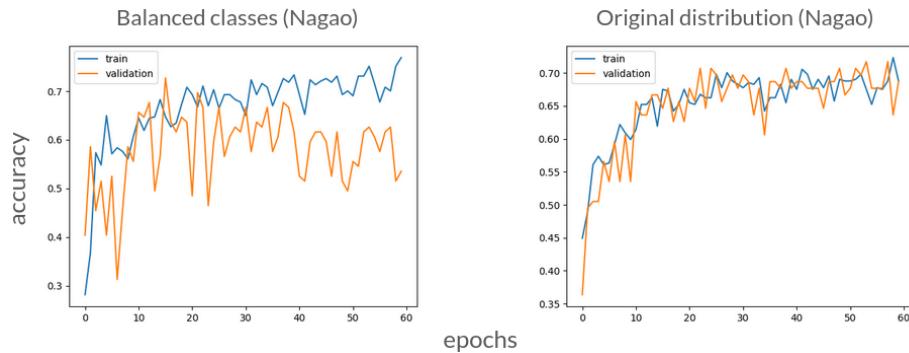
## APPENDIX

### A. Data Augmentation

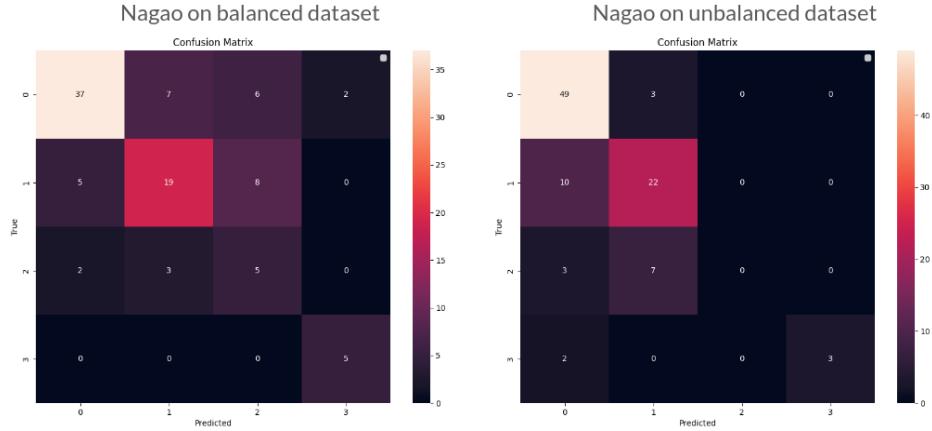


**Figure 4:** Nagao's CNN on different data augmentations

### B. Balancing



**Figure 5:** Effects of data balancing on overall accuracy

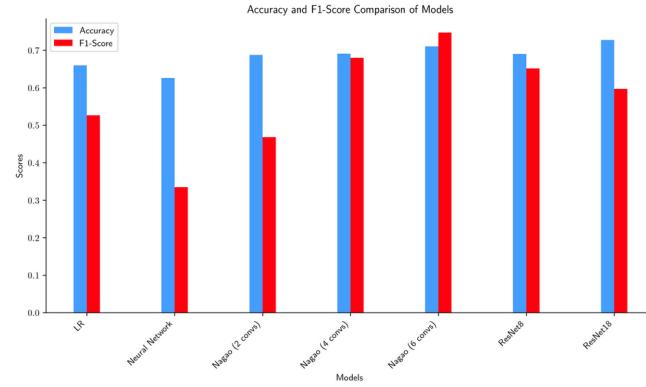


**Figure 6:** Effects of data balancing on class confusion

### C. Architectures

Depth	Block Type	Accuracy	F1-Score
8	Residual	0.52	0.37
8	NonResidual	0.72	0.55
14	Residual	0.68	0.50
14	NonResidual	0.63	0.42
20	Residual	0.71	0.57
20	NonResidual	0.70	0.55

**Table III:** ResNet performance with depth and block type



**Figure 7:** Comparison of architecture performances (ordered by architecture parametrization)