Deep Learning Methods for Green Fluorescent Protein Cell Classification

Hugo May, Dalil Koheeallee, Emilien Guandalino
École Polytechnique Fédérale de Lausanne, Switzerland

***Abstract*—Cell classification from microscope imaging is often a very time-consuming task for many biologists although it is essential when carrying out the experiments. Yet, very few performant tools are available and they often need a human verification due to their low reliability. In this project, we aim to detect the presence of the enhanced Green Fluorescent Protein (eGFP) in lateral amygdalae (LA), baso-lateral amygdalae (BLA) or basal amygdalae (BA) cells from coronal sections of mice brain. A common tool used to address this task is threshold-based classification which yields low accuracies due to changing background level across images, and requires human proofreading. In this project, we applied deep learning methods to classify GFP cells using a Convolutional Neural Network (CNN) and reached over 95% accuracy on human-labeled examples.**

I. Introduction

To understand the functional properties of biological complexes (from molecules to cell networks), a standard procedure in biology is to induce modifications aimed at increasing, decreasing or deleting the role of the object under investigation. Therefore, by estimating the effect of these manipulations, scientists infer and hypothesize on its functions. Microscopy imaging of biological samples is one of the most common datasets produced in biology. It aims at characterizing differences in molecular, subcellular, cellular, cell-network, compositions. In particular, the tracking of the object under study is often possible by coupling it with fluorescent probes that allow for its visualization through microscopy imaging. The enhanced green fluorescent protein (eGFP) it is known to be among the most used fluorophores for such studies, and it emits light at a known wavelength (in green, emission peak at 509 nm) in response to a particular excitation – also in the form of light (number). Thus, the sample containing the modifications of interest will also emit green light under appropriate excitation whereas the control with no modifications will not. A frequent experimental scenario is comparing populations of cells in biological tissues and their properties, such as the expression of specific proteins. Therefore researchers often require distinguishing eGFP positive cells versus eGFP negative cells from the acquired images. As experimental conclusions highly depend on this classification, it is essential for biologists to have a very high accuracy on this classification. Current methods rely on threshold-based classification, i.e. if the mean intensity of the cell on the GFP wavelength channel is higher than a certain threshold, the cell is classified as positive and otherwise as negative). One main limitation of this technique is that it is not precise enough to exempt human proofreading. A machine-learning-based classifier has recently emerged on the image processing software QuPath but consists only of a general classifier and not specific for GFP classification and offers very few GFP related-features such as mean, standard deviation, minimum and maximum intensity on GFP channel. The purpose of this project, is to develop a deep learning based classifier for GFP cells which is both performant and generalizable.

II. Experimental Background

In order to conduct our project, we were provided with microscopic images of coronal sections of mice brain. We focused on the amygdalae, circled in red in Figure 1.

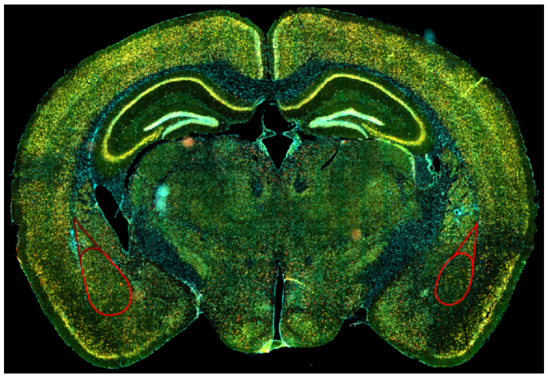

Fig. 1: Whole brain slice image

This large image is then further processed to yield hundreds of cropped images containing cells, which will serve as data input for our classifier.

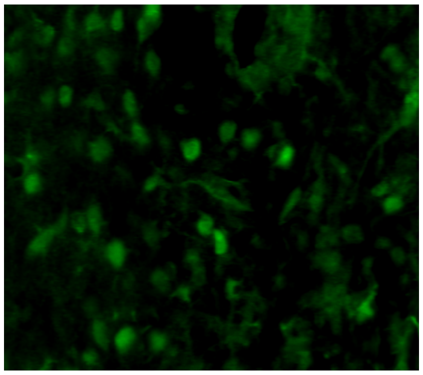

Fig. 2: Zoom on a whole brain slice

From inspecting Figure 2, we can observe 3 relevant cases which are depicted below. In panel A, we have a cell which is clearly positive and easy to classify. In panel B, the cell is negative as its emissions at 509 nm are similar to ambient background, and is also easy to classify. On the last panel, we have an example of a GFP- cell but which happens to have very high auto-fluorescence, meaning that the cell is emitting green light but this emission is not due to the presence of eGFP.

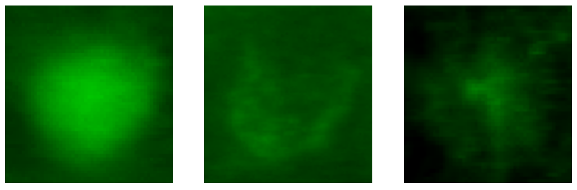

Fig. 3: Commonly encountered cells

Cases similar to panel C make the task of classifying cells very hard for simple methods like threshold-based classifiers, and will usually require expert intervention. In some very rare cases, even an expert can have issues classifying some cells. For those cells, supervised learning methods will also suffer some uncertainty.

III. Data pre-processing

A. Cell Extraction

In this project, we studied 129 slide scanner images of coronal sections of mice brain captured with an Olympus VS120 whole slide scanner and a 20x magnification. Our work was made to be compatible with QuPath analysis, which runs cell detection to give us centroid coordinates. It decomposes in two main parts : first a groovy script that can be run through the QuPath script panel creates a 64x64 pixels .tif image around each cell of the selected images, and a csv file for each image containing the label, the centroid coordinates and the image of origin of each cell. This script only preserves neuronal cells, whose nucleus area is superior to $50\mu m^2$, which filters out around half of the cells. We chose the size of 64x64 for our images because we computed that with this value we could entirely capture at least 99% of the remaining cells. This value depends on the resolution of the microscope. Our Python code then reads each image and converts them to 8-bit png images by selecting the channel corresponding to the emission wavelength of eGFP.

From this processing, we were able to generate a training set of around 126,000 cell images and a test set of 31,500. The images were splitted randomly to create those 2 sets. In the train set we have 8,774 eGFP+ and 117,367 eGFP-, and in the test set we have 2,225 eGFP+ and 29,457 eGFP+ cells. In both cases, the positive rate is around 7%, leading to an unbalanced dataset. Since we have such a large amount of training samples, we were able to use a large validation set, of about 30,000 samples, or one fourth of our training samples.

B. Data Augmentation

Our dataset is overwhelmingly GFP- and we would like to balance it which we have done by generating horizontal and vertical random flips and rotations of the original positive samples. This adds more diversity in our dataset and improves the robustness of the model on small changes in orientation and made it better able to handle new, unseen data.

After the augmentation, we have over 27 000 eGFP+ and 117 000 eGFP- for the train set.

As we will see later in the result section this was one of the major sources of improvement in performance.

IV. Deep Learning Model

We are considering an image classification problem, with two classes : GFP+ assigned to 1 and GFP- assigned to 0.

A. Architecture

Convolutional Neural Networks (CNN) are efficient tools for image processing, which we will train and feed into a Fully Connected Network (FCN) for classification. Since we are dealing with a large dataset, we can afford to train a complex model with many layers and tens of thousands of parameters. Below is the general structure of each layer block of our final network, which we will justify in the following sections.

Convolutional Layers :

- Convolution
- Batch Normalization
- Activation Function
- Pooling

Fully Connected Layers :

- Linear
- Dropout
- Activation Function

As it is standard for such problems, we will use Rectified Linear Unit (ReLU) activation function and the Softmax function for the final pass of our network.

B. Training

For all of the following results, the training was done using a batch size of 32 and over 10 epochs, which was enough to converge without overfitting. We used Adam as an optimizer with a learning rate $\alpha = 0.001$ and weight decay of 1e5.

C. Analysis Metrics

To evaluate the performance of different network architectures, training parameters and optimization techniques we used three different metrics. First is our loss function, for which we minimize the Cross Entropy Loss, a standard for binary classification. We also assessed the total accuracy of each model, but since our test set is unbalanced our main metric of evaluation in the end was the F1 score.

D. Results

1) Augmented Data: The first performance improvement came from balancing our data. We achieved this by doing both data augmentation, as explained earlier, and by using a random weighted sampler. Based on the number of samples per class, weights are generated and will determine how frequently each sample is taken from our training dataset. This enables us to train our model more fairly, and avoid the trap of always predicting the most common label. The following results are given for a simple, unoptimized network fed with unbalanced data:

Dataset	Loss	Accuracy	F1 Score
Non Balanced	0.1429	0.9495	0.419
Balanced	0.1264	0.9549	0.496

As we can see, for non augmented data, the test accuracy is already quite high, but the F1 Score is low. This is because our model predicts GFP- almost all the time. After balancing out our train set, we yield better results on the test set.

2) Baseline Performance: To measure the efficiency of different design choices and optimizations, we trained a simple model with a single convolutional layer and a single linear layer on augmented data.

Model	Loss	Accuracy	F1 Score
Baseline	0.1370	0.9534	0.5510

This will serve as reference for further analysis of different techniques.

3) Large CNN vs Large FCN: To determine the most performant architecture for our model, we initially trained two opposite networks : one with a large convolutional part and one with a large fully connected part. Both of those models were more accurate than the initial one, but were now much more complex with over 100,000 parameters. In the end, we fine tuned a combination of two intermediate convolutional and fully connected network to get the best performance.

Model	Loss	Accuracy	F1 Score
Large CNN	0.135508	0.9547	0.587
Large FCN	0.151827	0.9437	0.599
Combination	0.115469	0.9540	0.6141

Our final model uses 4 layers of convolution and 4 linear layers of 500 parameters each. We also found that adding an Average Pooling layer before Max Pooling gives slightly better performances.

4) Batch Normalization: After determining the architecture of our model, we further improved performance by normalizing the inputs, both at the beginning of the network and after each convolution before the activation function.

Model	Loss	Accuracy	F1 Score
Batch Norm	0.1156	0.9550	0.6143

This significantly improves our performance over the baseline model.

5) Dropout: Another optimization that we made is to dropout some nodes during the training of each linear layer. If applied in the right proportions, this avoids overfitting and enables our model to better generalize to test samples. After trying out multiple values, we found that $p = 0.25$, meaning 25% of the nodes are randomly dropped on each layer, gave the best performance increase.

Model	Loss	Accuracy	F1 Score
Dropout	0.1063	0.9607	0.5803

This gave us a good increase of 3% compared to baseline.

6) Final Model: Combining all the previously mentioned optimizations, we trained our final model to give the best performance we could achieve.

Model	Loss	Accuracy	F1 Score
Final	0.1153	0.9530	0.6351

Figure 4 displays the training losses and accuracies from this model over 10 epochs. Plotting this graph enables us know with good confidence that our model has converged without overfitting. Figure 5 shows the comparison of the different models, isolating each optimization. The final model, which combines all methods is the most performant.

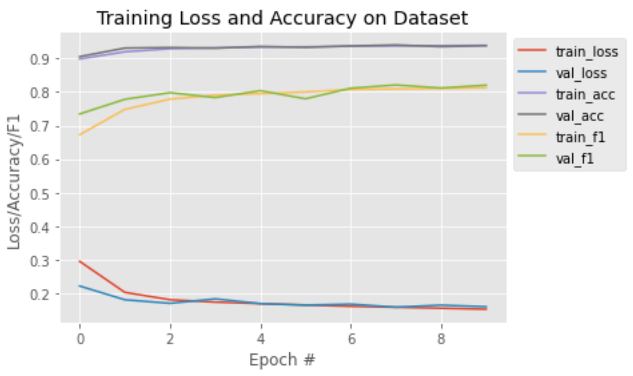

Fig. 4: Training Loss and Accuracy on Dataset

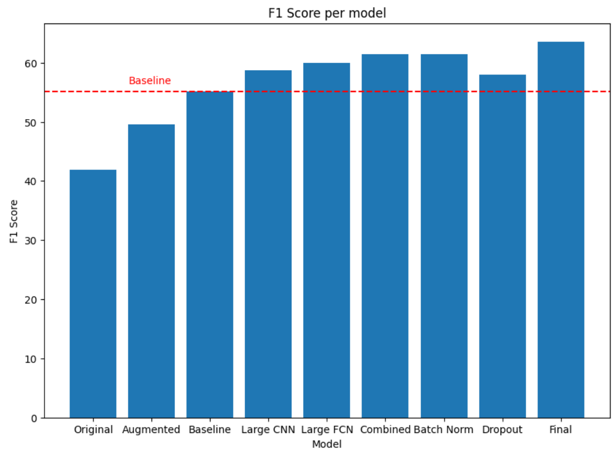

Fig. 5: F1 score per model

Unfortunately, despite trying many different models, we were not able to obtain a better performance. The task at hand is very hard and the remaining images that our model misclassifies are also non trivial to label for humans.

V. Conclusion

In this project we optimized a CNN followed by an FCN to reach an accuracy of 95% and a F1-score of 63% for GFP cells classification. Despite that most of the GFP- cells are well-classified, the model tends to over-predict GFP- and leads to some misclassification of GFP+ cells. Thus, we cannot say that this model can run without human proofreading although with an accuracy of 95%, human proofreading can be performed much quicker. With data augmentation, we tried to make our model as robust as possible to different types of images. Nevertheless our model can still not adapt to different magnification and different resolution as it takes only 64x64 images and cells must fit in this square for the model to perform well. We could imagine a future implementation, the addition of an algorithm able to transform an image bigger or smaller to a 64x64 image. It could also be interesting to increase the robustness of our model by feeding the model with different cell types and cell density from different tissues also containing GFP. Another improvement could be to train our model with cells coming from different types of tissues such as epithelial tissues therefore making it even more general.

Acknowledgements

We would like to thank the Gräff Lab for hosting our project as part of the ML4Science program. In particular we specially thank Giulia Santoni for her precious technical advice and for providing us with adequate data to realize our project.