CLASSIFYING T CELL ACTIVITY WITH CONVOLUTIONAL NEURAL NETWORKS

by

Zijie (Jay) Wang

A thesis submitted in partial fulfillment of

the requirements for the degree of

Bachelor of Science

(Statistics with Honors)

at the

UNIVERSITY OF WISCONSIN-MADISON

2019

1 Introduction

Immunotherapy is a type of cancer treatment that uses the body's own substances to boost natural defense against cancer. T cells, the most important immune cells in the human body, are a promising target for immunotherapies because of their diverse cytotoxic and immune-modulating activity. Immunotherapies that enhance T cell functions are currently used in clinical cancer treatments [1]. Some are also in development for additional diseases including HIV and diabetes [2,3]. Because of the different behaviors of T cell subtypes, it is required to assess the effect of various T cell compartments on patients and select the appropriate cell populations during immunotherapies. However, current T cell profiling methods, such as surface protein expression and cytokine production, rely on exogenous contrast agents. These methods require a long processing period, and some of them have destructive effects on cells. Therefore, a label-free and non-destructive pipeline to determine T cell functions will be more amenable to subtyping T cells for immunotherapies.

Autofluorescence imaging is appealing because it only relies on endogenous contrast and is non-destructive. Previous studies have shown success in using autofluorescence imaging on identifying macrophages in vivo [4]. However, this method usually only generates grayscale cell images, which is less quantitative and informative than other T cell profiling pipelines. In addition, current methods to determine cell subtypes with autofluorescence imaging require a large amount of human labor to label output images. In this study, we aim to develop a novel automated framework that utilizes autofluorescence cell images to assess T cell functions at the single-cell level.

Machine learning is a collection of statistical models that computers use to make or improve predictions based on data. Studies have shown that machine learning is promising for compensating for less informative fluorescent image data and automating cell subtype classification. For example, researchers have used a regularized logistic regression to assess the macrophage activation state [5] and a support vector classifier to determine heterogeneous cell populations [6]. Recently, more advanced machine learning models, such as the convolutional neural network (CNN), were explored with cellular image data to classify cell phenotypes [7] and differentiate immune cells from cancer cells [8]. In addition, CNNs can also be used to solve problems that are related to cell subtype classifications, such as cell segmentation [9] and cell sorting [10].

We here propose to use transfer learning from a pre-trained deep CNN to classify T cell activation state at the single-cell level. Instead of extracting a small set of features from raw images before applying the CNN [8], this approach utilizes cell images by treating raw autofluorescence images as input data. Also, it takes advantage of a model which has been trained on generic images, and only retrains partial layers. Therefore, comparing to end-to-end CNN training [7], our approach is more computationally efficient and requires fewer training samples. To overcome the inevitable artifacts of microscopy images and the shortage of training data, we carefully developed an image pre-processing pipeline with image augmentation to improve model performance. Because T cells have subtle differences from donor to donor in real-life immunotherapy applications, we designed a rigorous donor-specific cross-validation scheme to train and evaluate our models. For the same reason, we had held out all images from one cell donor and only used them to assess the final performance of our best model.

We explored the most advanced model with a gradient of several increasingly simpler models, which help us understand when and why deep learning is needed over simpler models. Even though the transfer learning approach outperforms other models across different donors, the performance gap is smaller in some than others. It provides insights for alternative image-based T cell classification methods. All of our code is organized in Jupyter notebooks [11] with reproducible examples and comprehensive instructions, which can serve as instructive examples for the audience who are not familiar with machine learning application on autofluorescence microscopy images.

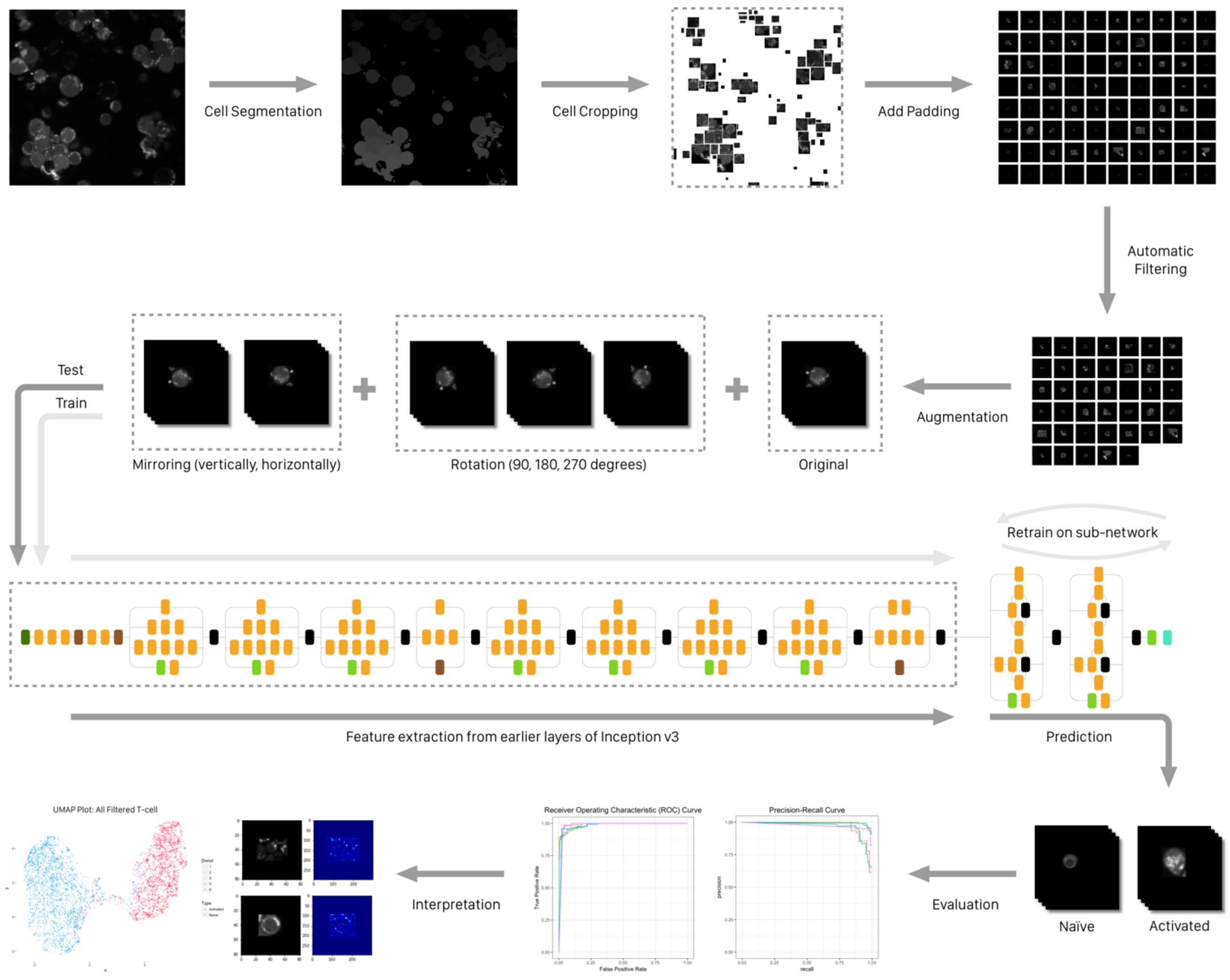
Cell Segmentation
Cell Cropping
Add Padding
Automatic Filtering
Augmentation
Original
Rotation (90, 180, 270 degrees)
Mirroring (vertically, horizontally)
Test
Train
Retrain on sub-network
Feature extraction from earlier layers of Inception v3
Prediction
Naïve
Activated
Evaluation
Receiver Operating Characteristic (ROC) Curve
Precision-Recall Curve
Interpretation
UMAP Plot: All Filtered T-cell

Fig. 1. T Cell Classification Overview.

2 Results

2.1 Overview

The goal of our study is to classify individual T cells as activated or quiescent (inactivated), using only cropped autofluorescence cell images. The overall workflow for our pre-trained CNN model is described in Fig. 1. This workflow starts from image pre-processing and proceeds to model training and model evaluation. In addition to the transfer learning model, we explored a frequency classifier, three Lasso logistic regression models with different features, a simple one-layer neural network and a simple CNN model. These models follow a similar workflow to the pre-trained CNN model.

2.2 Cross-validation Results

T cell microscopy images appear to be slightly different from donor to donor. A trained model that can be generalized to new donors will be useful in the clinic. Therefore, to assess our classifier's performance on cell images from new donors, we designed a nested cross-validation scheme to train, tune and test all models. For the same reason, we held out an entire donor, donor 4, from our image dataset and only use it to evaluate the final best model. Due to this cross-validation design, the same model could have various optimal hyper-parameters for different test donors. Therefore, the final model performance is grouped by test donors (Fig. 2). Also, Precision-recall (PR) curves and Receiver operating characteristic (ROC) curves are plotted to further assess the test performance (Fig. 3, Fig. 4). Based on these results, two pre-trained CNN models outperform other classifiers.

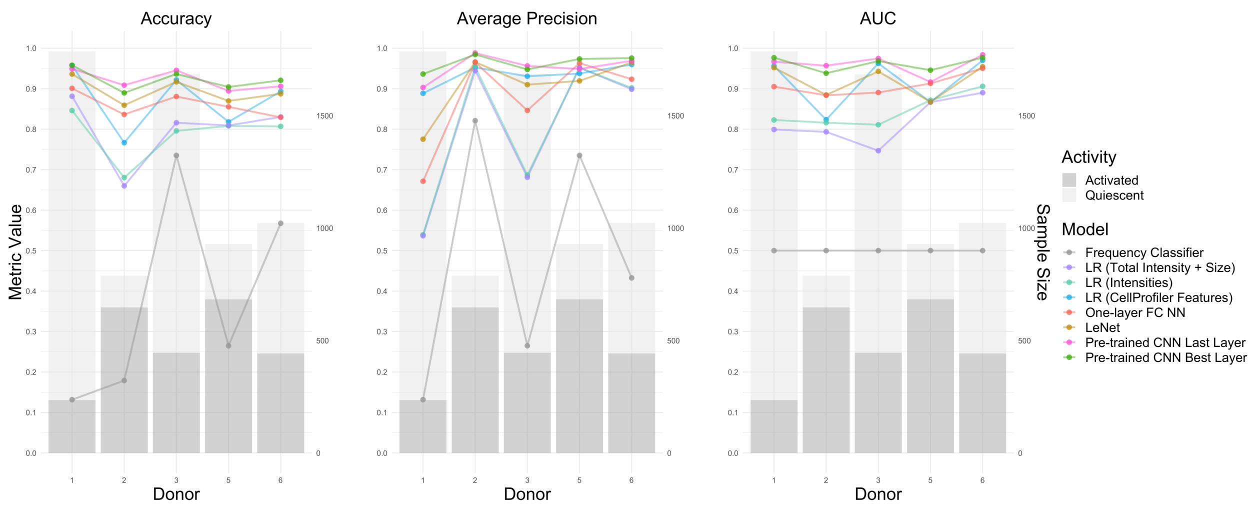

Fig. 2. Model Test Performance Summary.

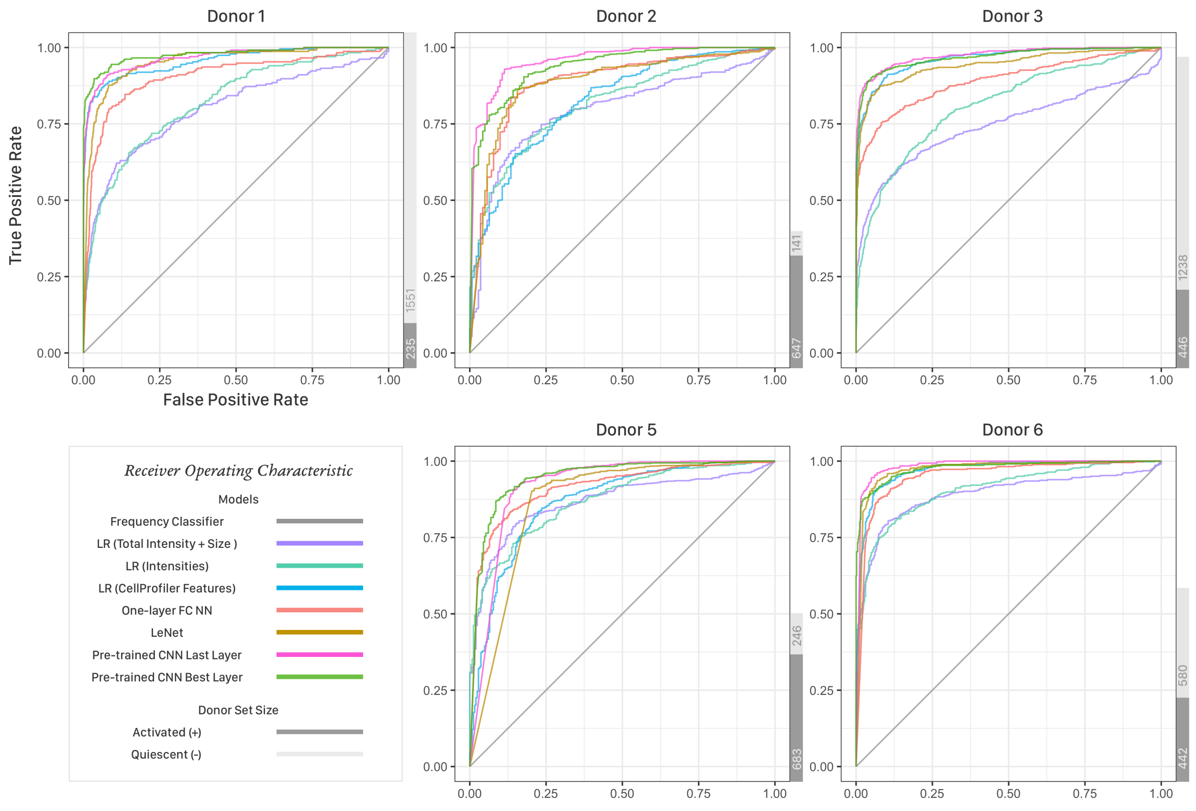

Fig. 3. Model Test Performance - PR Curves.

A frequency classifier, which uses the frequency of positive samples in the training set as the probability score of the activated label, is used as our baseline model. The average accuracy of all test donors is 37.56% (Table 1). It implies that the majority class in the test set and training

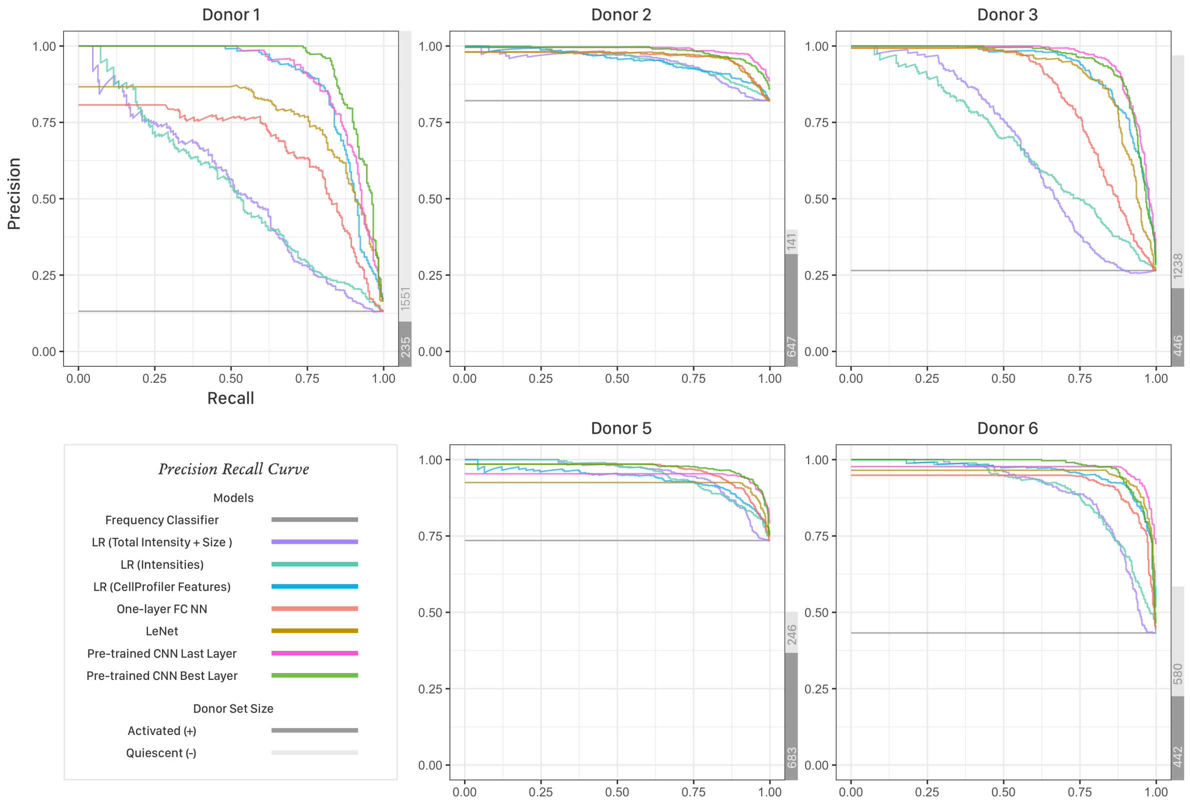

Fig. 4. Model Test Performance - ROC Curves.

set are more likely to be different. For example, there are more activated cells from donor 1 while there are more quiescent cells from the combination of donor 2, 3, 5 and 6. This sets up a good baseline model since classifiers that fail to utilize other features rather than the label count will be prone to perform poorly.

Three logistic regression models using different features all give better classification than the baseline model. The logistic regression fitted with the image pixel matrix leads to an average accuracy of 78.74% (Table 2). Among those 6,724 pixel features, over 5,000 features were reduced by the Lasso regularization. To interpret this model, one can plot the exponential of each pixel's coefficient to visualize the odds ratios. As shown in Fig. 5, this model learned the shape of cells, and cells with larger size were more likely to be classified as activated. The logistic regression using only two features, mask size and total intensity, gives slightly better performance with an average accuracy of 79.93% (Table 3). For all test donors, the optimal coefficient of cell mask size is negative whereas the coefficient of total intensity is positive. After adjusting the standard deviation of two features, this model is more sensitive to total intensity. This can be interpreted as after fixing the total intensity, cells with a larger size, visually dimmer cells, are more likely to be quiescent. Finally, the logistic regression model with CellProfiler [12] attributes, such as mean intensity value and cell perimeter, yields 87.14% average accuracy (Table 4). After adjusting the deviation, attributes that are related to image intensity and cell area are the most significant.

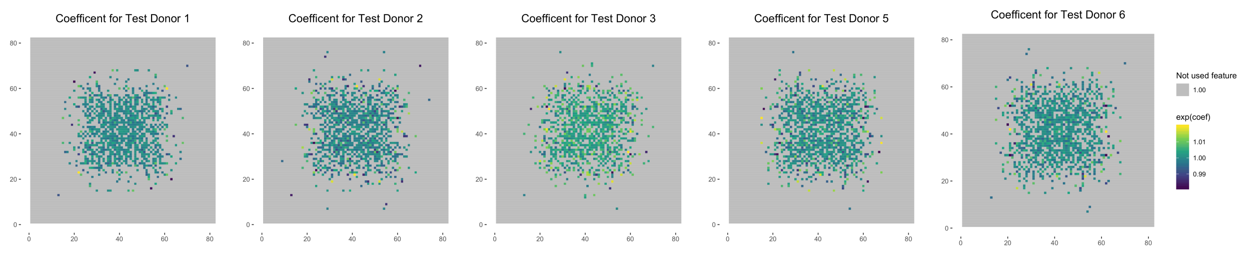

Fig. 5. Coefficient Visualization for Logistic Regression Model with Image Pixel Features.

Table 1: Performance of Frequency Classifier

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	13.16%	13.16%	100.00%	13.16%	50.00%	235	1551
2	17.89%	0.00%	0.00%	82.11%	50.00%	647	141
3	73.52%	0.00%	0.00%	26.48%	50.00%	446	1238
5	26.48%	0.00%	0.00%	73.52%	50.00%	683	246
6	56.75%	0.00%	0.00%	43.25%	50.00%	442	580

In addition to linear classifiers, non-linear models with image pixel as input data were developed and tested. A simple neural network with one hidden layer was fine-tuned on learning rate, batch size and the number of hidden layer neurons. Even though its average accuracy of 86.05% (Table 5) is slightly lower than the best logistic regression, it has a more stable performance across five test donors. In comparison, the simple CNN LeNet has a more complex

Table 2: Performance of Logistic Regression (Image Pixel Matrix)

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	84.60%	43.79%	60.00%	53.90%	82.27%	235	1551
2	68.02%	94.99%	64.45%	95.38%	81.61%	647	141
3	79.57%	61.75%	60.09%	68.67%	81.10%	446	1238
5	80.81%	88.12%	85.42%	95.20%	87.16%	683	246
6	80.68%	73.43%	86.97%	90.19%	90.59%	442	580

Table 3: Performance of Logistic Regression (Size and Total Intensity)

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	88.13%	55.35%	50.64%	53.67%	79.92%	235	1551
2	65.99%	96.56%	60.74%	94.46%	79.34%	647	141
3	81.59%	68.89%	55.61%	68.12%	74.68%	446	1238
5	80.92%	89.32%	84.11%	95.20%	86.68%	683	246
6	83.02%	78.17%	84.49%	89.86%	89.02%	442	580

Table 4: Performance of Logistic Regression (CellProfiler Attributes)

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	95.74%	86.98%	79.57%	88.85%	95.61%	235	1551
2	76.65%	91.56%	78.83%	95.24%	82.33%	647	141
3	92.16%	94.60%	74.66%	93.07%	96.26%	446	1238
5	81.81%	81.81%	96.78%	93.74%	86.70%	683	246
6	89.33%	82.33%	95.93%	95.97%	97.01%	442	580

architecture [13]. After selecting the best learning rate and batch size, LeNet has reached an average accuracy of 89.39% (Table 6).

Our most advanced models using a pre-trained CNN outperform all methods discussed above. These two models use cell images as input and require a previously trained CNN. For one model, we only retrained the last layer, whereas we tuned the retraining region as a hyper-

Table 5: Performance of Simple Neural Network

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	90.09%	59.12%	80.00%	67.14%	90.51%	235	1551
2	83.63%	96.59%	83.15%	96.53%	88.38%	647	141
3	88.06%	81.94%	70.18%	84.63%	89.06%	446	1238
5	85.53%	88.21%	92.71%	96.33%	91.30%	683	246
6	82.93%	72.53%	97.30%	92.34%	95.03%	442	580

Table 6: Performance of LeNet

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	93.62%	77.13%	73.19%	77.53%	95.21%	235	1551
2	85.91%	96.21%	86.24%	96.60%	88.45%	647	141
3	91.69%	93.97%	73.32%	90.99%	94.27%	446	1238
5	86.98%	88.18%	95.02%	91.93%	86.73%	683	246
6	88.75%	80.45%	97.74%	96.51%	95.44%	442	580

Table 7: Performance of Pre-trained CNN (Last Layer)

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	94.90%	78.57%	84.26%	90.29%	96.68%	235	1551
2	90.86%	97.52%	91.19%	98.82%	95.69%	446	1238
3	94.54%	95.85%	82.96%	95.63%	97.47%	647	141
5	89.45%	90.01%	96.34%	94.87%	91.65%	683	246
6	90.61%	83.02%	98.42%	96.90%	98.37%	442	580

parameter for the other model. The average accuracy is 92.07% for the first model (Table 7) and 92.18% for the latter model (Table 8).

2.3 Assessing Generalization with a New Donor

In order to evaluate our best model's ability to generalize to T cell images from a new individual, images from donor 4 were completely held out during the entire cross-validation

Table 8: Performance of Pre-trained CNN (Best Layers)

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
1	95.80%	82.26%	86.81%	93.62%	97.66%	235	1551
2	88.96%	95.60%	90.73%	98.45%	93.83%	647	141
3	93.65%	95.69%	79.60%	94.76%	96.82%	446	1238
5	90.42%	91.60%	95.75%	97.35%	94.58%	683	246
6	92.07%	90.38%	91.40%	97.57%	97.66%	442	580

Table 9: Performance of Pre-trained CNN (Best Layers) on Donor 4

Donor	Accuracy	Precision	Recall	Average Precision	AUC	Activated Count	Quescent Count
4	98.59%	99.15%	96.68%	99.67%	99.89%	482	1569

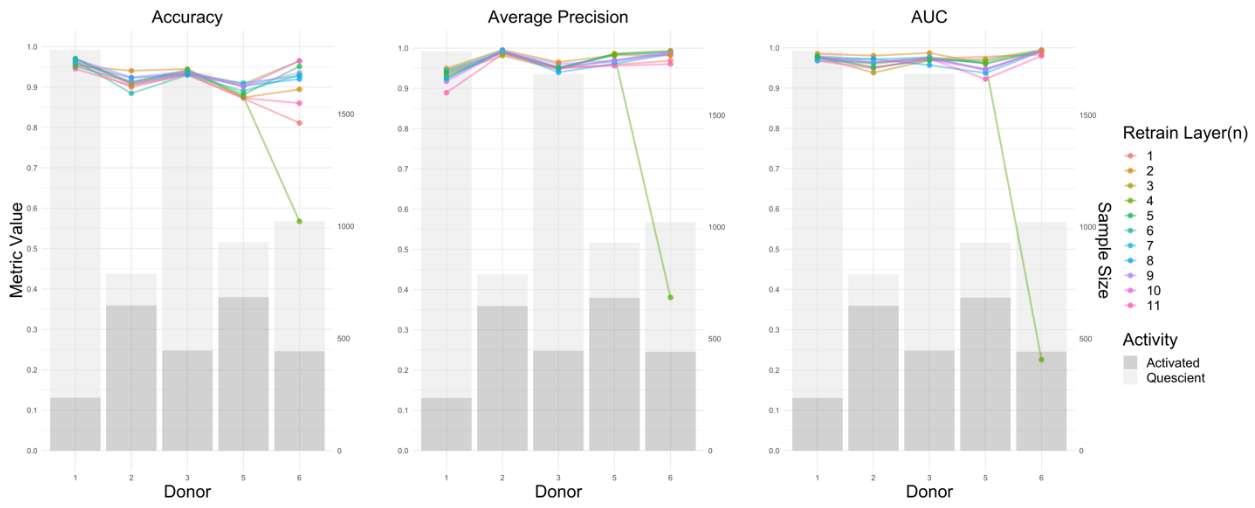

Fig. 6. Performance of Pre-trained CNN with Different Retraining Layers.

process and all of the model implementation and study design. After determining the pre-trained CNN model with optimal training layers as our best model, we applied the same nested Cross-validation scheme to train, tune and test it on images from donor 4. It gives an accuracy of 98.59% (Table 9). The performance metrics in Table 9 are significantly higher than their counterparts in Table 8. One reason is that this evaluation has training data from 5 donors, whereas previous tests only consist of training images from 4 donors.

2.4 Optimizing Layers of the Pre-Trained CNN

We define *n*, ranging from 1 to 11, as the number of last retraining Inception modules in the pre-trained CNN model. After tuning *n* along with other hyper-parameters through 3,520 nested cross-validation inner loop jobs, we tested pre-trained CNN models with all possible *n* values and associated optimal hyper-parameters for each test donor. As shown in Fig. 6, despite the

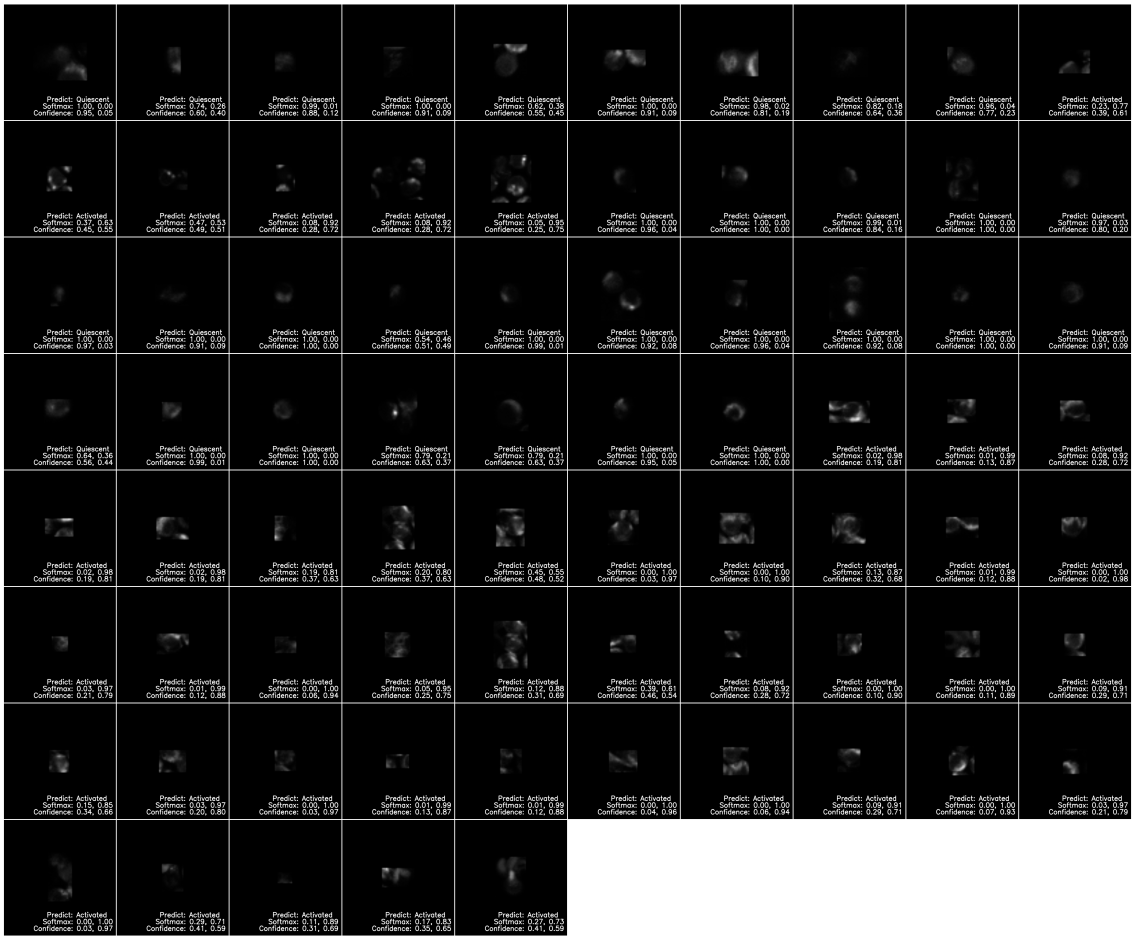

Fig. 7. Best Model Misclassified Images from Donor 1.

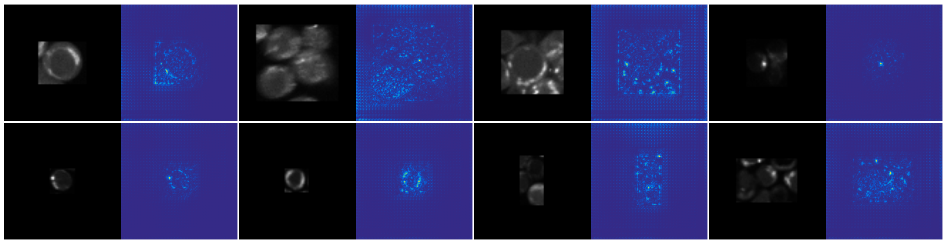

Fig. 8. Saliency Maps of the Last Layer of Best Model with Images from Donor 1.

outlier of donor 6 with *n* as 11, different *n* values give similar metrics. It suggests that retraining more layers in a pre-trained CNN does not significantly improve model performance.

2.5 CNN Interpretation

T cell images that the best model failed to classify correctly were visualized with model's classification confidence calibration. Fig. 7 shows all misclassified images for test donor 1, where the majority images include artifacts or multiple cells. Also, Saliency maps were used to interpret why the best model makes decisions [14]. Fig. 8 suggests that the model has learned to locate paddings and is more sensitive to high-intensity regions.

3 Discussion

The method we propose, retraining a pre-trained CNN model with T cell autofluorescence microscopy images, can classify activated and quiescent T cells. More importantly, its classification performance is significantly higher than other machine learning models that are commonly used for microscopy image classification. Also, its classification decision is sensitive to high-intensity regions on the image, which is in good agreement of how bioengineers manually analyze this type of autofluorescence cell images.

Based on the misclassified images, the performance of pre-trained CNN model is limited by the quality of cropped single-cell images. Developing a better filter to detect images with artifacts not only can further improve classification accuracy, but also contribute to other studies using microscopy cell images. Similarly, there are recent studies on developing novel cell segmentation methods using state-of-the-art machine learning algorithms [9,15].

Unlike the common transfer learning approach where only the last layer of a pre-trained CNN is retrained, we thoroughly explored the effect of retraining more than the last layer. The results suggest that retraining more layers does not improve classification performance. Possible reasons include the limited sample size and relatively monotonic cell image representations. This provides important insights for the application of this study: given the extra compute costs and implementation challenges, retraining only the last layer is sufficient. If users are interested in optimizing the hyper-parameter *n*, we suggest taking a larger "step size" in the grid search.

The results of our meticulous model comparison indicate that more complex classifiers tend to have better and more stable performance. However, more advanced models such as pre-trained CNNs are less interpretable than linear models. For logistic regression models, we can make statistical inference on every feature representation. However, it is much harder to interpret neural networks. Saliency maps help locate which region on the input image influence the classification the most, but more than thousands of learned coefficients remain unexplained. Therefore, in applications that require a full understanding of how a classifier makes decisions, so users can trust the system, logistic regression models are recommended.

Due to the computing costs, each model in our study was only tuned and tested once. This leads to a limitation of our study in which model performance, especially for the CNNs, is variable based on initializations, hardware, etc. Therefore, slight differences in performance should not be over-interpreted. In future studies, we are interested in designing an efficient evaluation scheme that takes performance variance into account.

4 Method

4.1 Cell Preparation and Imaging

All of our T cell autofluorescence images come from Walsh et al.'s study [16]. Peripheral blood was drawn from 6 healthy donors followed by extractions of bulk CD3+ T cells or an isolated CD3+ CD8+ T cell subset. Then, T cells were divided into quiescent and activated groups, where the activated group was stimulated with activating antibodies. Two T cell populations were cultured separately before generating NAD(P)H images using autofluorescence imaging. Antibodies were used to validate T cell type and activation state. Finally, these grayscale microscopy images were labeled with donor ID and T cell activity.

4.2 Image Processing

Cell segmentation was performed with the CellProfiler program [12]. Each cell was cropped according to the bounding box of its segmented mask. Cell short NAD(P)H lifetime, which had been generated from autofluorescence imaging, was used to filter out other visually indistinguishable cells (e.g., red blood cells). To validate and improve cell segmentation, an automatic filtering framework is designed: it detects very dim images and images with no cells by thresholding the combination of image entropy and total intensity (Fig. 9). The threshold values (4, 4.7 and 3500 in Fig. 9) were chosen based on the distribution of entropy and intensity with Gaussian approximation. This filter was so conservative that there was no false positive detection.

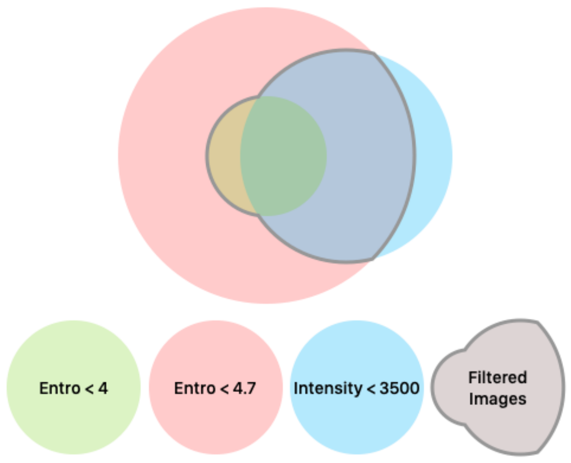

Fig. 9. Dim Image Filtering with Entropy and Intensity.

Since some advanced machine learning algorithms require the input to be square images with uniform size, all activated and quiescent cell images were padded with black borders. The padding size was chosen based on the largest image in the dataset after removing extremely large outliers. In this dataset, all padded images have dimension 82 × 82. Also, augmented images were introduced by rotating each original image by 90, 180, 270 degrees and flipping horizontally and vertically. This image processing pipeline was implemented using the Python package OpenCV [17].

4.3 Nested Cross-validation

Multiple classifiers with a gradient of increasing complexity were carefully designed and tested. We used the same leave-one-donor-out testing scheme to measure the performance for all models. For example, for test donor 1, a logistic regression model is trained on all images including augmented ones from donor 2, 3, 5, and 6 before testing on all unaugmented images from donor 1. By testing in this way, the classification result can tell how well each model performs on newly added data (e.g., new donors).

Following the leave-one-donor-out test design, we want hyper-parameter selections to be independent of the test donor as well. Therefore, a nested cross-validation scheme is developed (Fig. 10). Within the inner loop, a test donor is completely held out, and 4-fold cross-validation is performed to measure the average performance of each hyper-parameter combination (grid search). For models requiring early stopping, the validation set in the 4-fold cross-validation is used to assess the stopping criteria. Since training continues as long as the performance on images from the validation donor improves, the best hyper-parameter choice arguably could be optimistic. However, due to limited training samples, we decided to maximize the number of images used for training instead of partitioning additional images for early stopping. In the outer loop, a model with its best hyper-parameter is trained and then tested on the independent test donor. For example, Tables 1–9 report the test metrics from the outer loop.

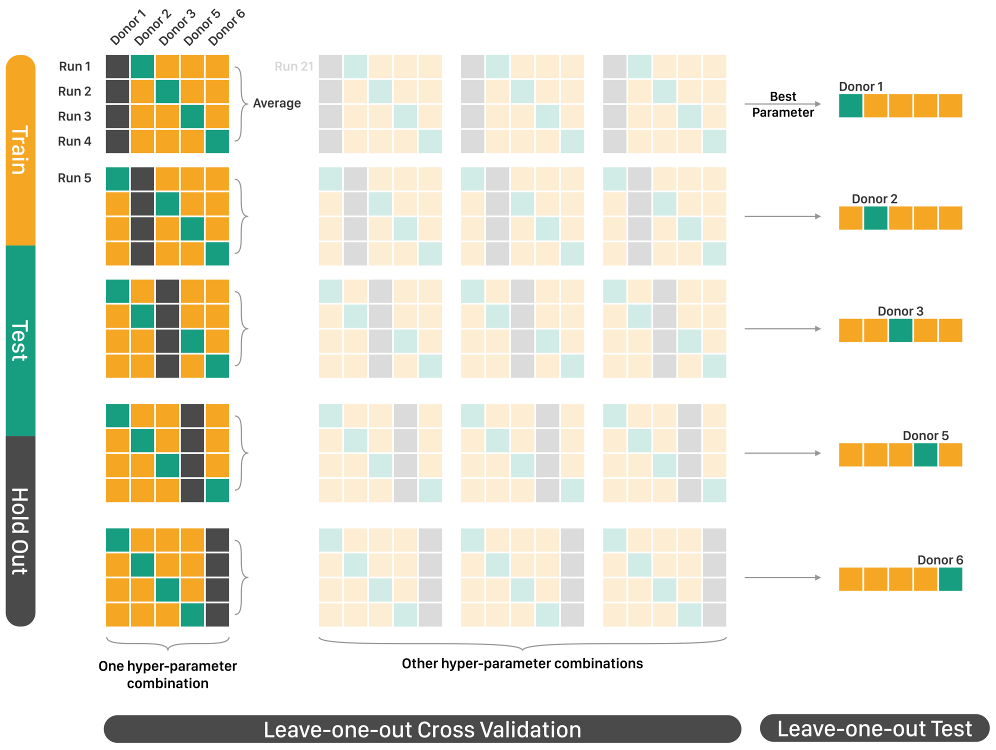

Fig. 10. Nested Cross-validation Scheme.

4.4 Linear Classifier

A trivial frequency classifier is introduced as a baseline model. This model computes the positive sample percentage in the training set, then it uses this frequency as a positive class prediction score (between 0 and 1) for all samples in the test set.

Logistic regression is a standard statistical model used to classify microscopy images [5]. Three logistic regression models with Lasso regularization were fitted and tested using the Python package Scikit-learn [18]. An image intensity matrix with dimension 82 × 82 and values from 0 to 255 was used to fit the first regression model. The second model was fitted with two scalar features: cell size and image total intensity, where cell size was computed using the pixel count in the cell mask generated by CellProfiler. The last regression model used 123 features relating to cell intensity, texture and area, which were extracted from cell images using a CellProfiler pipeline with modules *MeaureObjectSizeShape*, *MeasureObjectIntensity* and

MeasureTexture. The Lasso regularization parameter λ was tuned for all three classifiers with nested cross-validation.

4.5 Simple Network Classifier

In addition to linear models, a simple neural network with one hidden layer was developed using the Python package Keras and with the Tensorflow backend [19,20]. The input layer used the image pixel matrix with dimension 82 × 82. Network hyper-parameters – neuron numbers, learning rate and batch size – were tuned using nested cross-validation.

Also, we trained a simple convolutional network LeNet from scratch [13]. This network architecture has three sequential convolutional layers and two pooling layers. Among these layers, the default number of neurons provided in the original paper was used. The image pixel intensity matrix was used to feed this network, and the input layer was modified to support 82 × 82 one-channel images. Similarly to the previous network model, the learning rate and batch size were tuned.

4.6 Pre-trained CNN Classifier

Using the Inception v3 deep learning network with pre-trained ImageNet weights, a transfer learning classifier was developed [21,22]. Instead of retraining the whole network end-to-end, we took advantage of the pre-trained weights and only trained the last n inception modules, where n was treated as a hyper-parameter. Inception modules are mini-networks that constitute the overall Inception v3 architecture. The popular practice of transfer learning with Inception v3 is only retraining the last Inception module ($n = 1$). In our study, however, we used the nested cross-validation scheme to tune n along with learning rate and batch size. Images were resized with bilinear interpolation to fit the input layer dimension (299×299×3), and three-channel images were generated by merging three copies of the same grayscale image. Early stopping with patience 10 was used during training, which means we stop the training if the loss function fails to improve in 10 consecutive epochs. This model was implemented using Keras with the Tensorflow backend. It was tuned on Cooley instances in Argonne National Laboratory with GPU support.

4.7 Network Interpretation

After choosing the best parameter combinations for the pre-trained CNN models, we were interested in interpreting the classifications. Softmax is a function that is usually used in neural networks to map the output real number (Logits) to a new number between 0 and 1. Studies have found that using the Softmax score from deep neural networks as a confidence calibration does not match the real accuracy [23]. Therefore, we used temperature scaling to better calibrate the predictions [23]. After training, for each donor, the temperature T is optimized on the validation set of nested Cross-validation. Then, we applied T to scale the Logits before Softmax computation and used the new Softmax output to infer classification confidence.

Besides confidence calibration, we used saliency maps to further analyze what morphology feature is used in classification [14]. Saliency map is a straightforward and efficient way to detect how prediction value changes with respect to a small change in input cell image pixels. We generated saliency maps of the output layer for the best model for test donor 1 with few randomly sampled input images from donor 1. Also, saliency maps of each Inception module are visualized for both pre-trained Inception v3 network and our best-retrained network for a few randomly sampled images from donor 2. This interpretation helps us assess whether the classification basis is intuitive, so we can decide if we can trust the model in future applications.

5 Reflection

Through this independent study, my analytical skills have been significantly improved. I am more familiar with data processing, especially with image data. I have gained invaluable experience of implementing a series of classification models ranging from simple logistic regressions to deep convolutional neural networks. Also, I have become comfortable with deploying thousands of jobs on remote GPU servers. Besides practical programming skills, I have a better knowledge of applying different techniques to interpret logistic regressions and neural networks. In addition, I have a much better appreciation of the interpretability of statistical learning models. Given the demand of users in practice, a simple classifier with good interpretability sometimes is preferred over complex "black-box" models with high accuracy. This tradeoff, which I first encountered in this project, directly motivates me to further study machine learning interpretability and human-centered AI in my PhD study.

In addition to my refined analytical ability, I have practiced many subtle skills such as collaborations and communications, which are extremely helpful for my future research. I have learned how to explain statistical concepts such as variance-bias tradeoff and ensemble methods to our bioengineer collaborators. Also, I have become more comfortable with picking up knowledge from other fields that is essential for my interdisciplinary research. For example, through collaborations, I know where to find resources to help me better understand the motivation of immunotherapy. Moreover, I gained critical communication practice including scientific writing and research presentations. I have written many successful grant proposals for this project, which lead to the 2018 Honors Senior Thesis Summer Research Grant, as well as the 2019 University Book Store Academic Excellence Awards. Also, we are currently working on writing the manuscript for this project. Besides writing, I got many opportunities to present this project to different audience as posters and talks through the 2018 Undergraduate Symposium, the 2019 Senior Honors Thesis Symposium, and the 2019 Great Lakes Bioinformatics Conference. Finally, I believe that my graduate study will benefit from my practice in collaborations and communications through this project.

Even though this independent study is such a fantastic experience, there are few things that I would do differently if I could re-do this project. First, I would start using reproducible research techniques from the beginning stage of this project. Professor Karl Broman's talk on reproducible research really resonates with my experience. For example, I struggled in re-generating figures for my manuscript since I had not saved either some old data or R scripts that were used to plot these figures. Since the beginning of this project, I should have been backing up and organizing all my code, data, and outputs periodically. It would save me a great amount of time if I want to revalidate certain methods or reproduce the outputs. To make this project more reproducible for other researchers, we decided to publish our code using Jupyter Notebooks (our working repository: https://github.com/gitter-lab/t-cell-classification). I will definitely use this code releasing infrastructure for my future analysis-heavy projects. However, I would start preparing for these notebooks earlier, so I could "advertise" them in my posters and talks and get feedbacks. Other than making my research more reproducible, I would get more opportunities to share my early stage results with the community. During the 2018 Undergraduate Symposium, I have recieved many helpful feedbacks from statistics professors as well as my peers. I think early feedbacks could keep me on the "right" research trajectory and avoid wasting time on trying ideas that have already been invalidated by experienced researchers.

6 Acknowledgements

I would like to thank Alex Walsh and Professor Melissa Skala for sharing their T cell autofluorescence images. I would also like to express my great gratitude to Professor Anthony Gitter, my research advisor, for his patient guidance and enthusiastic encouragement.

Reference

1. D. M. Pardoll, "The blockade of immune checkpoints in cancer immunotherapy," Nat. Rev. Cancer **12**, 252–264 (2012).
2. N. P. Restifo, M. E. Dudley, and S. A. Rosenberg, "Adoptive immunotherapy for cancer: harnessing the T cell response," Nat. Rev. Immunol. **12**, 269–281 (2012).
3. N. Marek-Trzonkowska, M. Mysliwiec, A. Dobyszuk, M. Grabowska, I. Techmanska, J. Juscinska, M. A. Wujtewicz, P. Witkowski, W. Mlynarski, A. Balcerska, J. Mysliwska, and P. Trzonkowski, "Administration of CD4+CD25highCD127- Regulatory T Cells Preserves -Cell Function in Type 1 Diabetes in Children," Diabetes Care **35**, 1817–1820 (2012).
4. J. M. Szulczewski, D. R. Inman, D. Entenberg, S. M. Ponik, J. Aguirre-Ghiso, J. Castracane, J. Condeelis, K. W. Eliceiri, and P. J. Keely, "In Vivo Visualization of Stromal Macrophages via label-free FLIM-based metabolite imaging," Sci. Rep. **6**, (2016).
5. N. Pavillon, A. J. Hobro, S. Akira, and N. I. Smith, "Noninvasive detection of macrophage activation with single-cell resolution through machine learning," Proc. Natl. Acad. Sci. **115**, E2676–E2685 (2018).
6. B. Guo, C. Lei, H. Kobayashi, T. Ito, Y. Yalikun, Y. Jiang, Y. Tanaka, Y. Ozeki, and K. Goda, "High-throughput, label-free, single-cell, microalgal lipid screening by machine-learning-equipped optofluidic time-stretch quantitative phase microscopy," Cytometry A **91**, 494–502 (2017).
7. W. J. Godinez, I. Hossain, S. E. Lazic, J. W. Davies, and X. Zhang, "A multi-scale convolutional neural network for phenotyping high-content cellular images," Bioinformatics **33**, 2010–2019 (2017).
8. C. L. Chen, A. Mahjoubfar, L.-C. Tai, I. K. Blaby, A. Huang, K. R. Niazi, and B. Jalali, "Deep Learning in Label-free Cell Classification," Sci. Rep. **6**, (2016).
9. D. Bannon, E. Moen, E. Borba, A. Ho, I. Camplisson, B. Chang, E. Osterman, W. Graf, and D. Van Valen, "DeepCell 2.0: Automated cloud deployment of deep learning models for large-scale cellular image analysis," bioRxiv (2018).
10. N. Nitta, T. Sugimura, A. Isozaki, H. Mikami, K. Hiraki, S. Sakuma, T. Iino, F. Arai, T. Endo, Y. Fujiwaki, H. Fukuzawa, M. Hase, T. Hayakawa, K. Hiramatsu, Y. Hoshino, M. Inaba, T. Ito, H. Karakawa, Y. Kasai, K. Koizumi, S. Lee, C. Lei, M. Li, T. Maeno, S. Matsusaka, D. Murakami, A. Nakagawa, Y. Oguchi, M. Oikawa, T. Ota, K. Shiba, H. Shintaku, Y. Shirasaki, K. Suga, Y. Suzuki, N. Suzuki, Y. Tanaka, H. Tezuka, C. Toyokawa, Y. Yalikun, M. Yamada, M. Yamagishi, T. Yamano, A. Yasumoto, Y. Yatomi, M. Yazawa, D. Di Carlo, Y. Hosokawa, S. Uemura, Y. Ozeki, and K. Goda, "Intelligent Image-Activated Cell Sorting," Cell **175**, 266-276.e13 (2018).
11. K. Thomas, R.-K. Benjamin, P. Fernando, G. Brian, B. Matthias, F. Jonathan, K. Kyle, H. Jessica, G. Jason, C. Sylvain, I. Paul, A. Damián, A. Safia, W. Carol, and J. D. Team, "Jupyter Notebooks – a publishing format for reproducible computational workflows," Stand Alone 87–90 (2016).
12. A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, P. Golland, and D. M. Sabatini, "CellProfiler: image analysis software for identifying and quantifying cell phenotypes," Genome Biol. **7**, R100 (2006).
13. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE **86**, 2278–2324 (1998).

14. K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," ArXiv13126034 Cs (2013).
15. M. Weigert, U. Schmidt, T. Boothe, A. Müller, A. Dibrov, A. Jain, B. Wilhelm, D. Schmidt, C. Broaddus, S. Culley, M. Rocha-Martins, F. Segovia-Miranda, C. Norden, R. Henriques, M. Zerial, M. Solimena, J. Rink, P. Tomancak, L. Royer, F. Jug, and E. W. Myers, "Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy," bioRxiv (2018).
16. A. Walsh, K. Mueller, I. Jones, C. M. Walsh, N. Piscopo, N. N. Niemi, D. J. Pagliarini, K. Saha, and M. C. Skala, "Label-free Method for Classification of T cell Activation," bioRxiv (2019).
17. G. Bradski, "The OpenCV Library," Dr Dobbs J. Softw. Tools (2000).
18. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res. **12**, 2825–2830 (2011).
19. F. Chollet and others, *Keras* (2015).
20. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (2015).
21. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," ArXiv151200567 Cs (2015).
22. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09* (2009).
23. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," ArXiv170604599 Cs (2017).