Machine Learning for Analysis of Microscopy Images: A Practical Guide

Vadim Zinchuk[1,3] and Olga Grossenbacher-Zinchuk[2]

[1]Department of Neurobiology and Anatomy, Kochi University Faculty of Medicine, Kochi, Japan
[2]Clinical & Medical Affairs, Ziemer Ophthalmic Systems AG, Port, Switzerland
[3]Corresponding author: *zinchuk@kochi-u.ac.jp*

The explosive growth of machine learning has provided scientists with insights into data in ways unattainable using prior research techniques. It has allowed the detection of biological features that were previously unrecognized and overlooked. However, because machine-learning methodology originates from informatics, many cell biology labs have experienced difficulties in implementing this approach. In this article, we target the rapidly expanding audience of cell and molecular biologists interested in exploiting machine learning for analysis of their research. We discuss the advantages of employing machine learning with microscopy approaches and describe the machine-learning pipeline. We also give practical guidelines for building models of cell behavior using machine learning. We conclude with an overview of the tools required for model creation, and share advice on their use. © 2020 by John Wiley & Sons, Inc.

Keywords: convolutional neural networks • deep learning • image analysis • machine learning • microscopy

How to cite this article:
Zinchuk, V., & Grossenbacher-Zinchuk, O. (2020). Machine learning for analysis of microscopy images: A practical guide. *Current Protocols in Cell Biology*, *86,* e101. doi: 10.1002/cpcb.101

INTRODUCTION

Machine Learning (ML), referring to the creation of learning algorithms to solve specific problems, has demonstrated remarkable progress in various fields. Machine learning methodology is applied to large quantities of data and, therefore, can be particularly useful in medicine and biology.

The accelerated growth of ML in biomedical imaging is the result of several factors including: (a) the accumulation of large volumes of data generated by digital imaging devices, including microscopes; (b) the availability of powerful yet cost-effective hardware and software computational tools that can be applied to process data; and (c) the ability to utilize data from different microscopy sources by enhancing image resolution. The combination of these factors has made ML an important new tool for analyzing new details of cell-biological phenomena on multiple size scales.

The purpose of this review is to help researchers with no background in informatics to incorporate ML models into their workflow without becoming data scientists. We provide the most relevant information needed to understand the principles behind the creation of ML models and describe the ML microscopy image analysis pipeline. We also share practical insights into model creation. Finally, we describe the hardware and software tools needed for building the models. In addition, we have compiled a list of online resources with information on the model building tools and several multi-purpose image libraries (Table 1).

WHY USE MACHINE LEARNING FOR MICROSCOPY IMAGE ANALYSIS?

A Fundamentally Different Solution

Modern bioinformatics provides very capable tools for various tasks. Most algorithms

CURRENT PROTOCOLS
A Wiley Brand

Current Protocols in Cell Biology e101, Volume 86
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpcb.101

Table 1 Representative Online Resources Providing Information on the N+Model Building Tools and Multi-Purpose Image Libraries

Resource	URL
Python programming language. Relatively simple to learn, built for easy code readability. Commonly used for model creation.	*https://www.python.org*
CVonline: Image databases. A collated list of image and video databases useful for computer vision research and algorithm evaluation.	*http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm*
Micro-Net: Fluorescence cell segmentation dataset. A unified DL framework for the segmentation of cellular objects in fluorescence and histology images.	*https://warwick.ac.uk/fac/sci/dcs/research/tia/data/micronet/*
The fluorescence microscopy denoising dataset. Sets of images showing various degrees of noise.	*https://drive.google.com/drive/folders/1aygMzSDdoq63IqSk-ly8cMq0_owup8UM*
Biomedical image segmentation. Image libraries representing various image acquisition modalities, biological structure types, magnification levels, and image-acquisition parameters.	*http://www.cs.bu.edu/~betke/BiomedicalImageSegmentation/*

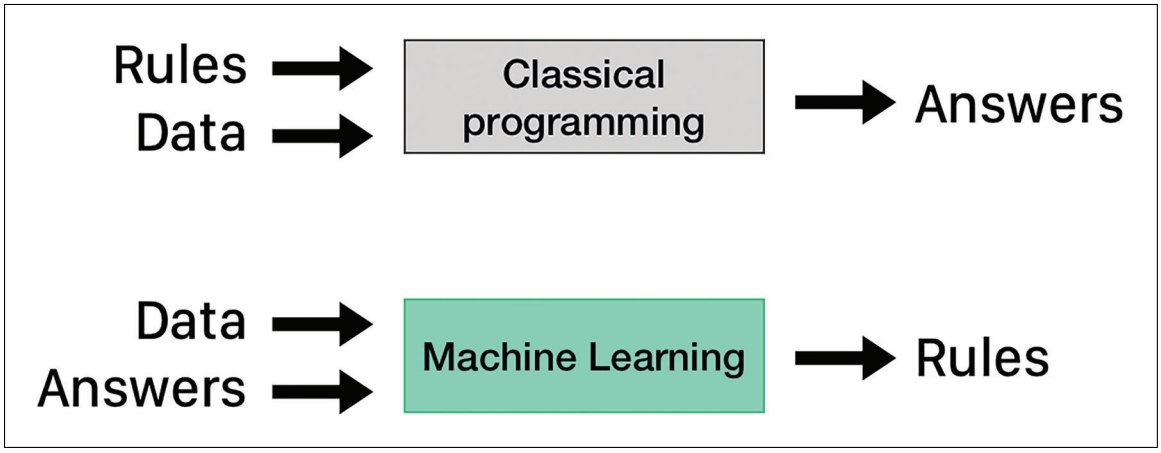

Figure 1 While classical programming gets answers while analyzing rules and data, ML learns processing rules from examples of data and answers, thus providing a fundamentally different solution.

used for analysis of images, however, are developed for very specific cases. These algorithms cannot be used for other situations without re-adjusting a variety of parameters. In some cases, their use even requires re-writing whole software packages from scratch (Sommer, Straehle, Köthe, & Hamprecht, 2011).

ML takes a completely different, more generalized approach. Instead of relying on manual changes to software parameters or predefined steps, it learns processing rules from examples (Domingos, 2012; also see Fig. 1).

Major Advantages

The major advantage of using ML for microscopy image analysis compared to imaging in general is that microscopy provides a very controllable environment where performance can be showcased by using carefully prepared training image sets. As a result, it allows focusing specifically on the tasks the model is trained for.

A Range of Successful Applications

ML has been applied with great success in various fields of microscopy, such as bright-field (Rivenson et al., 2017), holographic (Rivenson & Ozcan, 2018), and fluorescence (Wang et al., 2019). It has proven to be particularly useful for tasks like image reconstruction and transformation. Remarkably, once the model is trained, the inference (the process of taking a trained model and employing it to make predictions) is non-iterative (not requiring repetitive steps) and very fast to compute without the need for any parameter optimization. In comparison with traditional

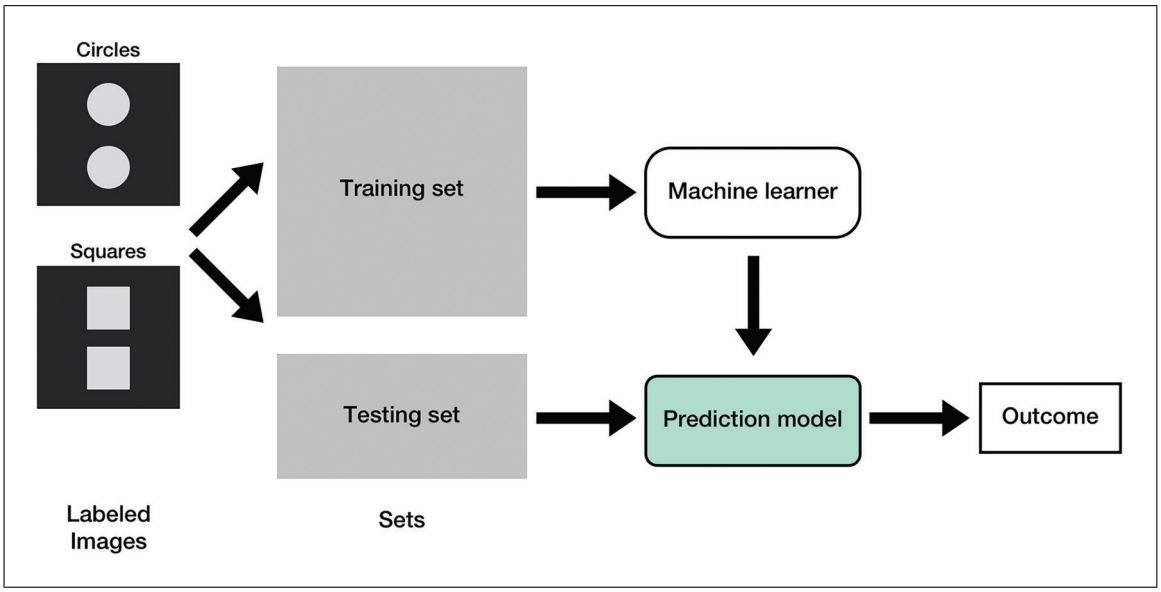

Figure 2 Using supervised learning, a model is created using labeled data sets.

solutions in microscopy, like, for example, deconvolution and convex optimization, a trained model, once optimized, presents significant advantages in terms of computational speed and inference time, even using modest computers and processors.

With the emergence of new processor architectures optimized for ML, this important advantage will become increasingly more evident. For example, the new processor architectures can perform inference tasks of neural networks in real time, even using mobile phones and other low-end consumer devices (Rivenson & Ozcan, 2018).

TYPES OF MACHINE LEARNING

Based on the availability of data and its characteristics, researchers can choose different approaches for creating ML models. Models can be created using supervised, unsupervised, semi-supervised, and reinforcement learning. The most popular models currently employed to analyze microscopy images are created using supervised learning.

In supervised learning, which is the focus here, researchers have a full set of data when training an algorithm. Therefore, the algorithm learns using a labeled data set, obtaining an answer key that the algorithm uses to evaluate its accuracy on training data (Fig. 2).

DEEP LEARNING

Recent progress in using ML has been led by the popularity of Deep Learning (DL; Litjens et al., 2017; Webb, 2018). DL is a subset of ML. In technical terms, it is still ML, but with different, more advanced, capabilities (Fig. 3).

The ML community coined the word "deep" for this type of neural network (NN) after implementing back-propagation, a method used for handling NN in a way that reduces error and produces good predictions. Before DL, there was only one hidden layer in NN. With the introduction of back-propagation, the number of hidden layers started to increase. As the number of layers in the hidden portion of NN became bigger, they went deeper and deeper. Hence, the name DL.

Why Use Deep Learning?

The popularity of DL is based on two major advantages:

- **Independence**. DL is built by creating a layered structure of algorithms called an artificial neural network (ANN). Success in application of ANN in computer vision, including analysis of microscopy images, is attributed to using convolutional neural networks (CNN; Fig. 4). Although conventional ML models do become better with time, they still require some guidance. If an ML algorithm returns an inaccurate prediction, then a programmer should intervene and make adjustments. In contrast, CNN in the DL model works independently of prior knowledge and human involvement, and its algorithms can determine on their own whether a prediction is accurate or not.
- **Efficiency** (biology-inspired). CNN is modeled after the connectivity pattern between neurons in the brain visual cortex.

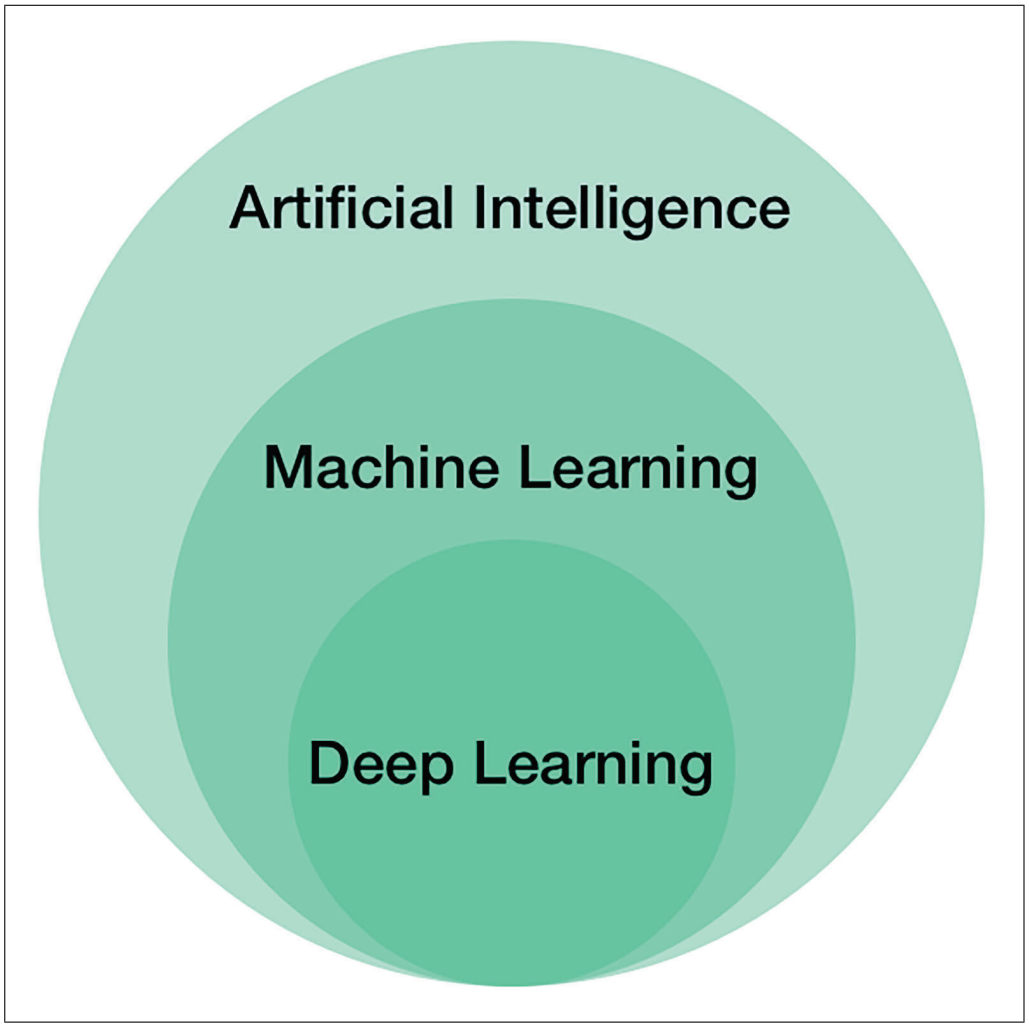

Figure 3 Visual representation of the relationship between artificial intelligence (AI), machine learning (ML), and deep learning (DL). Deep learning is a part of machine learning with extended capabilities. They both are parts of artificial intelligence.

Individual cortical neurons respond to stimulation only within a restricted region of the visual field. These visual fields, formed by different neurons, overlap to represent the entire field. Frank Rosenblatt laid out the foundation of NN by saying that "the aim was to formulate a brain analog useful for analysis" (Rosenblatt, 1957).

In essence, DL is the next logical evolutionary step of ML. A DL model can learn through its own method of computing, i.e., through its own "brain." It can make its own correct decisions without an engineer telling it what to do. This is what makes DL the closest to human-like artificial intelligence (AI).

THE MACHINE LEARNING PIPELINE

The purpose of the ML pipeline is to transform original raw data into units suitable as input for the respective ML algorithm and deliver a trained model capable of performing particular actions.

To understand the ML pipeline, we need to specify the meaning of its two crucial components—the classifier and learner. We also need to understand how the main goal of the ML process, generalization, is achieved.

Classifier

A classifier is a system that inputs a vector of discrete and/or continuous feature values and outputs a single discrete value, the class. For example, a classifier for recognizing types of shapes separates them into circles and squares.

Learner

A learner inputs a training set of examples and outputs a classifier. In our example of shapes, a proper learner will create a classifier that produces the correct output for

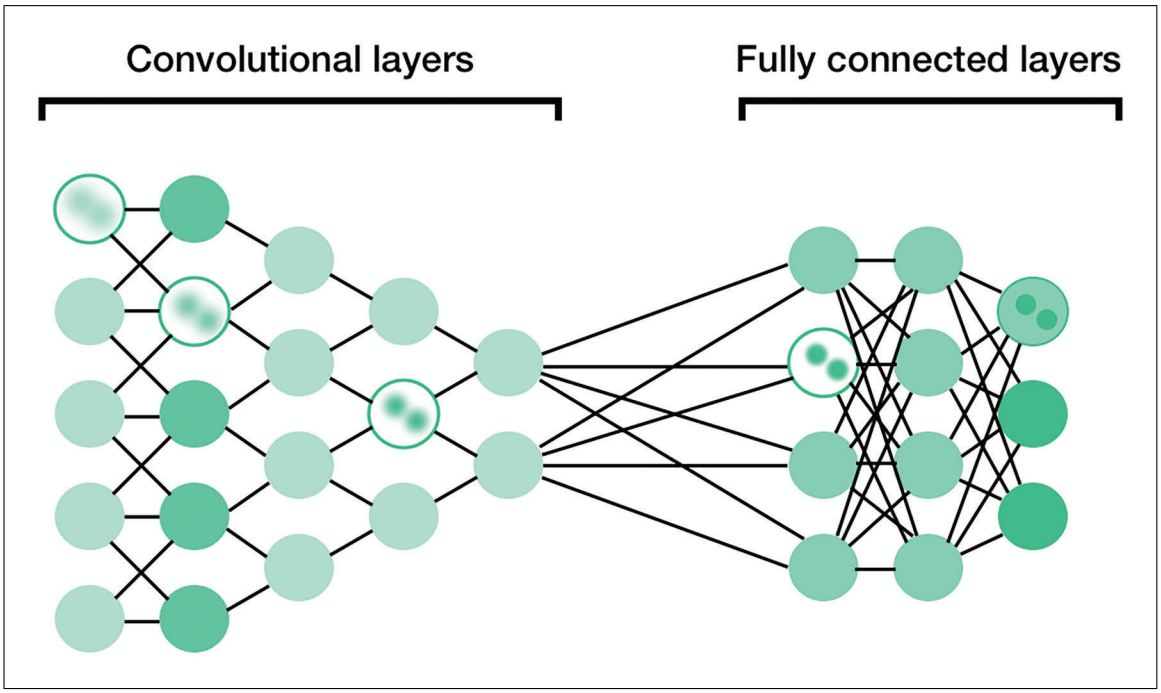

Figure 4 Schematic representation of CNN. Convolutional layers identify structural features of the images and integrate them into fully connected layers. A combination of layers of different types allows the network to recognize images with clarity.

future examples of unseen shapes of circles and squares.

The process of learning consists of a combination of three parts:

- **Representation**. A classifier must be represented in a language that the computer can understand.
- **Evaluation**. An evaluation function is required for distinguishing between good and bad classifiers.
- **Optimization**. Finally, the classifier needs to be optimized.

Generalization

The primary goal of ML is to generalize from a limited number of training examples in order to be able to make accurate predictions on large sets of data samples that were not observed during training (de Ridder, de Ridder, & Reinders, 2013). This is because, regardless of the volume of available data, it is highly unlikely that we will see those same examples again at the time of the test.

Generalization requires high-quality data and reliable representation of the feature that we would like to use to train the model for learning the unknown and underlying "true" mapping that exists from input to output.

Generalization means going from something very specific to something very broad. It resembles the way humans learn, and is directly related to the transfer of knowledge. For example, as humans, we do not memorize specific computer programs when learning to code. Instead, we learn general rules to solve problems with computer code for any case that may need it. Similarly, when studying tissue morphology, we learn about the properties of representative cells and then apply this knowledge to imagine the structure of whole tissues and organs containing them.

For cell-biological applications dealing with microscopy images, the ML pipeline consists of the following steps: (a) image pre-processing, (b) object detection, (c) feature extraction, (d) classifier training, and (e) classification (Fig. 5). As stated previously, supervised learning is widely used in biological studies utilizing ML tools.

Image Pre-Processing

This first step of the ML pipeline aims to remove artifacts produced by the microscope or acquisition method. This step is crucial, as it improves the quality of data.

For example, background noise in fluorescence images, consisting of signal-dependent photon noise and signal-independent read noise, plus the true rate of photon emission, should be removed or reduced to ensure reliable calculations of colocalization coefficients (Zinchuk, Wu, Grossenbacher-Zinchuk, & Stefani, 2011; also see Current Protocols

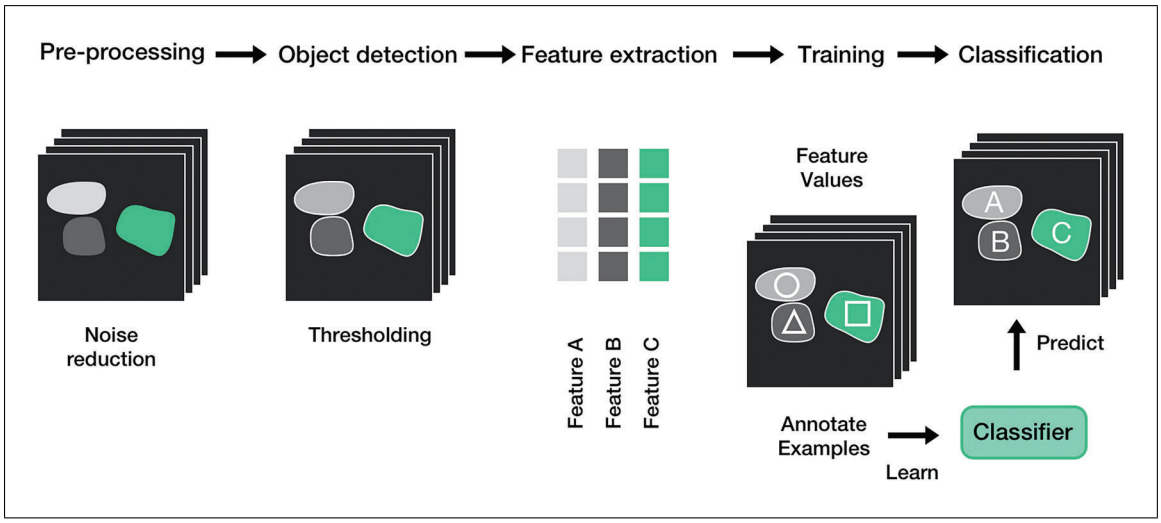

Figure 5 The ML pipeline for analysis of microscopy images. Steps include image pre-processing, object detection, feature extraction, classifier training, and classification. Training with a classifier, which learns from a representative set of annotated examples, yields a classification. Classification is then used for creating a model.

article Zinchuk & Grossenbacher-Zinchuk, 2018).

Pixel noise, resulting from low light exposure, should be removed using smoothing filters (Lindblad, Wählby, Bengtsson, & Zaltsman, 2004).

Image-restoration techniques can provide a higher resolution and improve results in fluorescence images obtained from microscopes operated at higher frame rates, shorter exposure times, and lower light intensities compared to the current state-of-the-art methods (Weigert et al., 2018).

Object Detection

The next step is to define the objects of interest in the image, which will form the basis for classification. This can be done by separating the objects of interest (e.g., cells) from the background, at the level of image pixels or by using whole unsegmented images (Fig. 6).

Object detection is based either on regional properties (bright regions can be segmented from the background by intensity thresholding) or on contours (edges of objects can be detected based on the local image gradient). As a rule, however, several approaches need to be applied for object detection, as no universal method can solve all segmentation problems.

Feature Extraction

After objects in images have been determined, they need to be described in quantitative terms suitable for identifying them by the classifier algorithm. It is important to remember that the performance of an ML pipeline relies primarily on the proper collection of relevant features (Tarca, Carey, Chen, Romero, & Drăghici, 2007; de Ridder et al., 2013). Defining the right set of features is, therefore, key to the success of a ML project.

Since raw image pixel intensities conceal the information on spatial and spectral patterns and can even contain unnecessary information on the absolute orientation of cell objects, they are not suitable as features (Huh, Lee, & Murphy, 2009). It is other descriptive features related to pixel intensities that are needed for suitable classification.

Two types of features are most commonly used to describe cell objects in microscopy images:

(1) **Contour features**. These describe the contour based on the segmentation mask, e.g., roughness or circularity (Liu, Mundra, & Rajapakse, 2011).

(2) **Texture features**. These quantify the distribution of pixel intensities within each object, such as granularity (Chebira et al., 2007) or pixel co-occurrence patterns (Haralick, 1979).

Many useful features are abstract representations of images and therefore difficult to distinguish upon visual examination. These features can be automatically determined by the learning algorithm and will differ depending on the specific marker and measurement. Several multi-purpose libraries have been introduced to assist with adaptations of specific features for particular applications (Held et al., 2010; Jones et al., 2008; Shariff, Kangas, Coelho, Quinn, & Murphy, 2010).

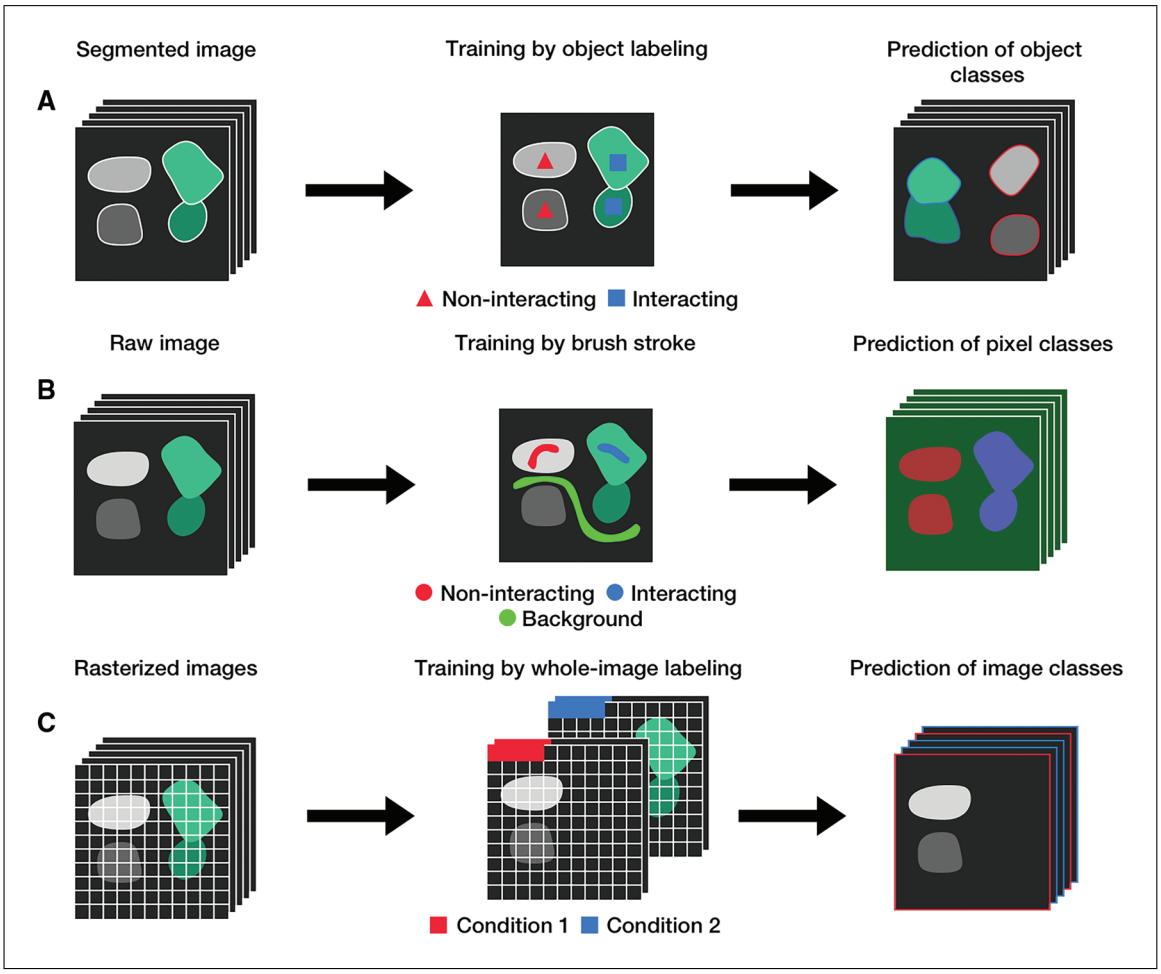

Figure 6 An example of image classification to distinguish between interacting and non-interacting molecules by supervised ML at the levels of (**A**) cell objects, (**B**) image pixels, and (**C**) whole images. (**A**) Each element is labeled according to its properties to learn a classifier. The classifier is then applied to unknown data to predict the cases (Held et al., 2010). (**B**) Pixels of cell elements and background are annotated by brush strokes according to pre-defined cases. Features of the labeled pixels and their local neighborhood are used to learn a classifier. Then, the classifier is used to predict unseen images in pixel-wise fashion (Sommer et al. 2011). (**C**) Image features characterize an image as a whole and classification outputs a class membership per image. Applied when segmentation of objects is difficult to achieve due to the high complexity of the objects of interest (Shamir et al., 2010).

The most efficient way to extract features is to automate the engineering process. One method for this is to generate a large number of candidate features and then select the best based on the information gain related to the class. It is important to realize, however, that features that seem irrelevant in isolation may become relevant in combination. Thus, there is virtually no replacement for trial and error and wisdom put into selecting the right features.

Interestingly, simply adding more features to the set does not necessarily improve the performance of an ML model. This is explained by the fact that the increase in dimensionality with each feature usually renders the classification task exponentially more complex. This is known as the "curse of dimensionality" (Hastie, Tibshirani, & Friedman, 2009).

Classifier Training

Classifier training produces a classifier by learning from representative training examples, which results in classification.

Classification

Classification is the process of predicting the class of given data points. Classes are sometimes called targets/labels or categories. Classification is the task of approximating a mapping function f from input variables x to discrete output variables y. It belongs to the category of supervised learning where the targets are provided with the input data.

Two types of learners exist in classification:

(1) **Eager learners**. This type of classifier creates a classification model based on the

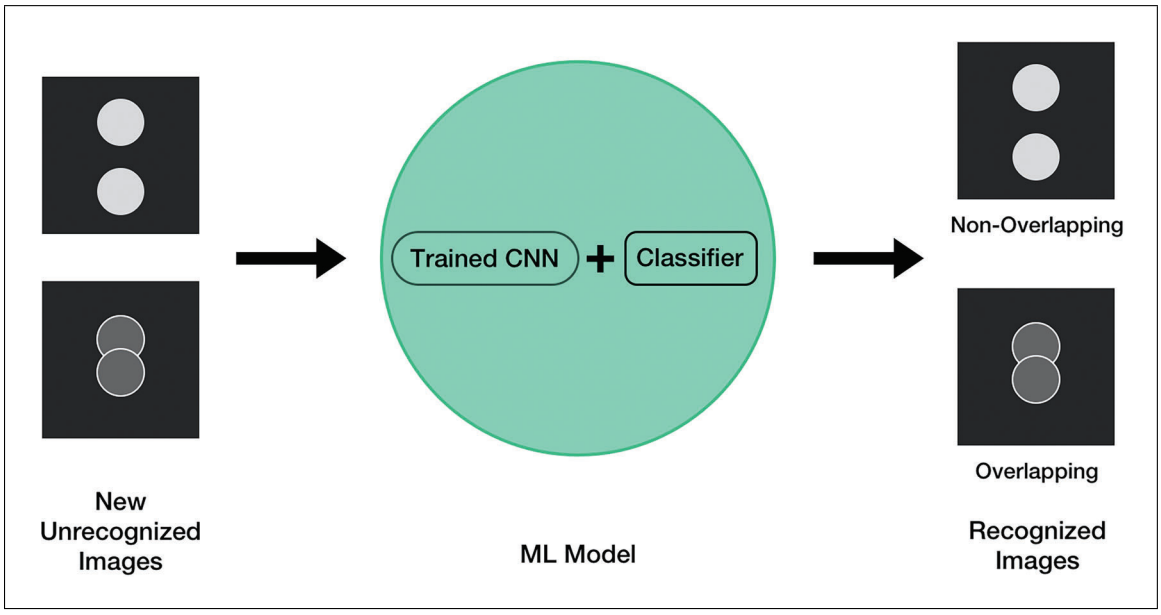

Figure 7 Application of ML model. Challenged with new unrecognized images, an ML model, consisting of trained CNN and a classifier, uses sets of images with annotated features to distinguish subjects of interest (non-overlapping and overlapping objects as an example) with high accuracy.

given training data before receiving data for classification. To work, it must be able to commit to a single hypothesis that covers the entire instance space. Due to the model architecture, eager learners take a long time to train and less time to predict.

(2) **Lazy learners**. This type of classifier stores the training data and waits until testing data appears. When it does, classification is conducted based on the most related data in the stored training data set. Compared to eager learners, lazy learners have less time for training but spend more time predicting.

After the classifier has been trained, it is important to evaluate its applicability. Several methods can be used for this task:

- **Holdout**. The given data set is divided into two parts: Train (80%) and Test (20%), respectively. The Train set will be used to train the model and the unknown Test set will be used to test its predictive properties.
- **Precision and recall**. Precision is the fraction of relevant instances among the retrieved instances. The recall is the fraction of relevant instances that have been retrieved divided by the total number of relevant instances. Precision and recall are used to measure relevance.
- **Cross-validation**. Overfitting is when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This is a common problem in machine learning that can affect most models. *k*-fold cross-validation is used to verify that the model is not overfitted. In this method, the data is divided into two *k* mutually exclusive sets of approximately the same size. One type is used for training, while the other is kept for testing. This process is repeated throughout all of the *k*-folds.

After establishing the ML pipeline, new images can be predicted automatically (Fig. 7).

MEASURING AND OPTIMIZING MODEL PERFORMANCE

Total Error

The performance of ML can be measured by calculating a total error, which is the ratio of incorrect classifications divided by the number of objects. Depending on the learning task, it may be beneficial to split the total error into false-positive and false-negative errors, thus enabling targeted optimization strategies.

Accurate evaluation of the performance of an ML method requires a comprehensive and representative set for the specific goal.

How Many Objects are Needed?

Determining the number of objects needed to train a good learner is, perhaps, the most important challenge in setting the entire pipeline. Unfortunately, there is no universal solution to this, because the number depends on the applied method and the variability within the

data set. In practical settings, some applications produce acceptable results by training with as few as 10 objects per class. However, most applications will require significantly more. Regardless of the learning algorithm, an increase in the number of features will require more training examples (de Ridder et al., 2013; Hastie et al., 2009).

Since the most important criterion for a learner is the ability to generalize (Hastie et al., 2009), the success in generalization can be estimated by splitting the annotated reference data into three subsets:

(1) Objects used for initial learning.
(2) Objects used to improve the parameter settings of the learner.
(3) An independent test.

This procedure prevents overfitting and ensures good generalization (de Ridder et al., 2013; Hastie et al., 2009).

Using a Limited Number of Training Objects

Because the number of training objects is often limited, it is important to use the available ones as efficiently as possible. A procedure called *k*-fold cross-validation was developed for this task (Ambroise & McLachlan, 2002). With it, the training set is divided into a user-defined number of *k* subsets. All of the subsets except one are used for the initial training of the learner. The remaining subset is used for estimating the performance of the learner and optimizing its parameters. This is repeated for all subsets of the data, usually 5 to 10 times.

Overrepresentation of a Class

When the ratio of representation of classes in the data set is different and a specific class is overrepresented, the resulting learner may perform inadequately on predicting the less-abundant classes. This issue can be addressed by sub-sampling only a portion of training objects from the abundant classes while preserving all training objects from the less-abundant classes, or by specialized learning algorithms (Kotsiantis, Kanellopoulos, & Pintelas, 2006).

Expected Accuracy

The overall accuracy depends on the quality of data and the number of various parameters used in the study. Many microscopy applications have achieved total accuracies of 90% and higher, corresponding to the levels achieved by trained humans.

IMPORTANT GUIDELINES

Data Quality is Paramount

ML is designed to generalize from examples, but it will do so only from the variability present in the training data. Slight differences in the data can make a significant impact on the quality of the learner. Thus, close attention should be paid to the data quality and the reproducibility of conditions under which the data has been acquired.

Importantly, ML is not a one-shot process of creating a data set and running a learner, but a sequence of repetitive steps consisting of running the learner, analyzing results, and modifying the data and/or the learner. The selection of features is usually the most difficult and time-consuming part, because it is application specific, while learners can be general purpose (Domingos, 2012).

More Data is Crucial

Since ML is all about letting data do the job, the amount of data is of paramount importance. If you have gathered the best possible features, but the classifier is still not accurate enough, then the best way to improve your model is to simply add more data, rather than to try to improve the learning algorithm using the existing data sets.

Adding more data should be well thought out, however, as too much data will take a classifier too long to learn. Thus, you should carefully balance the accuracy of the classifier with computational costs. The best solution usually comes from experimenting with different learners and data sets, taking into account the specific needs of the project.

Data Alone is not Enough

To be able to generalize the data, every learner must contain some knowledge or assumptions beyond the provided data sets (Wolpert, 1996). Therefore, data alone is not enough, regardless of how much of it you have (Domingos, 2012).

The best learners are those that do not simply have assumptions as to their permanent features, but allow them to be stated explicitly and varied extensively, and to be incorporated into the process of learning automatically (Domingos, 2012; Richardson & Domingos, 2006).

Select Features Thoughtfully

When deciding on the features, consider general-purpose sets first. These sets work well for the majority of cell-biological assays

(Held et al., 2010; Jones et al., 2008). Lately, sets for more specific use, e.g., a dataset for Poisson-Gaussian denoising with real fluorescence microscopy images, have started appearing as well (Zhang et al., 2018). If general-purpose sets seem unsatisfactory, consider designing custom sets with specialized features.

When designing custom sets, try to select independent features that correlate well with the class. This will make learning much easier.

Maintain Steady Experimental Conditions

It is crucial to keep the experimental conditions unchanged. This applies to all steps of experiments, such as sample preparation, microscope set-up, and image acquisition and handling. If necessary, add control tests during experiments and evaluate data quality and reproducibility using automated quality control (Zeder, Kohler, & Pernthaler, 2010). It is important to take these measures as early as possible within the course of the experiment, to ensure consistency of the data.

Always Maintain Optimal Image Focus

Keeping an image in focus is perhaps the single most impactful feature for obtaining high-quality microscopy data sets. It is therefore advisable to apply autofocus options to one's microscope system to avoid acquiring out-of-focus images.

Ensure Constant Illumination Intensity

In addition to maintaining good image focus, maintaining illumination intensity is also extremely important. Changing levels of illumination can result in different levels of image noise, which can affect classification. It has also been shown that when image features change due to experimental conditions (rather than to biological conditions), the reliability of ML methods is significantly compromised (Shamir, Delaney, Orlov, Eckley, & Goldberg, 2010).

CHALLENGES

Although ML has obvious advantages, it also faces several specific challenges:

- **Object crowding and overlapping.** These are perhaps some of the most difficult challenges to solve. Object crowding and overlapping occur when images contain a large number of objects of interest that overlap by grouping with each other. This is frequently observed in microscopy images when searching for colocalization. Overlapping as a result of colocalization may have important biological implications for researchers, but for image-recognition algorithms, it creates a hurdle for automatic detection and segmentation of individual objects. This challenge has been partially addressed by carefully designing task-specific objective functions and limiting requirements for accurate delineation of object boundaries (Sirinukunwattana et al., 2016). More substantial improvements are still required.
- **Low quality of data annotations**. Obtaining high-quality annotations of training image sets is crucial for creating efficient ML models (Russakovsky et al., 2015). This is frequently impossible to achieve for microscopy images, however. Low-quality data annotations can be caused by (a) insufficient annotations, when not enough information about images is provided; (b) imbalanced annotations, when some images have significantly more information than others; or (c) inconsistent annotations, when information about images is provided in a way that is difficult to use when forming data sets.
- **Limited database**. Since models are usually created to solve very specific problems, researchers frequently have a limited number of images to annotate. There are several ways to increase the number of images. One of them is to use data augmentation. The data sets can be enlarged using label-preserving transforms, such as translation, rotation, reflection, and random crops (Ciresan, Meier, & Schmidhuber, 2012). The problem, however, is that the augmented sets are potentially highly correlated. Another way is to generate additional images using semi-automated annotation methods, but this is not easy to accomplish in practical settings (Weese & Lorenz, 2016). Finally, the database can be improved using selected crowdsourcing, when a group of researchers from different institutions with access to specific types of images volunteers to annotate a large-scale data set (Estellés-Arolas & González-Ladrón-de-Guevara, 2012). Although this is an acceptable solution, some discrepancies can arise due to individual interpretations of the same type of image.
- **Image artifacts**. Microscopy images are acquired using several steps including tissue collection, embedding, sectioning, staining, and imaging. These are all prone to errors. The errors can lead to visual artifacts, such as uneven tissue distribution, contamination, or incompletely stained regions. These

undesired outcomes impact image quality and are challenges for automated image computing, including DL.

- **Batch effects**. Due to different preparations and microscope devices, color, staining-intensity, or magnification variations can occur between batches of samples, even those from the same source. These batch effects can cause bias in the performance evaluation of predictive models (Kothari, Phan, Stokes, & Wang, 2013). This is especially important for fluorescence microscopy experiments, because the intensity of fluorescence staining tends to fade with time and exposure to light. Therefore, algorithms developed for microscopy image analysis need to be robust to take into consideration these important variations. Although some solutions exist to remedy this problem, they require manual tinkering with the relevant parameters (Bejnordi et al., 2016).

MODEL BUILDING TOOLS

Hardware

The creation of ML models requires very capable hardware and specific software frameworks. The most important resources are listed in Table 1.

CPU

The CPU (central processing unit) is designed to execute a few very complex operations very efficiently. However, ML requires a different type of computation. Most computational tasks involved in the training of models are matrix multiplications. These are very small and easy, but there are many of them, which can overpower the CPU.

GPU

Fortunately, another major processor chip has been introduced into computers that is significantly better suited for ML tasks. This chip, called the graphics processing unit (GPU), has the core complexity to do basic operations but can do them at scale all at the same time.

GPUs first appeared in gaming applications in the early 1970s. In the late 1980s, they were added to consumer computers. Nowadays, the GPU is a standard component of all computers. What makes a GPU unique is how it handles commands—it is the exact opposite of a CPU.

The GPU utilizes a parallel architecture. While a CPU is very good at handling one set of very complex instructions, a GPU excels at handling many sets of very simple instructions.

Several years ago, the ML community began to realize that these architectural features of GPUs may be beneficial for using them in training algorithms. This is because GPUs show massive speed improvements over CPUs for training models.

Which Processor to Use?

The choice of a processor depends on the priorities for the ML process, which range from speed to accuracy and reliability. As a rule, if a good GPU is available, it is the number one choice due to its superior speed. However, if speed is not an issue, then CPU can work just as well. Besides, one needs to consider differences in your model-training and model-implementation needs. A GPU can be superior for training, but the CPU will be powerful enough to use once the model has been built.

Software

In addition to hardware, the other powerful engine behind the popularity of DL is the broad availability of open-source software. These framework libraries support the use of GPU for important operations, including DNN. Currently, the most popular packages are:

- **Tensorflow**

  Provides C++ and Python interfaces, developed by Google and is used by Google research (Abadi et al., 2016).
- **Theano**

  Provides a Python interface, developed by the MILA lab in Montreal (Bastien et al., 2012).
- **Keras**

  Written in Python. Developed with a focus on enabling fast experimentation. Capable of running on top of Tensorflow and Theano.
- **Caffe**

  Provides C++ and Python interfaces, developed by graduate students at UC Berkeley (Jia et al., 2014).

Which Software Frameworks to Use?

When deciding on the frameworks to use, researchers need to ask themselves the following questions:

- What kind of model is going to be created, ML or DL?
- What programming language will be used?
- What kind of hardware it will be used on and whether you plan to make it available for others to use as well (scale)?

As DL models are very popular now, the frameworks of choice are Tensorflow and Caffe. Their advantage is the ability to write efficient algorithms for image annotation.

As for the language, the most popular software packages rely on Python, a general-purpose language built for easy code readability. It is relatively simple to learn, which is why it is so popular among researchers.

When selecting the hardware, you need to decide what kind of resources your task requires. If your task is relatively small and needs only sequential processing, then a small system is satisfactory. You may settle for just a CPU. Modern CPUs, such as the Intel i7 series, can execute training at an average of 100 examples per second. In many cases, this is good enough.

If your task is more intensive and includes manageable data, then a powerful GPU would be a better solution. In a case like this, a portable computer with an appropriate GPU will do the job. If your notebook computer is equipped with a high-end GPU, like Nvidia GTX 1080 or Radeon Pro 555X, then you can achieve training in the range of 10,000 to 15,000 examples per second. Another option is to build your own PC with a reasonable CPU and a powerful GPU, but you need to remember that CPU must not bottleneck GPU. A bottleneck occurs when there is a huge difference in processing speed between CPU and GPU. In the given example, the Intel i7 series CPUs will work very well with either the Nvidia GTX 1080 or Radeon Pro 555X.

If you are trying to solve more complex learning problems, then you may consider building your own DL system or using a commercial cloud service.

For projects of a particularly large scale, it may be appropriate to use a GPU cluster or opt for multi-GPU computing. Since these solutions can be expensive, their costs should be considered as well.

Distribution

Finally, you need to decide whether you are planning to use the created ML model for yourself and your team only, or are considering making it available for other researchers as well (i.e., scale it). Scaling means that your model will work for many scientists, in different locations, and perform at an acceptable speed. If you plan to scale your model, you need to design your project with the following in mind:

- You need a lot of data to train your model, especially in the case of creating DL models.
- All data sets should be prepared before the training of the model starts.
- When training DL models, you should use a powerful GPU.

Scaling can be challenging and should be considered only when appropriate resources are available.

CONCLUSION AND OUTLOOK

Taking into account the current growth of tools employing ML in analytical microscopy, it is not hard to predict that this trend will continue and expand further. We expect that soon ML will tackle very ambitious tasks and gradually start replacing the majority of manually programmed pipelines used in microscopy image analysis today. By doing so, it will introduce new approaches that will dramatically increase processing output and precision of the essays.

The following directions will likely take the central stage in shaping up the future use of ML in microscopy image analysis:

- Identifying cells without labeling them.

A recent study described a DL approach called “in silico labeling” (Christiansen et al., 2018). With it, an ML model can find and predict features in images of unlabeled cells. Methods based on this approach should be able to uncover important information about cells that would otherwise be difficult or impossible to obtain, and open a whole avenue for totally new DL approaches.

- Integrating deeply into the image-acquisition process.

ML tools are starting to become deeply integrated into the image-acquisition process (Conrad et al., 2011). This process will likely continue further and accelerate. For example, we can expect ML tools to become incorporated into cameras and vision sensors of microscopes. As a result, ML will help to automate the design of an image-processing pipeline in automated optical inspection systems. It will also enable the creation of highly elaborated workflows limited only by human imagination.

- Enabling ML on portable devices.

The development of new and affordable processor architecture optimized specifically for DL will streamline the creation of ML models and enable the involvement of more researchers in the field. It will also allow using significantly less powerful devices, such as tablets and mobile phones, for model creation.

LITERATURE CITED

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. Available at http://download.tensorflow.org/paper/whitepaper2015.pdf.

Ambroise, C., & McLachlan, G. J. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(10), 6562. doi: 10.1073/pnas.102102699.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., … Bengio, Y. (2012). Theano: New features and speed improvements. Available at https://arxiv.org/pdf/1211.5590v1.pdf.

Bejnordi, B. E., Litjens, G., Timofeeva, N., Otte-Höller, I., Homeyer, A., Karssemeijer, N., & van der Laak, J. A. (2016). Stain specific standardization of whole-slide histopathological images. *IEEE Transactions on Medical Imaging*, *35*(2), 404–415. doi: 10.1109/TMI.2015.2476509.

Chebira, A., Barbotin, Y., Jackson, C., Merryman, T., Srinivasa, G., Murphy, R. F., & Kovačević, J. (2007). A multiresolution approach to automated classification of protein subcellular location images. *BMC Bioinformatics*, *8*(1), 210. doi: 10.1186/1471-2105-8-210.

Christiansen, E. M., Yang, S. J., Ando, D. M., Javaherian, A., Skibinski, G., Lipnick, S., … Finkbeiner, S. (2018). In Silico labeling: Predicting fluorescent labels in unlabeled images. *Cell*, *173*(3), 792–803.e719. doi: 10.1016/j.cell.2018.03.040.

Ciresan, D. C., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. Available at https://arxiv.org/abs/1202.2745.

Conrad, C., Wunsche, A., Tan, T. H., Bulkescher, J., Sieckmann, F., Verissimo, F., … Ellenberg, J. (2011). Micropilot: Automation of fluorescence microscopy-based imaging for systems biology. *Nature Methods*, *8*(3), 246–249. doi: 10.1038/nmeth.1558.

de Ridder, D., de Ridder, J., & Reinders, M. J. T. (2013). Pattern recognition in bioinformatics. *Briefings in Bioinformatics*, *14*(5), 633–647. doi: 10.1093/bib/bbt020.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87. doi: 10.1145/2347736.2347755.

Estellés-Arolas, E., & González-Ladrón-de-Guevara, F. (2012). Towards an integrated crowdsourcing definition. *Journal of Information Science*, *38*(2), 189–200. doi: 10.1177/0165551512437638.

Haralick, R. M. (1979). Statistical and structural approaches to texture. *Proceedings of the IEEE*, *67*(5), 786–804. doi: 10.1109/PROC.1979.11328.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. New York: Springer-Verlag.

Held, M., Schmitz, M. H. A., Fischer, B., Walter, T., Neumann, B., Olma, M. H., … Gerlich, D. W. (2010). CellCognition: Time-resolved phenotype annotation in high-throughput live cell imaging. *Nature Methods*, *7*(9), 747–754. doi: 10.1038/nmeth.1486.

Huh, S., Lee, D., & Murphy, R.F. (2009). Efficient framework for automated classification of subcellular patterns in budding yeast. *Cytometry A*, *75*(11), 934–940. doi: 10.1002/cyto.a.20793.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., … Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. doi: 10.1145/2647868.2654889.

Jones, T. R., Kang, I. H., Wheeler, D. B., Lindquist, R. A., Papallo, A., Sabatini, D. M., … Carpenter, A. E. (2008). CellProfiler Analyst: Data exploration and analysis software for complex image-based screens. *BMC Bioinformatics*, *9*(1), 482. doi: 10.1186/1471-2105-9-482.

Kothari, S., Phan, J. H., Stokes, T. H., & Wang, M. D. (2013). Pathology imaging informatics for quantitative analysis of whole-slide images. *Journal of the American Medical Informatics Association*, *20*(6), 1099–1108. doi: 10.1136/amiajnl-2012-001540.

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, *30*, 25–36.

Lindblad, J., Wählby, C., Bengtsson, E., & Zaltsman, A. (2004). Image analysis for automatic segmentation of cytoplasms and classification of Rac1 activation. *Cytometry Part A*, *57A*(1), 22–33. doi: 10.1002/cyto.a.10107.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., … Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, *42*, 60–88. doi: 10.1016/j.media.2017.07.005.

Liu, S., Mundra, P. A., & Rajapakse, J. C. (2011). Features for cells and nuclei classification. *Paper presented at the 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 30 Aug.–3 Sept. 2011.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1), 107–136. doi: 10.1007/s10994-006-5833-1.

Rivenson, Y., Göröcs, Z., Günaydin, H., Zhang, Y., Wang, H., & Ozcan, A. (2017). Deep learning microscopy. *Optica*, *4*(11), 1437–1443. doi: 10.1364/OPTICA.4.001437.

Rivenson, Y., & Ozcan, A. (2018). Toward a thinking microscope. *Optics and Photonics News*, *29*(7), 34–41. doi: 10.1364/OPN.29.7.000034.

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton: (Project Para). *Cornell Aeronautical Laboratory report, 85-460-1*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. doi: 10.1007/s11263-015-0816-y.

Shamir, L., Delaney, J. D., Orlov, N., Eckley, D. M., & Goldberg, I. G. (2010). Pattern recognition software and techniques for biological image analysis. *PLOS Computational Biology*, *6*(11), e1000974. doi: 10.1371/journal.pcbi.1000974.

Shariff, A., Kangas, J., Coelho, L. P., Quinn, S., & Murphy, R. F. (2010). Automated image analysis for high-content screening and analysis. *Journal of Biomolecular Screening*, *15*(7), 726–734. doi: 10.1177/1087057110370894.

Sirinukunwattana, K., Raza, S. E. A., Tsang, Y., Snead, D. R. J., Cree, I. A., & Rajpoot, N. M. (2016). Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transactions on Medical Imaging*, *35*(5), 1196–1206. doi: 10.1109/TMI.2016.2525803.

Sommer, C., Straehle, C., Köthe, U., & Hamprecht, F. A. (2011). Ilastik: Interactive learning and segmentation toolkit. *Paper presented at the 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. 30 March–2 April 2011.

Tarca, A. L., Carey, V. J., Chen, X.-W., Romero, R., & Drăghici, S. (2007). Machine learning and its applications to biology. *PLOS Computational Biology*, *3*(6), e116. doi: 10.1371/journal.pcbi.0030116.

Wang, H., Rivenson, Y., Jin, Y., Wei, Z., Gao, R., Günaydın, H., … Ozcan, A. (2019). Deep learning enables cross-modality super-resolution in fluorescence microscopy. *Nature Methods*, *16*(1), 103–110. doi: 10.1038/s41592-018-0239-0.

Webb, S. (2018). Deep learning for biology. *Nature*, *554*(7693), 555–557. doi: 10.1038/d41586-018-02174-z.

Weese, J., & Lorenz, C. (2016). Four challenges in medical image analysis from an industrial perspective. *Medical Image Analysis*, *33*, 44–49. doi: 10.1016/j.media.2016.06.023.

Weigert, M., Schmidt, U., Boothe, T., Müller, A., Dibrov, A., Jain, A., … Myers, E. W. (2018). Content-aware image restoration: Pushing the limits of fluorescence microscopy. *Nature Methods*, *15*(12), 1090–1097. doi: 10.1038/s41592-018-0216-7.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, *8*(7), 1341–1390. doi: 10.1162/neco.1996.8.7.1341.

Zeder, M., Kohler, E., & Pernthaler, J. (2010). Automated quality assessment of autonomously acquired microscopic images of fluorescently stained bacteria. *Cytometry Part A*, *77A*(1), 76–85. doi: 10.1002/cyto.a.20810.

Zhang, Y., Zhu, Y., Nichols, E., Wang, Q., Zhang, S., Smith, C., & Howard, S. (2018). A Poisson-Gaussian denoising dataset with real fluorescence microscopy images. Available at http://export.arxiv.org/pdf/1812.10366.

Zinchuk, V., & Grossenbacher-Zinchuk, O. (2018). Mobile quantitative colocalization analysis of fluorescence microscopy images. *Current Protocols in Cell Biology*, *78*(1), 4.37.31–34.37.12. doi: 10.1002/cpcb.43.

Zinchuk, V., Wu, Y., Grossenbacher-Zinchuk, O., & Stefani, E. (2011). Quantifying spatial correlations of fluorescent markers using enhanced background reduction with protein proximity index and correlation coefficient estimations. *Nature Protocols*, *6*(10), 1554–1567. doi: 10.1038/nprot.2011.384.