Avoiding a replication crisis in deep-learning-based bioimage analysis

Deep learning algorithms are powerful tools for analyzing, restoring and transforming bioimaging data. One promise of deep learning is parameter-free one-click image analysis with expert-level performance in a fraction of the time previously required. However, as with most emerging technologies, the potential for inappropriate use is raising concerns among the research community. In this Comment, we discuss key concepts that we believe are important for researchers to consider when using deep learning for their microscopy studies. We describe how results obtained using deep learning can be validated and propose what should, in our view, be considered when choosing a suitable tool. We also suggest what aspects of a deep learning analysis should be reported in publications to ensure reproducibility. We hope this perspective will foster further discussion among developers, image analysis specialists, users and journal editors to define adequate guidelines and ensure the appropriate use of this transformative technology.

Romain F. Laine, Ignacio Arganda-Carreras, Ricardo Henriques and Guillaume Jacquemet

Microscopy is a leading technology used to gain fundamental insight for biological research. Today, a typical microscopy session may generate hundreds to thousands of images, generally requiring computational analysis to extract meaningful results. Over the last few years, deep learning (DL) has increasingly become one of the gold standards for high-performance microscopy image analysis[1,2]. DL has shown the capacity to efficiently perform a wide range of image analyses, such as image classification[3,4], object detection[5,6], image segmentation[7–9], image restoration[10,11], super-resolution microscopy[10,12–15], object tracking[16,17], image registration[18] and the prediction of fluorescence images from label-free imaging modalities[19].

For image analysis, DL usually uses algorithms called artificial neural networks (ANNs). Unlike classical algorithms, before an ANN is used, it first needs to be trained (Fig. 1). During training, the ANN is presented with a range of data from which it attempts to learn how to perform a specific task (image denoising, for instance). More specifically, the ANN builds a model of the mathematical transformation that needs to be applied to data to obtain the desired output. Here, the model parameters (called weights) can be seen as the instructions to carry out the learned task. Once the weights of a model are optimized, the model can be used to perform the task—a step called inference or prediction. ANNs can therefore be considered as non-linear transformation machines, performing sequential mathematical operations on the input data. As we inspect deeper into these sequences of operations, it becomes difficult to understand what features of the original images are used. For that reason, they are often thought of as 'black boxes' as, for most users, only the input images and output predictions are readily available.

The training data provided to the ANN commonly consist of a large set of representative input images and their expected results. For instance, in denoising, the training dataset comprises matched pairs of noisy and high signal-to-noise ratio (SNR) images (Fig. 1). This type of training using paired image labels is commonly referred to as supervised training. On the other hand, for so-called self-supervised training, pre-processing steps directly generate the training pairs, and therefore the users only need to provide input images. Training is typically the most challenging, time-consuming and resource-greedy part of the process and can take minutes to weeks depending on the size of the training dataset and the type of ANN. It often requires specialized knowledge, dedicated training datasets and access to powerful computational resources, such as graphical processing units, to run and optimize ANN training. In comparison, using DL models (predictions) can be straightforward (parameter-free one-click solutions) and fast (seconds to minutes) even on a local machine. Multiple tools are in development to facilitate the training and use of DL for bioimage analysis, including both online and offline, commercial and open-source solutions[8,20–28]. Choosing the most appropriate tool out of these options largely depends on what tasks or combination of tasks need to be performed, the scale of the analysis and the level of computational skills required to run them.

In this Comment, we suggest a set of best practices for implementing and reporting on the use and development of DL image analysis tools. These suggestions are primarily built from our own experiences in developing and using DL tools. Here we cover traditional image analysis tasks, such as cell segmentation, as well as more recent approaches for image denoising and restoration.

The upsides and downsides of using DL for bioimaging analysis

Learning how to perform an analysis from example data is both the principal strength and the main weakness of DL. By learning directly from the data, the ANN tries to identify the most suitable way to perform the analysis, leading to models with excellent performances for that particular dataset (Fig. 2). However, trained DL models are only as good as the data, and the parameters used to train them. In particular, data augmentation, a method used to artificially increase training dataset size using controlled image manipulations — such as rotations, mirroring, noise addition, shearing and so on — can significantly influence model performance. Eventually, a DL model will only perform reliably on images similar to those used during training[29]. How similar the images need to be depends on the network used, and aspects to consider encompass microscope types, label types and the SNR or optical aberrations. If a training dataset is inadequate for the desired task, the resulting model will produce unwanted results

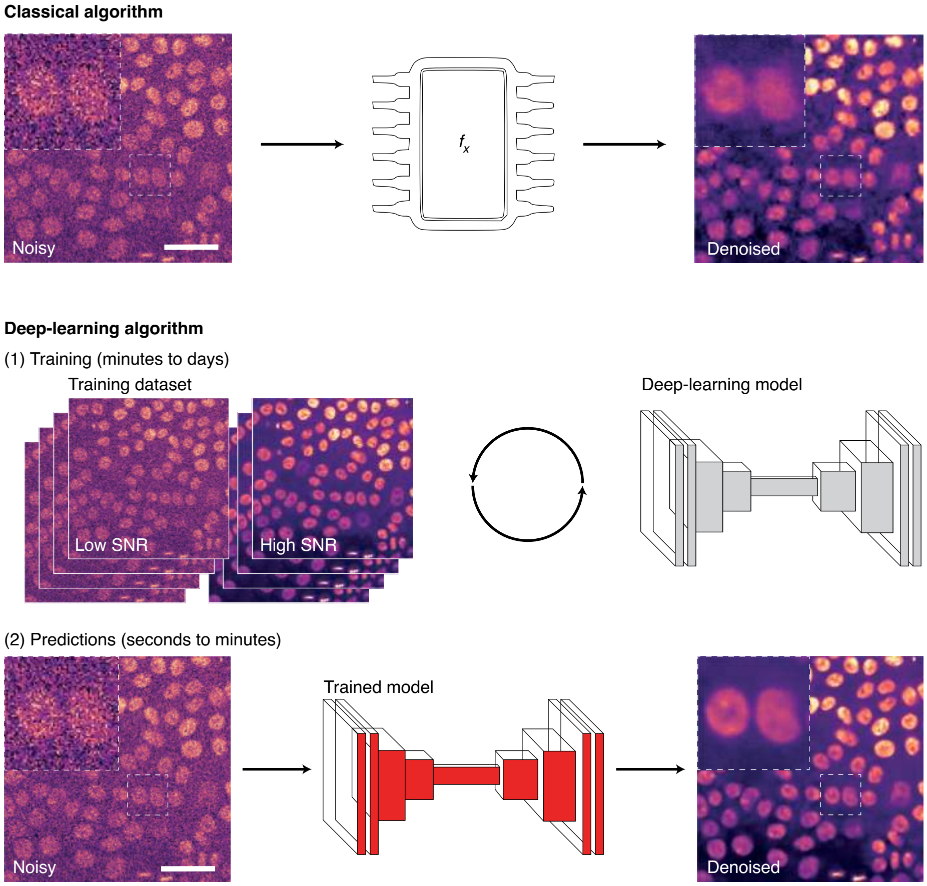


Fig. 1 | Use of classical or DL algorithms to analyze microscopy images. The critical steps required when using classical or DL-based algorithms to analyze microscopy images, using denoising as an example. When using a classical algorithm, researcher efforts are put into designing mathematical formulae (f_x) that can then be directly applied to the images. When using a DL algorithm, a model first needs to be trained using a training dataset. Next, the model can be directly applied to other images and generate predictions. The microscopy images displayed are breast cancer cells labeled with silicon rhodamine (SiR)-DNA to visualize the nuclei and imaged using a spinning disk confocal microscope. The denoising in the classical algorithm example was performed using PureDenoise implemented in Fiji[61,62]. The denoising in the DL algorithm example was performed using CSBDeep content-aware restoration (CARE) implemented in ZeroCostDL4Mic[10,20]. Scale bars, 50 µm.

that are often difficult to detect without detailed analysis of the network output. For instance, unsuitable segmentation models will lead to under- and over-segmentation results, while inappropriate image denoising and restoration models may lead to poor performance, image degradation and hallucinations (Fig. 2; see refs. [29,30] for reviews on this topic). In this case, non-DL-based approaches are likely to produce better results (Fig. 2). Users should be particularly cautious when using models based on generative adversarial networks (GANs), as they are designed to produce very realistic images while being prone to hallucinations, which are especially difficult to identify by eye.

One powerful approach is to produce general models with high reusability potential using a large and diverse training dataset. For example, popular nuclei or cell segmentation models have been made publicly available[27,31,32] (Figs. 2 and 3). However, this is only possible when large heterogeneous pre-curated datasets are available, which are challenging to produce or find. In particular, large object detection or segmentation datasets are very time-consuming to create as they require experts to annotate hundreds to thousands of images manually, with three-dimensional datasets being particularly difficult to generate. The curation of such datasets would be greatly facilitated by creating a centralized repository where training datasets generated to analyze microscopy data using DL would be available (for example, as done by the Papers with Code initiative; https://paperswithcode.com/datasets). This would also help to produce and disseminate benchmark datasets[14,33] that would then be accessible for both algorithm developers and life scientists.

As DL models are becoming accessible through public repositories (so-called model zoos, such as bioimage.io) or web interfaces[27,32], it becomes straightforward to use the models directly to analyze new data. This has the advantages of speeding up DL uptake but, unless the researcher can confirm that their own data were well-represented within the training dataset used initially (which can be very difficult to do), the performance of such portable models on the new data often remains unclear. Therefore, despite its incredible potential, the application of DL in microscopy analysis has raised concerns[30,34,35] due to a lack of transparency and understanding of its limitations, especially for generalizability. In addition to this, DL is developing at an incredible rate, which places a significant burden on users to determine the most appropriate tools for their needs. It remains challenging

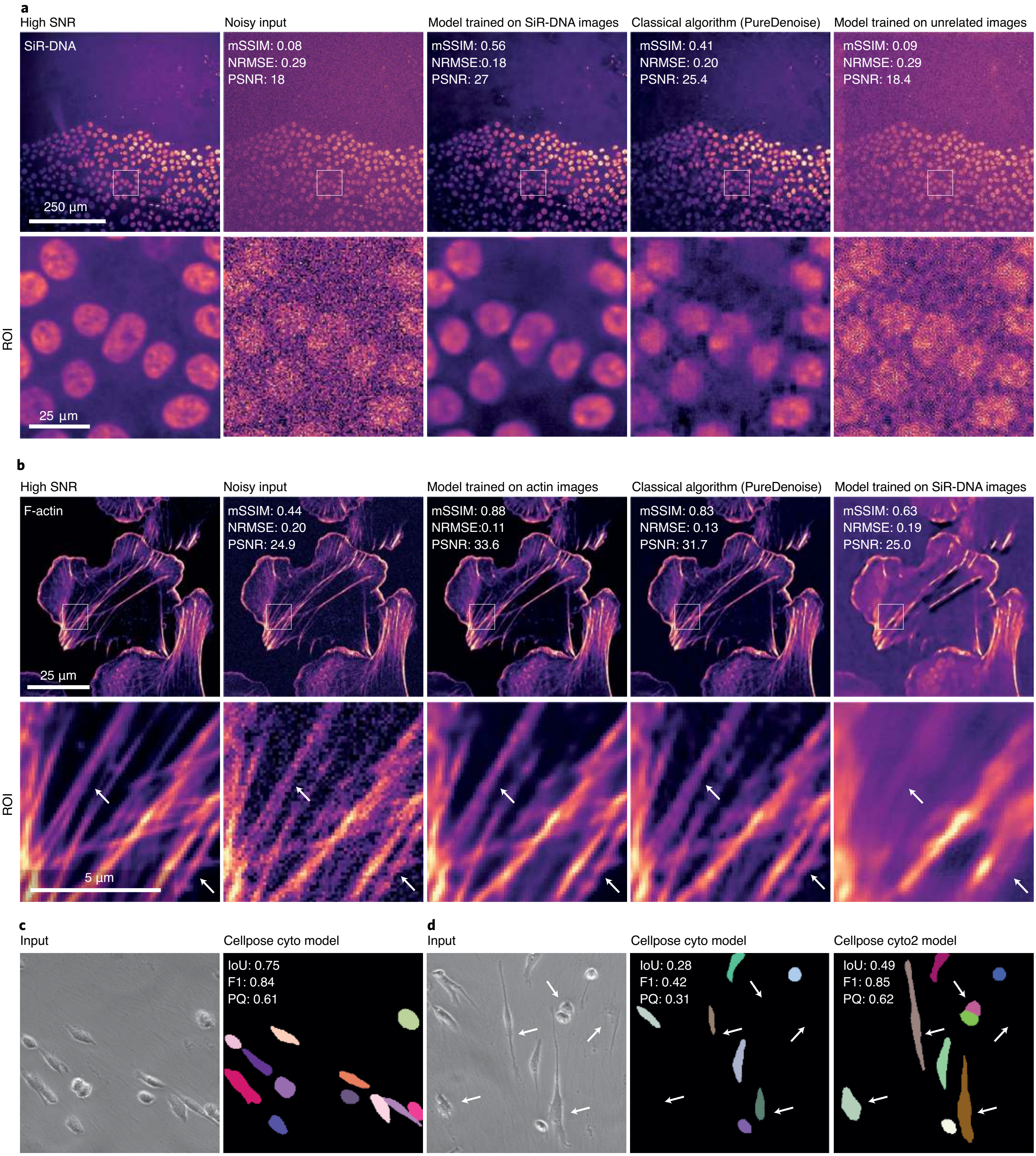


Fig. 2 | Artifacts, metrics and performance of DL in bioimaging. DL methods can offer excellent performances but only when the model used matches the data to be analysed. **a**,**b**, Noisy images of cells stained to visualize their nuclei (**a**) or F-actin (**b**) were acquired using a spinning disk confocal microscope and denoised using two different CARE models[10,20] or PureDenoise. One CARE model was trained to denoise these images, while the other was trained to denoise structured illumination microscopy images of F-actin. Note that in both cases, the appropriate CARE model outperforms PureDenoise[62], while the inappropriate CARE model fails to denoise these images correctly. In **b**, the inappropriate CARE model was trained to denoise the nuclei images shown in **a**. **c**,**d**, Examples to highlight how segmentation models can offer variable performance even on similar images. Images of cells migrating on cell-derived matrices were acquired using a brightfield microscope and segmented with cellpose pre-trained models[32]. Note how the cellpose cyto model performs well in **c** but poorly in **d**. Also, note how the cellpose cyto2 model, a model trained with additional data, performs better than the cyto model in **d**.

to assess the validity and performance of a range of approaches that are often difficult to compare, especially when widely accepted benchmark datasets are unavailable.

We propose that many of these concerns can be significantly alleviated by the careful assessment of DL models performance and consideration in the choice of tool, and by following reporting guidelines to ensure transparency.

Assessing DL model predictions

Currently, the most unambiguous way to assess the quality of DL model predictions is to compare them to ground-truth images or labels (Figs. 2 and 3). Here we primarily focus on image restoration and segmentation tasks, but similar concepts also apply to other image-to-image DL-based image analysis. Segmentation results can be compared to manually annotated masks. In this case, expert manual annotations remain the gold standard to evaluate segmentation. Denoising results can be compared to matching high-SNR images acquired with high laser power or long exposure times[10,14], or by computationally introducing noise to high-SNR data[15]. The comparison between the model prediction and the ground-truth dataset is scored using various metrics (see Box 1). These analyses are typically performed after a model has been trained. In publications, DL models are often evaluated using data similar to those used during training to demonstrate the models' capabilities, but this does not represent a general performance level. Therefore, we argue that when using a model produced by others, it is the end user's responsibility to assess the specific performance of that model on their images (Figs. 2 and 3). This involves generating ground-truth images or investing time to manually annotate a few images to ensure that sufficient material is available for this essential quality control step. For instance, when planning to use a denoising DL model, users can acquire a few corresponding high-SNR images to ensure that the chosen denoising strategy works appropriately. Additionally, using such a dataset, users can also compare the performance of various tools to find the most suitable for the job (Figs. 2 and 3).

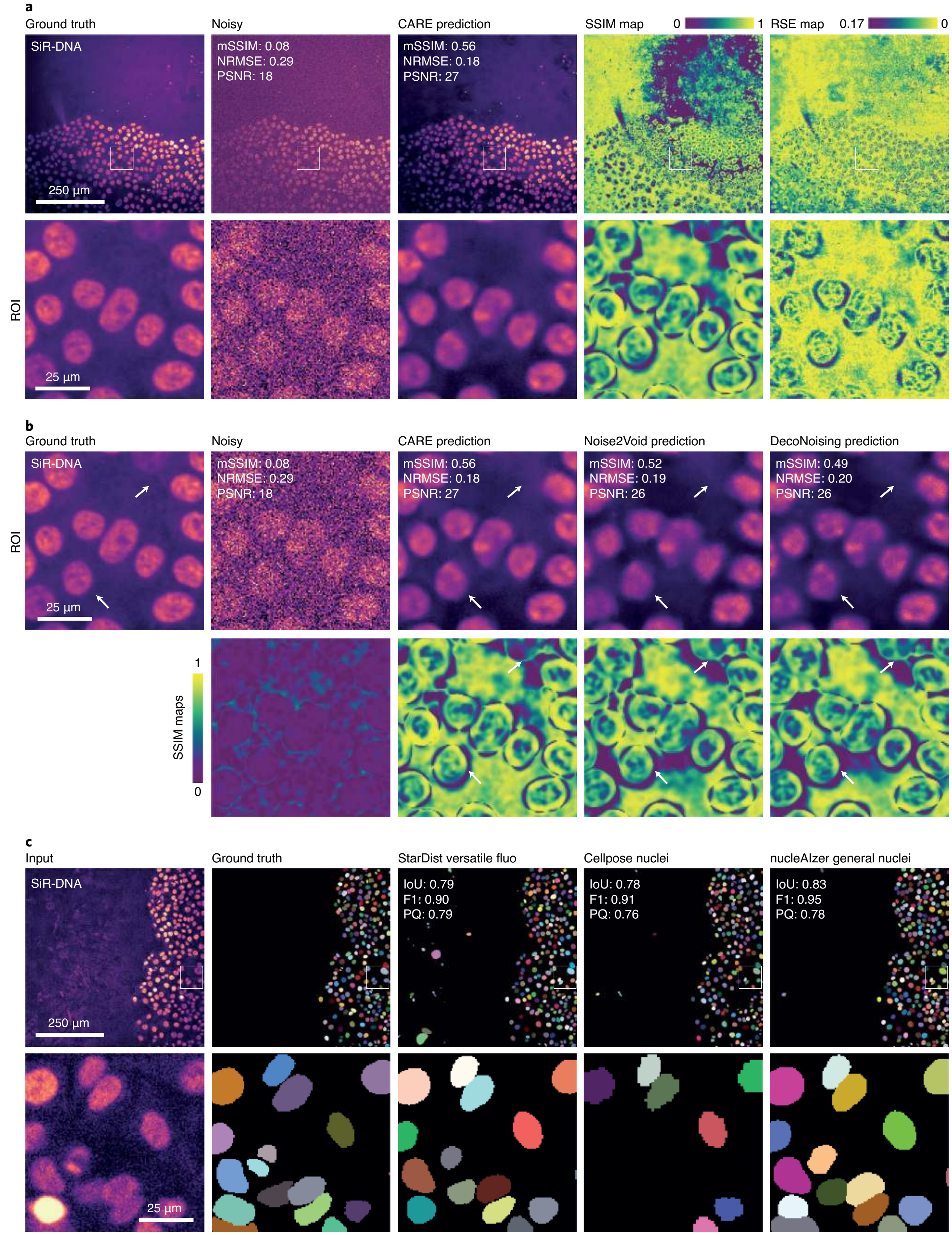
It is important to note that single high-SNR images or manually annotated labels are not strictly speaking ground truth. Indeed, regardless of the acquisition parameters used, single high-SNR images will always be affected by noise. Labels manually annotated by a single expert will also contain errors and bias. In both cases, repetition and averaging can improve the quality of the training data. For instance, averaging multiple high-SNR images of the same field of view will reduce the noise of the target image and produce a better proxy for the ground-truth images. Similarly, to avoid bias, it may be beneficial to combine the annotations of multiple experts or even to crowdsource the annotation process[36].

When comparing DL predictions to ground truth, it is important to visually assess the network output for artifacts; however, it is equally important to quantitatively estimate similarity with the expected results. Box 1 presents a list of commonly used metrics and their appropriate uses depending on the tasks performed by the DL model. In addition, we provide a Jupyter notebook, as part of the ZeroCostDL4Mic platform[20], to easily compute some of these metrics directly in the cloud. One of the most straightforward image metrics used to assess denoising, restoration and image-to-image translation predictions is the root square error (RSE), which calculates the sum of the square differences between predictions and the expected ground truth on a pixel-by-pixel basis. RSE is an easy-to-understand metric but does not report on structures, only on intensities. So other image similarity metrics, such as the structural similarity index measure (SSIM[37]), are also commonly used (Box 1 and Figs. 2 and 3). Additionally, these metrics can be presented as maps that spatially render the discrepancies between the DL predictions and ground-truth images. Such maps are especially useful to check for reconstruction artifacts that may be linked to specific structures in the images (Figs. 2 and 3). Other metrics, such as intersection over union (IoU), which measures the overlap between two binary masks, can assess the quality of segmentation outputs. Instance segmentation results can be further evaluated using additional scores such as F1 score or panoptic quality[38], reflecting the ability of the algorithm to identify each object in the image correctly. Other metrics have also been developed to assess other image processing tasks, such as image registration[39] or super-resolution reconstructions[40], but are not described here in detail.

When using metrics to assess DL predictions, an issue that often arises is to decide when the metric scores are good enough. This is often less of a problem for segmentation tasks where predictions and ground-truth images can reach a good agreement (IoU and F1 scores of 0.8 and above). However, assessing the quality of denoising and image-to-image translation predictions may be more challenging. We found that the approach of comparing both the prediction and raw images to the ground-truth images to be particularly useful to evaluate denoising. This allows users to check that the predictions are more similar to the ground-truth images than the raw input data. If this is not the case, the DL model used is not improving the dataset toward the target image and should be reconsidered. This, however, should not replace a careful visual inspection of the data, as an increase in metric is not always a sign of higher image quality (Fig. 2b).

We recommend that efforts should be put into generating ground-truth data as much as possible, and it is almost always possible to do so. But in rare cases, when ground-truth images are not available, a careful visual inspection of the results may be the only option to assess the performance of a DL model. While less desirable, this solution may be sufficient if the results are already well-characterized and well-understood by the researcher, such as when denoising known cellular structures. However, when studying novel phenomena, this approach should be avoided and observations cross-validated, especially if the structures observed after denoising

Fig. 3 | Using quality metrics to assess the performance of DL models. a,**b**, Noisy images of breast cancer cells labeled with SiR-DNA were denoised using CARE (**a**,**b**)[10], Noise2Void (**b**)[11] and DecoNoising (**b**)[63], all implemented in ZeroCostDL4Mic[20]. In **a** and **b**, the SSIM maps (yellow: high agreement; dark blue: low agreement; 1 indicates perfect agreement) and RSE (yellow: high agreement; dark blue: low agreement; 0 indicates perfect agreement) highlight the differences between the CARE prediction and the corresponding ground-truth image. Note that the agreement between these two images is not homogenous across the field of view and that these maps are helpful to identify spatial artifacts. Panel **b** shows a magnified region of interest from **a**, showcasing how image similarity metrics can compare different DL models trained using different algorithms but using the same training dataset. Note that in this example, all three algorithms improved the original image but to a different extent. Importantly, these results do not represent the algorithms' overall performance to train these models but only assess their suitability to denoise this specific dataset. White arrows highlight areas of poor agreement between the predictions and the ground-truth image. **c**, An example to highlight how segmentation metrics can be used to evaluate the performance of segmentation pre-trained models[27,31,32]. Of note, these results do not reflect the overall quality of these pre-trained models but only assess their suitability to segment this dataset. ROI, region of interest.


a
Ground truth
Noisy
CARE prediction
SSIM map
0
1
RSE map
0.17
0
SiR-DNA
mSSIM: 0.08
NRMSE: 0.29
PSNR: 18
mSSIM: 0.56
NRMSE: 0.18
PSNR: 27
250 µm
ROI
25 µm
b
Ground truth
Noisy
CARE prediction
Noise2Void prediction
DecoNoising prediction
SiR-DNA
mSSIM: 0.08
NRMSE: 0.29
PSNR: 18
mSSIM: 0.56
NRMSE: 0.18
PSNR: 27
mSSIM: 0.52
NRMSE: 0.19
PSNR: 26
mSSIM: 0.49
NRMSE: 0.20
PSNR: 26
ROI
25 µm
1
SSIM maps
0
c
Input
Ground truth
StarDist versatile fluo
Cellpose nuclei
nucleAIzer general nuclei
SiR-DNA
IoU: 0.79
F1: 0.90
PQ: 0.79
IoU: 0.78
F1: 0.91
PQ: 0.76
IoU: 0.83
F1: 0.95
PQ: 0.78
250 µm
25 µm

Box 1 | Common quality metrics used to assess denoising and segmentation DL models

All the metrics described here enable the comparison of the prediction generated by DL models to ground-truth images or labels. Their respective use depends on the type of task performed by the DL model. In some cases, several metrics might be available and can be used together.

Image similarity metrics (denoising, restoration and image-to-image translation)

Several metrics can be used to assess how similar two images are:

(1) The RSE map displays the root of the squared difference between two images. In this case, a smaller RSE is better. A perfect agreement between target and prediction will lead to an RSE map showing zeroes everywhere.

$$\mathrm{RSE}\,(i,j) = \sqrt{\left(P\,(i,j) - \mathrm{GT}\,(i,j)\right)^2}$$

where $P(i,j)$ is the prediction value at pixel (i,j) and $\mathrm{GT}(i,j)$ is the ground-truth value at the same pixel. These images are typically normalized before evaluation of the metric.

(2) The normalized root mean squared error (NRMSE) gives the average difference between all pixels in the images compared to each other. Good agreement between target and prediction yields low NRMSE values.

$$\mathrm{NRMSE} = \sqrt{\frac{1}{N}\sum\nolimits_{i,j}\left(P\,(i,j) - \mathrm{GT}\,(i,j)\right)^2}$$

where N is the total number of pixels. These images are typically normalized before evaluation of the metric.

(3) The Pearson correlation coefficient (PCC) represents the degree of linear correlation between two images. A high correlation between target and prediction translates into a PCC close to 1.

(4) The SSIM evaluates whether two images contain the same structures based on contrast, luminance and structural content concepts. It is a normalized metric, and a SSIM of 1 indicates a perfect similarity between the two images. The SSIM maps are generated by calculating the SSIM metric in each pixel but also considering the surrounding pixels. The mSSIM is the SSIM value calculated across the whole image[37].

(5) The peak signal-to-noise ratio (PSNR) is a metric that estimates discrepancies between two images with respect to the peak signal amplitude of the prediction image. It is usually calculated in decibels and the higher the score, the better the agreement.

Segmentation metric

(1) Image segmentation aims at defining areas of interest in an image based on their identity (foreground versus background being the most common one). A segmentation step typically provides a binary mask image where the pixels in the segmented area have a value of 1 (foreground) while the rest of the pixels have a value of 0 (background).

(2) The IoU metric is a method that can be used to quantify the overlap between two binary masks. Therefore, when using IoU to assess the performance of a segmentation algorithm compared to ground-truth masks, the closer to 1, the better the performance.

$$\mathrm{IoU} = \frac{P \cap \mathrm{GT}}{P \cup \mathrm{GT}}$$

where ∪ represent the union of two binary images (number of pixels that are foreground in either image), ∩ represents the intersection of two binary images (number of pixels that are foreground in both images simultaneously), and P is the predicted image.

Instance segmentation metrics (also used for classification and object detection tasks)

Instance segmentation aims to identify objects of interest in an image, both from the background and each other. An instance segmentation step commonly provides a label image where each identified object has a unique pixel intensity representing its identity, and the background is commonly set to have a pixel intensity of 0. Several metrics can be used to assess the quality of instance segmentation results, some of which are outlined below.

(1) Typically, an IoU value is first calculated between the DL prediction and a ground-truth image on a per-object basis. This allows identifying true and false positives, as well as false negatives. True positives are objects that are correctly identified. In contrast, false positives are segmented objects that are not present in the ground-truth image, and false negatives are objects missed by the segmentation algorithm. A particular object is considered as detected when its segmentation mask has an IoU with a ground-truth object mask that is above a user-defined threshold (for instance, $\mathrm{IoU} > 0.5$). The number of false-positive ($\mathrm{Nb}_{\text{False positive}}$) and false-negative ($\mathrm{Nb}_{\text{False negative}}$) results are then calculated as follows:

$$\mathrm{Nb}_{\text{False positive}} = \mathrm{Nb}_{\text{Prediction}} - \mathrm{Nb}_{\text{True positive}}$$

where $\mathrm{NB}_{\text{True positive}}$ represents the number of true positives and $\mathrm{Nb}_{\text{Prediction}}$ and $\mathrm{Nb}_{\text{GT image}}$ refer to the number of objects present in the predicted image and the ground-truth image, respectively.

(2) Precision is defined as the number of correctly segmented objects divided by the total number of detected objects. Precision is a metric used to assess the cost associated with false positives. The closer the precision is to 1, the better the performance.

$$\mathrm{Precision} = \frac{\mathrm{Nb}_{\text{True positive}}}{\mathrm{Nb}_{\text{True positive}}+\mathrm{Nb}_{\text{False positive}}} = \frac{\mathrm{Nb}_{\text{True positive}}}{\mathrm{Nb}_{\text{Prediction}}}$$

(3) Recall calculates how many of the actual positives the model captures by labeling them as true positive. Recall can be used as a metric to assess the cost associated with false negatives. The closer the recall is to 1, the better the performance.

$$\mathrm{Recall} = \frac{\mathrm{Nb}_{\text{True positive}}}{\mathrm{Nb}_{\text{True positive}} + \mathrm{Nb}_{\text{False negative}}}$$

(4) The F1 score combines both the precision and recall scores in a single metric and is calculated as follows:

$$\mathrm{F1} = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Other metrics such as ‘accuracy’ or ‘PQ’ (where 1 indicates perfect agreement[38]) can also be used to score the quality of instance segmentation results.

are not easily visible in the raw data. Thus, there is a need for developing metrics or implementing evaluation methods that can assess the quality of predictions when no ground-truth images are available. Therefore, it may be valuable for developers to include in their tools ways to easily assess DL model uncertainty using, for instance, Monte Carlo dropout strategies[41] (see ref. [42] for an in-depth review on this topic). Other possible methods include using particular network architectures such as variational autoencoders to extract a distribution of network output from a single input, allowing to estimate the model's variability and uncertainty quantitatively and spatially[43].

Choosing a DL tool

With the increasing availability of networks, models and software, it becomes challenging to identify the most suitable tool to answer a biological question. We do not recommend any particular software or tool simply because each user's needs are distinct (for an excellent review of DL-based segmentation tools, see ref. [9]). Instead, we present a few pointers to help readers sieve through the literature based on what developers have reported in their work and reports from early adopters.

First, we recommend choosing a well-documented and well-maintained tool that matches the user's preferred interface. Available DL tools now span various web interfaces[27,32], standalone software[22,26,32,44], plugins for popular image analysis software[10,11,25,45], online notebooks[20] and Python packages[46]. Each platform requires a different level of technical skill to use. In addition, the level of detail of the documentation provided by the developers can vary significantly and ranges from annotated code to online video tutorials and detailed step-by-step guides. This will limit accidental misuse of the tool and help users to understand the tools and their capabilities. Additionally, a substantial existing user base and online forums discussing troubleshooting are signs of a healthy and helpful tool. It also provides a wealth of information about user experience, as well as tips and tricks.

We advise being wary about works that do not provide source code and associated data for users to reproduce the results on example data. It is typically free and easy to make these elements publicly available via common platforms (such as GitHub). We support works that themselves encourage open science. We also believe that example data are instrumental as they allow users to test and learn how to use a tool properly before applying it to their data.

As discussed above, it is essential to carefully assess the performance of DL-based tools on the dataset of interest. Therefore, we also recommend using tools that offer purposely built evaluation and sanity-check strategies, such as those found in the StarDist Python package[31], the Noise2Void Fiji plugin[11] and the ZeroCostDL4Mic notebooks[20]. We also strongly encourage users to consider how the chosen tool can be used within their preferred image analysis pipeline. DL-based analyses will often constitute only a small part of the overall analysis process, and therefore, the pipeline as a whole should be considered before selecting a tool.

When training DL networks using a new algorithm or software, one feature to look for is the availability of strategies to identify and prevent overfitting. Overfitting occurs when a model becomes too specialized to the training dataset and does not generalize well to new data. In practice, this means that the trained model may not perform well on new data even if they are similar to those used during training. Overfitting can be detected by monitoring how the performance of the model evolves over training time on the training dataset and a set-aside validation dataset. When more training leads to an improvement in performance on the training dataset but an otherwise worsening of the performance on the validation dataset, this is a sign that overfitting is occurring, which can be typically visualized by plotting so-called loss curves over training time. Overfitting may be prevented by increasing the training dataset's diversity using, for instance, data augmentation[47,48] or strategies such as reducing the model complexity, adding regularization (L1, L2) or early stopping during training[49]. DL tools dedicated to training would enormously benefit from these features as they simplify the assessment and potential improvement on model optimization for the user.

Another feature to look for when choosing a tool to train DL models is the possibility to perform transfer learning. Transfer learning enables the use of existing models as a starting point when training a new model. This allows users to take advantage of previously learned features present in these trained models, instead of starting the training process from scratch. In other words, transfer learning enables users to fine-tune an existing model using their data. This approach can considerably accelerate training and reduce the size of the necessary training dataset while producing models with higher performance for a specific task[20,50].

Finally, when testing a new tool, it is often informative (and even often appreciated) to get in touch with developers and contribute to tool improvement when bugs are discovered or to report issues in some specific configuration that may not have been encountered at the development stage. We feel that the importance of this conversation is sometimes understated, even though it promotes good tools, open-mindedness and multidisciplinary research while building trust in the methods.

Reporting the use of DL in publications

As previously done for other transformative technologies, we believe that the bioimaging community needs to discuss and flesh out guidelines for reporting DL use for bioimaging in publications[51–54]. This is especially important as the reporting of more traditional image analyses and acquisitions pipelines is still raising concerns[51,55–57]. It is beyond the intention of the present work to propose guidance to developers on evaluation and reporting when proposing new DL algorithms, and we refer the readers to recent work that has initiated this conversation within the computer science community[58]. Instead, we focus on what would be useful to report when using DL tools.

Due to the wealth of hyperparameters, architecture choices and data manipulation available with DL, incorrectly trained or incorrectly evaluated DL models can be easily generated and lead to suboptimal results. This highlights the importance of reporting the steps leading to the generation of a particular model clearly and appropriately. Indeed, standard guidelines will increase confidence in the use of DL and promote transparency and reproducibility. Such guidelines will also help reviewers assess manuscripts using DL for image analysis, especially if this technology is unfamiliar to them. Below, we listed several suggestions to contribute to this critical discussion.

- Naturally, the algorithm used should be reported, and the appropriate paper(s) cited. We also recommend indicating the version of the algorithm used or, failing that, the date at which the tool was obtained, as most analytical tools change over time and each update may lead to varying performance on the same data. For DL, this is currently not a widespread habit, especially because both the network and dataset may change over time (acquiring more data to expand the training dataset, for instance).
- Similarly, when using models trained by others, it is advisable to indicate the version of the model used (Fig. 2c).

If not available, we recommend providing the date when the model was obtained and used.

- A DL model performance is entirely dependent on the dataset used at the training stage (Fig. 2). When training dedicated DL models, the training dataset should be clearly described in the material and methods (including, but not limited to, specimen, microscope, magnification, numerical aperture of the objective, light detector used, exposure and illumination conditions, as recommended in other work[55]).
- Training datasets should be deposited in a suitable and semi-permanent data repository (such as Zenodo, BioImageArchive or the Image Data Resource[59]). This would help create and disseminate diverse benchmark datasets and make them accessible to both tool developers and users.
- When training a DL model, we recommend indicating the key hyperparameters used and the main underlying libraries (for example, TensorFlow and PyTorch). We recommend that DL models with reusability potential are deposited in a suitable repository (such as Zenodo) and linked to a model Zoo (such as TensorFlow hub or bioimage.io), along with their associated metadata.
- If custom code was generated to run the algorithm or process the data (pre- or post-processing steps, for instance), it should also be shared with the paper and archived (for example, using GitHub or Zenodo).
- The steps taken to validate the DL model used should be clearly described. This includes the type of validation (that is, indicating the evaluation metric used and what score was achieved), the number and the origin of the images used for evaluation (it is often considered imperative for evaluation data to be completely absent from training data to have bearings on how well the model generalizes to new data), and explaining why the result was deemed acceptable. If space allows, we also recommend providing evaluation examples as supplementary figures.
- When performing predictions using a DL model, the tool used to run the model should be indicated (with the version again), and appropriate paper(s) cited. Indeed, several tools offer the possibility to run DL models and may involve different pre- or post-processing steps that can influence the results obtained.

Concluding remarks

DL tools are transforming the way we analyze microscopy images. As with all new methods, proper use, validation and reproduction are needed before the methods can be trusted. While we believe these methods offer many advantages, we think that DL cannot be used on any dataset without prior validation. This is especially important as users risk falling into the artificial intelligence hype when other techniques may be more appropriate, more robust and sometimes quicker to analyze their images. Importantly, due to the complexity of operations performed in DL, not knowing precisely how the images are manipulated may affect how they can be reliably analyzed downstream of DL. As an example, it is hard to estimate whether it is appropriate to quantify absolute image intensities following DL-based denoising due to potential non-linearity with respect to the input data. Similarly, although image-to-image translation and resolution improvement using DL are very promising approaches, they remain prone to undetected artifact generation due to the inherent addition of data to the input data[60] from the training dataset, raising concerns of validity. When using such approaches, it is particularly powerful to use tools that can spatially map the area in the resulting image that is likely to contain artifacts (for example, to produce a map of prediction confidence, as is done with DivNoising[43]).

Here, we presented arguments toward the importance of validating any model using a purposefully built evaluation dataset containing ground-truth target images or labels. Similarly, the use of DL models should be reported appropriately to ensure reproducibility and transparency. This is a challenging task for DL as many components, both internal (hyperparameters) and external (training dataset) to the network used, can dramatically influence the results obtained. With the increasing availability of networks and models, we also stress the importance of finding ways to identify what might be a 'good tool'. We believe that a good tool is not only a performant one, but that transparency of what it does to the data, usability and reliability are equally important. The responsibility of proper use of DL in microscopy is now equally shared between users and developers. Spiderman's Uncle Ben has never been more right than today: "With great powers comes great responsibility". Finally, this article is not intended to set strict standards in place, but rather to serve as a starting point for further discussions between users, developers, image analysis specialists and journal editors to define appropriate use of these otherwise powerful techniques. ❐

Romain F. Laine[1,2,10],
Ignacio Arganda-Carreras[3,4,5],
Ricardo Henriques[1,2,6] and
Guillaume Jacquemet[7,8,9] ✉

[1]MRC-Laboratory for Molecular Cell Biology, University College London, London, UK. [2]The Francis Crick Institute, London, UK. [3]Computer Science and Artificial Intelligence Department, University of the Basque Country (UPV/EHU), San Sebastian, Spain. [4]Ikerbasque, Basque Foundation for Science, Bilbao, Spain. [5]Donostia International Physics Center (DIPC), San Sebastian, Spain. [6]Instituto Gulbenkian de Ciência, Oeiras, Portugal. [7]Turku Bioscience Centre, University of Turku and Åbo Akademi University, Turku, Finland. [8]Faculty of Science and Engineering, Biosciences, Åbo Akademi University, Turku, Finland. [9]Turku Bioimaging, University of Turku and Åbo Akademi University, Turku, Finland. [10]Present address: Micrographia Bio, Translation and Innovation Hub, London, UK.

✉*e-mail: guillaume.jacquemet@abo.fi*

Published online: 4 October 2021
https://doi.org/10.1038/s41592-021-01284-3

References

1. Moen, E. et al. *Nat. Methods* **16**, 1233–1246 (2019).
2. von Chamier, L., Laine, R. F. & Henriques, R. *Biochem. Soc. Trans.* **47**, 1029–1040 (2019).
3. Krizhevsky, A., Sutskever, I. & Hinton, G.E. *Adv. Neural Inf. Process. Syst.* https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (2012).
4. Ouyang, W. et al. *Nat. Methods* **16**, 1254–1261 (2019).
5. Redmon, J. & Farhadi, A. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 7263–7271 (IEEE, 2017).
6. He, K., Gkioxari, G., Dollár, P., & Girshick, R.B. 2017 IEEE International Conference on Computer Vision (ICCV) 2980–2988 (2017).
7. Ronneberger, O., Fischer, P. & Brox, T. in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (eds Navab, N., Hornegger, J., Wells, W. & Frangi, A.) (Springer, 2017).
8. Falk, T. et al. *Nat. Methods* **16**, 67–70 (2019).
9. Lucas, A. M. et al. *Mol. Biol. Cell* **32**, 823–829 (2021).
10. Weigert, M. et al. *Nat. Methods* **15**, 1090–1097 (2018).
11. Krull, A., Buchholz, T.-O. & Jug, F. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 2124–2132 (IEEE, 2019).
12. Wang, H. et al. *Nat. Methods* **16**, 103–110 (2019).
13. Speiser, A. et al. *Nat. Methods* **18**, 1082–1090 (2021).
14. Qiao, C. et al. *Nat. Methods* **18**, 194–202 (2021).
15. Fang, L. et al. *Nat. Methods* **18**, 406–416 (2021).
16. Wen, C. et al. *eLife* **10**, e59187 (2021).
17. Newby, J. M., Schaefer, A. M., Lee, P. T., Forest, M. G. & Lai, S. K. *Proc. Natl Acad. Sci. USA* **115**, 9026–9031 (2018).
18. Nan, A., Tennant, M., Rubin, U. & Ray, N. *Proc. Machine Learn. Res.* **121**, 527–543 (2020).
19. Ounkomol, C., Seshamani, S., Maleckar, M. M., Collman, F. & Johnson, G. R. *Nat. Methods* **15**, 917–920 (2018).
20. von Chamier, L. et al. *Nat. Commun.* **12**, 2276 (2021).
21. Ouyang, W., Mueller, F., Hjelmare, M., Lundberg, E. & Zimmer, C. *Nat. Methods* **16**, 1199–1200 (2019).
22. McQuin, C. et al. *PLoS Biol.* **16**, e2005970 (2018).
23. Haberl, M. G. et al. *Nat. Methods* **15**, 677–680 (2018).
24. Bannon, D. et al. *Nat. Methods* **18**, 43–45 (2021).
25. Gómez-de-Mariscal, E. et al. *Nat. Methods* (in the press).
26. Belevich, I. & Jokitalo, E. *PLoS Comput. Biol.* **17**, e1008374 (2021).
27. Hollandi, R., Szkalisity, A. & Toth, T. *Cell Syst.* **10**, 453–458 (2020).
28. Waibel, D. J. E., Boushehri, S. S. & Marr, C. *BMC Bioinformatics* **22**, 103 (2021).

29. Möckl, L., Roy, A. R. & Moerner, W. E. *Biomed. Opt. Express* **11**, 1633–1661 (2020).
30. Belthangady, C. & Royer, L. A. *Nat. Methods* **16**, 1215–1225 (2019).
31. Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018 – 21st International Conference, Granada, Spain, September 16–20, 2018: Proceedings Pt II* (eds Frangi, A. F. et al.) 265–273 (Springer, 2018).
32. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. *Nat. Methods* **18**, 100–106 (2021).
33. Zhang, Y. et al. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 11702–11710 (2019).
34. Antun, V., Renna, F., Poon, C., Adcock, B. & Hansen, A. C. *Proc. Natl Acad. Sci. USA* **117**, 30088–30095 (2020).
35. Hoffman, D. P., Slavitt, I. & Fitzpatrick, C. A. *Nat. Methods* **18**, 131–132 (2021).
36. Spiers, H. et al. *Traffic* **22**, 240–253 (2021).
37. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. *IEEE Trans. Image Process.* **13**, 600–612 (2004).
38. Kirillov, A., He, K., Girshick, R., Rother, C. & Dollár, P. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 9404–9413 (2019).
39. Hermosillo, G. *Int. J. Comput. Vis.* **50**, 329–343 (2002).
40. Culley, S. et al. *Nat. Methods* **15**, 263–266 (2018).
41. Gal, Y. & Ghahramani, Z. *Proc. 33rd Intl Conf. Machine Learning, PMLR* **48**, 1050–1059 (2016).
42. Abdar, M. et al. *Inf. Fusion* **76**, 243–297 (2021).
43. Prakash, M., Krull, A. & Jug, F. Preprint at https://arxiv.org/abs/2006.06072 (2021).
44. Berg, S. et al. *Nat. Methods* **16**, 1226–1232 (2019).
45. Buchholz, T.-O., Prakash, M., Krull, A. & Jug, F. *Computer Vision – ECCV 2020 Workshops* (2020).
46. Gibson, E. et al. *Comput. Methods Programs Biomed.* **158**, 113–122 (2018).
47. Shorten, C. & Khoshgoftaar, T. M. *J. Big Data* **6**, 60 (2019).
48. Perez, L. & Wang, J. Preprint at https://arxiv.org/abs/1712.04621 (2017).
49. Moradi, R., Berangi, R. & Minaei, B. *Artif. Intell. Rev.* **53**, 3947–3986 (2020).
50. Wang, Y. et al. Preprint at *bioRxiv* https://doi.org/10.1101/2021.02.01.429188
51. Aaron, J. & Chew, T.-L. *J. Cell Sci.* **134**, jcs254151 (2021).
52. Bustin, S. A. et al. *Clin. Chem.* **55**, 611–622 (2009).
53. Füllgrabe, A. et al. *Nat. Biotechnol.* **38**, 1384–1386 (2020).
54. Klionsky, D. J. et al. *Autophagy* **12**, 1–222 (2016).
55. Heddleston, J. M., Aaron, J. S., Khuon, S. & Chew, T.-L. *J. Cell Sci.* **134**, jcs254144 (2021).
56. Jost, A. P.-T. & Waters, J. C. *J. Cell Biol.* **218**, 1452–1466 (2019).
57. Huisman, M. et al. Preprint at https://arxiv.org/abs/1910.11370 (2021).
58. Dodge, J., Gururangan, S., Card, D., Schwartz, R. & Smith, N. A. Preprint at https://arxiv.org/abs/1909.03004 (2019).
59. Williams, E. et al. *Nat. Methods* **14**, 775–781 (2017).
60. Manton, J. D. Preprint at https://arxiv.org/abs/2104.06558 (2021).
61. Schindelin, J. et al. *Nat. Methods* **9**, 676–682 (2012).
62. Luisier, F., Vonesch, C., Blu, T. & Unser, M. *Signal Process.* **90**, 415–427 (2010).
63. Goncharova, A. S., Honigmann, A., Jug, F. & Krull, A. Preprint at https://arxiv.org/abs/2008.08414 (2020).

Acknowledgements

R.F.L. would like to acknowledge the support of the Medical Research Council (MRC) Skills Development Fellowship (MR/T027924/1) and MRC grant funding (MR/V039229/1). This study was supported by grants awarded by the Academy of Finland (G.J.), the Sigrid Juselius Foundation (G.J.), the Cancer Society of Finland (J.I.) and Åbo Akademi University Research Foundation (G.J., CoE CellMech), and by Drug Discovery and Diagnostics strategic funding to Åbo Akademi University (G.J.). The Cell Imaging and Cytometry Core Facility (Turku Bioscience, University of Turku, Åbo Akademi University and Biocenter Finland) is acknowledged for services, instrumentation and expertise. R.H. is supported by Gulbenkian Foundation and received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 101001332), the European Molecular Biology Organization Installation Grant (EMBO-2020-IG-4734) and the Wellcome Trust (203276/Z/16/Z) (R.H.). I.A.-C. would like to acknowledge the support of the 2020 Leonardo Grant for Researchers and Cultural Creators, BBVA Foundation.

Author contributions

G.J. was responsible for conceptualization and visualization; R.F.L., I.A.-C., R.H. and G.J. all contributed to writing.

Competing interests

The authors declare no competing interests.

Additional information

Peer review information *Nature Methods* thanks Jiji Chen and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.