InstantDL - An easy-to-use deep learning pipeline for image segmentation and classification

Dominik Waibel[1,2], Sayedali Shetab Boushehri[1,3], Carsten Marr[1]

[1]Institute of Computational Biology, Helmholtz Zentrum München - German Research Center for Environmental Health, Neuherberg, Germany
[2]Technical University of Munich, School of Life Sciences, Weihenstephan, Germany
[3]Roche Innovation Center Munich, Roche Diagnostics GmbH, Penzberg, Germany

Abstract

Motivation

Deep learning contributes to uncovering and understanding molecular and cellular processes with highly performant image computing algorithms. Convolutional neural networks have become the state-of-the-art tool to provide accurate, consistent and fast data processing. However, published algorithms mostly solve only one specific problem and they often require expert skills and a considerable computer science and machine learning background for application.

Results

We have thus developed a deep learning pipeline called InstantDL for four common image processing tasks: semantic segmentation, instance segmentation, pixel-wise regression and classification. InstantDL enables experts and non-experts to apply state-of-the-art deep learning algorithms to biomedical image data with minimal effort. To make the pipeline robust, we have automated and standardized workflows and extensively tested it in different scenarios. Moreover, it allows to assess the uncertainty of predictions. We have benchmarked InstantDL on seven publicly available datasets achieving competitive performance without any parameter tuning. For customization of the pipeline to specific tasks, all code is easily accessible.

Availability and Implementation

InstantDL is available under the terms of MIT licence. It can be found on GitHub: https://github.com/marrlab/InstantDL

Contact

carsten.marr@helmholtz-muenchen.de

1 Introduction

Deep learning has revolutionised image processing (LeCun, Bengio, and Hinton 2015). On specific tasks such as cell segmentation (Caicedo, Roth, et al. 2019; Hollandi et al. 2020), cell classification (Cireşan et al. 2013; Buggenthin et al. 2017; Christian Matek et al. 2019) or in-silico staining (Ounkomol et al. 2018; Christiansen et al. 2018), deep learning algorithms have led to breakthroughs in biomedical image analysis. They now achieve higher accuracy than trained experts (Esteva et al. 2017; McKinney et al. 2020; Christian Matek et al. 2019) and outperform humans at data processing speed and prediction consistency (Tschandl et al. 2019; Liu et al. 2019).

However, machine learning algorithms are mostly developed to solve one specific problem. Moreover, applying them often requires a strong computer science and machine learning background. This makes it difficult for biomedical researchers to identify and use appropriate algorithms. Standardizing and generalizing deep learning methods and thereby reducing the necessary expertise in computer and data science will lead to broader application and acceleration of research.

We thus provide InstantDL, an easy-to-use deep learning pipeline, which automates pre- and post-processing for convenient application. It can be used for semantic segmentation (i.e. the classification of each pixel into a particular class), instance segmentation (i.e. the detection and classification of objects), pixel-wise regression (i.e. in-silico staining) and image classification (i.e. to discriminate cancerous from healthy cells). Only eleven parameters have to be set in order to run the pipeline. Since biomedical datasets are often sparsely annotated, as manual annotation is laborious and costly, we provide pre-trained models for efficient transfer learning.

InstantDL is benchmarked on seven publicly available (see section 6) datasets: multi-organ nuclei segmentation, nuclei detection in divergent images, lung segmentation from CT scans, in-silico prediction of a mitochondrial and nuclear envelope staining, cell classification in digitized blood smears, and cancer classification on histopathology slides. Without any hyperparameter tuning, we achieve competitive results.

InstantDL is aimed at users with a basic understanding of machine learning, knowing suitable loss-functions and understanding how to split data into train and test set. The code however is open source and well documented for those who want to customize the pipeline to their needs. By providing a debugged, tested, and benchmarked pipeline we help reduce errors during code development and adaptation, and contribute to reproducible application of deep learning methods.

2 Methods

InstantDL offers the four most common tasks in medical image processing: Semantic segmentation, instance segmentation, pixel-wise regression, and classification (Litjens et al. 2017; Maier et al. 2019). In the following we describe the algorithms implemented in InstantDL to address these tasks and how they can be applied within the pipeline.

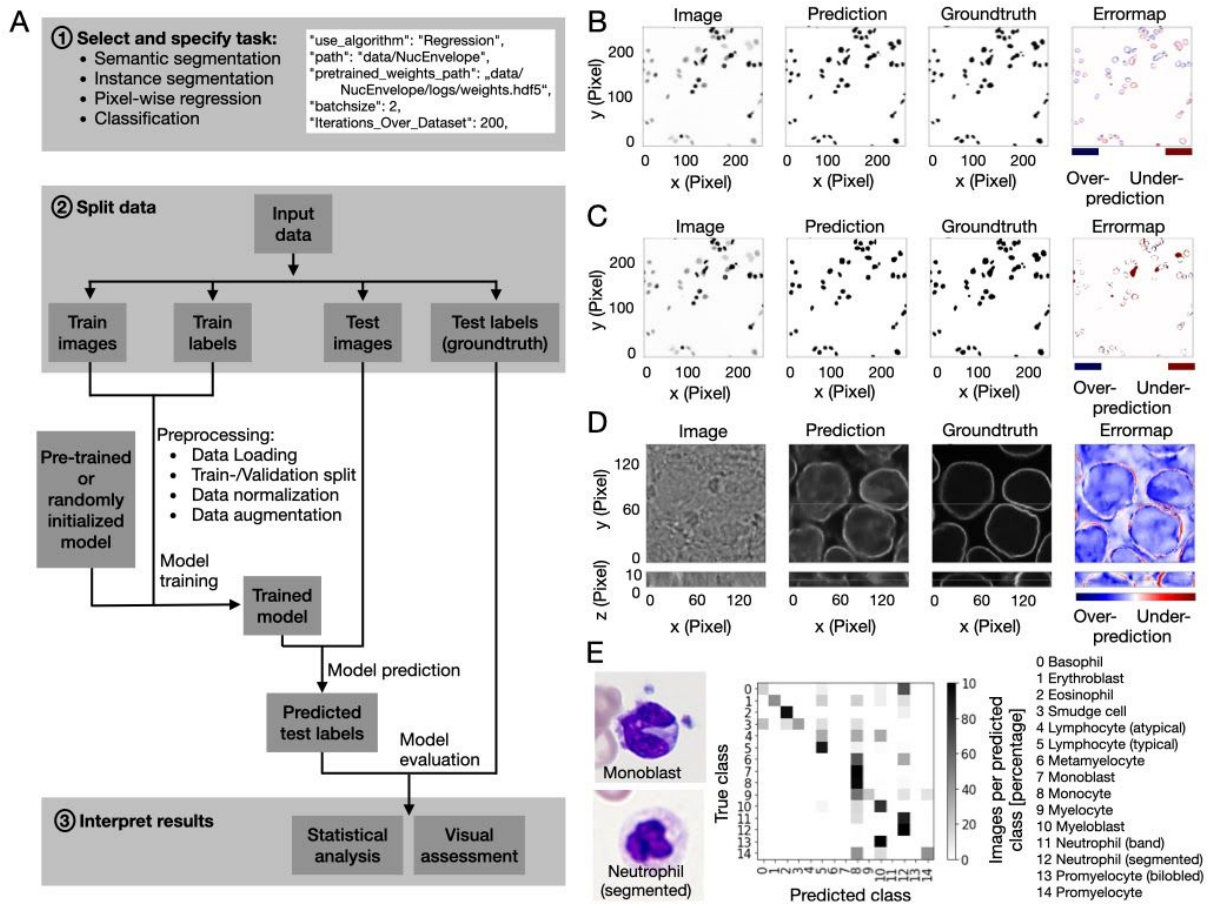

Figure 1. InstantDL provides an easy to use framework for the analysis of biomedical images.

(A) Flow diagram of the pipeline with indicated user action highlighted in gray. (1) One out of four tasks (semantic segmentation, instance segmentation, pixel-wise regression, classification) is selected by the user. Up to ten parameters can be set in the configuration file to adapt the pipeline to the task. A code snippet illustrates task selection and six of the ten parameter settings in the configuration file: selected task ("use_algorithm"), path to folder ("path"), if pretrained weights should be used the path to these ("pretrained_weights_path") should be set, batch size ("batchsize") and epochs chosen ("Iterations_Over_Dataset"). (2) Input data is split into train and test sets. The user specifies these by putting the data in the corresponding folders. After executing the python configuration file the pipeline will automatically load the data from the train folder, create a 20 percent validation split, normalize and augment the data (see Methods for details). Training is initiated with either a pre-trained or randomly initialized model. After training, the model predicts test labels: segmentation masks, pixel values or labels for the images in the test set according to the chosen task. (3) Results can be interpreted by the user via statistical and visual assessment of the predicted outcome by comparing it to the ground truth in the test set.

(B) Example output for a 2D semantic segmentation task: Cell nuclei in a brightfield image (left) are segmented with InstantDL (Prediction) using the U-Net, and compared to the original annotation (Groundtruth). The Errormap indicates over- and under-predicted pixels. The image is part of the 2018 Kaggle nuclei segmentation challenge dataset (Caicedo, Goodman, et al. 2019).

(C) Example output for a 2D instance segmentation task (same image as in (B)): A binary mask is predicted for each object in the image using InstantDLs Mask-RCNN algorithm and compared to the groundtruth.

(D) Example output for a 3D pixel-wise regression task using a U-Net. From stacks of bright-field images (Image) (Ounkomol et al. 2018) the pipeline predicts a nuclear envelope (Prediction) that resembles the true staining (Groundtruth). The first row shows the x-y-plane, the bottom row the x-z plane of the 3D volume.

(E) Example output for a classification task of benign and leukemic blood cells in blood smears from 200 individuals (C. Matek et al. 2019). We show two exemplary microscopy images (left) of two white blood cell classes, a monoblast and a neutrophil. The white blood cell type is predicted with a ResNet50. The confusion matrix (middle) shows that most of the 15 classes can be well predicted, in accordance to Matek et al. (2019).

Semantic segmentation

One of the standard approaches for detecting image patterns is semantic segmentation (Minaee et al. 2020). For each pixel in the input image a class label is predicted by the algorithm. The U-Net (Ronneberger, Fischer, and Brox 2015) is a commonly used architecture for semantic segmentation with numerous applications in biomedicine (Falk et al. 2019; Isensee et al. 2018). It consists of a symmetric contractive path to capture context and an expansive path to capture fine localizations (Ronneberger, Fischer, and Brox 2015). The U-Net outputs continuous values between 0 and 1 for each pixel, which can be interpreted as probabilities to belong to a given class. InstantDL allows for two classes (background vs. foreground) and thresholds the output using Otsu' method (Otsu 1979; Caicedo et al. 2017). In InstantDL we made minor changes to the architecture from (Ronneberger, Fischer, and Brox 2015): (i) we use padded convolutions to receive the same output and input dimensions (ii) and have implemented dropout layers in the encoder to enable uncertainty estimation.

Instance segmentation

Instance segmentation is used to detect objects (instances) within an image (Minaee et al. 2020; Caicedo et al. 2017). We implemented the Mask-RCNN (Kaiming et al. 2017) in InstantDL for this task. It first detects a bounding box for each object in the image and then performs a segmentation in each bounding box. Our Mask-RCNN is based on a ResNet50 from Abdullah's (2017) implementation. The Mask-RCNN requires instance level ground truth: For each image in the training set, a set of labels has to be created, each containing a segmentation mask of one single instance. An algorithm to create instance level ground truth from binary semantic segmentation ground truth is provided as a jupyter-notebook with the pipeline.

Pixel-wise regression

Tasks where no pixel-wise class labels but a continuous pixel value is desired (such as in-silico staining, (Ounkomol et al. 2018)) are called pixel-wise regression. InstantDL uses the same U-Net implementation as for semantic segmentation. The only difference is that the U-Net output is not interpreted as probabilities to belong to one or another class, but is regarded as a regression. We thus use continuous labels for training and the mean-squared-error as regressive loss as proposed previously (Ounkomol et al. 2018).

Image classification

Here, the task is to classify each image into one of a specific number of given classes. For this task a residual network (He et al. 2016) is implemented. These architectures are widely used for biomedical image classification (Brinker et al. 2018; Liu et al. 2019). Residual networks use residual blocks, a reformulation of layers as learning residual functions, which enable the use of many layers, while ensuring convergence (He et al. 2016). We use a slightly modified ResNet50 with 50 layers (*Keras-Contrib* n.d.) in InstantDL, where we have added dropout layers to enable uncertainty estimation.

3 The InstantDL pipeline

Data preparation

Data has to be manually split in a train and a test set (see Fig. 1A) according to the user's hypothesis: For one dataset a random split might be suitable (Waibel et al. 2018), while for others a split on patient or tissue slide level is methodically appropriate (Christian Matek et al. 2019). InstantDL can process stacked images, enabling the prediction from multiple channels. Input and ground truth images must have the same filename including the file ending.

	Semantic segmentation	**Instance segmentation**	**Pixel-wise regression**	**Classification**
Input image	2D & 3D	2D	2D & 3D	2D
Labels	Binary images	Images with float pixel values	Images with float pixel values	Labels in a .csv file
Output	Binary images	Binary masks	Images with float pixel values	Labels in a .csv file
Architecture	U-Net	Mask RCNN	U-Net	ResNet50

Table 1: Overview on the image processing tasks implemented in InstantDL, required input, label, and output format.

Pipeline Settings

After data preparation the user specifies tasks and parameters in the configuration file (see Fig. 1A). A maximum of eleven parameters have to be set. These are: The task (e.g., Semantic segmentation, instance segmentation, regression, or classification), the path to the project directory containing the train and test files, the pretrained weights for model initialization, the batchsize, the number of iterations over the dataset (e.g., epochs), the data augmentations, the loss function, the number of classes, if the images should be resized during import, if the uncertainty should be calculated after training, and if the model should be automatically evaluated after training. Within the pipeline one .json file serves as a config file.

After setting these parameters, the user executes the configuration file which starts the training with InstantDL on the desired task using the training data with the pre-trained weights and the chosen configurations.

Transfer learning

Pre-trained weights can be used to initialize a training process, a practice called transfer learning. The choice of weights can have a huge influence on the performance of the algorithm: Transferring from natural images such as ImageNet to the medial domain seems not to be always beneficial (Ngiam et al. 2018) while using pre-trained weights from a medical dataset can improve performance (Hénaff et al. 2019). Pre-trained weights for 2D

nuclei segmentation, 2D lung segmentation, 3D in-silico staining and for classification of white blood cells as well as ImageNet weights can be loaded.

Data augmentation

Data augmentation is a method commonly used in machine learning to artificially increase the variance in the training dataset and thereby train the network to generalize better (Shorten and Khoshgoftaar 2019). We implemented spatial and color augmentations. The user can choose the desired augmentations, which are then randomly applied online, while importing the input images.

Model training

InstantDL reads data from the corresponding folders and prepares for training and testing. This includes the following steps: i) Model initialization with pretrained weights, if selected. ii) Import of data and normalization, split of validation data from the data contained in the train data folder, shuffle of training data, batch creation and online data augmentation. iii) Training of the model for the given number of epochs using early stopping, which can be monitored live using tensorboard and automated saving of the best model. iv) Prediction of labels from the test dataset. v) For semantic segmentation, pixel-wise regression and classification uncertainty can be calculated after training.

Model evaluation

The trained model is evaluated on the unseen test images and labels (i.e. the groundtruth). For semantic segmentation, instance segmentation and pixel-wise regression, the network predictions are saved as image stacks to ease evaluation of large datasets with limited CPU capabilities. This also allows an initial manual, qualitative evaluation and quality control. In a second step the predictions can be quantitatively evaluated. For that, accuracy, mean relative and absolute error, pixel-wise Pearson correlation coefficient and Jaccard index over all pixels of the test set are calculated. Boxplots to visualize quantitative model performance are generated. The standard quantitative evaluation output plots (i) the input images side-by-side to the corresponding labels and predictions and (ii) an error map between the labels and predictions to visualize training performance (see example evaluation Fig. 1B-D). For classification the predicted labels in the test set are compared to the true labels and multiple error scores (Jaccard index, mean absolute error, mean squared error, area under curve) are calculated. A confusion matrix and a receiver operating characteristic (ROC) curve are automatically visualized (Fig. 1E). All evaluation steps are implemented in the pipeline and can be set to be executed after testing. Additionally postprocessing (e.g. statistical analysis and visual assessment) is accessible in jupyter notebooks for customization, which are provided with InstantDL.

Uncertainty quantification

Neural networks predictions can be unreliable when the input sample is outside of the training distribution, the image is corrupted or if the model fails. Uncertainty estimation can measure prediction robustness, adding a new level of insights and interpretability of results (Gal and Ghahramani 2016; Lakshminarayanan, Pritzel, and Blundell 2017). Bayesian inference can be approximated in deep Gaussian processes by Monte Carlo dropout (Gal and Ghahramani 2016). We have implemented Monte Carlo dropout for semantic

segmentation, pixel-wise regression and classification in the pipeline. During the inference phase, 20 different models are created using Monte Carlo dropout and model uncertainty is calculated on the test set. For pixel-wise regression and semantic segmentation, the pipeline saves an uncertainty map. From this, pixel uncertainty can be plotted and saved to the project folder or average pixel uncertainty for an image can be calculated (Fig. 2A,B). For classification InstantDL adds the uncertainty score to the results file where a score close to zero indicates certain predictions, and score close to 1 indicates high uncertainty (Fig. 2C,D).

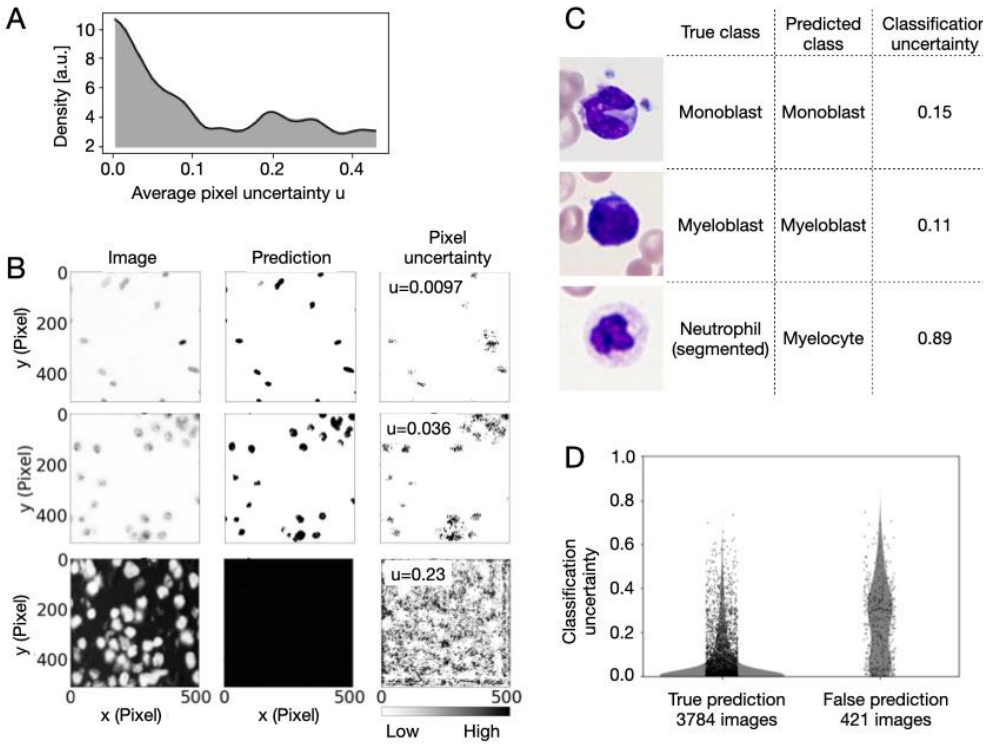

Figure 2: Uncertainty of semantic segmentation, pixel-wise regression and classification can be estimated with InstantDL.

(A) The distribution of average pixel uncertainty u for 162 images of a semantic segmentation task (Caicedo, Goodman, et al. 2019). The distribution is approximately bi-modal.

(B) Three exemplary semantic segmentations from the data visualized in (A) with InstantDLs pixel uncertainty estimation. Correct predictions correspond to a low average pixel uncertainty u (top and middle row), while a high average pixel uncertainty indicates failed segmentations (bottom row). Regions with ambiguous predictions are indicated by high pixel uncertainty (right column).

(C) For the prediction of white blood cells classes (Christian Matek et al. 2019), classification uncertainty indicates incorrect predictions.

(D) The distributions of classification uncertainty for correct and false predictions differ significantly (p-value < 0.001, Mann-Whitney rank test).

Implementation

InstantDL is written in Python 3.6 and leverages Tensorflow 1.14.0 and Keras 2.1.5 (Chollet and Others 2015), which provide an excellent framework for our pipeline due to the modular, composable and user friendly design. The pipeline can be run locally ensuring data protection, on a cluster or with Google Colab (Bisong 2019), which has successfully been

used for deep learning projects ("Deep Learning to Detect Skin Cancer Using Google Colab" 2019) making it usable for those with limited computational resources. We provide the pipeline as one package in a Docker image ("Docker Documentation" 2020) to simplify installation and application.

4 Results

To evaluate InstantDL broadly, we applied it to seven publically available datasets (four of which come from data science challenges) and compared its performance to published results. If no test set was provided, we took 20% of the data to create our own test set. This was done on the highest level of abstraction, for example on patient level or tissue-slide level whenever possible, otherwise the data was randomly split. We used the same evaluation metrics as published in the respective papers (Jaccard index, AUC, Pearson correlation) to compare our results appropriately.

For pre-processing, we transformed the images to .tiff files and classification labels to a .csv file to adapt them to the InstantDL requirements. Training was performed by saving the best model using early stopping. As data augmentation we used horizontal and vertical flipping. For pixel-wise regression we used mean squared error loss, for semantic segmentation we used binary cross entropy loss and for classification we used categorical cross entropy loss. For instance segmentation binary cross entropy was used as segmentation loss in combination with the localization and classification loss in the Mask-RCNN (Kaiming et al. 2017).

We evaluated the performance of semantic segmentation and instance segmentation on three datasets. In the first dataset we segmented nuclei in microscopy images contained in the Data Science Bowl 2018 (Caicedo, Goodman, et al. 2019) dataset. Using InstandDL instance segmentation we reached a median Jaccard index of 0.60 (25-75%ile: 0.61 to 0.58 estimated from bootstrapping), while using semantic segmentation we reached a median Jaccard index of 0.16 (25-75%ile: 0.15 to 0.17). The winner of the challenge reached a Jaccard index of 0.63 while the median participant reached 0.42 (solid and dotted line, Fig. 3A). The second task was the multi-organ nuclei segmentation challenge. Here, 30 microscopy images of various organs with hematoxylin and eosin staining are provided (Kumar et al. 2020, 2017). We reached a median Jaccard score of 0.57 (25-75%ile: 0.56 to 0.59) with InstantDL's semantic segmentation and 0.29 (25-75%ile: 0.28 to 0.30) with instance segmentation. The winner of the challenge reached 0.69 and the median participant scored 0.63 (solid and dotted line, Fig. 3B). Thirdly, we benchmarked InstantDL on lung CT images from the Vessel-12 challenge (Rudyanto et al. 2014). Using instance segmentation we reached an area under the receiver operator curve (AUC) of 0.94 (25-75%ile: 0.94 to 0.94), and 0.90 (25-75%ile: 0.88 to 0.92) with instance segmentation. The winner of the challenge reached a score of 0.99 and the median participant 0.94 (solid and dotted line, Fig. 3C).

To evaluate InstantDL's performance for pixel-wise regression, we predicted the 3D nuclear envelope and mitochondria staining from brightfield images (Ounkomol et al. 2018). For the nuclear envelope staining prediction, we achieved a median pixel-wise Pearson correlation

to the real staining of 0.85 and 0.78 for the prediction of mitochondria staining (Fig. 3D), similar to the published result.

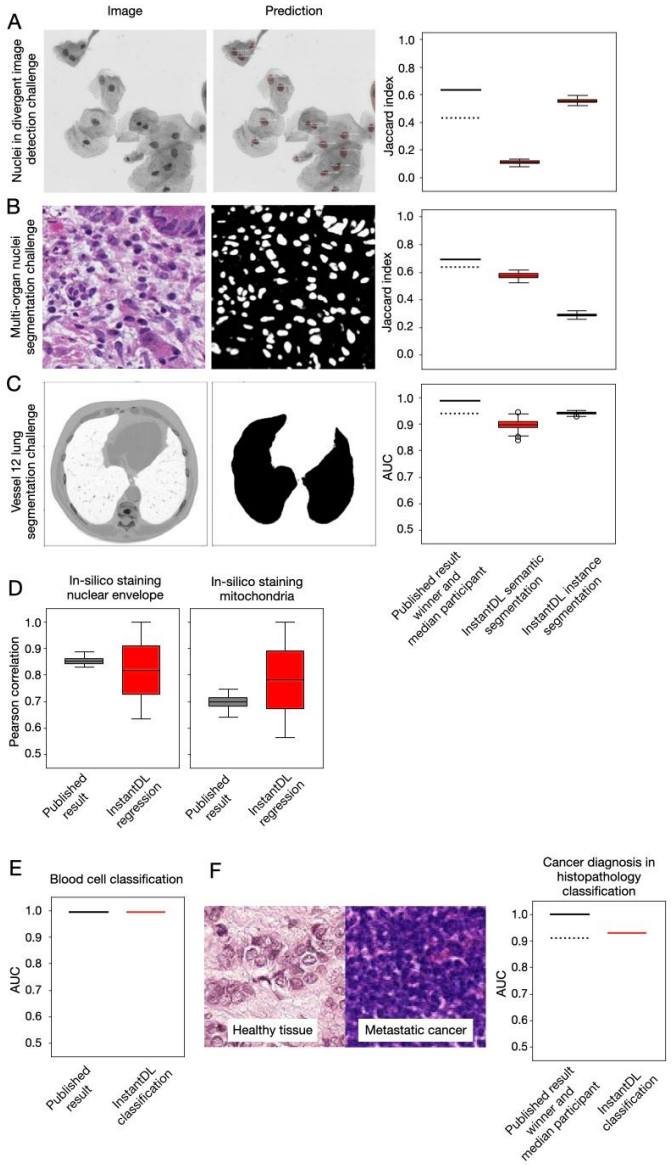

Figure 3: InstantDL achieves competitive performance on published datasets and computer vision challenges without hyperparameter tuning.

(A) InstantDLs instance segmentation achieves competitive results on the nuclear detection challenge dataset (Caicedo, Goodman, et al. 2019), which contains a variety of experimental conditions, cell types, magnifications, and imaging modalities. We show one exemplary image from the dataset and the corresponding prediction using InstantDL's instance segmentation. The winner of the challenge achieved a Jaccard index of 0.63 (solid line), while the median participant achieved 0.42 (dotted line). InstandDLs instance segmentation achieved a median Jaccard index of 0.60 without hyperparameter tuning. We estimate the Jaccard index distribution by bootstrapping, sampling 100 times half of the test set. Boxes indicate the median and the 25/75%ile of the distribution, whiskers indicate the 1.5 interquartile range.

(B) For the challenge of segmenting nuclei in microscopy images of multiple organs with hematoxylin and eosin staining (Kumar et al. 2020, 2017), the winner achieved a Jaccard index of 0.69 (solid line) and the median participant 0.63 (dotted line). InstantDL using instance segmentation reached a Jaccard index of 0.29, and 0.57 using semantic segmentation.

(C) Evaluation of instance segmentation of lung CT images from the Vessel-12 dataset (Rudyanto et al. 2014). The winner of the challenge reached an area under the receiver operating characteristic curve (AUC) of 0.99, while the median participant reached 0.94. InstantDL reached an AUC of 0.90 with semantic segmentation, and 0.94 with instance segmentation.

(D) InstantDL's pixel-wise regression performs similarly well as the published approach ((Ounkomol et al. 2018) for in-silico staining of bright-field images in three dimensions, but with a higher variability. We achieved a median pearson correlation of 0.85 for nuclear envelope staining and 0.78 for mitochondria staining.

(E) For classification of leukemic blast cell images vs. benign white blood cell images (C. Matek et al. 2019; Shen et al. 2019), InstantDL achieved an AUC of 0.99, while Matek et al. report 0.99.

(F) Classification of metastatic cancer in small image patches taken from larger digital pathology scans on histopathological images ("Histopathologic Cancer Detection" n.d.). InstantDL achieved an AUC of 0.93 while the winner of the challenge achieved an AUC of 1.0 and the median participant 0.91.

Classification performance was evaluated on two datasets. In Matek et al. images of single white blood cells from 100 leukemic and 100 non-leukemic individuals were classified into leukemic blast cell images vs. benign white blood cell subtype images (Christian Matek et al.

2019). We reached an AUC of 0.99 where Matek et al. reached 0.99 (Fig. 3E). The second task was to classify metastatic cancer image patches of digital pathology scans, taken from the Histopathologic Cancer Detection Kaggle challenge ("Histopathologic Cancer Detection" n.d.). We reached an AUC of 0.93, while the winner reached 1.00 (solid line in Fig. 3F) and the median participant scored 0.91 (dotted line, Fig. 3F).

5 Discussion

We present InstantDL, a deep learning pipeline for semantic segmentation, instance segmentation, pixel-wise regression and classification of biomedical images. InstantDL simplifies the access to the advantages of deep learning algorithms for biomedical researchers with limited computer science background. The only requirement is a solid understanding of the data (and how appropriately split it into training and test set), as well as of the task and loss function that should be optimized during training the model (see e.g. (Goodfellow, Bengio, and Courville 2016; LeCun, Bengio, and Hinton 2015)). The pipeline is designed for maximum automation to make training and testing as convenient and as easy as possible. However, some parameter settings depend on the dataset properties and therefore cannot be automated. After setting a maximum of 11 parameters, the pipeline can be run without further user interactions. We included state-of-the-art analysis metrics that are accessible out of the box. Moreover, we included uncertainty prediction to provide an additional level of interpretability of predictions. We tested the performance of InstatDL on a variety of publicly available data sets and achieved competitive results without any hyperparameter tuning.

	InstantDL	**Open ML**	**Google Cloud AI**	**ImJoy**	**ZeroCostDL4 Mic**
Host	Local, on cluster, or Google Colab	Web based	Web based	Web based	Web based (Google Colab)
Data privacy	Yes, if run locally	No (shared with upload)	Limited (hosted in the cloud)	Limited (hosted in the cloud)	Limited (hosted in the cloud)
Target audience	Biomedical researchers	Researchers and developers	Enterprises	Researchers and developers	Biomedical researchers
Developed for	Biomedical images	All kinds of data	All kinds of data	All kinds of data	Biomedical images
Customizability of Code	Open source	Open source	Code not available	Possible through writing a plugin	Open source
Cost	Free	Free	Payment plan	Payment plan	Free

Table 2. Comparison of InstantDL to other deep learning pipelines.

Other deep learning frameworks such as OpenML (Vanschoren et al. 2013), Google Cloud AI (Yilmaz, Aydin, and Demirbas 2014), ImJoy (Ouyang et al. 2019) and ZeroCostDL4Mic (Von Chamier et al. 2020) provide similar tools for applying deep learning methods on user data (Table 2). However, they all require data upload to a cloud system. OpenML, e.g., offers an online ecosystem of datasets and machine learning models, but the dataset will be made publically available with upload. Google Cloud AI offers a framework for data analysis with predefined, not-modifiable models and a fee will be accrued. ImJoy and ZeroCostDL4Mic are browser based applications, requiring upload of the data to the web. Our pipeline can easily be installed and run locally on a computer or server, ensuring data privacy and security. Additionally it can be used on cloud solutions, such as Google Colab. As the code is publicly available it is convenient to tune and extend InstantDL.

Applied to microscopy images of single cells, deep learning-powered image processing can contribute to uncovering and understanding of a wide variety of molecular and cellular processes (Van Valen et al. 2016; Moen et al. 2019). With InstantDL, we hope to empower biomedical researchers to conduct reproducible image processing with a convenient and easy-to-use pipeline.

6 Availability of data and materials

InstantDL is published on GitHub: https://github.com/marrlab/InstantDL. There, we provide code, extensive documentation, and instructive examples of how to run the pipeline.

For reproducing our results, all data used for illustration and benchmarking is available for download at https://hmgubox2.helmholtz-muenchen.de/index.php/s/YXRD4a7qHnCa9x5

7 Competing interests

None.

8 Authors' contributions

Dominik Waibel implemented the pipeline and conducted experiments. He wrote the manuscript and created the figures with Carsten Marr and Sayedali Shetab Boushehri. Sayedali Shetab Boushehri revised and refactored the code, added docker installation and tested the pipeline. Carsten Marr supervised the study.

Acknowledgements and Funding

We thank Niklas Köhler and Nikos Chlis (Munich) for contributing to the initial concept for InstantDL. We thank Daniel Schirmacher (Zürich), Lea Schuh, Johanna Winter, Moritz Thomas, Matthias Hehr, Benjamin Schubert, Niklas Kiermeyer, and Benedikt Mairhörmann (all Munich) for valuable feedback on the manuscript and using InstantDL. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 862811 (RSENSE). Carsten Marr was supported by the BMBF, grant 01ZX1710A-F (Micmode-I2T). Sayedali Shetab Boushehri is a member of the Munich School for Data Science (MUDS).

References

Abdulla, Waleed. 2017. “Mask R-CNN for Object Detection and Instance Segmentation on Keras and TensorFlow.” *GitHub Repository*. Github. https://github.com/matterport/Mask_RCNN.

Bisong, Ekaba. 2019. “Google Colaboratory.” In *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, edited by Ekaba Bisong, 59–64. Berkeley, CA: Apress.

Brinker, Titus Josef, Achim Hekler, Jochen Sven Utikal, Niels Grabe, Dirk Schadendorf, Joachim Klode, Carola Berking, Theresa Steeb, Alexander H. Enk, and Christof von Kalle. 2018. “Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review.” *Journal of Medical Internet Research* 20 (10): e11936.

Buggenthin, Felix, Florian Buettner, Philipp S. Hoppe, Max Endele, Manuel Kroiss, Michael Strasser, Michael Schwarzfischer, et al. 2017. “Prospective Identification of Hematopoietic Lineage Choice by Deep Learning.” *Nature Methods* 14 (4): 403–6.

Caicedo, Juan C., Sam Cooper, Florian Heigwer, Scott Warchal, Peng Qiu, Csaba Molnar, Aliaksei S. Vasilevich, et al. 2017. “Data-Analysis Strategies for Image-Based Cell Profiling.” *Nature Methods* 14 (9): 849–63.

Caicedo, Juan C., Allen Goodman, Kyle W. Karhohs, Beth A. Cimini, Jeanelle Ackerman, Marzieh Haghighi, Cherkeng Heng, et al. 2019. “Nucleus Segmentation across Imaging Experiments: The 2018 Data Science Bowl.” *Nature Methods* 16 (12): 1247–53.

Caicedo, Juan C., Jonathan Roth, Allen Goodman, Tim Becker, Kyle W. Karhohs, Matthieu Broisin, Csaba Molnar, et al. 2019. “Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images.” *Cytometry. Part A: The Journal of the International Society for Analytical Cytology* 95 (9): 952–65.

Chollet, François, and Others. 2015. “Keras.” 2015. https://keras.io.

Christiansen, Eric M., Samuel J. Yang, D. Michael Ando, Ashkan Javaherian, Gaia Skibinski, Scott Lipnick, Elliot Mount, et al. 2018. “In Silico Labeling: Predicting Fluorescent Labels in Unlabeled Images.” *Cell* 0 (0). https://doi.org/10.1016/j.cell.2018.03.040.

Cireşan, Dan C., Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. 2013. “Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks.” *Medical Image Computing and Computer-Assisted Intervention: MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention* 16 (Pt 2): 411–18.

“Deep Learning to Detect Skin Cancer Using Google Colab.” 2019. *International Journal of Engineering and Advanced Technology* 8 (6): 2176–83.

“Docker Documentation.” 2020. Docker Documentation. May 12, 2020. https://docs.docker.com/.

Esteva, Andre, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. 2017. “Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks.” *Nature* 542 (7639): 115–18.

Falk, Thorsten, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, et al. 2019. “U-Net: Deep Learning for Cell Counting, Detection, and Morphometry.” *Nature Methods* 16 (1): 67–70.

Gal, Yarin, and Zoubin Ghahramani. 2016. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.” In *International Conference on Machine Learning*, 1050–59.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. “Deep Residual Learning for Image Recognition.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–78.

Hénaff, Olivier J., Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali

Eslami, and Aaron van den Oord. 2019. “Data-Efficient Image Recognition with Contrastive Predictive Coding.” *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1905.09272.

“Histopathologic Cancer Detection.” n.d. Accessed April 28, 2020a. https://www.kaggle.com/c/histopathologic-cancer-detection/overview.

———. n.d. Accessed April 22, 2020b. https://www.kaggle.com/c/histopathologic-cancer-detection/overview.

Hollandi, Reka, Abel Szkalisity, Timea Toth, Ervin Tasnadi, Csaba Molnar, Botond Mathe, Istvan Grexa, et al. 2020. “nucleAIzer: A Parameter-Free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer.” *Cell Systems*, May. https://doi.org/10.1016/j.cels.2020.04.003.

Isensee, Fabian, Jens Petersen, Andre Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, et al. 2018. “nnU-Net: Self-Adapting Framework for U-Net-Based Medical Image Segmentation.” *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1809.10486.

Kaiming, H., G. Georgia, D. Piotr, and G. Ross. 2017. “Mask R-Cnn.” *Proceedings / IEEE International Conference on Computer Vision. IEEE International Conference on Computer Vision*.

Keras-Contrib. n.d. Github. Accessed October 25, 2019. https://github.com/keras-team/keras-contrib.

Kumar, Neeraj, Ruchika Verma, Deepak Anand, Yanning Zhou, Omer Fahri Onder, Efstratios Tsougenis, Hao Chen, et al. 2020. “A Multi-Organ Nucleus Segmentation Challenge.” *IEEE Transactions on Medical Imaging* 39 (5): 1380–91.

Kumar, Neeraj, Ruchika Verma, Sanuj Sharma, Surabhi Bhargava, Abhishek Vahadane, and Amit Sethi. 2017. “A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology.” *IEEE Transactions on Medical Imaging* 36 (7): 1550–60.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. 2017. “Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles.” In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 6402–13. Curran Associates, Inc.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. “Deep Learning.” *Nature* 521 (7553): 436–44.

Litjens, Geert, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. 2017. “A Survey on Deep Learning in Medical Image Analysis.” *Medical Image Analysis* 42 (December): 60–88.

Liu, Xiaoxuan, Livia Faes, Aditya U. Kale, Siegfried K. Wagner, Dun Jack Fu, Alice Bruynseels, Thushika Mahendiran, et al. 2019. “A Comparison of Deep Learning Performance against Health-Care Professionals in Detecting Diseases from Medical Imaging: A Systematic Review and Meta-Analysis.” *The Lancet Digital Health* 1 (6): e271–97.

Maier, Andreas, Christopher Syben, Tobias Lasser, and Christian Riess. 2019. “A Gentle Introduction to Deep Learning in Medical Image Processing.” *Zeitschrift Fur Medizinische Physik* 29 (2): 86–101.

Matek, Christian, Simone Schwarz, Karsten Spiekermann, and Carsten Marr. 2019. “Human-Level Recognition of Blast Cells in Acute Myeloid Leukaemia with Convolutional Neural Networks.” *Nature Machine Intelligence* 1 (11): 538–44.

Matek, C., S. Schwarz, K. Spiekermann, and C. Marr. 2019. “Human-Level Recognition of Blast Cells in Acute Myeloid Leukemia with Convolutional Neural Networks.” *bioRxiv*. https://www.biorxiv.org/content/10.1101/564039v1.abstract.

McKinney, Scott Mayer, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, et al. 2020. “International Evaluation of an AI System for Breast Cancer Screening.” *Nature* 577 (7788): 89–94.

Minaee, Shervin, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. 2020. “Image Segmentation Using Deep Learning: A Survey.” *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/2001.05566.

Moen, Erick, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. 2019. “Deep Learning for Cellular Image Analysis.” *Nature Methods* 16 (12): 1233–46.

Ngiam, Jiquan, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V. Le, and Ruoming Pang. 2018. “Domain Adaptive Transfer Learning with Specialist Models.” *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1811.07056.

Otsu, Nobuyuki. 1979. “A Threshold Selection Method from Gray-Level Histograms.” *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1): 62–66.

Ounkomol, Chawin, Sharmishtaa Seshamani, Mary M. Maleckar, Forrest Collman, and Gregory Johnson. 2018. “Label-Free Prediction of Three-Dimensional Fluorescence Images from Transmitted Light Microscopy.” *bioRxiv*, May, 289504.

Ouyang, Wei, Florian Mueller, Martin Hjelmare, Emma Lundberg, and Christophe Zimmer. 2019. “ImJoy: An Open-Source Computational Platform for the Deep Learning Era.” *Nature Methods* 16 (12): 1199–1200.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. “UNet Convolutional Networks for Biomedical Image Segmentation.” *arXiv:1505. 04597 [cs]*, May. http://arxiv.org/abs/1505.04597.

Rudyanto, Rina D., Sjoerd Kerkstra, Eva M. van Rikxoort, Catalin Fetita, Pierre-Yves Brillet, Christophe Lefevre, Wenzhe Xue, et al. 2014. “Comparing Algorithms for Automated Vessel Segmentation in Computed Tomography Scans of the Lung: The VESSEL12 Study.” *Medical Image Analysis* 18 (7): 1217–32.

Shen, Li, Laurie R. Margolies, Joseph H. Rothstein, Eugene Fluder, Russell McBride, and Weiva Sieh. 2019. “Deep Learning to Improve Breast Cancer Detection on Screening Mammography.” *Scientific Reports* 9 (1): 12495.

Shorten, Connor, and Taghi M. Khoshgoftaar. 2019. “A Survey on Image Data Augmentation for Deep Learning.” *Journal of Big Data* 6 (1): 60.

Tschandl, Philipp, Noel Codella, Bengü Nisa Akay, Giuseppe Argenziano, Ralph P. Braun, Horacio Cabo, David Gutman, et al. 2019. “Comparison of the Accuracy of Human Readers versus Machine-Learning Algorithms for Pigmented Skin Lesion Classification: An Open, Web-Based, International, Diagnostic Study.” *The Lancet Oncology* 20 (7): 938–47.

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. “OpenML: Networked Science in Machine Learning. SIGKDD Explorations 15, 2 (2013), 49--60.”

Van Valen, David A., Takamasa Kudo, Keara M. Lane, Derek N. Macklin, Nicolas T. Quach, Mialy M. DeFelice, Inbal Maayan, Yu Tanouchi, Euan A. Ashley, and Markus W. Covert. 2016. “Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments.” *PLoS Computational Biology* 12 (11): e1005177.

Von Chamier, L., J. Jukkala, C. Spahn, and M. Lerche. 2020. “ZeroCostDL4Mic: An Open Platform to Simplify Access and Use of Deep-Learning in Microscopy.” *BioRxiv*. https://www.biorxiv.org/content/10.1101/2020.03.20.000133v1.abstract.

Waibel, Dominik, Janek Gröhl, Fabian Isensee, Thomas Kirchner, Klaus Maier-Hein, and Lena Maier-Hein. 2018. “Reconstruction of Initial Pressure from Limited View Photoacoustic Images Using Deep Learning.” In *Photons Plus Ultrasound: Imaging and Sensing 2018*, 10494:104942S. International Society for Optics and Photonics.

Yilmaz, Y. S., B. I. Aydin, and M. Demirbas. 2014. “Google Cloud Messaging (GCM): An Evaluation.” In *2014 IEEE Global Communications Conference*, 2807–12.