Original Research Article

Number of necessary training examples for Neural Networks with different number of trainable parameters

Th.I. Götz \textsuperscript{c,e,f,*}, S. Göb \textsuperscript{c}, S. Sawant \textsuperscript{c}, X.F. Erick \textsuperscript{c}, T. Wittenberg \textsuperscript{c}, C. Schmidkonz \textsuperscript{a,f}, A.M. Tomé \textsuperscript{d}, E.W. Lang \textsuperscript{b}, A. Ramming \textsuperscript{e}

\textsuperscript{a} Clinic of Nuclear Medicine, University Hospital Erlangen, 91054 Erlangen, Germany
\textsuperscript{b} CMI, Group, Biophysics, University of Regensburg, 93040 Regensburg, Germany
\textsuperscript{c} Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits IIS, Erlangen, Germany
\textsuperscript{d} IEETA, DEIT, Universidade de Aveiro, 3810-193 Aveiro, Portugal
\textsuperscript{e} Department of Industrial Engineering and Health, Technical University of Applied Sciences Amberg-Weiden, Weiden, Germany

ARTICLE INFO

Keywords:
Cell segmentation
Pruning
Deep neural networks
Training sample size
DNN complexity

ABSTRACT

In this work, the network complexity should be reduced with a concomitant reduction in the number of necessary training examples. The focus thus was on the dependence of proper evaluation metrics on the number of adjustable parameters of the considered deep neural network. The used data set encompassed Hematoxylin and Eosin (H&E) colored cell images provided by various clinics. We used a deep convolutional neural network to get the relation between a model’s complexity, its concomitant set of parameters, and the size of the training sample necessary to achieve a certain classification accuracy. The complexity of the deep neural networks was reduced by pruning a certain amount of filters in the network. As expected, the unpruned neural network showed best performance. The network with the highest number of trainable parameter achieved, within the estimated standard error of the optimized cross-entropy loss, best results up to 30\% pruning. Strongly pruned networks are highly viable and the classification accuracy declines quickly with decreasing number of training patterns. However, up to a pruning ratio of 40\%, we found a comparable performance of pruned and unpruned deep convolutional neural networks (DCNN) and densely connected convolutional networks (DCCN).

Introduction

Modern deep convolutional neural networks (DCNN) can easily encompass millions of parameters, quite a number of them being redundant or close to zero in magnitude. As such, highly complex network architectures put a heavy load to the necessary hardware as well as to computation time and require a large training sample. Especially in the medical area, data samples are often small lacking sufficient data to train complex networks, thus incurring the risk of overfitting. Moreover, it is difficult to know how many training examples are necessary for a neural network with a given number of parameters. Hence, efforts have increased recently to reduce network complexity. One way to do so is network pruning, whereby one tries to simplify the network architecture without impairing network performance. Removing weak or redundant weights speeds up learning and occasionally also improves prediction accuracy. Anyway, pruning first tries to identify the most promising network units (neurons, channels, filters) to be removed. Afterwards, the pruned network model needs further training and fine-tuning to recover the base model’s prediction performance.\textsuperscript{15}

In practice, analytical proxies like the number of adjustable parameters or floating point operations (FLOPs) are commonly used to quantify pruning efficacy. Such DCNNs with reduced network complexity have several advantages to offer: Concerning server computations, less complex models reduce bandwidth usage, power consumption, and operational costs, while computations on embedded systems or edge devices guaranty privacy, low latency, and better customization.\textsuperscript{37}

Work related to pruning

When considering network pruning, often heuristics are employed to identify candidate units to be removed. Generally, either data-agnostic saliency criteria are computed or data-aware techniques are considered. The former group is, for example, represented by methods considering the Hessian matrix to identify weights to be removed without harming the prediction accuracy.\textsuperscript{15} As such techniques encompass heavy computations, alternatively weights may be grouped according to some similarity measures and each group is then replaced by its prototype weight.\textsuperscript{50}

\textsuperscript{*} Corresponding author.
\textsuperscript{E-mail address:} theresa.goetz@biologie.uni-regensburg.de (T.I. Götz).

http://dx.doi.org/10.1016/j.jpi.2022.100114
Accepted 12 March 2021
Available online 06 July 2022
2153-3539/© 2022 Published by Elsevier Inc. on behalf of Association for Pathology Informatics. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
If weight values, i.e. their $L_2$-norms, represent a saliency criterion, pruning them iteratively provides another simple alternative.13 The grain of salt with such techniques is the sparse matrices they create, which don't help speeding up inference. These drawbacks can be leveraged by applying structural pruning methods.30 The latter commonly deploy $L_1$-norm Lasso constraints and consider channel- or filter-based pruning approaches.31,35 More generally, a Group Lasso approach55 to pruning dealt with variable selection problems at the group level. Smoothing functions were introduced to alleviate problems arising with gradient computations at the origin. Enforcing sparsity even further, the $L_1$-norm constraint has been replaced in some studies by a $L_{1/2}$-norm regularization. The latter penalty yielded better sparsity than the $L_1$-norm regularization at similar computational costs. Chang et al.4 proposed a network pruning method based on the $L_{1/2}$ penalty, which reduced incorrect pruning by increasing the sparsity of the pre-trained models. Also in Wang et al.,6 the $L_{1/2}$-norm constraints have been considered at the channel level, while channel importance has been evaluated by a genetic algorithm. Structured sparsity also often needs some control based on attention mechanisms as pointed out in Torfi et al.,34 whose work is an extension of the work of Wen et al.62 Similarly, the work of Lin et al.35 considered regularized structured filter pruning by incorporating 2 different regularizers into the objective function to fully coordinate global outputs and local filter pruning operations. Also a novel regularization-based pruning method, named IncReg, was proposed by Wang et al.,57 which incrementally assigns different regularization factors to different weights based on their relative importance. Further pruning methods on the filter pruning level considered the scaling factor of batch normalization layers as salient feature65 or removed filters close to their geometric median.16 Recently structured Dirichlet filter pruning has been proposed by Adamczewski and Park.1 The authors assigned Dirichlet distributions over every channel's convolutional layers and estimated by variance inference the parameters of the distribution. The latter allowed them to identify irrelevant filters of the architecture. As a by-product, the method provided interpretable features. Filter pruning methods also have close connections to low rank network matrix decomposition techniques.345 Swaminathan et al.52 proposed a sparse low rank weight matrix decomposition, while considering the significance of both input and output nodes of a layer. Recent work by Yeom et al.48 also explored connections between pruning and matrix decomposition methods by developing a new energy-aware pruning technique. The preserved energy corresponded to the summed singular values of the filter decomposition and filters were pruned on the basis of their energy content. Concerning data-driven pruning methods on the weight level, the average percentage of zeros of nodes19 served as saliency criterion for weight importance and network trimming. Structured pruning at the channel level was considered by He et al.17 by minimizing the reconstruction error for input data. Also the entropy of channel activation was considered a suitable saliency measure for channel removal34 as well as cross entropy. Concerning loss functions, their derivatives can serve as cost measures for feature dropping and were deployed by Molchanov et al.41 to prune network structures based on grouped feature map activation. Similarly, the magnitude of gradients has also been considered a proper saliency measure for network pruning.31

Work related to cell segmentation

A seminal recent review of deep learning methods for cellular image analysis is provided by Moen et al.40 highlighting to biologists the use of deep learning techniques for biological image understanding. If image classification is intended, biological images often lack appropriate annotation with class labels, hence transfer learning deems most suitable here.34,64 Transfer learning resorts to an image classifier that has been trained on a generally huge image data set, such as ImageNet, and only re-trains the final fully connected layer with an often small image data set of interest.10,64 Changes in cell morphology may not always be captured by a labeled training data set. Here, deep learning can be used to extract feature vectors rather than labels. These features can then be clustered according to some similarity measure and appropriately classified.24,44,49 Image classification can also be employed to identify cell states. A recent study trained a classifier with images labeled with a fluorescence marker for cell differentiation. The trained classifier was then employed to identify differentiated cells directly from corresponding bright field images.48,49 Yet another study applied a deep learning classifier to fluorescence images to determine characteristic spatial patterns in the images which indicated protein localization in large data sets from yeast27,28,65 and humans.61

Another large field of image analysis, where deep learning techniques can be employed favorably, is image segmentation. Classical segmentation methods often rely on simple thresholding,23 but with moderate success only. Modern segmentation techniques come in 2 flavors: semantic segmentation and instance-based segmentation. The former partitions a cell image into semantically meaningful parts, like cytoplasm, nucleus, cytoskeleton etc. and labels all pixels belonging to the latter appropriately. Instance-based segmentation, instead, focuses on every instance of a class in any given cell image. Early examples of the latter approaches are given by the software packages U-Net47 and DeepCell.56 They consider instance-based segmentation as a pixel-level classification task. The trained image classifier then generates predictions about class membership pixel-wise thereby grouping pixels into categories like cell interior, cell edges, and background, for instance. A rather more classical segmentation technique is watershedding, whereby deep learning has recently been employed to learn a distance measure. This allows to build a mask encompassing those pixels which have at least a minimal distance from the background of the image.27 A recent application to segment single cells in images performed surprisingly well.59 Other approaches to image segmentation are based on object detection techniques, whereby the bounding box of every object is identified. Deep learning algorithms like Faster R-CNN16 and Retinanet36 combine bounding box detection with a suppression of non-maximal pixels to avoid redundant bounding box predictions.18,21,22,55 A rather different approach to image segmentation considers the segmentation problem as a vector embedding problem.6,45 Thereby, all pixels belonging to an object are assigned by a discriminative loss function to the same vector, while pixels of different objects belong to different vectors. All objects are finally identified through clustering techniques applied on the embedding space.14,39,66 Recently, most of these techniques have been generalized to be applicable to 3D data sets.1,2,64,53 These deep learning-based image segmentation techniques are applicable in many fields of science. They help to automate common computer vision workflows and render possible segmentation tasks that previously deemed impossible. For example, a precise quantification of localization-based live-cell reporters became possible by way of an accurate identification of the cytoplasm in mammalian cells.42,56 A recent study explored this image segmentation method to investigate the mechanisms of cell size control during the fission of yeast cells.8 Another exciting application concerns the use of instance-based segmentation in pathology images.25,26,29,30 There interactions between tumor cells and immune cells were investigated employing spatial proteomics methods in a formalin-fixed and paraffin-embedded substrate.26

In this work, filter pruning methods are deployed to reduce network complexity and the question of the number of necessary training examples for various numbers of trainable parameters is investigated, using the example of a cell segmentation network.

Method

In this section, the used data set is described and its preprocessing is explained. Moreover, we sketch the state-of-the-art neural network applied, and provide an overview of the number of trainable parameters.

Data set

In this work, part of the data set provided by Kumar et al.29, henceforth called the Kumar data set for simplicity - was used. It encompassed Hematoxylin and Eosin stain (H & E) cell images provided by various clinics (for
Fig. 1. Cell image of size 250 × 250 pixels with corresponding segmentation.62

an example see Fig. 1). Table 1 collects all information about the various images employed in this study.

The Kumar data set consisted of an equal number of images for every tissue class. Only part of the data was used in this study, whereby a balanced data set such that each tissue class was represented by one image for training and another image for testing. While selecting them, care was taken that the images came from different clinics and, if possible, represented different tumor types. This selection should render the classification task as hard as possible. Furthermore, the trained networks should be suitable for transfer learning on cell images of other clinics. All images were subdivided into overlapping image patches of size (51 × 51 = 2601 pixels). In total, the data set encompassed 14 654 944 such image patches. However, only Ntr = 915 934 overlapping image patches were taken of every input image and constituted the size of the training sample for an unpruned DCNN. Furthermore, the number Np of training samples was chosen to match the number Np = 915 934 of adjustable parameters of our deep convolutional neural network (DCNN). The pixels of every image were classified as belonging either to class cell or class background, whereby the goal of the classification effort was to properly classify the center pixel of every image patch. During the training process, the total number Npr of image patches were separated into 70% training examples and 30% validation examples using a cross-validation method. Also from every tissue, class 1 image was kept out of bag for testing and estimating standard errors. Thus, the out of bag test set consisted of 7 images, each for every tissue class, from which a corresponding number of patches was drawn for testing.

Table 1

| Patient ID TCGA | Organ | Disease type | Usage           |
|-----------------|-------|--------------|-----------------|
| A7 – A13F – 01Z | Breast| Breast invasive carcinoma | Pre-training |
| A7 – A13F – 01Z | Breast| Breast invasive carcinoma | Pre-training |
| AR – A14K – 01Z | Breast| Breast invasive carcinoma | Training |
| E2 – A185 – 01Z | Breast| Breast invasive carcinoma | Testing |
| HE – 7128 – 01Z | Kidney| Kidney renal papillary cell carcinoma | Training |
| B0 – 5711 – 01Z | Kidney| Kidney renal clear cell carcinoma | Testing |
| 38 – 6178 – 01Z | Liver | Lung adenocarcinoma | Training |
| 21 – 5784 – 01Z | Liver | Lung squamous cell carcinoma | Testing |
| G9 – 6336 – 01Z | Prostate| Prostate adenocarcinoma | Training |
| CH – 5767 – 01Z | Prostate| Prostate adenocarcinoma | Testing |
| DK – A266 – 01A | Bladder| Bladder urothelial carcinoma | Training |
| G2 – A2EK – 01A | Bladder| Bladder urothelial carcinoma | Testing |
| AY – ABY K – 01A | Colon | Colon adenocarcinoma | Training |
| NH – A87F – 01A | Colon | Colon adenocarcinoma | Testing |
| KB – A93J – 01A | Stomach| Stomach adenocarcinoma | Training |
| RD – A8SN9 – 01A | Stomach| Stomach adenocarcinoma | Testing |

Fig. 2. Architecture of the DCNN for cell segmentation similar to Xing et al.62
accuracy. Thus, the study explored possibilities to reduce network complexity resulting in a reduced number of needed training examples. For this task, we chose a neural network with several million parameters and another with less than 1 million parameters. The selected network architectures perfectly suited this purpose, as we only wanted to achieve a classification of the central pixel of every image patch.

In Fig. 4, the training and evaluation process is illustrated. First, the DCNN and DCCN were implemented and randomly initialized. The networks were then pre-trained for 10 epochs employing the breast data set indicated in Table 1 of the previous section. The data set, used for pre-training, consisted of only 2 breast carcinoma cell images recorded in a clinic. These images were partitioned into 1,831,868 image patches. This set of image patches was used to train the set of $N_p = 915,934$ adjustable parameters of our network models and provided us with a pre-trained neural network that served as the basis for all further experiments. To reduce network complexity, the pre-trained neural networks were pruned by applying different pruning ratios, starting with 0% pruning and increasing to 90% pruning in steps of 10%. Thereby, pruning was done by adopting the structured $L_1$ pruning method implemented by PyTorch.

This method is based on Li et al.\textsuperscript{32} and uses the $L_1$ norm of filters as an indicator of the importance of the filter. It is assumed that if the $L_1$ norm of a filter is small, the importance of the filter is also low. This norm is calculated for all filters of a conv layer and then 30% of the least important filters are eliminated. This procedure is carried out one after the other for each conv layer.

All layers of the pruned networks were then further trained for 30 epochs employing various numbers of training examples $N_tr(r)$. The latter have been computed by taking the following multiples $r$ of the number $N_p$ of trainable parameters of the pruned DCNN architecture:

$$N_{tr}(r) = \frac{r}{C_1} N_p,$$

where $r = 2, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001$ (1)

For the DCCN computations were performed using only $r$ values between 0.5 and 0.001, because of the demanding compute time and the missing number of training samples. Note that this resulted in different numbers of training examples depending on the chosen pruning ratios. Additionally, all pruned neural networks were trained for further comparability with identical training sample sizes as given in Table 3. The latter have been deduced from the number $N_p$ of parameters of the unpruned DCNN or DCCN, respectively.

After the training process, every trained network was applied to segment the test data sample. When increasing $N_{tr}$, every larger training data set contained the smaller training data set used before. Also care was taken to ensure balanced training data sets with a roughly equal number of pixels per class (cell, non-cell) in each set $X$. To ensure that the data sets $X$ are representative, the distributions of the mutual information between all images of a data set were chosen to be similar.

Despite facing a 2-class classification problem here, we formulate the optimization problem for $K$ classes. Hence, training was guided by a softmax cross-entropy loss function

$$\mathcal{L}(y_i, \hat{y}_i) = -\frac{1}{N} \sum_k \sum_n \left[ y_{in} \hat{y}_{ink} + y_{in} \ln \left( \sum_{k'} \exp(\hat{y}_{ink'}) \right) \right]$$

Fig. 3. Architecture of the DCCN for cell segmentation similar to Huang et al.\textsuperscript{20}

Table 2

| Layer | DCNN Parameters | DCCN Parameters |
|-------|-----------------|-----------------|
| conv 1 | 1225           | 9536            |
| conv 2 | 31,300         | 33,280          |
| conv 3 | 1,44,080       | 1,047,552       |
| FC 1   | 738,104        | 730,988         |
| FC 2   | 2050           | 1026            |
| Total number | 916,759 | 4,794,754 |

Fig. 4. Principal training and evaluation process.
where $N$ denotes the number of pixels in input patch $x_i$. Furthermore, $y_{in} \in (0.1)^K$ denotes a 1-hot encoded ground-truth vector and $\tilde{y}_{in} \in \mathbb{R}^K$ represents a vector of class membership probabilities.

Evaluation metrics play an important role in assessing the outcomes of segmentation models. Hence, prediction quality was assessed deploying the following performance metrics: Dice Sørensen coefficient $DSC$, accuracy $acc$, sensitivity $Se$, and recall $Rec$, specificity $Sp$, and precision $Pre$.

In order to analyze the robustness of the training process, the unpruned DCNN was trained 7-fold with different random seed numbers and $N_p = 91539$ training examples, corresponding to a reduction factor $r = 0.1$. 

Table 4
Cross-entropy loss $L_{ce} = \log p(y_i|x)$. Segmentation accuracy $acc$, Sensitivity, Specificity, and Dice coefficient for different trained network configurations. Number of training examples: $N_p$, related cross-entropy loss $L_{ce}$, test accuracy $acc$. The best result per row, i.e. per reduction factor $r$, is highlighted as bold. Note that $x_{ce} = 1 \cdot 10^{-3}$ and $x_{acc} = 3 \cdot 10^{-3}$ are considered to hold for all DCNN network configurations.

| $r$   | $N_p$ | $N_{ce}$ | $N_{acc}$ | $N_{test}$ | $N_{val}$ | $N_{train}$ |
|-------|-------|----------|-----------|------------|-----------|-------------|
| 2     | 1363 | 835      | 952       | 802        | 933       | 0.176       |
| 1     | 915  | 934      | 828       | 952        | 0.176     | 1.648       |
| 0.5   | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.1   | 915  | 934      | 828       | 952        | 0.176     | 412.7       |
| 0.05  | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.01  | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.005 | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.001 | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |

$\text{Table 3}$
Reduction factor $r$ and corresponding number of training patterns $N_p(r)$. The number of training samples with the label cell $N_{cell}$ and the one with the label no cell $N_{acc}$ are given. Remember that the number of adjustable parameters of the unpruned DCNN amounted to $N_p = 915 343$ and for the DCNN amounted to $N_p = 4 \cdot 794 754$. 

| $r$   | $N_p$ | $N_{ce}$ | $N_{acc}$ | $N_{test}$ | $N_{val}$ | $N_{train}$ |
|-------|-------|----------|-----------|------------|-----------|-------------|
| 2     | 1363 | 835      | 952       | 802        | 933       | 0.176       |
| 1     | 915  | 934      | 828       | 952        | 0.176     | 1.648       |
| 0.5   | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.1   | 915  | 934      | 828       | 952        | 0.176     | 412.7       |
| 0.05  | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.01  | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.005 | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |
| 0.001 | 457  | 967      | 0.828     | 0.952      | 0.176     | 412.7       |

$\text{Table 4}$
Cross-entropy loss $L_{ce} = \log p(y_i|x)$. Segmentation accuracy $acc$, Sensitivity, Specificity, and Dice coefficient for different trained network configurations. Number of training examples: $N_p$, related cross-entropy loss $L_{ce}$, test accuracy $acc$. The best result per row, i.e. per reduction factor $r$, is highlighted as bold. Note that $x_{ce} = 1 \cdot 10^{-3}$ and $x_{acc} = 3 \cdot 10^{-3}$ are considered to hold for all DCNN network configurations.
Finally, all test images were segmented with the trained networks. The standard deviation over these predictions was taken as a measure of the inaccuracy of the predictions for all investigated network configurations, which could not be fully exploited due to an insufficient computing capacity.

Results

Test of reproducibility

As mentioned in the last section, in order to be able to distinguish networks with different pruning configurations, hence different numbers of trainable parameters, the reproducibility of the network architecture was examined first. In 7 different training runs of an unpruned network with \(N_p(r=1.0) = 915\,934\) training examples, each initialized with a different random seed, the predictions of the pixel class memberships achieved with a DCNN model a standard error for the accuracy of \(s_{acc} = 1.7 \cdot 10^{-3}\) and for the cross-entropy of \(s_{sce} = 4.3 \cdot 10^{-3}\), respectively. These numbers were rounded to \(s_{acc} = 2 \cdot 10^{-3}\) and \(s_{sce} = 4 \cdot 10^{-3}\) for convenience. For the DCNN model corresponding standard errors read \(s_{acc} = 2.8 \cdot 10^{-3}\) and \(s_{sce} = 1.3 \cdot 10^{-3}\), respectively, which were rounded to \(s_{acc} = 3 \cdot 10^{-3}\) and \(s_{sce} = 1 \cdot 10^{-3}\) for the sake of an easy comparison. These figures of merit were considered valid for all network configurations investigated in this study, as already explained above.

Varying the number of training examples

Next the unpruned DCNN was trained with various numbers of training examples. The latter were determined, given the pruning ratio \(\delta\), as \(N_p(r,\delta) = N_p(r)(1-\delta)\). Up to a pruning ratio \(\delta\) of 50% \(\delta = 0.5\) and \(N_p \geq N_p\), the accuracy remained practically constant with a maximal test accuracy of \(acc = 0.932 \pm 0.002\), and a minimal cross-entropy loss of \(L_{sce} = H(p,q) = 0.181 \pm 0.003\) was achieved. Note that 99% of the maximal test accuracy could already be obtained with only 9159 training patterns, corresponding to \(r = 0.01\) (see Table 4).

First, Fig. 5 illustrates the dependence of the test accuracies, achieved with the various DCNN pruning configurations, on the number of training examples. Corresponding result in case of the more complex DCCN are shown in Fig. 6. All networks were trained for best image segmentation by minimizing the cross-entropy loss. During training, the number of necessary training examples was varied for a given set of adjustable parameters as determined by the pruning ratio. The former was determined by the chosen network architecture via \(r \cdot N_p\), with \(r\) denoting a factor multiplying the number \(N_p\) of trainable parameters. The most complex unpruned network, either DCNN or DCCN, yielded best results among all training runs. It is remarkable to see that even a reduction of the number of training samples by a factor \(r = 0.001\) leads to a rather modest decrease of the accuracy by roughly 5% only for both network architectures. A similar observation holds for the Dice coefficient, which decreased by 6% and 1%, respectively.

With increased network pruning, however, both accuracy and Dice coefficient degrade substantially for strongly reduced training sample sizes \(r < 0.1\). With \(\delta = 0.9\), for example, we observed a decrease of the accuracy by 11% and 9%, respectively while Dice coefficients decreased to zero for both networks. This is amazing as a reduced network complexity results in a smaller number of adjustable model parameters. Obviously, a too strong reduction of the training sample size results in a worsening of the learning ability of the resulting network. Considering the cross-entropy loss, i.e. the objective that was optimized during training, it steadily increases with decreasing reduction factor \(r\). For the unpruned networks, cross entropy increased by 70% in case of a DCNN but only by 4% in case of a DCCN. However, the increase is especially strong for less complex networks at pruning ratios larger than 50%, where \(L_{sce}\) more than doubles in case of the DCNN, while the increase is only 22% in case of the DCCN.

Fig. 5. Dice coefficient and test accuracy for the DCNN with different pruning ratios. The network configurations were trained with different data sample sizes, scaled by the factor \(r\).
Reducing network complexity by pruning

Next, the dependence of the performance metrics on network complexity was studied. With large training sample sizes, i.e. $r = 1$ and $r = 2$, respectively, in case of the DCNN network and $r = 0.5$ in case of the DCCN network, classification accuracy remained practically unaffected. This was expected as there the largest numbers of training examples were used. The same held true for the Dice coefficient, with the exception of 40% and 50% pruning ratios in case of the DCNN network, which unexpectedly showed a sudden drop for all numbers of training examples. Lacking any

Table 5
Cross-entropy loss $L_{ce} = H(p, q)$, Segmentation accuracy $acc$, Sensitivity, Specificity, and Dice coefficient for different trained network configurations. Number of training examples: $N_{tr}$, related cross-entropy loss: $L_{ce}$, test accuracy: $acc$. The best result per row, i.e. per reduction factor $r$, is highlighted as bold. Note that $s_{ce} = 1 \cdot 10^{-3}$ and $s_{acc} = 3 \cdot 10^{-3}$ are considered to hold for all DCCN network configurations.

| $r$  | $N_{tr}$ | Sensitivity | Specificity | DiceCo | $L_{ce}$ | $N_{tr}$ | Sensitivity | Specificity | DiceCo | $L_{ce}$ |
|------|----------|-------------|-------------|--------|----------|----------|-------------|-------------|--------|----------|
| 0.5  | 2 397 977 | 0.846       | 0.940       | 0.792  | 0.923    | 0.384    | 1 198 689  | 0.838       | 0.941  | 0.787    |
| 0.1  | 479 475  | 0.823       | 0.952       | 0.796  | 0.928    | 0.382    | 239 738   | 0.829       | 0.943  | 0.786    |
| 0.05 | 239 738  | 0.838       | 0.944       | 0.795  | 0.926    | 0.384    | 119 869   | 0.835       | 0.946  | 0.768    |
| 0.01 | 47 948   | 0.818       | 0.947       | 0.795  | 0.927    | 0.386    | 23 974    | 0.754       | 0.942  | 0.740    |
| 0.005| 23 974   | 0.803       | 0.947       | 0.783  | 0.923    | 0.387    | 11 987    | 0.746       | 0.935  | 0.723    |
| 0.001| 4795      | 0.771       | 0.951       | 0.770  | 0.920    | 0.396    | 23 974    | 0.670       | 0.935  | 0.669    |
| 0.5  | 958 951  | 0.835       | 0.944       | 0.789  | 0.923    | 0.386    | 719 213   | 0.817       | 0.940  | 0.768    |
| 0.1  | 191 790  | 0.816       | 0.944       | 0.779  | 0.920    | 0.390    | 143 843   | 0.799       | 0.936  | 0.750    |
| 0.05 | 95 895   | 0.807       | 0.940       | 0.767  | 0.916    | 0.392    | 71 921    | 0.773       | 0.932  | 0.729    |
| 0.01 | 19 179   | 0.733       | 0.941       | 0.724  | 0.904    | 0.408    | 14 384    | 0.728       | 0.926  | 0.700    |
| 0.005| 9590      | 0.715       | 0.940       | 0.710  | 0.900    | 0.415    | 7192      | 0.647       | 0.942  | 0.672    |
| 0.001| 1918      | 0.595       | 0.939       | 0.613  | 0.882    | 0.435    | 1438      | 0.000       | 1.000  | 0.000    |
| 0.5  | 479 475  | 0.834       | 0.932       | 0.765  | 0.914    | 0.392    | 239 738   | 0.772       | 0.937  | 0.739    |
| 0.1  | 95 895   | 0.770       | 0.936       | 0.739  | 0.908    | 0.401    | 47 948    | 0.745       | 0.924  | 0.796    |
| 0.05 | 47 948   | 0.749       | 0.931       | 0.719  | 0.902    | 0.409    | 23 974    | 0.737       | 0.922  | 0.698    |
| 0.01 | 9590      | 0.686       | 0.927       | 0.670  | 0.889    | 0.424    | 4795      | 0.625       | 0.932  | 0.637    |
| 0.005| 4795      | 0.651       | 0.928       | 0.647  | 0.886    | 0.428    | 23 974    | 0.098       | 0.991  | 0.154    |
| 0.001| 959       | 0.000       | 1.000       | 0.000  | 0.818    | 0.500    | 479       | 0.000       | 1.000  | 0.000    |

Fig. 6. Dice coefficient and test accuracy for the DCCN with different pruning ratios. The network configurations were trained with different data sample sizes, scaled by the factor $r$. 

Reducing network complexity by pruning

Next, the dependence of the performance metrics on network complexity was studied. With large training sample sizes, i.e. $r = 1$ and $r = 2$, respectively, in case of the DCNN network and $r = 0.5$ in case of the DCCN network, classification accuracy remained practically unaffected. This was expected as there the largest numbers of training examples were used. The same held true for the Dice coefficient, with the exception of 40% and 50% pruning ratios in case of the DCNN network, which unexpectedly showed a sudden drop for all numbers of training examples. Lacking any
reasonable explanation, we consider the results for these 2 pruning ratios as outliers. Remarkably, up to a pruning ratio of 50% prediction accuracy remained rather high, amounting to a drop of a mere 0.2%. This is not the case for the corresponding Dice coefficient, which for the DCNN network decreased by almost 13% over the same range of network pruning (compare Table 4 but remember that the 50% values were considered outlier). More realistic might be the drop seen over the entire pruning range $\delta = 0 \rightarrow 0.9$, where the corresponding figures read 1% for the accuracy and 4% for the Dice coefficient. Remarkably, for the more complex DCCN network the corresponding decrease at $r = 0.5$ amounted to 1%, roughly for $\delta = 0 \rightarrow 0.5$ while it was seen to be 7% for $\delta = 0 \rightarrow 0.9$ (compare Table 5). The corresponding figures for the accuracy read 0.6% and 3%, respectively. For even larger pruning ratios, i.e. $r = 0.001$, both metrics degraded rapidly with decreasing training sample size and decreasing network complexity for both network architectures. Thus, the accuracy dropped by 9% for $\delta = 0 \rightarrow 0.9$ and 2% for $\delta = 0 \rightarrow 0.5$ in case of the DCNN network. For the DCCN architecture, the corresponding accuracies dropped by 11% for $\delta = 0 \rightarrow 0.9$ and 4% for $\delta = 0 \rightarrow 0.5$. This degradation was, however, especially visible for the Dice coefficient, which reduced to zero for $\delta = 0 \rightarrow 0.9$ for both architectures and yielded 25% for $\delta = 0 \rightarrow 0.5$ in case of DCNN and 13% under the same conditions in case of DCCN. Considering the cross-entropy loss, it again stood almost constant up to a pruning ratio of 50% before starting to increase with further decreasing network complexity. For instance, at $r = 1$ cross entropy increased in case of a DCNN by a mere 2% for $\delta = 0 \rightarrow 0.5$ but by 26% for $\delta = 0 \rightarrow 0.9$. At high reduction factors $r \approx 0.001$, the corresponding values were 12% and 52%, respectively. In case of more complex DCCN networks, we observed an increase by 3% for $\delta = 0 \rightarrow 0.9$ and by 0.3% for $\delta = 0 \rightarrow 0.5$ and $r = 0.5$. Again, for very parsimonious networks ($r = 0.001$) we, instead, observed an increase in $L_{CE}$ by 39% for $\delta = 0 \rightarrow 0.9$ and by 8% for $\delta = 0 \rightarrow 0.5$. The results for the cross-entropy loss $L_{ce}$ and the test accuracy $acc$ for different network configurations are summarized in Table 4 and Table 5.

Network training with fixed training set size

In order to be able to directly compare the performance of pruned neural networks, the different network configurations were trained once again

| $N_p$ | $acc$ | $L_{ce}$ | $Sensitivity$ | $Specificity$ | DiceCo |
|-------|-------|----------|---------------|---------------|--------|
| 0% Pruning, $N_p = 915 934$ | 0.206 | 0.976 | 0.214 | 0.789 | 0.931 |
| 10% Pruning, $N_p = 824 341$ | 0.207 | 0.976 | 0.215 | 0.789 | 0.931 |
| 20% Pruning, $N_p = 732 747$ | 0.208 | 0.976 | 0.216 | 0.789 | 0.931 |
| 30% Pruning, $N_p = 641 154$ | 0.209 | 0.976 | 0.217 | 0.789 | 0.931 |
| 40% Pruning, $N_p = 549 560$ | 0.210 | 0.976 | 0.218 | 0.789 | 0.931 |
| 50% Pruning, $N_p = 457 967$ | 0.211 | 0.976 | 0.219 | 0.789 | 0.931 |
| 60% Pruning, $N_p = 366 374$ | 0.212 | 0.976 | 0.220 | 0.789 | 0.931 |
| 70% Pruning, $N_p = 274 780$ | 0.213 | 0.976 | 0.221 | 0.789 | 0.931 |
| 80% Pruning, $N_p = 183 187$ | 0.214 | 0.976 | 0.222 | 0.789 | 0.931 |
| 90% Pruning, $N_p = 91 593$ | 0.215 | 0.976 | 0.223 | 0.789 | 0.931 |

Table 6
Cross-entropy loss $L_{ce}$ and statistical metrics (Sensitivity, Specificity, acc $\pm 10^{-3}$) for the segmentation prediction of different network configurations are presented. Note that here the number of training examples is the same for all tested pruning ratios. Again remember that the number of adjustable parameters of the unpruned DCNN amounted to $N_p = 916 959$. The rows marked yellow present results obtained with networks where the condition $N_p \geq N_h$ held.
for both network architectures with identical numbers of training examples. All results are collected in Table 6 and Table 7, respectively. Note that here the number of training examples is the same for all tested pruning ratios. Again remember that the number of adjustable parameters of the unpruned DCNN amounted to \( N_p = 4,794,754 \). The rows marked yellow present results obtained with networks where the condition \( N_{tr} \geq N_p \) held.

### Table 7

| \( N_{tr} \) | Sensitivity | Specificity | DiceCo acc | \( L_{ce} \) | Sensitivity | Specificity | DiceCo acc | \( L_{ce} \) |
|------------|-------------|-------------|------------|---------|-------------|-------------|------------|---------|
| 0% Pruning, \( N_{tr} = 915,934 \) | 2,397,377 | 0.846 | 0.940 | 0.792 | 0.923 | 0.384 | 0.838 | 0.941 | 0.787 | 0.922 | 0.385 |
| 479,475 | 0.823 | 0.931 | 0.796 | 0.928 | 0.382 | 0.848 | 0.939 | 0.791 | 0.922 | 0.386 |
| 239,738 | 0.828 | 0.944 | 0.795 | 0.926 | 0.384 | 0.827 | 0.946 | 0.791 | 0.924 | 0.385 |
| 47,948 | 0.818 | 0.949 | 0.795 | 0.927 | 0.386 | 0.772 | 0.941 | 0.750 | 0.912 | 0.394 |
| 23,974 | 0.803 | 0.947 | 0.783 | 0.923 | 0.387 | 0.756 | 0.940 | 0.737 | 0.909 | 0.400 |
| 47,95 | 0.771 | 0.951 | 0.770 | 0.920 | 0.396 | 0.712 | 0.935 | 0.695 | 0.896 | 0.417 |
| 60% Pruning, \( N_{tr} = 366,374 \) | 2,397,377 | 0.859 | 0.934 | 0.787 | 0.920 | 0.389 | 0.836 | 0.935 | 0.771 | 0.916 | 0.389 |
| 479,475 | 0.814 | 0.946 | 0.783 | 0.923 | 0.387 | 0.833 | 0.938 | 0.779 | 0.920 | 0.390 |
| 239,738 | 0.824 | 0.941 | 0.780 | 0.919 | 0.389 | 0.838 | 0.838 | 0.756 | 0.909 | 0.394 |
| 47,948 | 0.785 | 0.937 | 0.747 | 0.910 | 0.401 | 0.774 | 0.774 | 0.729 | 0.902 | 0.405 |
| 23,974 | 0.775 | 0.930 | 0.729 | 0.902 | 0.408 | 0.757 | 0.757 | 0.721 | 0.901 | 0.409 |
| 47,95 | 0.638 | 0.948 | 0.665 | 0.896 | 0.421 | 0.625 | 0.625 | 0.649 | 0.888 | 0.426 |
| 80% Pruning, \( N_{tr} = 183,187 \) | 2,397,377 | 0.843 | 0.935 | 0.775 | 0.918 | 0.387 | 0.805 | 0.948 | 0.777 | 0.924 | 0.386 |
| 479,475 | 0.828 | 0.933 | 0.764 | 0.914 | 0.392 | 0.801 | 0.937 | 0.755 | 0.913 | 0.391 |
| 239,738 | 0.824 | 0.927 | 0.754 | 0.910 | 0.397 | 0.774 | 0.936 | 0.737 | 0.908 | 0.396 |
| 47,948 | 0.762 | 0.925 | 0.718 | 0.900 | 0.411 | 0.758 | 0.923 | 0.711 | 0.898 | 0.410 |
| 23,974 | 0.738 | 0.925 | 0.705 | 0.897 | 0.416 | 0.723 | 0.920 | 0.686 | 0.890 | 0.421 |
| 47,95 | 0.622 | 0.938 | 0.639 | 0.889 | 0.426 | 0.675 | 0.921 | 0.654 | 0.882 | 0.432 |
| 90% Pruning, \( N_{tr} = 91,593 \) | 2,397,377 | 0.804 | 0.928 | 0.756 | 0.904 | 0.392 | 0.782 | 0.942 | 0.754 | 0.902 | 0.391 |
| 479,475 | 0.788 | 0.926 | 0.736 | 0.901 | 0.401 | 0.764 | 0.937 | 0.735 | 0.896 | 0.400 |
| 239,738 | 0.784 | 0.923 | 0.721 | 0.897 | 0.406 | 0.748 | 0.931 | 0.725 | 0.890 | 0.401 |
| 47,948 | 0.728 | 0.922 | 0.707 | 0.894 | 0.411 | 0.723 | 0.920 | 0.696 | 0.882 | 0.407 |
| 23,974 | 0.703 | 0.921 | 0.694 | 0.888 | 0.416 | 0.697 | 0.917 | 0.686 | 0.878 | 0.412 |
| 47,95 | 0.607 | 0.930 | 0.639 | 0.875 | 0.421 | 0.673 | 0.913 | 0.654 | 0.872 | 0.425 |

Fig. 7. Test accuracy and Dice coefficient for neural network configurations resulting from different pruning ratios. Each configuration was trained with an identical but variable number of training examples.
number of training samples $N_p = \text{max}$. While the accuracy remained constant in this range, the observed reduction of the Dice coefficient amounted to 1.5%. Only for pruning ratios above 50% both metrics declined more strongly, with the Dice coefficient responding earlier and stronger. Thus, we observed a decrease of the Dice coefficient by 3%, while the accuracy dropped by only 1% for $\delta = 0 \rightarrow 0.9$ at $N_p = \text{max}$. In case of the minimal number of training samples $N_p = \text{min}$, the Dice coefficient dropped to zero at $\delta = 0.9$, however, while the accuracy was reduced by 9%. For less strongly pruned networks, i.e. $\delta = 0 \rightarrow 0.5$, the Dice coefficient dropped by 6% while the accuracy remained constant. In general, the higher the number of training samples, the higher the prediction accuracy. However, the higher the pruning rate, the lower the accuracy.

For the more complex DCCN architecture, the following observations could be made: Considering first the situation, where the number of training samples was maximal ($N_p = \text{max}$) and the network pruning also reached its largest value ($\delta = 0.9$), the Dice coefficient dropped by 5% while the accuracy only decreased by 2% and the cross-entropy objective raised by 2% as well. If, instead, the number of training samples was minimal ($N_p = \text{min}$), the corresponding changes were $\Delta \text{DC}(N_p^{\text{min}}, \delta = 0.9) = 15\%$, $\Delta \text{acc}(N_p^{\text{min}}, \delta = 0.9) = 4\%$, $\Delta LCE(N_p^{\text{min}}, \delta = 0.9) = 9\%$. In case of less strongly pruned DCCN networks and a maximal number of training samples we got $\Delta \text{DC}(N_p^{\text{max}}, \delta = 0.5) = 0.6\%$, $\Delta \text{acc}(N_p^{\text{max}}, \delta = 0.5) = 1\%$, $\Delta LCE(N_p^{\text{max}}, \delta = 0.5) = 1\%$. If, however, the number of training patterns was minimal, the corresponding numbers read $\Delta \text{DC}(N_p^{\text{min}}, \delta = 0.5) = 10\%$, $\Delta \text{acc}(N_p^{\text{min}}, \delta = 0.5) = 3\%$, $\Delta LCE(N_p^{\text{min}}, \delta = 0.5) = 5\%$. Hence, given a maximal number of training samples, strong network pruning had only a minor effect on the performance metrics, which was even less pronounced in case of less strongly pruned networks. There all changes did not exceed 1%. But the effects were much stronger, if the number of training patterns was minimal. In Fig. 9, the predictions of different networks are illustrated. The x-axis corresponds to the number of training samples and the y-axis to the pruning ratio. The test accuracy for all neural network configurations are summarized in Fig. 10.

Analysis of computational parameters

The pruning of networks lead to a decrease in average inference time $\langle t_{\text{inf}}(\delta) \rangle$ and to a decrease in disk space $V_{\text{disk}}$ (see Table 8). This was verified

Fig. 8. Test accuracy and Dice coefficient for neural network DCCN configurations resulting from different pruning ratios. Each configuration was trained with an identical but variable number of training examples.

Fig. 9. Qualitative illustration of the predictions of different network configurations in dependence on the number of training examples and pruning ratios.

Fig. 10. Test accuracy and Dice coefficient for neural network DCCN configurations resulting from different pruning ratios. Each configuration was trained with an identical but variable number of training examples.
Discussion

The main goal of this work was to reduce network complexity with a concomitant reduction in the number of necessary training examples. Network optimization was driven by cross-entropy minimization, yielding a standard error as small as \( s_{\text{acc}} = 3 \times 10^{-3} \) in case of a DCNN architecture and \( s_{\text{acc}} = 1 \times 10^{-3} \) in case of the more complex DCCN architecture. Network predictions were evaluated employing standard statistical metrics like prediction accuracy on a test set, true-positive rate and true-negative rate. The focus thus was on the dependence of proper evaluation metrics on the number of adjustable parameters of the considered deep neural network. First, given a DCNN architecture with its concomitant number of adjustable parameters, the variance of the prediction accuracy was analyzed. With a standard error for the accuracy of \( s_{\text{acc}} = 9 \times 10^{-3} \), the uncertainty of the results is comparatively low. Next, the number of trainable parameters of the DCNN architecture was reduced by pruning a certain amount of weights in the network. For these parsimonious networks with reduced complexity of the weight connectivity, the training sample size was reduced by scaling down the number of training examples by a factor \( r \), which multiplies the number of adjustable parameters of the considered DCNN. Again performance was evaluated by estimating both the standard error of the accuracy \( s_{\text{acc}} \) and of the cross entropy \( s_{\text{ce}} \).

The predictions of different DCNN network configurations are illustrated in Fig. 9. As expected, the unpruned DCNN neural network showed best performance. This network with the highest complexity, hence the largest number of training examples \( N_{\text{tr}} \), corresponding to \( r = 1 \), achieved, within the estimated standard error of the optimized cross-entropy loss, best results, which roughly persisted up to 40% pruning. Despite the rather small standard error \( s_{\text{acc}} \), it appears that this observation holds across all reduction factors \( r \), i.e. across a rather large range of training sample sizes \( N_{\text{tr}} \). Remarkably, even with a pruning rate of 90%, \( \delta = 0.9 \), the achieved test accuracy amounted to 98% of the accuracy of the unpruned network. Even more astonishing, if for this network the number of training examples is reduced by a factor of \( r = 0.01 \) to \( N_{\text{tr}} = 9159 \), the trained network (acc = 0.905) still reached 98% of the maximum achievable accuracy (acc = 0.922). Again, this accuracy was achieved on the independent test set, hence overfitting was not an issue here despite the fact that this network had 10 times more adjustable parameters than training examples.

But with further decreasing numbers of training examples, corresponding to reduction factors lower than \( r < 0.01 \), no good results could be achieved for any network configuration and number of training examples. Taken the other way round, a remarkably high accuracy remained even when the training sample size was reduced by a factor of 100, corresponding to \( r = 0.01 \) for networks with pruning ratios \( \delta \leq 0.5 \). Only then prediction accuracy degraded rapidly, especially for pruning ratios \( \delta \geq 0.5 \). The same was true for the other metric, the Dice coefficient (DC) and was also reflected in an increasing minimal cross-entropy (CE) loss. The additional metrics sensitivity (Se) and specificity (Sp) provided an even more detailed picture. Sensitivity measured how many of the pixels belonging to class cell were correctly predicted, while specificity measured the percentage of correctly assigned background pixels. Remarkably, the specificity or true-negative rate (TNR) was rather high and remained so across all pruning rates. On the contrary, the sensitivity or true-positive rate (TPR) was smaller and decreased further with increasing pruning rate. Furthermore, sensitivity showed a much stronger dependence on the number of training examples and decreased considerably with decreasing \( N_{\text{tr}} \). Note that a high sensitivity indicates that a large number of all pixels belonging to a specific class was classified correctly. Also a high specificity indicates that a large number of pixels not belonging to a specific class were correctly classified accordingly. Fig. 10 provides a qualitative illustration of the accuracy when the number of adjustable parameters and/or the number of training examples changed. As long as the number of model parameters matched the number of training examples, accuracy was generally very high (acc > 90%). But even in case of an insufficient number of training examples, the test accuracy remained amazingly high. As this concerns the previously unseen test data set, overfitting can be safely excluded.

Considering networks of various complexities, i.e. different pruning ratios \( \delta \), the latter were also trained by a fixed but variable number of training examples. The latter was deduced from the number of adjustable parameters of the unpruned network by applying the reduction factor \( r \). If considered as a function of the number of training examples, a common observation is that as along as \( N_{\text{tr}} \leq N_{\text{tr}} \) holds, all metrics evaluated remained largely insensitive to \( N_{\text{tr}} \). These values are marked with a yellow background in Table 6. But if \( N_{\text{tr}} < N_{\text{tr}} \) is met, then some metrics like the sensitivity or the cross-entropy quickly degraded strongly with decreasing \( N_{\text{tr}} \) for all network configurations considered, while others like the specificity, Dice coefficient or the accuracy remained still pretty insensitive to the number of stimuli. Remember that the specificity represented the proportion of the non-cell pixels, and the sensitivity the proportion of cell pixels, that were correctly classified. The following Table 9 provides a comprehensive summary to the detailed results presented in Table 6.

### Table 8

| \( \delta \) | 0.0 | 0.10 | 0.20 | 0.30 | 0.40 | 0.50 | 0.60 | 0.70 | 0.80 | 0.90 |
|-------------|-----|------|------|------|------|------|------|------|------|------|
| \( V_{\text{in}} \) in KB (DCNN) | 3586 | 3172 | 2822 | 2376 | 1996 | 1634 | 1282 | 947  | 623  | 315  |
| \( V_{\text{in}} \) in KB (DCNN) | 19147| 15766| 12602| 9805 | 7356 | 5058 | 3423 | 2079 | 1096 | 467  |
| \( \langle t_{\text{inf}} \rangle \) in ms (DCNN) | 7.6  | 6.1  | 5.4  | 5.9  | 5.2  | 3.9  | 3.3  | 2.8  | 2.2  | 1.8  |
| \( \langle t_{\text{inf}} \rangle \) in ms (DCNN) | 47.6 | 43.1 | 40.6 | 35.2 | 31.9 | 22.3 | 18.5 | 16.3 | 11.1 | 9.8  |
as can be seen from Table 10, where percentage changes of all metrics are presented. The changes relate to a decrease of the number of training samples and a good classification performance was hardly hampered, while for larger pruning ratios some statistical metrics degraded noticeably. The conclusion is that $N_p \geq N_{pr}$ is not a necessary condition and that oversized networks often achieve equally good classification performance. Finally, remember that at a pruning ratio $\delta = 0.5$, where all evaluation metrics still achieved respectable values, average inference time was halved and the used disk space shrank to half of its size needed for the unpruned network. This may be of concern, if such evaluations have to be performed on edge devices.

**Conflict of interest**

None.

**References**

1. Adamczewski K, Park M. Distichlet pruning for neural network compression. Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR; 2021.
2. Bai M, Urtasun R. Deep watershed transform for instance segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 5221–5229.
3. Bao R, Yuan X, Chen Z, Ma R. Cross-entropy pruning for compressing convolutional neural networks. Neural Comput 2018;30:3128–3149.
4. Chang J, Sha J. Prune deep neural networks with the modified $l_0$ penalty. IEEE Access 2018;7:2273–2280.
5. Cichocki A, Zdunek R, Amari S-I. Nonnegative matrix and tensor factorization. IEEE Signal Process Mag 2008;14:1.
6. De Brabandere B, Neven D, Van Gool L. Semantic instance segmentation with a discriminative loss function. arXiv preprint 2017:arXiv 1708.02551:v1. arXiv:1708.02551.
7. Duncan S, Olsson TSG, Hartley M, Dean C, Rosa S. A method for detecting single mrna molecules in Arabidopsis thaliana. Plant Methods 2016;12(1):1-10.
8. Facchetti G, Knapp R, Flor-Paiva I, Chang F, Howard M. Reprogramming c2d-dependent geometry-based cell size control in fusion yeast. Curr Biol 2019;29(2):250–258.
9. Falk T, Mai D, Bensch R, et al. U-net: deep learning for cell counting, detection, and morphometry. Nat Methods 2019;16(1):67–70.
10. Godinez WJ, Honsain I, Lazic SE, Davies JW, Zhang X. A multi-scale convolutional neural network for phenotyping high-content cellular images. Bioinformatics 2017;33(13):1996–2002.
11. Golub M, Lemieux G, Liu M. Full deep neural network training on a pruned weight budget. 2019.
12. Huber MG, Churaz C, Tindall L, et al. Cdeep3m-play-and-play cloud based deep learning for image segmentation of light, electron and x-ray microscopy. bioRxiv 2018:353425.
13. Han S, Pool J, Tran J, Dally W. Learning both weights and connections for efficient neural network. Advances in Neural Information Processing Systems; 2015. p. 1135–1143.
14. Häring M, Großhans J, Wolf F, Eule S. Automated segmentation of epithelial tissue using multi-scale convolutional neural networks acceleration. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019. https://doi.org/10.1109/CVPR.2019.00047.
15. He Y, Liu P, Wang Z, Hu Zh, Yang Y. Filter pruning via geometric median for deep convolutional neural networks. bioRxiv 2018:313173.
16. Hassibi B, Stork DG. Second order derivatives for network pruning: optimal brain surgeon. Advances in Neural Information Processing Systems; 1993. p. 164–171.
17. He Y, Liu P, Wang Z, Hu Zh, Yang Y. Filter pruning via geometric median for deep convolutional neural networks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019. https://doi.org/10.1109/CVPR.2019.00047.
18. Hollandi R, Szakalit A, Toth T, et al. A deep learning framework for nucleus segmentation using image style transfer. bioRxiv 2019:580605.
20. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 4700–4708.

21. Huang Z, Huang J, Gong Y, Huang C, Wang X. Mask scoring r-cnn. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 6409–6418.

22. Johnson JW. Adapting mask-rcnn for automatic nucleus segmentation. arXiv preprint 2018b:1805.00501:1–7. arXiv:1805.00500.

23. Jourdren I, Delaveau T, Marquetet E, Jaqc G, Garcia M. Cosens, a new software dedicated to microscope-based 3d distance measurements: mna-mitochondria distance, from single-cell to population analyses. RNA 2010;16(7):1301–1307.

24. Kanasaiewamy C, Silva LM, Alexandre LA, Santos JM. High-content analysis of breast cancer using single-cell deep transfer learning. J Biomed Screen 2016;21(3):252–259.

25. Keren I, Bosse M, Marquez D, et al. A structured tumor-immune microenvironment in triple negative breast cancer revealed by multiplexed ion beam imaging. Cell 2018;174(6):1373–1387.

26. Khosdeli M, Winkelheimer G, Parvin B. Fusion of encoder-decoder deep networks improves delineation of multiple nuclear phenotypes. BMC Bioinform 2018;19(1):1–11.

27. Kraus OZ, Ba JL, Frey BJ. Classifying and segmenting microscopy images with deep multiple instance learning. Bioinformatics 2016;32(12):152–159.

28. Kraus OZ, Grys BT, Ba J, et al. Automated analysis of high-content microscopy data with deep learning. Mol Syst Biol 2017;13(4):924.

29. Kumar N, Verma R, Sharma S, Bhargava S, Vahadane A, Sethi A. A dataset and a technique for generalized nuclear segmentation for computational pathology. IEEE Trans Med Imaging 2017;36(7):1550–1560.

30. Lebedev V, Lempitsky V. Fast convnets using group-wise brain damage. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016. p. 2554–2564.

31. Li H, Kadar A, Durandovic I, Samet H, Graf HP. Pruning filters for efficient convnets. 2017.

32. Li H, Kadar A, Durandovic I, Samet H, Graf HP. Pruning filters for efficient convnets. CoRR 2016abcs:1508.08710:1–13. arXiv:1508.08710.

33. Li R, Wang X, Qian W, Song Y, Lei L. Robust and structural sparsity auto-encoder with L21-norm minimization. Neurocomputing 2023;425:71–81.

34. Li Y, Gu Sh, Van Gool L, Timofte R. Learning filter basis for convolutional neural network compression. Proceedings of the IEEE International Conference on Computer Vision; 2019. p. 5623–5632.

35. Lin Sh, Ji R, Li Y, Deng Ch, Li X. Towards compact convnets via structure-sparsity regularized filter pruning. IEEE Trans Neural Netw Learn Syst 2020;31:574–588.

36. Lin T-Y, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 2980–2988.

37. Liu J, Tripathi S, Kurup U, Shah M. Pruning algorithms to accelerate convolutional neural networks for resource efficient inference. 2017.

38. Newby JM, Schaefer AM, Lee PT, Forest MG, Lai SK. Convolutional neural networks for high-throughput microscopy images using deep learning. G3: Genes Genomes Genet 2017;7(5):1385–1392.

39. Pawlowski N, Caicedo JC, Singh S, Carpenter AE, Storkey A. Automating morphological profiling with deep convolutional networks. BioXiv 2016:085118.

40. Payer C, Stern D, Neff T, Bischof H, Unschler M. Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks. International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer; 2018. p. 3–11.

41. Ren S, He K, Girshick R, Sun J. Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Machine Intel 2016;39(6):1137–1149.

42. Renneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer; 2015. p. 234–241.

43. Simm J, Klambauer G, Arany A, et al. Repurposing high-throughput image assays enables biological activity prediction for drug discovery. Cell Chem Biol 2018;25(5):611–618.

44. Sommer C, Hoefler R, Samwer M, Gerlich DW. A deep learning and novelty detection framework for rapid phenotyping in high-content screening. Mol Biol Cell 2017;28(23):3428–3436.

45. Pärnamaa T, Parts L. Accurate classification of protein subcellular localization from high-throughput microscopy images using deep learning. A survey 2020;abs/2005.04275:1

46. Simm J, Klambauer G, Arany A, et al. Repurposing high-throughput image assays enables biological activity prediction for drug discovery. Cell Chem Biol 2018;25(5):611–618.

47. Parnamaz T, Parnamaz L. Accurate classification of protein subcellular localization from high-throughput microscopy images using deep learning. G3: Genes Genomes Genet 2017;7(5):1385–1392.

48. Pawlowski N, Caicedo JC, Singh S, Carpenter AE, Storkey A. Automating morphological profiling with deep convolutional networks. BioXiv 2016:085118.