Augmentation for Generalization

Modifiera för att generalisera

Joel Ottosson

Master's thesis
2022:E10

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Abstract

To be able to automatically segment cells in microscopic images would give biologist a new tool to gather crucial data in quantities not possible by manual work. This is however not a trivial problem and has proven to be very difficult, especially if the images are in 3D. One of the major challenges is the ability for the methods to generalize beyond the data it has previously been presented with. This thesis investigates how image augmentation can be used to mitigate this issue in the domain of 3D microscopy. It does so by training two state of the art deep learning models, Plantseg and Cellpose, with different augmentations and then test their ability to generealize on three data sets which can be considered typical for the field. The results show that augmentations have a small but positive effect on the models. If the un-augmented model is completely unable to segment the image, augmentations will not improve the results. However, if the model is performing poorly, but is still able to segment some cells, augmentation can greatly improve the results. No augmentation by itself stood out as having a greater effect than others. Instead the combination of all the augmentations gave the best results over all the experiments. Furthermore, the ability to generalize was strongly correlated with the difference in shape and size of the different data sets. Further research into the shape and size augmentations are hence encouraged. Implementation for the experiments can be found on Github here or the full link in the foot note [1].

[1] https://github.com/AllaVinner/Augmentation-in-3D-microscopy

Acknowledgements

I would like to thank my supervisors; Mikael Nilsson from Lund university and Anna Kreshuk from EMBL. Their collaboration made this project possible in the first place and have provided me with insight and guidance throughout the process. I would also like to thank everyone in the Kreshuk group at EMBL who have helped me through technical issues, code bugs, and in general provided a positive and constructive atmosphere.

Joel Ottosson
Heidelberg, 12th February 2022

Contents

1 Introduction **5**
- 1.1 Motivation . . . 5
- 1.2 Aim . . . 5

2 Theory **6**
- 2.1 What are images? . . . 6
- 2.2 Image tasks . . . 6
- 2.3 Function approximation with perceptrons . . . 7
- 2.4 Convolutional layers . . . 9
- 2.5 U-net . . . 9
- 2.6 Miscellaneous . . . 10
  - 2.6.1 The Residual Link . . . 10
  - 2.6.2 Batch normalization . . . 11
  - 2.6.3 Sphericity . . . 11

3 Models **12**
- 3.1 PlantSeg . . . 12
  - 3.1.1 The watershed algorithm . . . 13
  - 3.1.2 Multicut . . . 13
- 3.2 Cellpose . . . 13

4 Data **17**
- 4.1 FM . . . 17
- 4.2 Ovules . . . 17
- 4.3 SAM . . . 17
- 4.4 Data set comparison . . . 17

5 Augmentations **20**
- 5.1 Geometrical . . . 20
- 5.2 Contrast . . . 22
- 5.3 Noise addition . . . 22

6 Evaluation **24**
- 6.1 Adapted Rand Error . . . 24
- 6.2 Variation of information . . . 25
- 6.3 Segmentation categorization . . . 26

7 Experiments **28**

8 Results **29**

9 Discussion and further research **36**

A Appendix - Tables **40**

1 Introduction

1.1 Motivation

Deep learning has revolutionized the image analysis field and caused a paradigm shift as deep learning models are the state of the art in classification [3], semantic segmentation[32], and instance segmentation [9] benchmarks. Biological and medical images are no exceptions [18]. One important task in this domain is to be able to instance segment microscopy images of cells, and particularly in 3D. This ability would open a world of new opportunities as orders of magnitude more data would be available for just a percentage of the work. There are many models that attempt to tackle this challenge with varying results [9]. A common issue that arises is that the models tend to be bad at generalizing. They might work well on the data it is trained on, but poorly on similar data it hasn't seen before, e.g. data from another lab. Comparisons of different segmentation models have been made with the ability to generalize in focus [9], and in this paper two models stood out, Plantseg [29] and Cellpose [23]. Both models show the capacity to generalize beyond the training set, but they are far from perfect. A common approach to mitigate the generalization problem in general is to use augmentations [20]. Augmentations are a way of increasing the size of the training set by altering the training data while keeping the target intact. The question is if they can be used to improve the performance of segmentation models in this setting.

1.2 Aim

The aim of this report is to investigate how augmentation can be used to increase the generalizability of models in the domain of 3D cell-instance segmentation. It will be achieved by implementing augmentations on top of current state of the art models and evaluate their ability to generalize. Plantseg and Cellpose introduced in [29] and [23] are the two models chosen for the experiments. The report does not aim at evaluate the models performance against each other, but instead focus on how augmentations may effect different types of models. Furthermore, different data sets will be used for both training and evaluation, with the aim at investigating how augmentation effects the results across data sets.

2 Theory

2.1 What are images?

Most people have an idea of what an image is, but to make the concept useful we must make a more precise definition. A digital image consists of a grid of pixels where each pixel can display an array of colors. We can let the color displayed by a pixel be represented by a vector $y \in \mathbb{R}^f$, where typically f is equal to three for RGB image and one for grey scale images. we can then let a whole image be described as a mapping

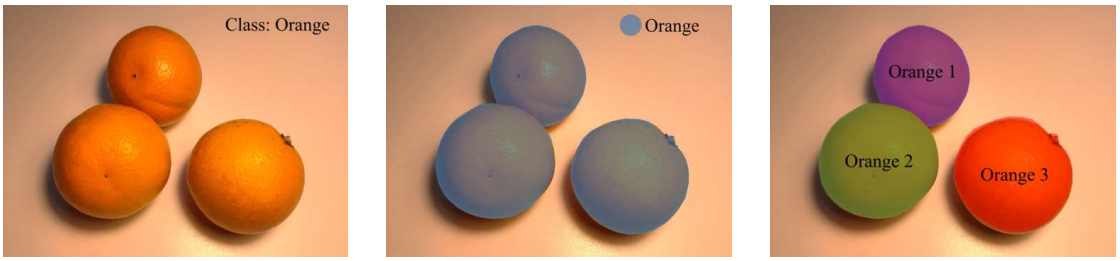
$$\begin{aligned} I : \mathbb{Z}^s &\mapsto \mathbb{R}^f \\ I(\mathbf{x}) &= y. \end{aligned} \tag{1}$$

s can be seen as the number of spatial dimensions while f is the number of feature dimensions. In our case, we will work with 3D grey scale images which means that $s = 3$ and $f = 1$. If a 2D image is made of a grid of pixels, a 3D image can either be seen as a stack of such 2D images or as having an extra dimension with pixels, making each pixel occupy a volume. In the later view, the name voxels are used instead of pixels to specify that we are dealing with a pure 3D image. In the real world, images tend to end, i.e. they have a finite width and height (and possibly depth). However, for mathematical convenience we will extend the domain of the image I by either reflecting the image in the boundaries, or by setting all pixels outside the scope to zero. The output space of the image is said to here be $\mathbb{R}^f$ while in reality this space is limited as well. A common way is to store the values as unsigned 8-bit digits (uint8) which means that each pixel has an integer value between 0 and 255. Furthermore, we will in this report consider uint16 and 32-bit floating point (float32) images. For a more detailed description about the general mathematical properties of images, the reader is referred to the book [22].

2.2 Image tasks

With a precise mathematical definition of images, we can go on to define various image analysis tasks. The most basic question to ask is probably "What is in this image?". This type of task is called image classification and works by assigning each image a label corresponding to the class that the image depicts. This is a classical image analysis task and have been research extensively [10], [31], [12]. The next step is to consider cases where there might be more than one object in an image, and when we want to know exactly where each object is; enter semantic segmentation. In this task, instead of assigning a class to every image, we assign a class to every pixel. Closely related to semantic segmentation is image regression, where instead of assigning a class to each pixel a value is assigned. An example of an image regression task is to predict the depth of each pixel in an image. The next step is instance segmentation which is similar to semantic segmentation in that it classifies each pixel with a class. However, it goes further and separates different instances of the same class. For an example of image classification, semantic segmentation, and instance segmentation see figure 1.

In this report, we will only consider the classes of cell and not-cell, i.e. foreground and background, in 3D which means that a semantic segmentation of such an image can be represented as an image $I_{sem} : \mathbb{Z}^3 \mapsto \mathbb{N}_{\{0,1\}}$ where 0 represents background and 1 foreground. Instance segmentation can similarly be represented as an image $I_{ins} : \mathbb{Z}^3 \mapsto \mathbb{N}_{\{0,\infty\}}$ where 0 represents background and larger numbers represents cells. Voxels with the same label > 0 are considered to belong to the same cell.



(a) (b) (c)

Figure 1: Illustrating image classification (a), semantic segmentation (b), and instance segmentation (c). Classification simply associate the image with one label, in this case *orange*. Semantic segmentation associates each pixel with a label, in this case either *orange* or *not-orange*. Instance segmentation is similar to semantically segmentation but in addition segments the different instances of oranges are separated.

The aim of this report is to investigate models that perform cell instance segmentation on 3D single channelled images, which means that we can formulate the overall problem as follows. Given a set of images $I_i : \mathbb{Z}^3 \mapsto \mathbb{N}_{[0,255]}$ and corresponding ground truths $G_i : \mathbb{Z}^3 \mapsto \mathbb{N}_{[0,\infty]}$ that we want to be able to segment. The goal is to find a function f that minimizes the expected value of some measure of distance between G_i and $f(I_i) = P_i$. As an expression, we are looking to solve

$$\min_f \mathbb{E}_i\big[L(f(I_i), G_i)\big] \tag{2}$$

where L is a distance function, also called the loss, of which there are various options that are discussed more closely in section 6. $\mathbb{E}_i$ is here referring to the expected value over the given set of images and corresponding ground truths.

2.3 Function approximation with perceptrons

Finding the exact function f as a solution to equation 2 is in many cases not possible, however, there are many ways it could be approximated. In a general sense, function approximation aims at approximating some function f by consider a family of functions f_θ, where $\theta_n \in \theta$ is a set of parameter values which specifies f_{θ_n}. f_{θ_n} is then iteratively updated as $\theta_n \to \theta_{n+1}$ where $f_{\theta_{n+1}}$ is, according to equation 2, a better approximation of f than f_{θ_n}. The strategy or algorithm by which θ_n is updates is called the learning algorithm [25].

A popular choice of f_θ is the perceptron P_θ which works by sending the input $x_i \in \mathbb{R}^{n_x}$ through a linear transform followed by a non-linear function called the activation function producing a n_y dimensional output. The transformation can be summarized as

$$y_i = P_{\theta_n}(x_i) = \sigma\big(W_n x_i + b_n\big), \quad \theta_n = \{W_n, b_n\} \tag{3}$$

where W and b are n_x by n_y and 1 by n_y matrices respectively and make up the parameters of the perceptron. There are many different types of activation functions, but in this report, we are most interested in two of them, the rectified linear unit (ReLU) and the logistic function. ReLU

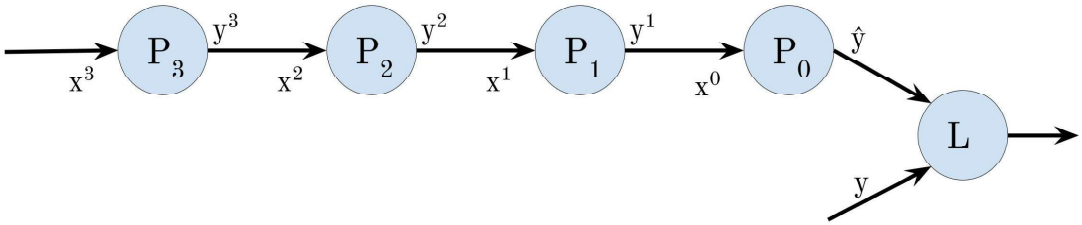


Figure 2: A visualization of an MLP with four perceptron layers. The output $\hat{y}$ is then depicted to be passed through a loss function together with the ground truth y. Here, the input to each layer is denoted with a x^l and the output with a y^l. Note that $y^{l+1} = x^l$.

is defined as $\sigma_{\text{ReLU}}(x) = \max(0, x)$ and the logistic function as $\sigma_{\log}(x) = (1 + \exp(-x))^{-1}$, where both are applied element wise if the input is not a scalar.

If we now let the output of a perceptron be the input to another perceptron, and so on, we get a structure called a *Multi Layer Perceptron* (MLP). The MLP has been proven to be able to approximate any function $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ within an arbitrary small error given that the depth is large enough [2] [13]. For the MLP, the set of parameters θ_n is the joint set of weights W and biases b from every layer. An illustration of the MLP can be seen in figure 2.

With a chosen family of functions, the next step is to define a learning algorithm. With the goal of minimizing L from equation 2, we can look at the partial derivative of L with regard to one of the parameters $\omega_k \in \theta_n$. $\frac{\partial L}{\partial \omega_k}$ expresses how L responds to small changes of ω_k, which means that if we know the derivative, we can move ω_k in a direction which makes L smaller. The question is then how we calculate $\frac{\partial L}{\partial \omega_k}$.

If we take figure 2 as inspiration and start with the perceptron closest to the loss function. The derivative with regards to one of the weights in the perceptron can be calculated using the chain rule as

$$\frac{\partial L(y, \hat{y})}{\partial \omega_k} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \omega_k}. \tag{4}$$

Since $\hat{y} = \sigma(Wx + b)$ where x is the output of the previous layer and ω_k is either part of W or b, we can find the partial derivative we are looking for as long as σ is differentiable [3]. Now lets consider a weight ω_k^l in layer l. Let x^l be the input to the layer and y^l the output, note that $y^{l+1} = x^l$. Then the derivative with regards to ω_k^l can be calculated as

$$\frac{\partial L}{\partial \omega_k} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial y^1} \cdots \frac{\partial y^{l-1}}{\partial \hat{y}^l} \frac{\partial y^l}{\partial \omega_k^l}. \tag{5}$$

The process of calculating the derivatives by using the chain rule back through the network is

[2] The theorem is called the universal approximation theorem and us usually stated regarding the width of the network, but a dual version of the theorem has been proven for the ReLU-activation which states that as long as the width of the network is $n + 4$, where n is the number of input dimensions, any function can be approximated given a deep enough network [13]

[3] ReLU is technically not differentiable in $x = 0$, however we can set it as 0, 1, or 1/2 in the few cases when it happens.

called back propagation. It is important to not that we are using vector calculus in equation 4 and 5 as both the nominator and denominator can be vectors in the derivatives. For the general case where x and y are vectors of length n_x and n_y. $\partial y/\partial x$ gives a matrix with shape n_y by n_x, where index element i, j is equal to $\partial y_i/\partial x_j$. Furthermore, the multiplications between the derivatives are standard inner products. Given the set of partial derivatives we can go about updating the parameters. However, the set of partial derivatives only gives us the direction of which the parameters should be updated, not the magnitude. This task is handled by a so-called optimizer that takes in the current set of partial derivatives, and possibly previous derivatives, and combines them to calculate the magnitude of the update. There are many optimizers to choose from and the reader is directed to [19] for an overview. The ones touched on in this report are the RAdam and SGD.

2.4 Convolutional layers

Even if the MLP is proven to be able to approximate any function, it might do so very inefficiently by requiring a large number of parameters, especially as the number of input and output dimensions grow [11]. As we are using images, the number of input and output dimensions are the number of voxels. This is a very high dimensional space that would make an MLP extremely inefficient. An approach to deal with this inefficiency is a so-called convolutional layer. The input and output shape of a convolutional layer applied on 3D images are (F_{in}, Z, Y, X) and a (F_{out}, Z, Y, X) respectively[4]. The F_{in} and F_{out} are referred to the feature dimension and is one for the first layer as we are dealing with grey scale images. Z, Y, and X are the spatial dimensions of the image. The convolutional layer consists of F_{out} number of kernels of shape (F_{in}, K_z, K_x, K_y) and an additional bias for each kernel. $K_{z,y,x}$ is called the kernel dimensions and are in general $\ll Z, Y, X$. To compute the output at position (f, z, y, x) we choose kernel f, place the centre of the kernel at position (z, y, x) in the input image, multiply each value in the kernel with the overlapping value in the input image, sum up all the products, and finally pass the sum through an activation function. As the convolutional layer only performs element wise multiplication, summation, and a final pass through an activation function, just like a perceptron, the partial derivatives with respect to the kernel values and biases can be computed via back propagation in a similar way to MLP. For a more in-depth explanation of convolutional layers see [4] and for an overview of how to calculate the gradient see [6].

2.5 U-net

Typical convolutional networks apply the layers in a sequence, an approach that has been successful in image classification [10], but less so in semantic segmentation. For these problems a new architecture has been created, the U-net [18]. U-net is a neural network architecture introduced in [18] that consists solely of convolutional layers, but which are not linked in a single sequence. Instead, we first have a so called contracting path that looks like a typical, sequential convolutional net. In this case we have four convolutional blocks where each block is followed by a pooling layer that reduces each spatial dimension by a factor of two. After the last pooling layer, the data is passed through yet another block of convolutional layers before entering the expanding path. The expanding path also consists of a series of four convolutional blocks but where each block is preceded by an up-convolutional layer that increases the spatial dimensions by a factor of two. In addition,

[4]Convolutional layers may change the spatial dimensions as well, however, this is not done in the investigated models and would hence only add complexity. For further reading see [4].

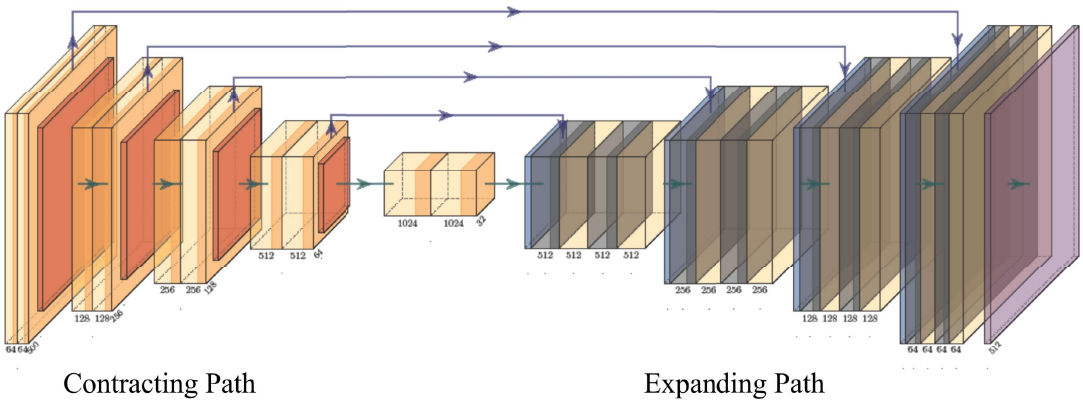


Figure 3: Architecture of a U-net as presented in [18]. The number on top of each layer refer to the number of feature dimensions present. Each yellow block represents a convolutional layer, the red blocks are pooling layers, and the green blocks are upconvoutional layers.

the data is concatenated along the feature dimension after each up-sampling with the output of the convolutional block from the contracting path at the same sampling level. Conceptually, this means that we combine more local information (from contracting path) with more global information (from expanding path). Furthermore, the feature dimensions of the data are increased to 64 at the first sampling layer and then doubled after every down sampling, the process is then reversed in the expanding path. In figure 3, we can see a visualization of the U-net architecture presented in [18]. Both Plantseg and Cellpose make use of U-nets as the main part of their models.

2.6 Miscellaneous

In this section, we introduce topics which did not fit in other parts of the theory but that are still relevant to the report.

2.6.1 The Residual Link

Numerous approaches have been proposed to make network learn more efficiently. One such approach, with great success, is the residual link introduced in [5]. Given a layer, or block of layers, denoted as L that takes an input x and outputs $y = L(x)$. The residual link of such a layer is defined as L_R where $L_R(x) = L(x) + x$. The main idea behind this link, is that if x is already the desired output, L_R simply needs to push all its weights to zero which they hypothesis is an easier task than to make the weights in L find a specific configuration to creates the unit function. Of course, it is rarely the case that x is already the target output, however, it has been shown experimentally that such layers are easier to optimize. The residual link was first introduced for convolutional nets [5] but was later extended to U-nets [33].

2.6.2 Batch normalization

Another type of layer which has been shown to improve the accuracy and speed of training is batch normalization layers [2]. When passing data through a neural network, each input is normally not passed by itself, but stacked into batches. This means that the shape of the input array is (B, F, Z, Y, X) where B is the number of samples in the batch. For an input $I(b, f, z, y, z)$ the output $O(b, f, z, y, z)$ is given as

$$O(b, f, z, y, x) = \gamma_f \frac{I(b, f, z, y, x) - \mu_f}{\sqrt{\sigma_f^2 - \epsilon}} + b_f. \tag{6}$$

Here, μ_f and σ_f is the mean and standard deviation calculated for each feature channel over the b, z, y, and x dimensions of the batch. ϵ is a small value to keep the denominator from being zero. γ_f and b_f are parameters that are learned through back propagation for each feature channel. In this report batch normalization is used in the convolutional blocks for both the Plantseg and Cellpose model.

2.6.3 Sphericity

Sphericity is a measure of how similar a 3D object is to a sphere. It is defined as

$$S = \frac{\pi^{1/3}(6V)^{2/3}}{A}$$

where V is the volume of the object and A is the area. The measure range between zero and one, and only reaches one for a perfect sphere [27]. In this project, the sphericity of each cell is computed. This is done by approximating the surface with triangles and then computing the area and volume. The implementation used the `reconstruct surface` function from the `pyvista` library [24]. For full implementation see the Github repository [7].

3 Models

3.1 PlantSeg

PlantSeg is an end-to-end instance segmentation method designed for microscopy. The method is split into two sub-tasks. First, a 3D semantic segmentation is performed on the image that predicts the borders of each cell. In principle, this step could be executed by any kind of function which outputs a probability map over the boundaries, however in the original paper [29] a U-net was used. The boundary map is then passed to the Multicut algorithm [8], a graph partitioning algorithm, that performs the final instance segmentation.

The 3D images used are often too large in terms of memory to pass into the network. To deal with this issue the image can be divided into patches which are individually predicted and then stitched together by the model. PlantSeg supports varies patch sizes as well as batching and the paper suggest a patch shape of $(100, 100, 80)$ voxels and a batch size of four. The input is then standardized by subtracting the mean of the whole data set and dividing by the standard deviation. A U-net was used as the main architecture of the model where each convolutional block consists of a batch normalization layer and a convolutional layer with a ReLU activation function. The convolutional layers have a kernel of spatial shape $(K_z, K_y, K_z) = (3, 3, 3)$, stride of one, and a mirror padding of size one to maintain the spatial shape of the data[5]. The last convolutional block uses a sigmoid function as activation function to create a patch array with values between zero and one. The patches are then stitched together to form the complete boundary probability map of shape $(1, Z, Y, X)$. The boundary predictions are then sent to the instance segmentation step. Here, the boundary prediction is thresholded at $v = 0.4$ before a distance transform is computed. The distance map is then filtered with a Gaussian filter before applying the watershed algorithm to create a set of super pixels. Super pixels are an instance segmentation of the image, but where the segments are not considered to be cells. To get the final cell segmentation the super pixels are merged by the Multicut algorithm [8]. Multicut could in principle be run directly on the distance map, but due to time complexity this is not feasible [29]. A schematic visualization of the Plantseg workflow can be seen in figure 4.

During the training of Plantseg, only the first step, the semantic segmentation, is considered. To create the ground truths, that are compared to the network predictions, the cell boundaries are found in the ground truth instance segmentation by passing it through the `sklearn`'s[6] `find_boundary` function. The boundaries are then blurred with a Gaussian filter to thicken the edges before a threshold of $v = 0.4$ is applied to convert the image in to binary. An example of the created ground truth along with the original segmentation can be found in figure 7 b) and a) respectively. Two of the data samples in the training data is taken out and used for validation while the rest of the data is used for training. Given the ground truth and the prediction, the loss is calculated by the binary cross-entropy loss and back propagated through the network. To update the parameters of the network an ADAM optimizer was used with $\beta_1 = 0.9$, $\beta_2 = 0.9999$ as parameter settings with an initial learning rate of $lr = 0.0002$. The learning rate was then updated with a factor of 0.2 when the model stopped improving on the validation set 30 times in a row. The network stops the training process when the learning rate goes below $\eta = 10^{-6}$.

[5] For discussion about stride and padding in convolutional layers see [4].

[6] Sci-kit learn is a python library with many functions relevant for machine learning tasks. For more information, see their home page https://scikit-learn.org/stable/.

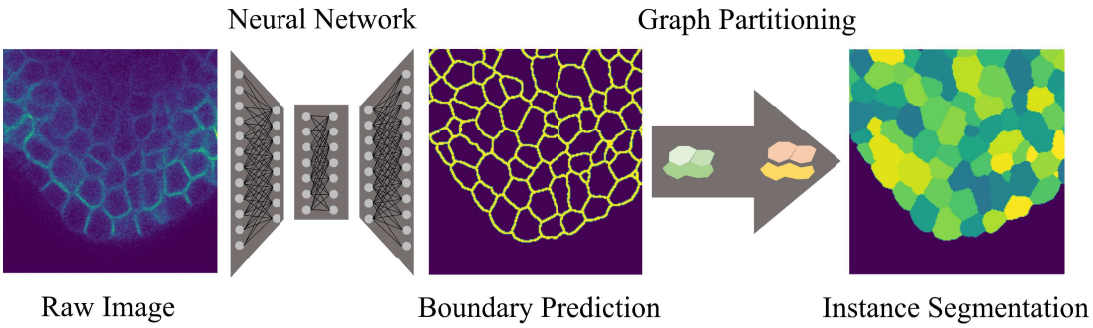


Figure 4: Visualization of the Plantseg's two step instance segmentation process. First the raw input images are passed through a neural network to create a boundary prediction. The boundary prediction is the passed through a graph partitioning algorithm which in this project is Multicut to create the final instance segmentation.

3.1.1 The watershed algorithm

The watershed algorithm is a type of image segmentation algorithms that are based on mathematical morphology. Plantseg uses this method to convert a cell-boundary probability map into a segmentation consisting of super-pixels. The implementation used are outlined in [17] as the *Watershed definition by topological distance*. The algorithm takes inspiration from geography and views a grey-scale image as a landscape where a higher grey-scale value equates to a higher altitude. The minima of each valley is then imagined to function as a source of water which slowly fills up the landscape. When water from two different sources meet, a water shed is built to keep the water separated. When the whole landscape is flooded, we are left with a set of watersheds which defines the outlines of the resulting segmentation. Since each source will give one segmentation label, the method tends to over-segment due to large number of local minima [17].

3.1.2 Multicut

The watershed segmentation is the fed into the Multicut algorithm [8] together with the boundary probability map. Mutlicut is a general graph partitioning algorithm which views each super pixel as a node, the bordering super pixels as neighbours, and the probability map values between the super-pixels are set to be the edge values. The algorithm then finds the cuts of minimum costs which decides which super pixels are to be merged. For full details and implementation see [8].

3.2 Cellpose

Cellpose is an end-to-end instance segmentation tool specialized for cell and nuclei segmentation [23]. The design is fundamentally for 2D images but can, and is in this report, extended to 3D. Like PlantSeg, Cellpose consists of two parts. First a semantic segmentation and an image regression is performed. The segmentation predicts a cell probability (not to confuse with cell boundary) while the regression predicts a vector field over the image. The segmentation and vector field are then

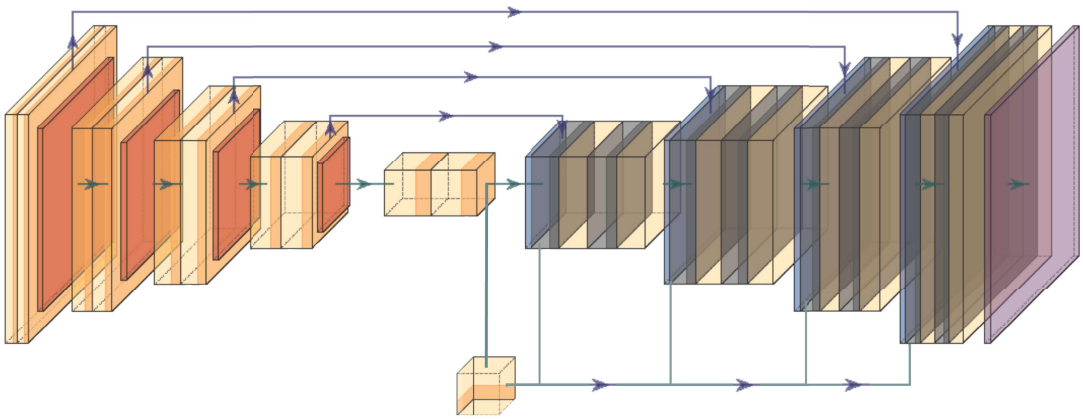


Figure 5: Architecture of the Cellpose neural network. Note the similarities to the original U-net image 3, and also the difference in form of the style being added to each level in the expanding path.

passed on to a second step which performs the final instance segmentation.

Cellpose requires the input images to be 2D, which means that the 3D images are sliced into 2D slices before being pre-processed by linearly scaling such that the 1 percentile maps to zero and the 99 percentile maps to 1. The images are then cropped to a set size of $(224, 244)$ before being passed through a residual U-net. The convolutional blocks are made up of a convolutional layer, a batch normalization layer, and finally a ReLU activation layer, similar to that of Plantseg, but in another order. The final convolutional block outputs a $(3, 224, 244)$ array where the first channel is passed through a sigmoid activation function to map the values between zero and one, while the other two channels are left without an activation. Cellpose uses the U-net architecture as its base network, however, it adds some alterations to the original structure. First, the links between the contracting and the expanding paths are not concatenated but simply added, this is to reduce the number of parameters. Secondly, a global average pooling is performed on the lowest level convolutional output of the contracting path. This creates a '*style*' vector which is linearly projected and concatenated to each level of the expanding path. For an overall illustration of the architecture see figure 5. Cellpose also make use of ensambling as it trains four identical models at once and averaging out their results. Furthermore, Cellpose performs test time augmentations in the form of resizing. This means that each model performs multiple augmentations of the data and process each one. The predicted result is then averaged out. Finally, Cellpose makes a final region of interest quality estimation of the instance segmentation to remove any which are deemed to be bad.

The outputs are then stitched together to create a complete probability map and vector field of this 2D slice. The complete 2D predictions are made for all the slices to create a 3D prediction. This is repeated with the z-, y-, and x-axis as the slicing axis. The three results are averaged out to create the final 3D cell probability map and vector field which is then passed to the instance segmentation step.

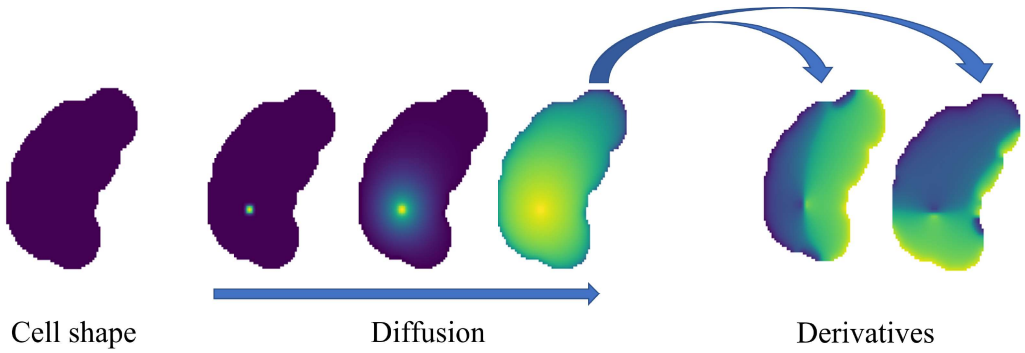


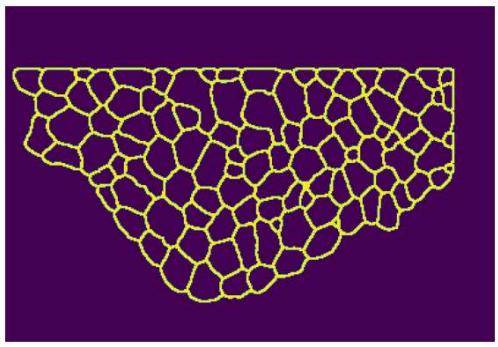
Figure 6: A cell instance in a raw image is converted into a binary map of zeros with the center voxel set to one. The center voxel acts as a heat source and fills the cell instance with a scalar field. The spatial gradients are then calculated to create the final vector field.

In the instance segmentation step the cell probability map is thresholded at $v = 0.5$ to create a binary image. The predicted foreground voxels are considered to be part of a cell. The position of each cell-voxel is then iteratively updated in direction of the vector field at their current position. This process repeats until all the voxels has converged to some set of points. The voxels which have converged to the same point are considered to belong to the same cell which then defines the final instance segmentation.

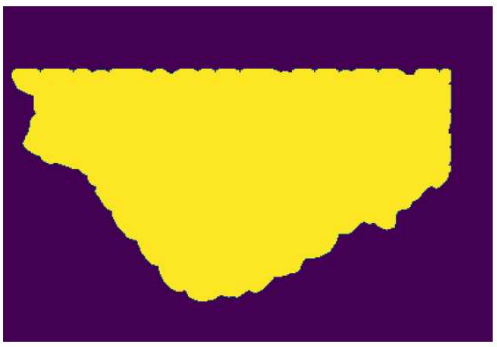
It is only the first step, the semantic segmentation, which is being trained. The ground truth cell probability map is created by simply setting the background of the label to zero and the remaining voxels to one. The creation of the vector field is a bit more involved with the core idea taken from heat diffusion. Given a cell instance, all voxel values in the image are initially set to zero. A voxel in the center of the cell is set to one and will work as the heat source. All voxels which do not correspond to the instance will work as a sink and is permanently kept at zero. Then the heat diffusion starts. At each iteration, each voxel, except for the source and sinks, is set to the average value of its eight neighbours. The simulation will then play out over 200 iterations after which the gradient is calculated in each voxel which will define the ground truth vector field. An illustration over the process can be found in figure 6 and an example of the three channelled target can be found in figure 7 c), d) and e).

Given the prediction and the ground truth each channel will produce a loss which is linearly combined to produce a total loss which is back propagated through the network. The semantic segmentation loss is calculated via binary cross-entropy and the two vector outputs are calculated using the L_2 norm. A stochastic gradient with momentum was used as the optimizer with a weight decay of $\omega = 10^{-5}$, and momentum at $m = 0.9$. The learning rate was linearly increased during the first 10 epochs from 0 to 0.2 and was then kept steady until the 400th epoch where it was halved every 10th epoch for another 100 epochs.

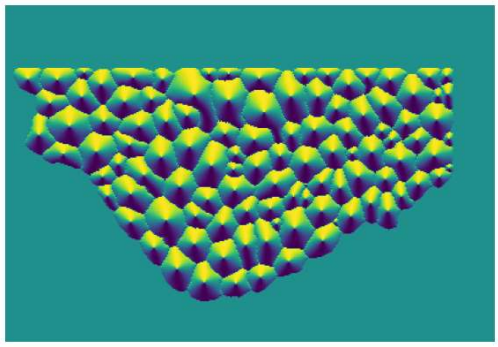
(a) Original labels.



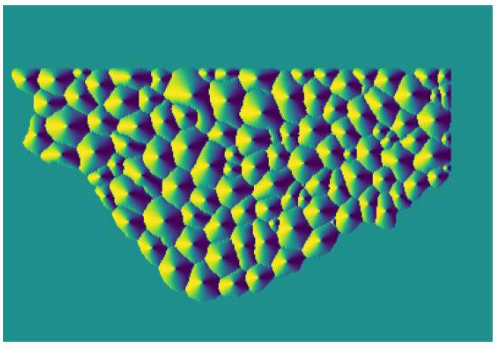
(b) Cell boundary used by Plantseg.



(c) Cell probability used by Cellpose.



(d) Y component of vector field used by Cellpose.



(e) X component of vector field used by Cellpose.

Figure 7: Labels along with the created targets used by Plantseg and Cellpose.

4 Data

The data used in this report consists of raw 3D images and their labelled ground truths. The images are of different shapes but are all single channelled which means that they can be represented by a (Z, Y, X) array. The raw images are stored and processed as unsigned 8-bit integers, while the labels are stored as unsigned 16-bit integers. The integer in the labels corresponds to different instances in the image, or more precisely cells and background. The background is by convention set to one while the cells are index from two and up. Three different data sets have been selected for this project, that depict different parts of the plant *Arabidopsis thaliana*. The data sets was chosen as they were used either in [9] or [29], and are considered to be different enough to make it difficult for the models to generalize, but similar enough to not leave the task impossible.

4.1 FM

The data set consists of ten 3D confocal images floral meristems (FM) from Arabidopsis Thaliana [16]. The data set was chosen since it has been used previously in [9] to compare different 3D image segmentation algorithms. The data set consists of images from six different plants where each plant was photographed every four hours for a period up to 72 hours. As was done in [9], ten images were selected from two of the plants, specimen 1 and 6. Six images were taken from specimen 1 at time points 0h, 24h, 32h, 72h, 120h, and 132h, and the remaining four images were taken from specimen 6 at 26h, 44h, 56h, and 69h. The image from specimen 1 taken at 132h can be seen in figure 8 a). Note how it appears to be cells in the background which are not part of the labelling. This is due to the fact that these cells are not part of the meristem of the *Arabidopsis Thaliana*.

4.2 Ovules

The data sets consists of 24 3D confocal images and annotation of the ovules of the species *Arabidopsis Thaliana* [26]. The specimens are taken from all stages of development and the data set has previously been used in [29].

4.3 SAM

The data set consists of twelve 3D confocal membrane image and their annotated segmentation and was introduced in [28]. Furthermore, it was used in the 3D-segmentation comparison paper [9] and is hence considered an appropriate data set for this project. The images depict six Arabidopsis thaliana shoot apical meristems (SAM) where the image of each specimen was selected at two different time points. Originally the SAM data set consists of 124 images, taken at different time points from six different specimens. To reduces the data set size and to make it more compatible with the other data sets, two images were selected from each specimen to create the final twelve images.

4.4 Data set comparison

The images in all three data sets are taken from *Arabidopsis Thaliana* plants but depict different parts of the plant and at different times in the development. Figure 8 shows samples from each of the three data sets where we can see quite clearly how they differ. Note also how there exist cells in the images which are not segmented. These are cells from the *Arabidopsis Thaliana* which are

Table 1: Meta data over the data sets used. Shapes, number of cells, and cell sizes are measured in voxels, while sphericity and background are measured as percentages. The numbers inside parenthesis are the standard deviation of the corresponding value over the data set.

Quantity	FM	Ovules	SAM
# images	10	24	12
Shape min	(305, 274, 400)	(354, 592, 391)	(149,512, 512)
Shape max	(634, 700, 700)	(315,1020,1020)	(312,512,512)
# cells	1241 ±915	1114 ±496	914 ±300
Cell size	27814 ±11200	42515 ±10314	19255 ±4200
Sphericity	83 ±8	54 ±23	86 ±6
Background	67 ±13	83 ±6	73 ±6

not part of the meristem, ovules, and shoots respectively. In the evaluation, the background will be ignored and hence the model will not be penalised for predicting, or not predicting, these cells.

In table 1 we can see basic meta data over the data sets. The sizes of the images vary more within each data set in comparison to in between them. The Ovules data set can be said to have slightly larger images then the other two. Furthermore, the FM data set stand out as the only one where all three dimensions are of similar size, while in the other two sets, the first dimension is considerably smaller. The number of cells in each image is about the same over the data sets, however there is more variation in the FM data set compared to SAM. The cell sizes differ between the data sets where the average cell in SAM is more the half the size of the average cell in Ovules. By the sphericity we can see that the FM and SAM cells are similar with high values while, the Ovules' value is smaller. The variation in FM and SAM are furthermore smaller, suggesting that the cells here are similar while they differ more with in the Ovules data set. All the images in the data sets are scarcely populated by cells as the background column shows. FM and SAM fill up the images with about the same density while Ovules are a bit emptier.

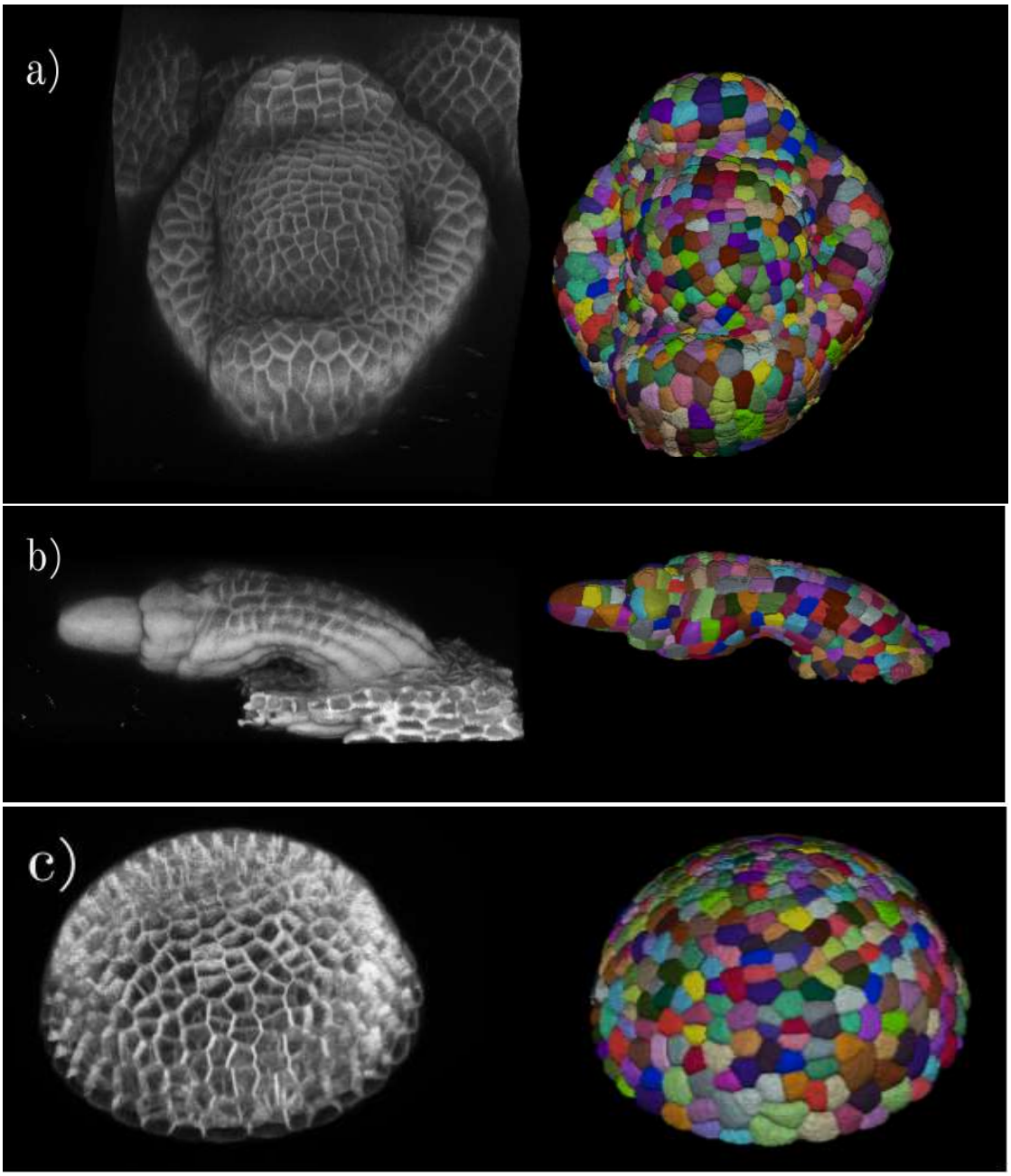


Figure 8: Samples from the three data sets FM, Ovules, and SAM, a), b), and c) respectively. The images are visualized using the 3D visualization tool napari [21]. The left side of the images shows the raw image while the right shows the hand annotated segmentation. The colors in the segmentation have no intrinsic meaning other that voxels colored with the same color are said to belong to the same cell.

5 Augmentations

The goal of the models is to create a function f which takes in a 3D image I and outputs a (correct) instance segmentation of that image, $S = f(I)$. More precisely we want to find f according to equation 2. I, however, cannot be any image, but is sampled from some set of images D, where D is the set of all images that we would like to be able to segment. In this project we are only interested in microscopy images of cells, a fact that drastically reduces our domain size. Unfortunately, we do not have access to all of D, we only have access to a subset $D_T \subset D$ which we will refer to as the training set. We can now use D_T as an approximation for D to be able to in turn approximate f. However, two main issues arise. First, D_T is very limited. In our data sets we have at most 24 images to use as D_T, where the rest must be inferred by the model. Secondly, D_T will most likely be sampled from D with some bias. This bias may come from how the images are taken, or from how the samples which are photographed are chosen. The data sets in this project differ in both at what time in the development the images were taken and what part of the plant was photographed. This bias may lead to a model which handles input from D_T well but is almost useless on images from $D \setminus D_T$. For this project, the hypothesis is that the models will work well on the data it is trained on but suffer on the two data sets which it hasn't seen before. One way to address these two issues is to apply data augmentations. Data augmentations work by altering the input to expand our training set in such a way that the target can still be inferred. This means that we artificially add data points to D_T which makes it harder for the network to over fit on the training data and in turn perform better on the evaluation data.

The augmentations used in this report are divided into three categories, geometrical, contrast, and noise. For the experiments, two categories are added containing non or all the augmentation categories. They are referred to as unit and combine respectively. For examples of the different categories see figure 10 and for the specific parameters used in each augmentation see table 2. For other augmentations approaches and a more comprehensive list, the reader is referred to the survey [20].

5.1 Geometrical

Geometrical augmentations alter the input to an image I, rather than on the output, i.e. voxel values. Referring back to equation 1, a geometrical augmentation $\mathcal{A}_G$ alters the input as

$$y = I\big(\mathcal{A}_G(\mathbf{x})\big), \quad \mathbf{x}' = \mathcal{A}_G(\mathbf{x}). \tag{7}$$

Geometrical augmentations change the expected segmentation which means that any geometrical augmentation performed on the input needs to be followed by a corresponding augmentation on the target. This target augmentation consists of both the original geometrical augmentation and a possible value augmentation. The value augmentation is added when the values in the target are related to the under-laying grid. In this report that applies to two of the channels in the target of the Cellpose model. The two channels represent a vector field in 2D and should stay the same in relation to the under-laying grid. Figure 9 illustrates this situation. The figure shows the y-components of the vector field and should point towards the centre of the cell. In the figure, yellow regions represent a downwards facing vectors while dark blue regions represent upwards facing vectors. In image (a), we can clearly see that the yellow regions are on the top part of the cells while the dark blue ones are at the bottom. This is not true for the transformed image (b). To correct this, a value transform is applied, which in this case simply means to negate the values in

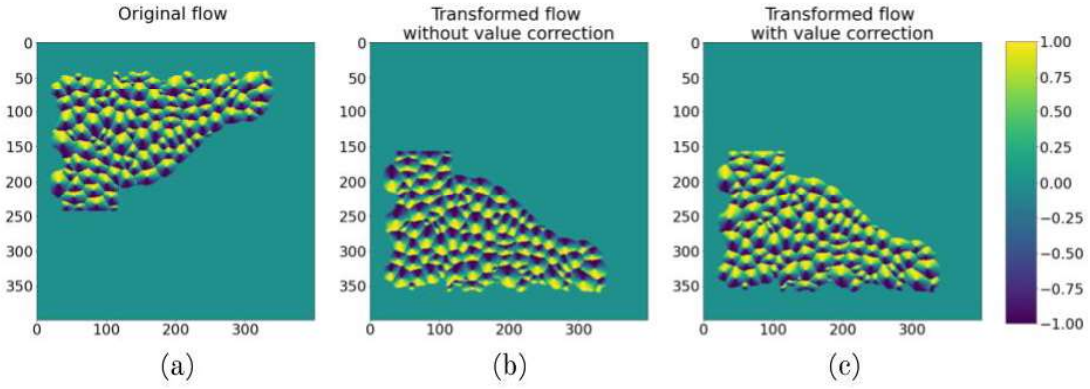


Figure 9: Images of the second channel of Cellpose's targets. The second channel represents the y-component of a flow vector. Image (a) shows the flow in the original, non-augmented case. (b) shows the flow after a geometrical augmentation in the form of a reflection in the x-axis. Image (c) shows the same image after the flip-augmentations with an additional value correction. Notice how the yellow part of each cell is in the top in both (a) and (c) while on the bottom in (b).

the images which leaves us with image (c) where, again, the yellow regions are in the top part of the cells.

Bellow follows a list of the geometrical augmentations used including the corresponding value transforms and augmentations specific parameters. L_x and L_y are used as notations for the x- and y-components of the vectors in the Cellpose targets.

Random Flip

A random flip reflects the image along one of the image axes with a given probability. Each orientation of the image is assumed to be equally likely and therefore the axis probability is set to 0.5. For all flipped axis, the corresponding channel in the Cellpose targets are negated.

Random 90° rotation

The image is rotated by 0°, 90°, 180°, or 270° chosen at random around an axis. Since the Cellpose training images are 2D, only rotations around the z-axis are considered. The flow is changed according to

$$\begin{pmatrix} L'_y \\ L'_x \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^k \begin{pmatrix} L_y \\ L_x \end{pmatrix}$$

where $90k$ is the number of the degrees in the rotation.

Elastic deformation

An image I can according to equation 1 be viewed as a function which takes in a grid position and outputs the value of that grid position. Let I_C be a extension of I where also positions inbetween grid points are accepted, the output will then be a linear combination of the surrounding grid

points. Three values (ϵ_x, ϵ_y, and ϵ_z) are then sampled from a normal distribution for every voxel which will define the distortion. High frequency components are then removed from the distortion by a Gaussian filter with a set standard deviation σ before being scaled by a factor α. The steps can be summarized as

$$\begin{aligned}\Delta_x &= \alpha g_\sigma \star \epsilon_x \\ \Delta_y &= \alpha g_\sigma \star \epsilon_y \\ \Delta_z &= \alpha g_\sigma \star \epsilon_z\end{aligned}$$

where g_σ is a Gaussian with a standard deviation of σ and $\star$ is a standard convolution. In the code the Scipy's gaussian filter was used.

With the smoothed and scaled distortions in place the new augmented values of each gridpoint $\mathbf{x} = (x, y, z)$ is set to

$$y = I\big(\mathcal{A}_D(\mathbf{x})\big) = I(x + \Delta_x, y + \Delta_y, z + \Delta_z).$$

In the implementation the σ is set to 50 and α to 2000, values which were set after a qualitative assessment. The deformation is applied to the input data, which means that the flow values in the Cellpose targets need to be altered. This alteration may be done by approximate the Jacobian at each voxel and rotate the vectors accordingly. However, due to time constraint, this was not implemented.

5.2 Contrast

Contrast augmentation changes the values of each voxel by considering all the voxel values in the image. We denote the augmentations as $\mathcal{A}_V$ and can be expressed as

$$y = \mathcal{A}_V\big(I(\mathbf{x}), I\big), \tag{8}$$

Since no augmentation is done on the input the target does not need to be altered.

Random Contrast

Update each voxel value by the equation below where v is the voxel value and μ the mean voxel value of the image. α is a sampled parameter which determine the strength of the contrast change. A value of 1 keeps the image as is, a value < 1 makes the image darker, and a value > 1 makes the image brighter. α is sampled uniformly from the range $(0.5, 1.5)$, which is the default range suggested in [30]. As an expression, the transform can be described as

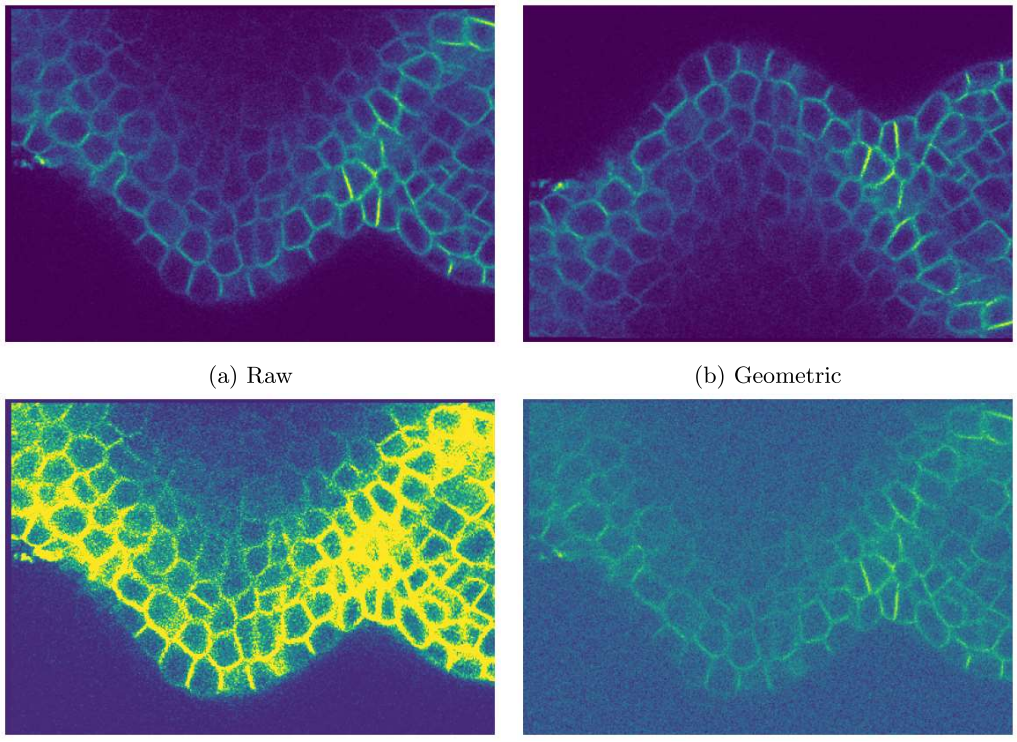
$$y = \mathcal{A}_V\big(I(\mathbf{x}), I\big) = \mu_I + \alpha\big(I(\mathbf{x}) - \mu_I\big).$$

5.3 Noise addition

Noise augmentation changes the value of each voxel, but in contrast to value augmentations, they do consider any properties regarding the whole image, but rather only uses the individual voxels. As an equation, they can be expressed as

$$y = \mathcal{A}_N\big(I(\mathbf{x})\big). \tag{9}$$

These augmentations do not have any effect on the target, which hence will remain as is.



(a) Raw (b) Geometric

(c) Contrast (d) Noise

Figure 10: Raw image along with the augmented versions according to each of the three augmentation categories.

Gaussian noise

For each voxel a distortion ϵ is sampled from a zero-centred Gaussian with a standard deviation of σ. This distortion is then added to the voxel value. In the implementation σ is sampled uniformly between 0 and 1, values set after a qualitative assessment. As an expression, the augmentation can be defined as

$$y = \mathcal{A}_N\big(I(\mathbf{x})\big) = I(\mathbf{x}) + \epsilon, \quad \epsilon \sim N(0, \sigma).$$

Poisson noise

Poisson noise adds a distortion ϵ to the value of each voxel where the distortion is drawn from a Poisson distribution. λ is drawn from a uniform distribution between zeros and one for every new input. The range was determined since it was suggested as the default settings in [29]. The augmentation can be expresses as

$$y = \mathcal{A}_N\big(I(\mathbf{x})\big) = I(\mathbf{x}) + \epsilon, \quad \epsilon \sim \text{Pois}(\lambda).$$

Table 2: Specification over the different parameters each augmentation uses.

Type	Augmentation	Parameter	Default value
Geometric	Flip	Execution probability	1.0
		Axis probability	0.5
	Rotation	Execution probability	1.0
	Elastic	Execution probability	0.1
		α	2000
		σ	50
Contrast	Contrast	Execution probability	0.1
		α-range	(0.5 - 1.5)
Noise	Gaussian	Execution probability	0.1
		σ-range	(0 - 1)
	Poisson	Execution probability	0.1
		λ-range	(0-1)

6 Evaluation

In this report we will investigate the performance of different instance segmentation algorithms, and to do so we need to have a way to measure this performance. Since the instance segmentation in our case is over a single class, we can view the problem as a clustering or partitioning problem.

A partitioning of a set D is defined as a set of subsets $C = \{C_k\}_{k=1}^{K}$ where the subsets are pairwise disjoint and the union over all the subsets gives the original set D [14]. In our case each subset represents the pixels which belong to, or are predicted to belong to, the same instance of a cell. We then want to find a distance metric between the ground truth partition (G) and a predicted partition (P). Furthermore, we will not consider the background to be part of the ground truth partitioning. This means that we will only consider the pixels in the ground truth and predictions which are labelled as cells in the ground truth. The reason for this is that the labelling of the ground truth is non-complete. Looking at the lower image in figure 8, we can clearly see that there are cells in the back of the image which are not labelled, and we do not want to punish an algorithm which segments these cells.

6.1 Adapted Rand Error

The adapted random error is a measure of the quality of a segmentation proposed in [15]. The measure is calculated as one minus the harmonic mean between a precision and recall. The former is defined as the probability that two voxels belong to the same cluster given that they are predicted to do so. The later is conversely defined as the probability that two voxels are predicted to belong to the same cluster given that they are in the ground truth. A more detailed explanation will be given below. The measure was implemented using *skimage*'s built-in function which is defined in [1]. Skimage refers to the two probabilities as precision and recall which we will be doing in this report as well, however, in [1], they are seen as measures of over and under segmentation respectively.

The measure is calculated by first assuming that we are given a ground truth segmentation $G = \{G_1, ..., G_{N_G}\}$ and a predicted segmentation $P = \{P_1, ..., P_{N_P}\}$. Then define q_{ij} as the probability that a voxel picked at random belongs to G_i and P_j, i.e. $|G_i \cap P_j|/N$. The probability that a voxel belongs to cluster G_i can then be expressed as the marginal probability $s_i = \sum_j q_{ij}$, and

conversely $t_j = \sum_i q_{ij}$ for the probability that a voxel belongs to P_j. If we now pick two voxels independently at random, the probability that both of them belongs to G_i is s_i^2. Taking the sum over all different clusters then gives us the probability that two randomly chosen voxels belongs to the same cluster, $P(S_G) = \sum_i s_i^2$, here S_G is the event that the two voxels belong to the same cluster in G. Analogously, the probability that two randomly chosen voxels belongs to the same cluster in P is $P(S_P) = \sum_j t_j^2$. We can now define the precision and recall as:

$$\text{precision} = \frac{P(S_G \cap S_P)}{P(S_P)} = \frac{\sum_{ij} q_{ij}^2}{\sum_j t_j^2} \tag{10}$$

$$\text{recall} = \frac{P(S_G \cap S_P)}{P(S_G)} = \frac{\sum_{ij} q_{ij}^2}{\sum_i s_i^2}. \tag{11}$$

The joint probability $P(S_G \cap S_P)$ can be calculated as $\sum_{ij} q_{ij}^2$, as the q_{ij}^2 is the probability that two voxels belongs to G_i and P_j and summing over all combinations of i and j gives the probability that the two voxels is in the same cluster in G (S_G) and in the same cluster in P (S_P). Finally, we will combine the precision and recall with the harmonic mean to get the final random adapted rand error:

$$L_{ARand} = 1 - \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}. \tag{12}$$

6.2 Variation of information

Variation of information is a true metric in a partition space and measures the information gained and lost by going from partitioning G to P. It is defined as

$$VI(G, P) = H(P|G) + H(G|P) \tag{13}$$

where the first term measures the information gained by using P to describe G while the second term measures the information lost in the same situation [14]. These properties makes $H(P|G)$ and $H(G|P)$ measures of over and under segmentation respectively [29].

To compute these measures, we need to define a couple of terms. First, the probability that a voxel w belongs to a cluster G_i is

$$P(w \in G_i) = P_G(i) = \frac{|G_i|}{|G|}$$

and similarly for P

$$P(w \in P_j) = P_P(j) = \frac{|P_j|}{|P|}.$$

Secondly, the probability that a pixel belongs to both G_i and P_j is

$$P(w \in G_i \cap P_j) = P(i, j) = \frac{|G_i \cap P_j|}{|G|}.$$

The conditional entropies are then computed as

$$\begin{aligned} V_u &= H(P|G) = -\sum_{i,j} P(i,j) \log \frac{P(i,j)}{P_G(i)} \\ V_0 H(G|P) &= -\sum_{i,j} P(i,j) \log \frac{P(i,j)}{P_P(j)}. \end{aligned} \tag{14}$$

In this project we will also used a normalized version of this measure. The normalized version can be expressed as

$$\begin{aligned} \Delta V_u &= H(P_i|G) - H(P_u|G) \\ \Delta V_o &= H(G|P_i) - H(G|P_u) \end{aligned} \tag{15}$$

where P_i is the partition, we want to evaluate and G its corresponding ground truth. P_u is also a partitioning of G which is done by a model trained under the same circumstances as P_i, except that that no, i.e. unit, augmentations was used during training. This means that all predictions made by a model trained with unit augmentations will get a normalized variation of information of zero. Positive values indicate a worse score than no augmentations, and negative values indicates a better score.

6.3 Segmentation categorization

A drawback of the above measures is that they are difficult to get an intuition about, or to be more precise, the values are only meaningful in relation to other values and even in those cases we can only say if one value is better than another. To try and give the reader a sense of the quality of the models, each cell in the ground truth is classified as *correct-*, *over-*, *under-*, *missed-*, or *divergent*-segmentation as described in [9].

The categories are determined by first finding the best match of each cell G_i in P, and the reverse for predictions P_j in G. More precisely, the matches are defined as

$$P_{G_i} = \arg\max_{P_j \in P} \frac{|G_i \cap P_j|}{|G_i|}; \quad G_{P_j} = \arg\max_{G_i \in G} \frac{|G_i \cap P_j|}{|P_j|} \tag{16}$$

and can be seen as the corresponding cell which covers a plurality of the coll volume.

The nest step is to define a set E which consists of all the pairs (G_i, P_j) where either G_i is the match of P_j or the reverse. E can be viewed as the edges in a graph where the clusters in G and P are the nodes. In this graph, cluster the nodes which are connected into new partitioning C_k. Given a cell in the ground truth segmentation G_i, localize the cluster C_k where G_i is contained. If the C_i only consists of one other element P_j, then G_i and P_j are each others matches, and we have a *correct* segmentation. If G_i is the only element from G while there are many from P, then we have an *over* segmentation. If the reverse is true with many from G and one from P, then we have an *under* segmentation. Finally, if there are multiple elements from both G and P in C_k then we have a *divergent* segmentation. However, cells in the ground truth which are matched with the background of the prediction are treated specially. First, two clusters which both containing the prediction background are not considered connected. If we now imagine that the prediction has missed to segment two cells, these two cells are not considered to belong to the same cluster. The categorization then goes as follows. If a cell G_i is in a cluster C_k where the only other element is the prediction background. The segmentation is categorized as a *missed* segmentation. On the other hand if C_k contains more elements then the prediction background, the segmentation is set as *divergent*.

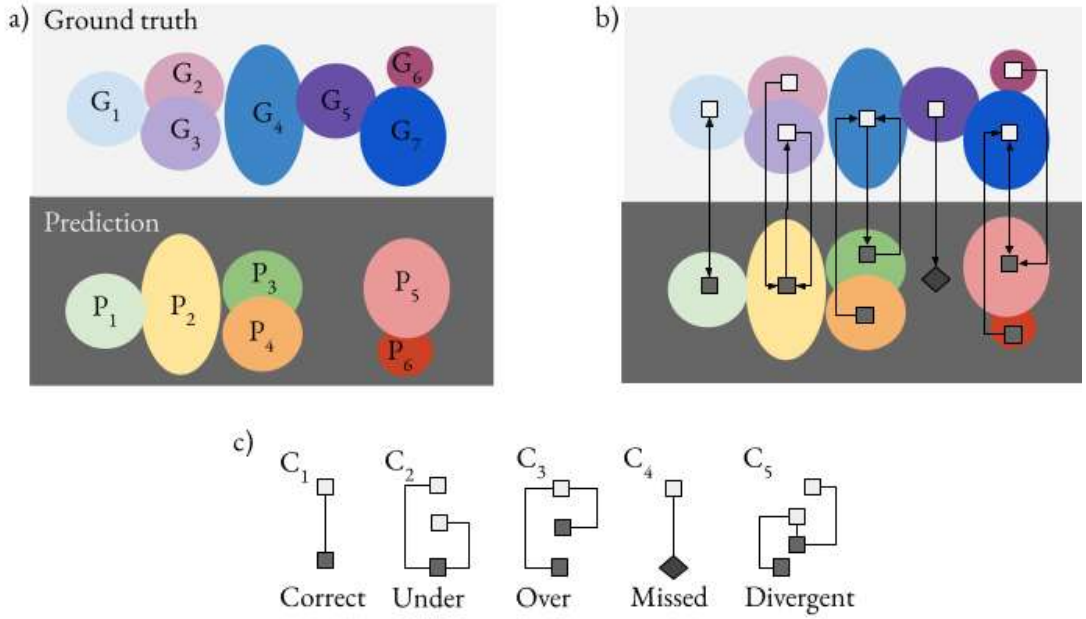


Figure 11: Illustration of the different segmentation types. Image a) shows the ground truth and prediction of an image where each colored ellipse represents a cell which are labelled as G_i and P_j respectively. Image b) shows the same cells where the match of each cell is represented with an arrow. Image c) shows the clustered version of the image b) labelled as C_k. Each cluster is labelled with the respective category.

In summary, if a cell G_i is only connected to another cell, the segmentation is *correct*, if it is only connected to the prediction background the segmentation is *missed*. If the cell is connected with multiple predictions, it is *over* segmented, while the reverse results in an *under* segmentation. Otherwise, the segmentation is considered *divergent*. Figure 11 gives a schematic view of the different categories and how they are determined.

7 Experiments

An experiment in this report consists of four parts. First the experiment is **set up**. Here the model, training data set, and augmentations are chosen. This is stored in a configuration file which is sent to the next step: **training**. In the training step, the train data is pre-processed to comply with the chosen model. Then the data is used to train the model from scratch using a local altered copy of the model's source code. The training step creates a trained model which is passed to the next step, **prediction**. Here all the data sets are passed one by one through the models including the instance segmentation steps until we have a predicted segmentation of each data set. These predictions are then compared to the ground truth segmentation in the **evaluation** step. The evaluation step calculates the metrics described in 6 for each image and stored as a tables which will be analysed below. The Adaptive Rand error will be used as an overall measure of the segmentation, while the variation of information will be used for estimating over and under segmentation. The segmentation categories will be used for a more qualitative approach, partly to get an intuition for how the model behaves but also to find and visualize the various categories.

The experiments were run using GeForce RTX 2080 Ti GPUs where each experiment took around 40 hours for both models when using one GPU. We have defined three categories of augmentations and for each experiment either non, one, or all of them where chosen. On top of that we have two models and three data sets to chose from making the total set of experiments $2 \cdot 3 \cdot 3 \cdot 5 = 90$ experiments.

8 Results

The experiments give us data from two models, three training sets, three evaluation sets, and five augmentation groups, for a total of 90 experiment where each experiment contains the evaluation metrics for each image from the specified evaluation set. For the complete results of the experiments, see table 4, 5, 6, and 7 in appendix A. In figure 12, we can see a visualization of the resulting AR-error where the columns corresponds to the training set and the rows to the evaluation set. Within each plot, the different augmentations are spread over the x-axis, and further divided into two colors, one for each model. Finally, the distribution of AR-error can be seen with a box plot along the y-axis.

If we look along the diagonal plots, we can see the results from the experiments that used the same data set for training and evaluation. Plantseg seem to do better over all the data sets compared to Cellpose, even if it performs a bit worse on FM than on SAM and Ovules. Cellpose, on the other hand, has its best results on SAM while performing significantly worse on Ovules. If we instead look at the off-diagonal plots row by row, we can see how difficult it is for the models to generalize to the different data sets. Cellpose can clearly be seen struggle to generalize to SAM, while Plantseg does this very well. FM is instead the most difficult data set for Plantseg to generalize to. Along each column in the figure, we see the results of experiments that have been trained on the same data set. Plantseg is doing well when trained on any of the data sets, while Cellpose trained on Ovules is completely unable to generalize to the other two data sets.

Within each plot we can compare the results from the different augmentations. In some situation, augmentation seem to do very little to the result as for Plantseg in c), e), and f), and Cellpose in b) and h). In other situations, any augmentation improves the results as for Plantseg in g) and d), and Cellpose in c). In Plantseg b) and h), and Cellpose a), d), f), and g), some augmentations do improve the results, while some augmentations worsen the results. For the individual augmentations, the combined augmentation gives the best results in plot b), f), and g) for Plantseg, and in a), c), d), g), and h) for Cellpose. Regarding the single augmentations, the geometric augmentation stands out in some cases where the other augmentations have very little effect. This occurs in Plantseg a), and h), and in Cellpose f) and i). The noise augmentation is also of interest as it have a relative significant effect on Cellpose in plot g) compared to the other single augmentations. Consequently, the augmentations seem to either have no impact or a slightly negative impact on the results when trained and evaluated on the same data. When trained and evaluated on different data, however, augmentations overall have a positive effect. Different augmentations are beneficial in different circumstances, while combining them all together have the best generalizing effect.

To visualize how different AR-error corresponds to different segmentations, figure 13 shows four example slices with the raw input image, ground truth labelling and Plantseg's and Cellpose's predictions. In a) both models achieve relatively low errors, 0.04 and 0.13 respectively, while in b) both perform relatively poorly, 0.94 and 0.99. It is also worth noticing the regions in b) which looks like cells but are not included in the ground truth labelling. These are cells which do not belong to the Ovules and the models do not get penalised for predicting those. Cellpose maxed out in both c) and d) with an AR-error of 1.0 while Plantseg managed to get 0.12 and 0.23. It is difficult to find any major flaws or even differences between the ground truth and the predictions in a). Both models create good segmentations as the AR-error suggests. In b), we can see that both models miss some cells in the bottom part of the left most structure, but apart from that the segmentation looks good even though both has close to the worst score possible. Plantseg segments both c) and d) very well while Cellpose is barely managing to segment some of the outer cells in

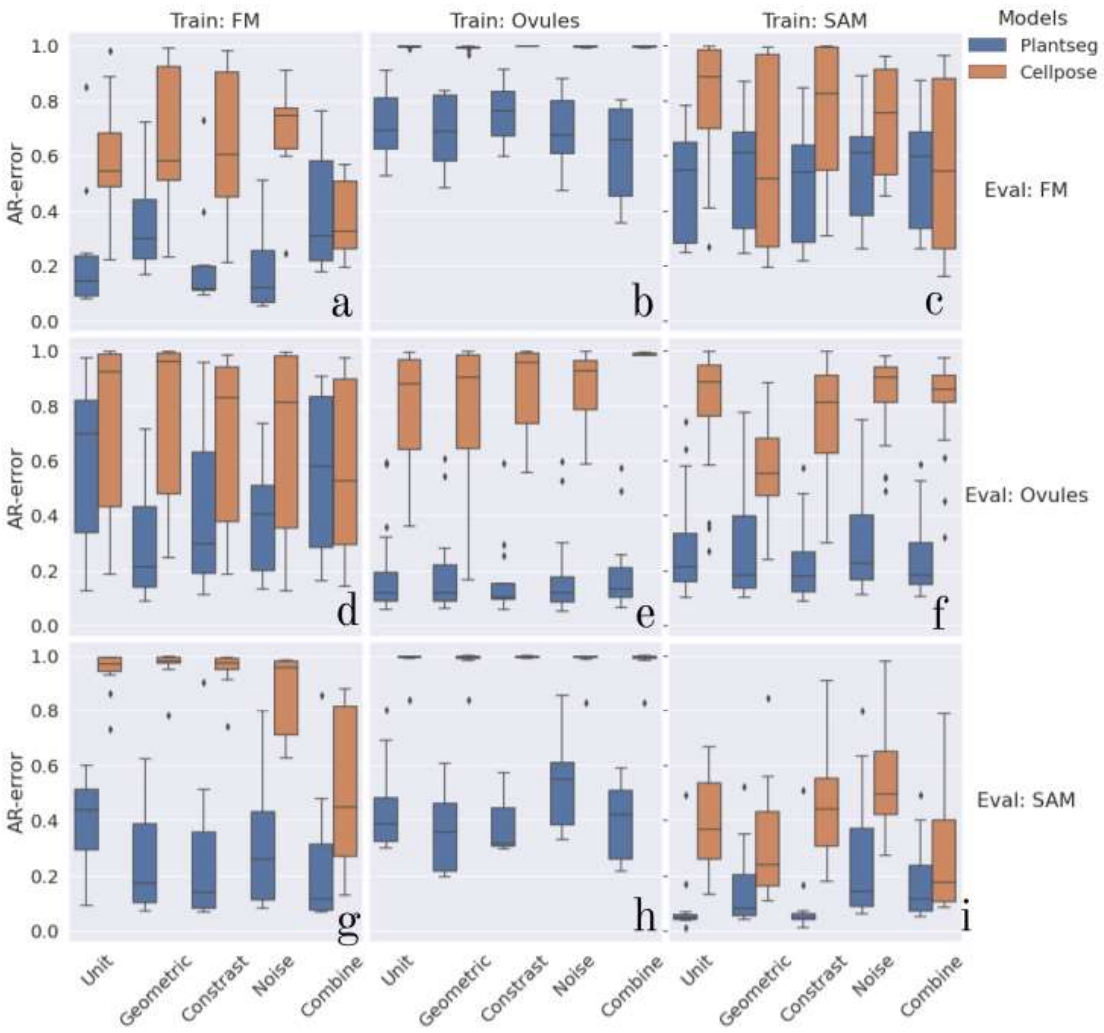


Figure 12: Figure showing the resulting AR-error for the different experiments. The experiments are divided by training set (column), evaluation set (row), augmentation (x-axis), and model (color). Each boxplot are created from the images the corresponding evaluation set. The AR-error goes between zero and one where lower is better.

c) and completely breaks down in d). This shows that the AR-error is good as separating a good model from a better model but struggles to compare models which are not performing that well.

The AR-error tells us about the overall segmentation quality, but not in an intuitive way as we saw regarding figure 13 where Cellpose had very similar AR-errors but where the segmentation quality varied greatly. In figure 14 and 15 we can see the segmentation categorise as percentages for the different experiments. Figure 14 shows the results for Plantseg. Here the proportion of correctly classified cells are very high across all plots. The errors are mostly due to under segmentations and a smaller part divergent segmentation. The variation within each plot is limited meaning that augmentations have a very little effect. The variance is instead bigger between plots where the off-diagonal predictions of the FM data set are worse than diagonal ones.

Figure 15 shows the categorise of the Cellpose predictions. Here the results are more varied where the mistakes come from both over, missed, and divergent classifications. This in contrast to Plantseg, where under segmentations were the dominant mistake. In plot g) we can see an increased number of correct segmentations when either noise or the combined augmentations were used. The augmentations seem to make no difference in number of over segmentations but reduces the missed and divergent segmentations. Similar things happen in plot c) and d). We can also compare b) and h) to b) and h) in figure 12 and see that the bad AR-error is mainly due to missed segmentations.

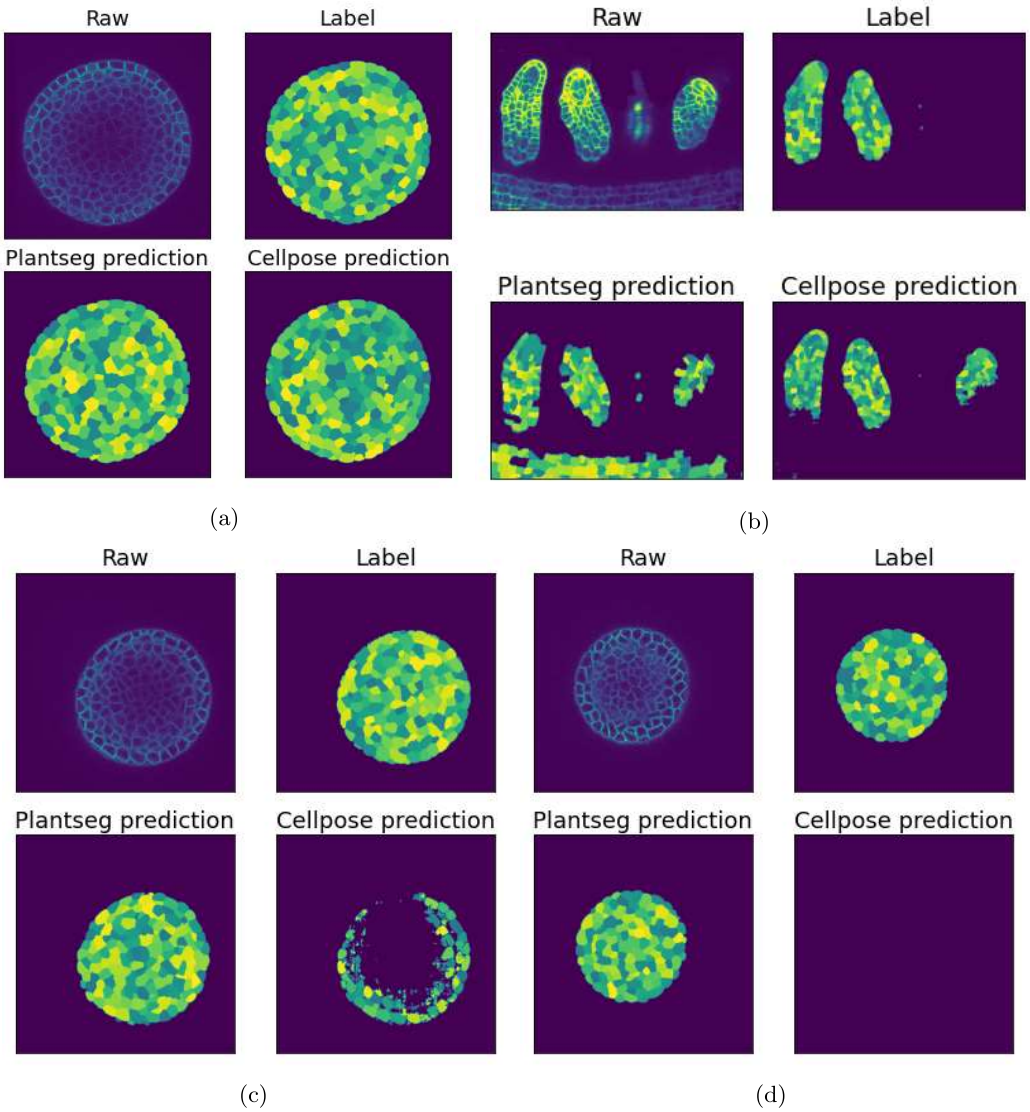


Figure 13: Four figures showing predictions of Plantseg and Cellpose in comparison to the ground truth labels and the input raw image. The sample from a) is taken from the SAM data set where Plantseg had a AR-error of 0.04 and Cellpose 0.13, both models had SAM as training set. The sample in b) is taken from the Ovules data set where both models was trained on FM and had high AR-errors of 0.94 and 0.99 respectively. c) and d) are SAM predictions made by models trained on Ovules. The AR-error for Plantseg was 0.12 and 0.23, and 1.0 in both example for Cellpose. In all figures, the models were trained without augmentations, and for visualization a 2D-slice was extracted.

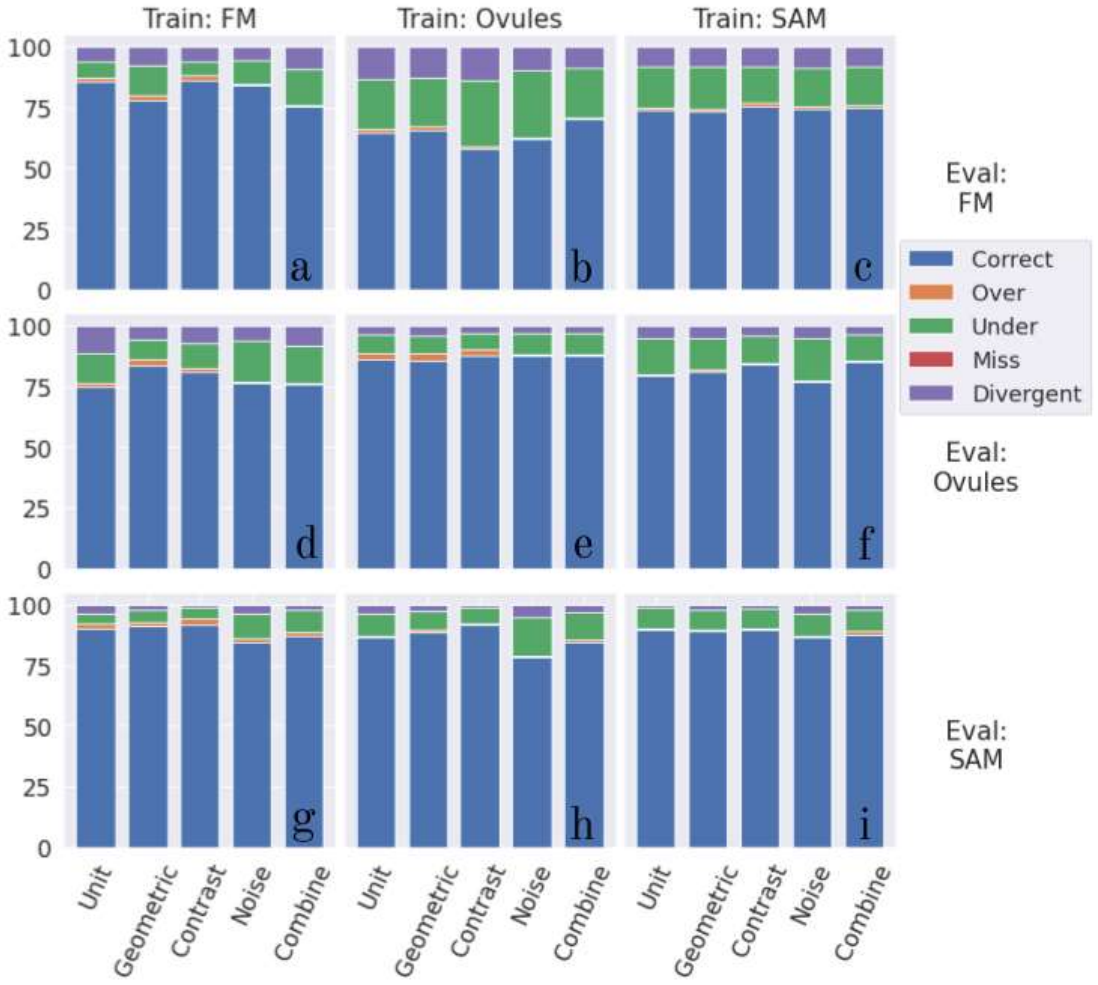


Figure 14: Results from the Plantseg experiments where the segmentation of each ground truth cell is classified as correct, over, under, miss, or divergent segmented.

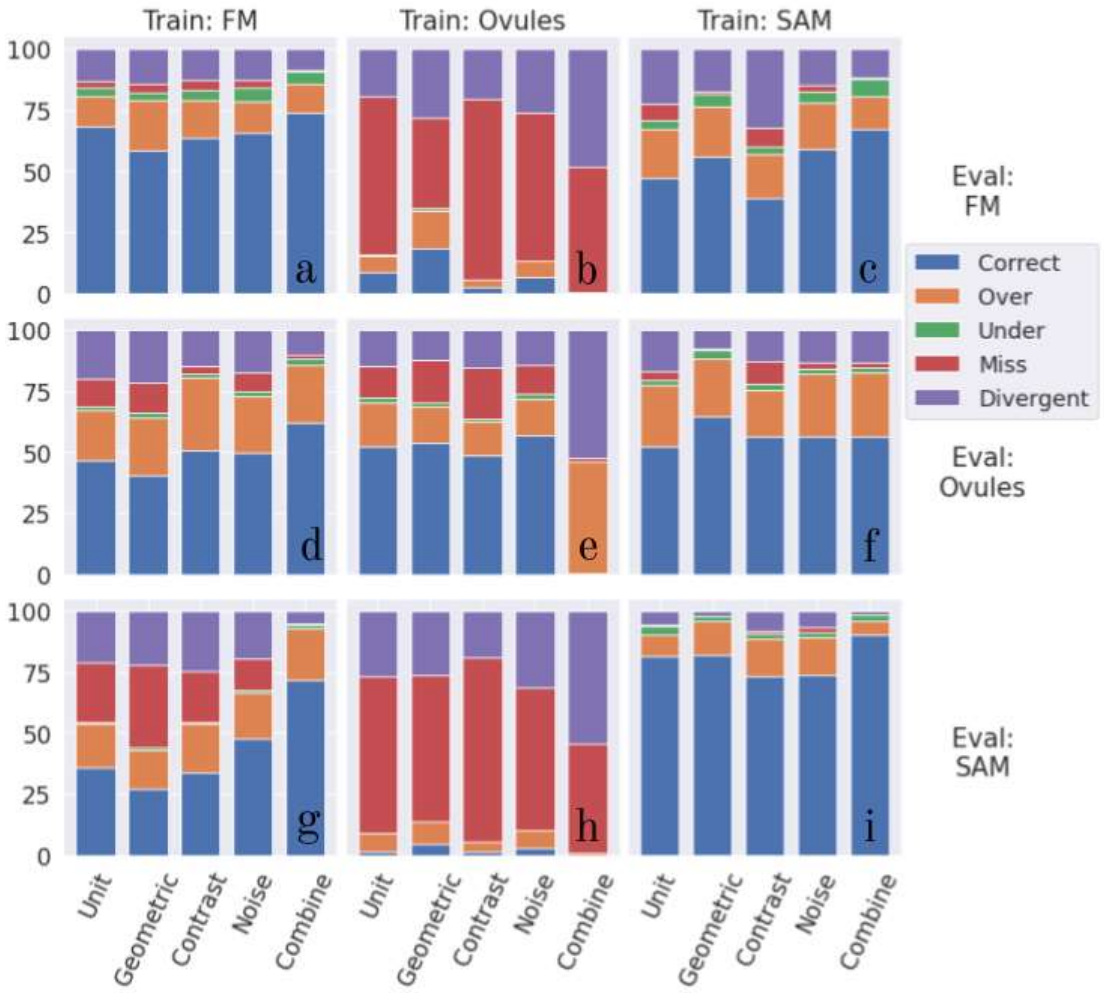


Figure 15: Results from the Cellpose experiments where the segmentation of each ground truth cell is classified as correct, over, under, miss, or divergent segmented.

We have seen, mainly in figure 12, how augmentations seem to improve the performance when trained and evaluated on different data, but worsen the performance when trained and evaluated on the same data. Furthermore, we saw in figures 14 and 15 that Plantseg mostly suffered from under segmentation while Cellpose struggled with over, missed, and divergent segmentations. To get another view on these values, figure 16 shows four scatter plots depicting the normalized variation of information values. Each dot in these graphs is representing a segmented image, and its position the VoI-over and VoI-under values, see equation 15. This normalization means that negative values correspond to an improvement in the values of that prediction compared to the non-augmented counter part. The row in the figure corresponds to the predictions of the two models and the columns to if the data comes from models trained and evaluated on same data, i.e. on-diagonal, or

trained and evaluated on different data, i.e. off-diagonal.

In plot a) we can see how the result from Plantseg trained and evaluated on the same data. Here most of the data falls very close to the center with some variation in the positive direction indicating a decrease in performance due to the augmentations. In plot b) we see a similar behaviour with most points clustered closely around the center, however, here the variation is along the negative part of the x-axis indicating a decrease in over segmentations. In plot c), we move over to Cellpose where again the data is centered around zeros with a slight tendency to positive values. In the final plot d), we can see the results of Cellpose evaluated on data sets it was not trained on. The results are spread out but with a clear tendency towards negative VoI-over values, but also a slight tendency towards positive VoI-under can be seen. There are no clear trends between the different augmentations, instead all of them are clustered around the center with similar tendencies as discussed above. The overall effects in the off-diagonal plots, is negative VoI-over values and positive VoI-under values, where the former is more prevalent. An average of the original VoI values, together with AR-error and the percentage of correct classification, over all the samples for the different augmentations and models can be found in table 3.

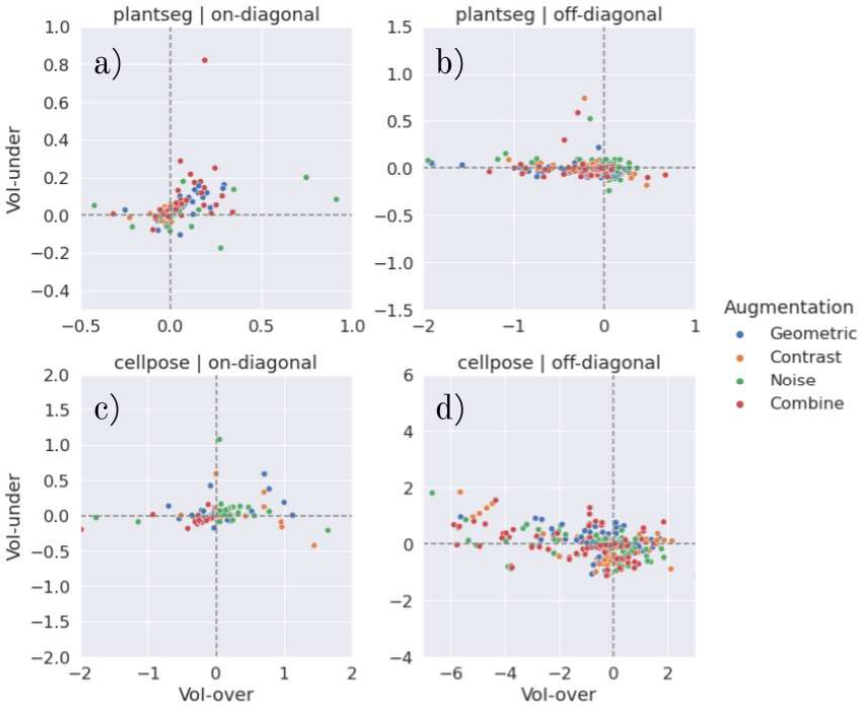


Figure 16: Scatter plots of the normalized VoI measures. The normalization makes negative values indicate an improvement in VoI in comparison to if the model had not used augmentation. The on-diagonal plots refer to models which are trained and evaluated on the same data while the off-diagonal plots are trained and evaluated on different data.

Table 3: Mean values with standard deviations in parenthesis for the different evaluation measures used. The values are computed over all experiment results by the given model using the given augmentation.

Model	Augmentation	AR-error	VoI-under	VoI-over	Correct
Cellpose	Unit	0.84 ± 0.24	0.99 ± 0.60	3.63 ± 2.97	0.39 ± 0.28
	Geometric	0.80 ± 0.27	0.95 ± 0.57	3.77 ± 2.95	0.37 ± 0.28
	Contrast	0.79 ± 0.25	0.99 ± 0.54	2.97 ± 2.93	0.43 ± 0.27
	Noise	0.83 ± 0.22	0.90 ± 0.44	3.27 ± 2.90	0.41 ± 0.27
	Combine	0.73 ± 0.29	0.99 ± 0.42	2.78 ± 3.06	0.48 ± 0.30
Plantseg	Unit	0.47 ± 0.25	0.57 ± 0.18	0.96 ± 0.50	0.78 ± 0.12
	Geometric	0.36 ± 0.23	0.55 ± 0.19	0.76 ± 0.35	0.82 ± 0.12
	Contrast	0.38 ± 0.26	0.57 ± 0.23	0.81 ± 47	0.80 ± 0.12
	Noise	0.43 ± 0.22	0.57 ± 0.19	0.91 ± 0.37	0.76 ± 0.12
	Combine	0.41 ± 0.26	0.55 ± 0.23	0.80 ± 0.38	0.80 ± 0.10

9 Discussion and further research

From the experiments done it is difficult to draw any strong conclusions. Figure 12 shows that the augmentation investigated could have s small effect such as for Plantseg in c), e), and f), or a larger positive effect as for Cellpose in g). Overall, it can be said that the use of augmentations over these microscopy data set do no harm, and are mostly beneficial, even if limited.

Discussing the models separately, Plantseg generalized well over all data sets without augmentations, however, including them did improve the results in many cases. There is no augmentation that on its own consistently increased Plantseg's values, instead the combination gave the best overall effect. An interesting result of Plantseg, is how figure 14 suggests that the biggest issue for Plantseg is under segmentations while table 3 suggests that it is over segmentations. We suspect that this effect stems from that even correctly segmented cells can add to the VoI values. The over all effect may hence be that the model over segments according to VoI. Furthermore, Figure 16 shows that the overall improvements gained by augmentations are made by mostly reducing the VoI-over segmentations. It is difficult to say why this occurs, however, it might simply arise since the VoI-over is in general greater than VoI-under, as can be seen for both models in table 3. This means that any improvement made is more likely to affect the VoI-over value more.

The results from Cellpose are more varied. Among the individual augmentations, geometric and noise each had situations where they had a big impact, e.g. figure 12 f) and g) respectively. Overall, the best strategy for generalizability seems to be the combined version for Cellpose as well. Comparing the different data sets, Cellpose trained on Ovules could not generalize at all to the other two data sets. We suspect that the reason for this is that the cells in Ovules are shaped differently than the cells in the other two sets. Something which is quantified under sphericity in table 1. The reasons why Cellpose struggle with this but not Plantseg, is that Cellpose works by predicting '*cells*', either as a probability or as a vector field, while Plantseg works by predicting boundaries. Cells is a more high-level feature than boundary, and will probably vary more in between data sets, making in more difficult for Cellpose to generalize in comparison to Plantseg. Evidence for this claim may be found in figure 13 b) where Plantseg has predicted the cells in the lower structure while Cellpose hasn't. If Cellpose has learned these more high-level concepts as what a correct cell looks like, then it can discard cells which do not fill its criteria. If Plantseg, on the other hand,

only learns more low lever concepts, it will segment the lower structure given that it locks similar locally. This possible difference in sensitivity to the scale of the features would be an interesting question to do further investigation on.

Size may play another role for Cellpose as well. When trained on Ovules, the model fails to generalize to the other two sets, and when trained on FM, it struggles to generalize to SAM. The cells in Ovules are bigger than the ones in FM which in turn are bigger than SAM, suggesting that Cellpose can generalize to cells which are bigger than the training cells but not the other way around.

Comparing Cellpose in b), g), and h) in Figure 12 to the same plots in figure 14 shows an other interesting result. In figure 12 all three plots are similarly bad for the first three augmentations. However, adding noise or the combined augmentation drastically improves the result in plot g), but not in b) and h). The reason for this difference can be seen in figure 14 where g) is no longer like the other two. The Cellpose values in g), compared to b) and h), are much better but apparently not good enough to have an impact on the AR-error. This suggests that augmentations cannot improve models that are completely off but can have a great effect on functioning but bad models.

Looking at the average values presented in table 3, we can see that the contrast augmentations have the greatest single effect on Cellpose while geometric is the best one for Plantseg. These values should, however, be treated with care as they are averaged over the data sets and as we have seen in figure 12, 14, and 15, the values varies greatly between the data sets. It is still noteworthy that all augmentations perform better than unit in all categories with only one exception, geometric augmentations on Cellpose when measuring the percentage of correctly classified, indicating a clear positive effect when applying augmentations.

Consequently, the experiments show that augmentations can have a beneficial effect on the generalizability of models, even if often very limited. It was also shown that the combination of augmentations gave the best overall improvements. For a model which already generalises well like Plantseg, the effect was smaller but still significant while they were crucial for Cellpose's performance in certain cases. The improvements made seem to come mostly from decreasing the amount of over and missed segmentations. The experiments show that the model's ability to generalize is dependent on the shape and size of the data, a topic worth investigating further.

References

[1] Ignacio Arganda-Carreras et al. "Crowdsourcing the creation of image segmentation algorithms for connectomics". In: *Frontiers in neuroanatomy* 9 (2015), p. 142.

[2] Nils Bjorck et al. "Understanding batch normalization". In: *Advances in neural information processing systems* 31 (2018).

[3] Zihang Dai et al. *Coatnet: Marrying convolution and attention for all data sizes*. 2021.

[4] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: *arXiv preprint arXiv:1603.07285* (2016).

[5] Kaiming He et al. "Deep residual learning for image recognition". In: (2016), pp. 770–778.

[6] Sakshi Indolia et al. "Conceptual understanding of convolutional neural network-a deep learning approach". In: *Procedia computer science* 132 (2018), pp. 679–688.

[7] Ottosson Joel. *Augmentation in 3D microscopy*. Version 1.0.0. Feb. 2021. DOI: 10.5281/zenodo.1234. URL: https://github.com/AllaVinner/Augmentation-in-3D-microscopy/.

[8] Jörg Hendrik Kappes et al. "Globally optimal image partitioning by multicuts". In: *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer. 2011, pp. 31–44.

[9] Anuradha Kar et al. "Assessment of deep learning algorithms for 3D instance segmentation of confocal image datasets". In: *bioRxiv* (2021). DOI: 10.1101/2021.06.09.447748. eprint: https://www.biorxiv.org/content/early/2021/06/10/2021.06.09.447748.full.pdf. URL: https://www.biorxiv.org/content/early/2021/06/10/2021.06.09.447748.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[11] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[12] Chenxi Liu et al. "Progressive neural architecture search". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 19–34.

[13] Zhou Lu et al. "The expressive power of neural networks: A view from the width". In: *Advances in neural information processing systems* 30 (2017).

[14] Marina Meilă. "Comparing clusterings—an information based distance". In: *Journal of multivariate analysis* 98.5 (2007), pp. 873–895.

[15] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. DOI: 10.1080/01621459.1971.10482356. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1971.10482356. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

[16] Yassin Refahi et al. "Research data supporting" A multiscale analysis of early flower development in Arabidopsis provides an integrated view of molecular regulation and growth control"". In: (2021).

[17] Jos BTM Roerdink and Arnold Meijster. "The watershed transform: Definitions, algorithms and parallelization strategies". In: *Fundamenta informaticae* 41.1, 2 (2000), pp. 187–228.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-net: Convolutional networks for biomedical image segmentation*. Springer, 2015.

[19] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[20] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of Big Data* 6.1 (2019), pp. 1–48.

[21] Nicholas Sofroniew et al. *napari/napari: 0.4.14*. Version v0.4.14. Feb. 2022. DOI: 10.5281/zenodo.5975474. URL: https://doi.org/10.5281/zenodo.5975474.

[22] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[23] Carsen Stringer et al. "Cellpose: a generalist algorithm for cellular segmentation". In: *Nature methods* 18.1 (2021), pp. 100–106.

[24] C Sullivan and Alexander Kaszynski. "PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)". In: *Journal of Open Source Software* 4.37 (2019), p. 1450.

[25] Markus Svensén and Christopher M Bishop. *Pattern recognition and machine learning*. 2007.

[26] Athul Vijayan et al. "A digital 3D reference atlas reveals cellular growth patterns shaping the Arabidopsis ovule". In: *Elife* 10 (2021), e63262.

[27] Hakon Wadell. "Volume, shape, and roundness of quartz particles". In: *The Journal of Geology* 43.3 (1935), pp. 250–280.

[28] Lisa Willis et al. "Cell size and growth regulation in the Arabidopsis thaliana apical stem cell niche". In: *Proceedings of the National Academy of Sciences* 113.51 (2016), E8238–E8246.

[29] Adrian Wolny et al. "Accurate and versatile 3D segmentation of plant tissues at cellular resolution". In: *Elife* 9 (2020), e57613.

[30] Adrian Wolny et al. *Sparse Object-level Supervision for Instance Segmentation with Pixel Embeddings*. 2021. arXiv: 2103.14572 [cs.CV].

[31] Saining Xie et al. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[32] Yuhui Yuan et al. "Segmentation transformer: Object-contextual representations for semantic segmentation". In: *arXiv preprint arXiv:1909.11065* (2019).

[33] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. "Road Extraction by Deep Residual U-Net". In: *CoRR* abs/1711.10684 (2017). arXiv: 1711.10684. URL: http://arxiv.org/abs/1711.10684.

A Appendix - Tables

Table 4: The adapted rand error and variation of information results from the experiments run with Plantseg. The values are taken as average over the evaluation set with one standard deviation shown as ±.

Plantseg					
Train set	Eval set	Augmentation	AR-error	VoI-under	VoI-over
FM	FM	Unit	0.24 ± 0.25	0.38 ± 0.09	0.52 ± 0.34
		Geometric	0.36 ± 0.20	0.49 ± 0.05	0.70 ± 0.22
		Contrast	0.22 ± 0.20	0.39 ± 0.09	0.48 ± 0.27
		Noise	0.19 ± 0.16	0.34 ± 0.11	0.46 ± 0.25
		Combine	0.41 ± 0.24	0.44 ± 0.05	0.75 ± 0.21
	Ovules	Unit	0.59 ± 0.29	0.50 ± 0.08	1.23 ± 0.70
		Geometric	0.28 ± 0.19	0.49 ± 0.09	0.65 ± 0.22
		Contrast	0.39 ± 0.26	0.51 ± 0.08	0.84 ± 0.49
		Noise	0.39 ± 0.18	0.55 ± 0.06	0.94 ± 0.33
		Combine	0.56 ± 0.28	0.46 ± 0.06	1.05 ± 0.44
	SAM	Unit	0.39 ± 0.19	0.51 ± 0.32	0.51 ± 0.16
		Geometric	0.25 ± 0.19	0.50 ± 0.37	0.41 ± 0.12
		Contrast	0.26 ± 0.26	0.57 ± 0.52	0.35 ± 0.06
		Noise	0.31 ± 0.23	0.56 ± 0.46	0.48 ± 0.13
		Combine	0.23 ± 0.24	0.52 ± 0.49	0.34 ± 0.06
Ovules	FM	Unit	0.72 ± 0.13	0.73 ± 0.11	1.43 ± 0.30
		Geometric	0.68 ± 0.14	0.70 ± 0.11	1.36 ± 0.30
		Contrast	0.75 ± 0.11	0.68 ± 0.06	1.62 ± 0.34
		Noise	0.69 ± 0.13	0.69 ± 0.08	1.42 ± 0.27
		Combine	0.61 ± 0.17	0.70 ± 0.09	1.18 ± 0.21
	Ovules	Unit	0.18 ± 0.15	0.36 ± 0.05	0.44 ± 0.12
		Geometric	0.18 ± 0.14	0.38 ± 0.07	0.46 ± 0.13
		Contrast	0.17 ± 0.15	0.37 ± 0.06	0.42 ± 0.11
		Noise	0.17 ± 0.14	0.36 ± 0.05	0.44 ± 0.11
		Combine	0.17 ± 0.12	0.42 ± 0.10	0.48 ± 0.15
	SAM	Unit	0.44 ± 0.17	0.66 ± 0.20	0.89 ± 0.27
		Geometric	0.37 ± 0.15	0.60 ± 0.23	0.69 ± 0.12
		Contrast	0.39 ± 0.15	0.58 ± 0.05	0.75 ± 0.16
		Noise	0.53 ± 0.15	0.57 ± 0.15	0.96 ± 0.27
		Combine	0.40 ± 0.14	0.61 ± 0.30	0.69 ± 0.14
SAM	FM	Unit	0.50 ± 0.21	0.62 ± 0.15	0.91 ± 0.22
		Geometric	0.54 ± 0.22	0.59 ± 0.12	0.98 ± 0.28
		Contrast	0.50 ± 0.22	0.63 ± 0.17	0.89 ± 0.25
		Noise	0.56 ± 0.20	0.60 ± 0.14	0.97 ± 0.26
		Combine	0.55 ± 0.20	0.60 ± 0.13	0.95 ± 0.22
	Ovules	Unit	0.28 ± 0.18	0.54 ± 0.12	0.76 ± 0.26
		Geometric	0.28 ± 0.20	0.52 ± 0.12	0.73 ± 0.29
		Contrast	0.22 ± 0.14	0.54 ± 0.14	0.65 ± 0.22
		Noise	0.30 ± 0.18	0.54 ± 0.12	0.83 ± 0.31
		Combine	0.20 ± 0.13	0.54 ± 0.14	0.62 ± 0.20
	SAM	Unit	0.09 ± 0.13	0.26 ± 0.10	0.27 ± 0.06
		Geometric	0.16 ± 0.15	0.28 ± 0.07	0.33 ± 0.10
		Contrast	0.09 ± 0.14	0.24 ± 0.10	0.26 ± 0.07
		Noise	0.27 ± 0.26	0.31 ± 0.07	0.49 ± 0.31
		Combine	0.20 ± 0.16	0.40 ± 0.21	0.33 ± 0.07

Table 5: The resulting segmentation categories as percentages from the experiments run with Plantseg. The values are taken as average over the evaluation set with one standard deviation shown as ±.

Plantseg							
Train set	Eval set	Augmentation	Correct	Under	Over	Missed	Divergent
FM	FM	Unit	86 ± 10	7 ± 6	1 ± 1	0 ± 0	6 ± 5
		Geometric	78 ± 8	12 ± 6	2 ± 1	0 ± 0	8 ± 3
		contrast	86 ± 10	6 ± 6	2 ± 1	0 ± 0	6 ± 4
		Noise	84 ± 9	10 ± 5	1 ± 0	0 ± 0	5 ± 4
		Combine	76 ± 7	15 ± 7	1 ± 0	0 ± 0	9 ± 3
	Ovules	Unit	75 ± 13	12 ± 6	1 ± 1	0 ± 0	11 ± 9
		Geometric	84 ± 8	8 ± 6	3 ± 1	0 ± 0	6 ± 5
		contrast	81 ± 12	10 ± 7	2 ± 1	0 ± 0	7 ± 7
		Noise	76 ± 11	17 ± 8	1 ± 0	0 ± 0	6 ± 5
		Combine	76 ± 11	15 ± 7	0 ± 0	0 ± 0	8 ± 6
	SAM	Unit	90 ± 5	4 ± 3	2 ± 1	0 ± 0	4 ± 3
		Geometric	91 ± 7	5 ± 6	2 ± 1	0 ± 0	2 ± 2
		contrast	92 ± 5	5 ± 5	2 ± 1	0 ± 0	1 ± 1
		Noise	85 ± 7	10 ± 5	1 ± 1	0 ± 0	4 ± 3
		Combine	87 ± 6	9 ± 5	1 ± 1	0 ± 0	2 ± 2
Ovules	FM	Unit	65 ± 12	20 ± 9	2 ± 1	0 ± 0	13 ± 5
		Geometric	66 ± 13	20 ± 10	2 ± 1	0 ± 0	13 ± 5
		contrast	58 ± 16	27 ± 13	1 ± 1	0 ± 0	14 ± 5
		Noise	62 ± 11	28 ± 11	0 ± 0	0 ± 0	10 ± 3
		Combine	70 ± 9	20 ± 8	1 ± 0	0 ± 0	9 ± 3
	Ovules	Unit	86 ± 7	8 ± 5	2 ± 1	0 ± 0	4 ± 4
		Geometric	86 ± 8	8 ± 5	3 ± 2	0 ± 0	4 ± 4
		contrast	88 ± 7	7 ± 5	2 ± 1	0 ± 0	3 ± 3
		Noise	87 ± 7	9 ± 6	1 ± 0	0 ± 0	3 ± 4
		Combine	88 ± 7	8 ± 6	1 ± 1	0 ± 0	3 ± 4
	SAM	Unit	87 ± 8	9 ± 5	1 ± 1	0 ± 0	4 ± 4
		Geometric	89 ± 5	8 ± 4	1 ± 1	0 ± 0	2 ± 2
		contrast	92 ± 2	6 ± 3	1 ± 1	0 ± 0	1 ± 1
		Noise	79 ± 9	16 ± 5	0 ± 0	0 ± 0	5 ± 5
		Combine	85 ± 6	11 ± 3	1 ± 1	0 ± 0	3 ± 3
SAM	FM	Unit	74 ± 8	17 ± 6	1 ± 1	0 ± 0	8 ± 3
		Geometric	73 ± 8	17 ± 7	1 ± 1	0 ± 0	8 ± 3
		contrast	75 ± 8	15 ± 6	1 ± 2	0 ± 0	8 ± 3
		Noise	74 ± 8	16 ± 6	1 ± 1	0 ± 0	8 ± 3
		Combine	75 ± 8	16 ± 7	1 ± 1	0 ± 0	8 ± 3
	Ovules	Unit	79 ± 11	15 ± 9	0 ± 0	0 ± 0	5 ± 5
		Geometric	81 ± 11	13 ± 8	1 ± 0	0 ± 0	5 ± 5
		contrast	84 ± 9	11 ± 7	1 ± 0	0 ± 0	4 ± 4
		Noise	77 ± 12	17 ± 10	0 ± 0	0 ± 0	5 ± 4
		Combine	85 ± 9	11 ± 7	1 ± 0	0 ± 0	4 ± 4
	SAM	Unit	90 ± 5	8 ± 4	0 ± 1	0 ± 0	1 ± 1
		Geometric	89 ± 5	8 ± 5	1 ± 1	0 ± 0	2 ± 2
		contrast	90 ± 5	8 ± 4	0 ± 1	0 ± 0	1 ± 1
		Noise	86 ± 8	9 ± 4	1 ± 1	0 ± 0	4 ± 4
		Combine	88 ± 6	9 ± 4	1 ± 1	0 ± 0	2 ± 2

Table 6: The adapted rand error and variation of information results from the experiments run with Cellpose. The values are taken as average over the evaluation set with one standard deviation shown as ±.

Cellpose					
Train set	Eval set	Augmentation	AR-error	VoI-under	VoI-over
FM	FM	Unit	0.60 ± 0.22	0.82 ± 0.14	1.09 ± 0.67
		Geometric	0.65 ± 0.27	0.97 ± 0.27	1.41 ± 1.12
		Contrast	0.64 ± 0.29	0.85 ± 0.23	1.31 ± 0.90
		Noise	0.69 ± 0.19	0.87 ± 0.13	1.09 ± 0.39
		Combine	0.37 ± 0.15	0.81 ± 0.14	0.70 ± 0.11
	Ovules	Unit	0.74 ± 0.31	1.11 ± 0.49	2.80 ± 2.67
		Geometric	0.77 ± 0.29	1.36 ± 0.66	2.88 ± 2.60
		Contrast	0.69 ± 0.30	1.36 ± 0.57	1.38 ± 0.96
		Noise	0.69 ± 0.31	1.20 ± 0.44	1.94 ± 1.63
		Combine	0.57 ± 0.31	1.16 ± 0.47	0.99 ± 0.61
	SAM	Unit	0.94 ± 0.08	0.63 ± 0.21	3.85 ± 1.87
		Geometric	0.97 ± 0.06	0.59 ± 0.18	4.89 ± 1.84
		Contrast	0.95 ± 0.07	0.74 ± 0.22	3.67 ± 1.51
		Noise	0.85 ± 0.15	0.74 ± 0.14	2.47 ± 1.28
		Combine	0.51 ± 0.29	0.87 ± 0.42	0.80 ± 0.33
Ovules	FM	Unit	1.00 ± 0.00	0.36 ± 0.26	8.02 ± 1.15
		Geometric	0.99 ± 0.01	0.72 ± 0.30	6.06 ± 1.96
		Contrast	1.00 ± 0.00	0.22 ± 0.17	8.89 ± 0.55
		Noise	1.00 ± 0.00	0.38 ± 0.18	8.18 ± 0.59
		Combine	1.00 ± 0.00	0.79 ± 0.58	8.36 ± 0.91
	Ovules	Unit	0.80 ± 0.21	0.84 ± 0.25	2.37 ± 2.00
		Geometric	0.80 ± 0.24	0.73 ± 0.20	2.64 ± 2.22
		Contrast	0.86 ± 0.18	0.71 ± 0.17	3.16 ± 2.24
		Noise	0.87 ± 0.13	0.77 ± 0.19	2.20 ± 1.42
		Combine	0.99 ± 0.00	3.55 ± 0.11	2.65 ± 0.51
	SAM	Unit	0.97 ± 0.06	0.35 ± 0.25	8.09 ± 0.59
		Geometric	0.98 ± 0.05	0.37 ± 0.21	7.75 ± 1.17
		Contrast	1.00 ± 0.00	0.21 ± 0.18	8.77 ± 1.07
		Noise	0.98 ± 0.05	0.41 ± 0.23	7.92 ± 0.80
		Combine	0.98 ± 0.05	0.90 ± 0.41	7.88 ± 0.58
SAM	FM	Unit	0.78 ± 0.26	0.94 ± 0.22	2.50 ± 1.96
		Geometric	0.59 ± 0.34	1.06 ± 0.50	1.64 ± 1.25
		Contrast	0.75 ± 0.26	0.98 ± 0.24	3.12 ± 2.78
		Noise	0.73 ± 0.21	0.98 ± 0.26	1.24 ± 0.35
		Combine	0.56 ± 0.33	0.95 ± 0.35	1.02 ± 0.43
	Ovules	Unit	0.80 ± 0.21	1.52 ± 0.59	1.70 ± 1.61
		Geometric	0.57 ± 0.18	1.16 ± 0.35	0.79 ± 0.22
		Contrast	0.74 ± 0.21	1.04 ± 0.45	1.96 ± 2.56
		Noise	0.84 ± 0.14	1.09 ± 0.32	1.49 ± 0.50
		Combine	0.82 ± 0.16	1.02 ± 0.28	1.40 ± 0.48
	SAM	Unit	0.40 ± 0.18	0.69 ± 0.39	0.60 ± 0.13
		Geometric	0.32 ± 0.23	0.71 ± 0.51	0.47 ± 0.08
		Contrast	0.47 ± 0.20	0.76 ± 0.55	0.64 ± 0.14
		Noise	0.54 ± 0.19	0.84 ± 0.67	0.69 ± 0.15
		Combine	0.27 ± 0.23	0.65 ± 0.45	0.40 ± 0.07

Table 7: The resulting segmentation categories as percentages from the experiments run with Cellpose. The values are taken as average over the evaluation set with one standard deviation shown as ±.

Cellpose							
Train nset	Eval set	Augmentation	Correct	Under	Over	Missed	Divergent
FM	FM	Unit	68 ± 14	4 ± 2	12 ± 5	3 ± 3	13 ± 6
		Geometric	58 ± 23	3 ± 2	20 ± 11	3 ± 6	14 ± 9
		Contrast	64 ± 21	4 ± 3	15 ± 10	4 ± 6	13 ± 8
		Noise	66 ± 11	6 ± 3	13 ± 4	3 ± 2	13 ± 5
		Combine	74 ± 7	5 ± 3	12 ± 3	0 ± 0	9 ± 4
	Ovules	Unit	47 ± 30	2 ± 2	20 ± 9	11 ± 16	20 ± 16
		Geometric	40 ± 28	2 ± 2	24 ± 11	13 ± 17	22 ± 17
		Contrast	51 ± 21	2 ± 2	30 ± 12	3 ± 3	15 ± 11
		Noise	50 ± 27	2 ± 3	23 ± 9	8 ± 10	17 ± 14
		Combine	62 ± 20	3 ± 3	23 ± 12	2 ± 2	10 ± 8
	SAM	Unit	36 ± 20	1 ± 1	18 ± 8	25 ± 20	21 ± 8
		Geometric	27 ± 18	1 ± 1	16 ± 5	34 ± 22	22 ± 7
		Contrast	34 ± 17	1 ± 1	20 ± 7	21 ± 16	25 ± 7
		Noise	48 ± 20	1 ± 1	19 ± 5	13 ± 9	20 ± 10
		Combine	72 ± 14	1 ± 1	21 ± 10	1 ± 1	5 ± 4
Ovules	FM	Unit	9 ± 12	0 ± 0	7 ± 6	65 ± 24	20 ± 11
		Geometric	18 ± 15	1 ± 1	16 ± 8	37 ± 23	28 ± 13
		Contrast	2 ± 2	0 ± 0	3 ± 3	74 ± 17	20 ± 12
		Noise	7 ± 8	0 ± 0	7 ± 5	60 ± 16	26 ± 11
		Combine	0 ± 0	0 ± 0	1 ± 1	51 ± 24	48 ± 23
	Ovules	Unit	52 ± 27	2 ± 1	18 ± 10	13 ± 16	15 ± 9
		Geometric	54 ± 26	2 ± 2	15 ± 7	17 ± 19	12 ± 7
		Contrast	49 ± 29	1 ± 1	14 ± 8	21 ± 20	16 ± 9
		Noise	57 ± 19	2 ± 1	15 ± 7	12 ± 12	14 ± 7
		Combine	0 ± 0	0 ± 0	46 ± 28	2 ± 1	52 ± 28
	SAM	Unit	1 ± 1	0 ± 0	7 ± 6	64 ± 23	27 ± 17
		Geometric	4 ± 4	0 ± 0	9 ± 7	60 ± 19	26 ± 9
		Contrast	1 ± 3	0 ± 0	4 ± 5	75 ± 18	19 ± 12
		Noise	3 ± 3	0 ± 0	7 ± 7	59 ± 18	31 ± 11
		Combine	0 ± 0	0 ± 0	1 ± 1	45 ± 17	55 ± 17
SAM	FM	Unit	47 ± 25	4 ± 4	20 ± 12	6 ± 7	23 ± 11
		Geometric	56 ± 25	5 ± 5	20 ± 15	1 ± 1	17 ± 13
		Contrast	39 ± 31	3 ± 4	18 ± 7	8 ± 8	32 ± 21
		Noise	59 ± 12	5 ± 2	19 ± 8	3 ± 2	15 ± 4
		Combine	67 ± 16	7 ± 4	14 ± 10	0 ± 0	12 ± 6
	Ovules	Unit	52 ± 23	2 ± 2	25 ± 9	3 ± 5	17 ± 18
		Geometric	65 ± 9	4 ± 2	23 ± 8	0 ± 1	8 ± 5
		Contrast	56 ± 24	3 ± 2	19 ± 10	9 ± 21	13 ± 11
		Noise	56 ± 12	2 ± 2	26 ± 9	2 ± 2	14 ± 5
		Combine	56 ± 13	2 ± 2	26 ± 9	2 ± 2	14 ± 6
	SAM	Unit	81 ± 6	3 ± 2	9 ± 3	1 ± 1	5 ± 3
		Geometric	82 ± 5	2 ± 1	14 ± 5	0 ± 0	2 ± 1
		Contrast	73 ± 6	2 ± 1	15 ± 3	1 ± 1	8 ± 3
		Noise	74 ± 6	2 ± 1	16 ± 4	2 ± 2	7 ± 2
		Combine	90 ± 6	2 ± 1	6 ± 4	0 ± 0	2 ± 2

Master's Theses in Mathematical Sciences 2022:E10
ISSN 1404-6342

LUTFMA-3464-2022

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/