Organoids Segmentation using Self-Supervised Learning: How Complex Should the Pretext Task Be?

Asmaa Haja
a.haja@rug.nl
Bernoulli Institute, University of Groningen, The Netherlands

Bart van der Woude
b.r.van.der.woude@student.rug.nl,
University of Groningen, The Netherlands

Lambert Schomaker
l.r.b.schomaker@rug.nl
Bernoulli Institute, University of Groningen, The Netherlands

ABSTRACT

Most popular supervised-learning approaches require large annotated data sets that are time-consuming and costly to create. Self-supervised learning (SSL) has proven to be a viable method for increasing downstream performance, through pre-training models on a pretext task. However, the literature is not conclusive on how to choose the best pretext task. This research sheds light on how the complexity of the pretext task affects organoid segmentation performance, in addition to understanding whether a self-prediction or innate relationship SSL strategy is best suited for organoid segmentation. Eight novel self-prediction distortion methods were implemented, creating eight simple and twenty-eight complex pretext tasks. Those were compared to two innate relationship pretext tasks: Jigsaw and Predict rotation. Results showed that the complexity of the pretext tasks does not correlate with segmentation performance. However, complex models (μF1 = 0.862) consistently, albeit with a small effect size, outperform simple models (μF1 = 0.848). Possibly due to acquiring a wider variety of learned features after pretext learning, despite not being necessarily more complex. Comparing SSL strategies showed that self-prediction models (μF1 = 0.856) slightly outperform innate relationship models (μF1 = 0.848). Furthermore, more pretext training data improves downstream performance under the condition that there is a minimum amount of downstream training data available. Too little downstream training data combined with more pretext training data leads to a decrease in segmentation performance.

CCS CONCEPTS

• Computing methodologies → Image segmentation.

KEYWORDS

Deep Learning, Self-supervised Learning, Pretext Task, Segmentation, Organoids

1 INTRODUCTION

Using deep-learning (DL) methods for biomedical image segmentation is of great value to medicine and biological research. From finding cancer in endobronchial ultrasounds [24] and breast mammography [12], to diagnosing acute ischemic stroke lesions in CT perfusion maps [21]. This research specifically focuses on DL for performing organoid segmentation. Organoids are in vitro grown tissue cultures mimicking the structure and functionality of in vivo organs. Researching organoids gives the opportunity to understand organ function, growth and their response to potentially useful drugs [5].

In the domain of biological image analysis, supervised learning methods have proven greatly successful [2]. Despite this success, these methods require copious amounts of annotated data [8, 9, 11, 22]. These data sets are challenging to acquire for biological imaging, as annotation is expensive, time-consuming and calls for expertise in this field [3, 8, 12, 17]. Similarly, creating data sets for clinical use cases is also limited as labeling often focuses on creating a data set that is suited for a single task, rather than a data set that has more potential use cases [8].

Self-supervised learning (SSL) tries to solve the problem of lacking annotated data by pre-training models on similar unlabeled data solving a pseudo-task, thereby having the pre-trained network learn relevant features of the data [6, 8, 9, 11, 15, 22]. Existing literature shows numerous well-established pseudo-tasks, hereafter called pretext tasks [8]. The process of pre-training through solving pretext tasks is called pretext learning. After pretext learning, the pre-trained model is trained on solving the actual task, called the downstream task, in a process called downstream learning [6, 8, 9, 11, 15, 22].

It has been well established that the semantic information of the data used by the model to solve the pretext task affects its learned features [8, 9, 15]. The latent feature representation encodes information that is used to solve the task it was trained on. In the case of pretext learning this means that non-relevant pretext tasks will potentially create features that are not beneficial, or even detrimental, for solving the downstream task [15].

Therefore, it is important to pick the right pretext task for the downstream task. However, research is lacking on how the complexity of the pretext task affects the downstream task performance. Here, complexity refers to the difficulty of the pretext tasks, often linked to the quantity and quality of features required to solve it. A more complex pretext task might negatively affect pretext task performance, but perhaps will be beneficial for solving the downstream task.
Furthermore, a lacking area of research is how the different SSL strategies (pretext task types) affect downstream task performance. Those strategies are defined based on how they augment or transform data in combination with their expected output type. A number of data transformation/augmentation techniques, categorized as the self-prediction strategy, are proposed to discuss this issue. These techniques are compared to two well-known innate relationship pretext tasks, i.e. jigsaw puzzle [15, 22] and predict rotation angle [6, 9, 22].

In essence, this research aims to shed light on the topic of how the complexity and the type of the pretext task affect the quality of the learned features after pretext learning, and thus organoid segmentation performance, with regard to the amount of data used in training. This research, thus, aims to answer the following research questions:

- How does the complexity of pretext tasks affect self-supervised learning of the segmentation of organoids?
- How does the pretext task strategy type affect self-supervised learning of the segmentation of organoids?
- What effect does the amount of training data, for both self-supervised and supervised learning, have on the quality of organoid segmentation in relation to the complexity of the pretext task and the pretext task strategy type?

## 2 LITERATURE REVIEW

Organoid segmentation, which is the downstream task for this research, is a form of semantic image segmentation. Semantic image segmentation is the process of recognizing and localizing objects in an image by classifying each pixel in the image to a specific object class from a predetermined set of classes [14].

[14] and [4] list some of the more traditional methods of image segmentation. For example, thresholding, watershed and active contours were commonly used to identify cells in an image. However, most of these traditional methods require expertise to set up and require researchers to account for imaging technique, scale and experimental conditions [4]. DL methods have proven to be a great alternative to traditional methods for semantic segmentation as they are more versatile and are more easily adapted to different experimental conditions [3, 4, 14]. Often also improving segmentation quality [4, 14].

This research uses a U-Net type architecture, which is a fully convolutional encoder-decoder type architecture, to segment the organoids from their background. Previous research successfully used a U-Net for segmentation purposes [3, 4, 14, 17]. Segmentation is useful for organoid research as it solves issues with the high dimensionality of organoid data, acquisition artefacts, low contrast, and bright-field noise [13]. DL methods used for segmentation, provide more stability and robustness at the drawback of requiring more effort to set up [16]. DL methods requiring more effort to set up refer to, among other things, requiring a large amount of annotated data. In order to address this issue this research uses a SSL approach.

SSL is the process of using pseudo-labels on unannotated data to train models to extract semantic information by creating meaningful feature representations of the input data in a process called pretext learning. After pretext learning, the pre-trained model is trained on annotated data to perform the downstream task. In this case, the research focuses on progenitor liver organoids segmentation. Through using SSL the model is able to perform better, despite being trained on limited annotated data, as the model has already learned to extract data-specific semantic information [6, 8, 9, 11, 15, 22]. As previously mentioned, there are a large number of clinical use cases per image data type. Having models pretrained on specific data could prove useful as the pretrained models can then be adapted to specific use cases. For example, models pre-trained on EEG data can be trained for emotion recognition [11], recognizing lesions [19] or analyzing sleep activity [1]. In this case, the proposed SSL approach could be adapted and used for many other research purposes in organoid research, and even biological image analysis in general.

There exists a number of categories of SSL strategies such as generative, contrastive, innate relationship and self-prediction [8]. This research focuses on innate relationship and self-prediction type strategies. In self-prediction strategies an input image is augmented and/or transformed on a portion of the image, creating a distorted image that serves as input during pretext learning. The pretext task consists of reconstructing the distorted image back to the original image, or ground truth (GT), using a reconstruction loss. The unaltered portions of the distorted image are meant to inform and aid the model in reconstructing back to GT. Alternatively, innate relationship strategies use a pretext task that has pseudo labels not related to the original data. Instead, the labels are directly related to the pretext task distortion method and expected output structure. In this case, the model uses the structural information of the data to solve the pretext task, which helps the model learn a solid feature representation.

## 3 METHODS

### 3.1 General approach

This research uses separately trained U-Net models. Before pretext learning, the encoder of the U-Net is transfer-learned from a ResNet50 model. These models were trained on self-prediction and innate relationship type pretext tasks during pretext learning. Each pretext task, for both self-prediction and innate relationship, has models trained using 10%, 30% or 50% of the available pretext training data. Each experimental condition pertaining to the pretext task and amount of data uses 5-fold cross validation. Self-prediction pretext tasks are divided into simple and complex tasks.

After pretext learning, the best performing model per aforementioned experimental condition is transferred to downstream task training. The encoder portion of the model is frozen and the model is trained on the task of organoid segmentation using either 10%, 50% or 100% of the available downstream training data. Downstream learning also uses 5-fold cross validation.

After pretext learning and downstream learning, 38 (pretext tasks) x 3 (pretext data amount) x 5 (folds) = 570 pretext trained models were obtained based on the pretext task, amount of data for pretext learning and 5-fold cross validation. Additionally, 38 (pretext tasks) x 3 (pretext data amount) x 3 (folds) = 1,026 downstream trained models were obtained based on pretext task, amount of data for pretext learning, amount of data for downstream learning and 3 rotations of 5-fold cross-validation.
These models were then tested on their performance on organoid segmentation and compared given their experimental conditions.

3.2 Data set

This research uses a data set consisting of images of progenitor liver organoids provided by the University Medical Centre Groningen (UMCG) in the Netherlands. The liver progenitor organoids were captured using light microscopes to create CZI images.

The observed organoids were grown under 2 different growing conditions: organoids grown in a complete medium for optimal growth and organoids grown in a medium without amino acids for stumped growth [7]. For each condition, 5 CZI images were taken at an interval of 24 hours for a total of 10 CZI images. Each CZI image consists of around 14 2D slices, which combined creates a 3D representation of the organoid structure. Of these 14 2D slices only the middle 4 slices showed relevant information, as the outer slices were out of focus.

The remaining 40 slices were 3828x2870 pixels in size. These large images were divided into smaller images, 636x636 pixels in size, called crops, using the sliding window method with a step increment of 60 pixels. Furthermore, images with less than 5% of relevant information were discarded. As an augmentation technique, images were rotated by 90, 180 and 270 degrees. The total data set consists of 101,589 crops.

In addition, for each organoid crop in the dataset, a corresponding mask was created using a Mask-RCNN trained on a similar dataset. These masks were used as labels in downstream task training. Having images with dimensions 636x636 pixels would be too large for a DL model to effectively handle. Therefore, before training, the images were resized to 320x320 pixels.

3.3 Model architecture

3.3.1 Data usage and implementation details. The models used in this research are trained on varying amounts of data. There are 40,631 images available for pretext learning, representing 40% of the total data set and both sets do not overlap. Each model trained on a pretext task and pretext learning data amount was then trained on the downstream task using either 10%, 50% or 100% of the available downstream task data, which was 4063, 20,318 or 40,636 images, respectively.

Similarly, downstream learning has 40,636 images available, which is 40% of the total data set. The downstream learning data set is distinct from the pretext learning data set and both sets do not overlap. Each model trained on a pretext task and pretext learning data amount was then trained on the downstream task using either 10%, 50% or 100% of the available downstream task data, which was 4063, 20,318 or 40,636 images, respectively.

The above-mentioned training was also done using k-fold cross-validation, where k=5. 5-fold cross-validation is the process of switching the validation set of the training data 5 times over the whole training data set. Using this method ensures that the division of training-validation data does not trap the model in a local minimum. The downstream task was divided using 5-folds, i.e. 20% validation data, but only swapped 3 times resulting in 3 models rather than 5. Figure 1 shows the division of data for pretext learning and downstream learning. The red vertical lines mark the amount of data used for training, and the curved arrows indicate the 5-fold cross-validation.

3.3.2 Loss functions. The Structural Similarity Index Measure (SSIM), used as a performance metric and reconstruction loss for self-prediction pretext learning, is a metric for comparing a reconstructed predicted image to the GT image [23]. The metric outputs a value ranging from [-1, +1], where +1 represents perfect similarity and -1 represents extreme dissimilarity. Rather than comparing pixels from two images, SSIM compares patches of the reconstructed image to corresponding patches of the GT. Using this pixel neighbourhood approach ensures a more human method of comparing image quality. The SSIM formula is a combination of calculating luminance, contrast and structure [23]:

\[
SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\mu_x^2 + \mu_y^2 + \sigma_x^2 + \sigma_y^2 + C_2}
\]

where \(x\) represents an image (or patch of an image) and \(y\) represents the corresponding GT image (again, or patch). \(\mu_x\) and \(\mu_y\) are the average pixel values of \(x\) and \(y\), respectively. \(\sigma_x^2\) and \(\sigma_y^2\) are the pixel value variances of \(x\) and \(y\), respectively.

\[
C_1 = (K_1L)^2
\]

\[
C_2 = (K_2L)^2
\]

Where \(K_1, K_2\) are constants to avoid a weak denominator, i.e. division by zero error, often \(K_1 = 0.01, K_2 = 0.03\). \(L\) is the dynamic range of pixel values, i.e. 255 in this case. \(\sigma_{xy}\) is the covariance of \(x\) and \(y\).

\[
\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y)
\]

Where \(N\) is the total number of pixels in the image(-patch).

Generally, the SSIM is calculated for sections of the image and in the end the global mean is calculated for the complete image [23]. The loss function is calculated using the formula:

\[
LossSSIM(x, y) = 1 - SSIM(x, y)
\]

The Sparse Categorical Cross Entropy (SCCE) is used as a classification loss for both the jigsaw puzzle and prediction rotation.
angle innate relationship pretext task. The SCCE is based on the Categorical Cross Entropy (CCE) loss function [20]:

\[
CCE(p, q) = - \sum_{x=1}^{N} p(x) \cdot \log q(x)
\] (6)

Where \( p \) is the label vector containing value 1 for the correct class and 0 for other classes. \( q \) is the predicted softmax class probabilities. \( N \) represents the number of classes. SCCE, as opposed to CCE, uses an integer value to represent the class label (e.g. [2]) where CCE uses a hot encoded label vector (e.g. [0,0,1,0]).

The Jaccard Distance is used as a segmentation loss for downstream learning and is based on the Jaccard Index, or Intersection over Union (IoU). IoU is a metric used to compare two images on the precision of their classification. The formula for the IoU is:

\[
IoU(x, y) = \frac{|x \cap y|}{|x| + |y| - |x \cap y|}
\] (7)

\( x \) represents an image, \( y \) represents the corresponding reference image.

The Jaccard Distance uses the IoU to express it as a loss function:

\[
JD(x, y) = 1 - \frac{|x \cap y|}{|x| + |y| - |x \cap y|}
\] (8)

3.3.3 Optimizer. Adam, or Adaptive Moment Estimation, was used as the optimizer for both pretext learning and downstream learning. Adam is an optimizer used to update learning rates over training steps [10]. Adam can be seen as an extension to stochastic gradient descent. Classical stochastic gradient descent has a single learning rate for all parameters in the model, but Adam has individual learning rates for all parameters [18]. Adam adapts learning rates based on the average first moment, the mean, as well as the average of the second moments of the gradients, the uncentered variance [18]. Overall, it is recommended to use Adam as the optimizer for most cases [18].

3.4 Pretext task

3.4.1 Self-prediction. The self-prediction SSL strategy is based on reconstructing distorted images to the original GT image using the unaltered portions of the image as information for reconstruction. In essence, images distorted using transformation techniques served as input paired with the GT as labels. Eight self-prediction transformation techniques are proposed:

- Blur (b): uses Gaussian blur to blur complete image;
- Drop (d): makes 4 randomly positioned boxes of 50x50 pixels black;
- Shuffle (s): swaps 4 randomly positioned boxes of 50x50 pixels;
- Rotate (r): rotates 4 randomly positioned circles with radius of 25 pixels by a random degree;
- Blur boxes (B): uses Gaussian blur to blur 4 randomly positioned boxes of 50x50 pixels;
- Drop pixels (D): turns 25% of pixels in 4 randomly positioned boxes of 50x50 pixels black;
- Shuffle and rotate (S): rotates and swaps 4 randomly positioned circles with radius of 25 pixels;
- Rotate boxes (R): rotates 4 randomly positioned boxes of 50x50 pixels by either 90, 180 or 270 degrees.

In this research, a pretext task using a singular transformation technique is considered a simple pretext task. A combination of two transformation techniques is considered a complex pretext task. The proposed pretext tasks, both simple and complex, are shown in Figure 2. Four sections of 50x50 pixels were distorted with no overlap per distortion technique. The distortion of four sections was chosen as it would be likely to distort an organoid in the image as well as leave sufficient information for reconstruction to GT.

3.4.2 Innate relationship. The innate relationship SSL strategy uses pseudo-labels generated based on the pretext task, rather than data dependent labels [8]. This research implements two of the most popular innate relationship strategies: solving a jigsaw puzzle [15, 22] and predicting image rotation angle [6, 22]. It is important to note that these pretext tasks are essentially (multi-class) classification tasks, rather than reconstruction tasks. Figure 3 shows examples of distorted images used as pretext task input for both innate relationship pretext tasks. As explained by [22], using a predict rotation angle pretext task on texture type data does not lead to good results. Moreover, rotation was also used as an augmentation technique for the data set. Therefore, the rotation is done on a section of the image. The pretext task consists of predicting the correct rotation angle of this section.

4 RESULTS

4.1 How does the complexity of pretext tasks affect self-supervised learning of the segmentation of organoids?

Figure 4 shows box plots of pretext performance of simple and complex self-prediction models measured using SSIM. These box plots include models of all pretext training data amounts. The models trained on simple tasks have overall higher pretext performance (\( \mu_{SSIM} = 0.960 \pm 0.024 \)) compared to models trained on complex tasks (\( \mu_{SSIM} = 0.951 \pm 0.027 \)). Similarly, Figure 5 shows SSIM pretext performance of simple and complex models further separated by pretext training data amount. It can be seen that simple self-prediction models consistently outperform complex self-prediction models on SSIM pretext task performance for all pretext training data amounts. These results show that models trained on reconstructing a single distortion to GT, rather than a combination of two distortions, have reconstructions more similar to GT. In turn, this suggests that simple tasks are overall less complex for the model to learn to solve compared to complex tasks, regardless of the variability of the complexity of pretext tasks.

After downstream learning, self-prediction models were tested on their downstream task performance. The F1-score was calculated by comparing predicted test set segmentation masks to their respective GT segmentation masks. Figure 6, using SSIM pretext performance, shows models of all training data amounts placed using aforementioned pretext performance as the x-axis and downstream F1-score performance as the y-axis separated by...
Figure 2: Top table presents proposed distortion methods of simple pretext tasks. Bottom table presents combinations of aforementioned pretext tasks, named complex pretext tasks.
simple and complex using color. The trend lines represent linear downstream F1-score change based on pretext performance, again, divided into simple and complex using color. There appears to be a slight rate of change in downstream performance given pretext performance for both the simple and complex conditions for SSIM pretext performance. These slopes, however, are very minimal. Overall, a Pearson correlation test on pretext SSIM and downstream F1-score shows a coefficient of 0.0514 with p-value of 0.356. Low coefficients and high p-values suggest no correlation between pretext performance and downstream performance. In addition, similar Pearson correlation tests on simple SSIM (coef. = 0.2194, p-value = 0.063), and complex SSIM (coef. = 0.0331, p-value = 0.601) show that no significant correlation was found. Therefore, pretext performance measured in SSIM does not seem to correlate with downstream performance measured in F1-score. In other words, results suggest that the complexity of the pretext task does not directly correlate to any difference in downstream performance.
Table 1: Average downstream F1-score for simple and complex models and amount of data used in training. Rows represent the amount of downstream training data, columns represent the amount of pretext training data, both are grouped by simple and complex self-prediction models. The overall block shows the average downstream F1-score over all training data amounts.

<table>
<thead>
<tr>
<th></th>
<th>10% pretext</th>
<th>30% pretext</th>
<th>50% pretext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple tasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10% downstream</td>
<td>0.831 ± 0.022</td>
<td>0.824 ± 0.030</td>
<td>0.770 ± 0.145</td>
</tr>
<tr>
<td>50% downstream</td>
<td>0.864 ± 0.011</td>
<td>0.865 ± 0.013</td>
<td>0.867 ± 0.022</td>
</tr>
<tr>
<td>100% downstream</td>
<td>0.868 ± 0.024</td>
<td>0.877 ± 0.013</td>
<td>0.866 ± 0.021</td>
</tr>
<tr>
<td>Complex tasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10% downstream</td>
<td>0.836 ± 0.014</td>
<td>0.828 ± 0.022</td>
<td>0.830 ± 0.019</td>
</tr>
<tr>
<td>50% downstream</td>
<td>0.871 ± 0.013</td>
<td>0.850 ± 0.010</td>
<td>0.870 ± 0.030</td>
</tr>
<tr>
<td>100% downstream</td>
<td>0.876 ± 0.021</td>
<td>0.876 ± 0.038</td>
<td>0.888 ± 0.017</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple tasks</td>
<td>0.848 ± 0.059</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex tasks</td>
<td></td>
<td>0.862 ± 0.031</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7: Box plots showing average downstream F1-score performance of self-prediction models separated into simple (red) and complex (blue).

Table 2: Average downstream F1-score performance for all pretext distortion methods and amount of data used in training. Rows show self-prediction, simple and complex combined, models grouped by distortion method used in the pretext task, e.g. pretext task b_d_4 is counted as blur (b) and drop boxes (d) distortion methods. Furthermore, performance was separated over columns based on pretext training data mount and grouped based on downstream training data amount. The block showing overall performance is the average downstream performance of distortion methods over all pretext and downstream training data amounts.

<table>
<thead>
<tr>
<th>Distortion method</th>
<th>Downstream 10%</th>
<th>Downstream 30%</th>
<th>Downstream 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pretext 10%</td>
<td>Pretext 30%</td>
<td>Pretext 50%</td>
</tr>
<tr>
<td>b</td>
<td>0.830 ± 0.126</td>
<td>0.822 ± 0.135</td>
<td>0.779 ± 0.137</td>
</tr>
<tr>
<td>d</td>
<td>0.843 ± 0.119</td>
<td>0.838 ± 0.118</td>
<td>0.837 ± 0.126</td>
</tr>
<tr>
<td>s</td>
<td>0.837 ± 0.125</td>
<td>0.825 ± 0.130</td>
<td>0.830 ± 0.126</td>
</tr>
<tr>
<td>r</td>
<td>0.839 ± 0.124</td>
<td>0.829 ± 0.131</td>
<td>0.836 ± 0.127</td>
</tr>
<tr>
<td>S</td>
<td>0.842 ± 0.124</td>
<td>0.833 ± 0.127</td>
<td>0.832 ± 0.127</td>
</tr>
<tr>
<td>R</td>
<td>0.833 ± 0.127</td>
<td>0.839 ± 0.121</td>
<td>0.832 ± 0.132</td>
</tr>
<tr>
<td>D</td>
<td>0.834 ± 0.125</td>
<td>0.819 ± 0.137</td>
<td>0.819 ± 0.136</td>
</tr>
<tr>
<td>B</td>
<td>0.825 ± 0.126</td>
<td>0.818 ± 0.132</td>
<td>0.819 ± 0.133</td>
</tr>
</tbody>
</table>

Self-prediction pretext tasks were complex pretext tasks (ranked top 15/36). In addition, despite blur (b) being the worst distortion method, there are a number of best-performing pretext tasks that use the blur distortion method.

Figure 8 shows examples of organoid segmentation using the 50% pretext - 100% downstream data amount models from the top three best and three worst performing self-prediction pretext tasks. The predicted mask of the best-performing models is closer to the true mask with higher F1-scores compared to the predicted masks of the worst-performing models. These examples show that the difference in F1-scores of predicted masks is mostly caused by segmenting (med F1 = 0.870) compared to simple models (med F1 = 0.862). In addition, complex models had higher performers. 5.95% of complex models outperformed the best performing simple model.

Analyzing tasks using a specific pretext distortion method, both simple and complex, proves useful for understanding the best pretext distortion method for downstream performance. Table 2 shows the average and standard deviation of downstream F1-score performance of collections of tasks using a specific distortion method. All self-prediction tasks, both simple and complex, are included in the average downstream F1-score performance of a distortion method. Drop boxes (d) (µF1 = 0.866 ± 0.020) and rotate circles (r) (µF1 = 0.866 ± 0.022) distortion methods were the two best performing distortion methods. Blur (b) (µF1 = 0.850 ± 0.035) was overall the worst performing distortion method.

Figure 9 shows the downstream F1-score performance of all tasks, both simple and complex, and their ranking from highest to lowest F1-score median. Notably, most of the top-performing
organoids that are not present in the true mask as well as not segmenting organoids that are in the true mask. Since organoids are a 3D structure, some should be ignored since they’re out of focus or unhealthy. In addition, masks of segmented organoids of the worst performing models are more jagged and less polished.

To summarize, although there is a high variability among the pretext models in complexity, complex models are indeed overall more complex for the model to learn to solve compared to simple models. In addition, results show that complex models outperform simple models on downstream F1-score performance of the segmentation of organoids. Moreover, complex models are ranked higher in median downstream performance compared to simple models. Although this effect is consistent through multiple visualizations, the effect size of the difference in downstream performance is small. Given that there appears to be no correlation between pretext performance and downstream performance, it cannot be concluded that the complexity of the pretext task directly relates to any difference in downstream performance. Differences between simple and complex downstream performance are therefore most likely due to the effect of (combinations of) distortion methods on learned features, rather than the complexity of the pretext tasks.

4.2 How does the pretext task strategy type affect self-supervised learning of the segmentation of organoids?

Figure 10 shows box plots of downstream F1-score performance separated by SSL strategy. The self-prediction strategy ($\mu F1 = 0.859 \pm 0.112$) seems to outperform the innate relationship strategy ($\mu F1 = 0.848 \pm 0.118$). The difference in median appears relatively small, however, self-prediction has better high-performing models while innate relationship has worse low-performing models. 18.5% of self-prediction models outperform the best innate relationship model and 5.6% of innate relationship models perform worse than the worst performing self-prediction model. Figure 11 shows the

<table>
<thead>
<tr>
<th>Downstream 10%</th>
<th>Innate relationship</th>
<th>Jigsaw</th>
<th>Rotation prediction</th>
<th>Pretext 10%</th>
<th>Pretext 30%</th>
<th>Pretext 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jigsaw</td>
<td>$0.800 \pm 0.153$</td>
<td>$0.809 \pm 0.147$</td>
<td>$0.816 \pm 0.125$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation prediction</td>
<td>$0.823 \pm 0.141$</td>
<td>$0.808 \pm 0.142$</td>
<td>$0.740 \pm 0.175$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Downstream 50%</td>
<td>Innate relationship</td>
<td>$0.863 \pm 0.093$</td>
<td>$0.877 \pm 0.095$</td>
<td>$0.875 \pm 0.097$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jigsaw</td>
<td>$0.864 \pm 0.095$</td>
<td>$0.869 \pm 0.095$</td>
<td>$0.863 \pm 0.119$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation prediction</td>
<td>$0.864 \pm 0.095$</td>
<td>$0.869 \pm 0.095$</td>
<td>$0.863 \pm 0.119$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>Innate relationship</td>
<td>$0.854 \pm 0.036$</td>
<td>$0.882 \pm 0.114$</td>
<td>$0.843 \pm 0.119$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jigsaw</td>
<td>$0.854 \pm 0.036$</td>
<td>$0.882 \pm 0.114$</td>
<td>$0.843 \pm 0.119$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation prediction</td>
<td>$0.882 \pm 0.114$</td>
<td>$0.843 \pm 0.119$</td>
<td>$0.843 \pm 0.119$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
average downstream F1-score performance for the Jigsaw puzzle and the rotation angle prediction pretext tasks. Models trained on the Jigsaw puzzle (j) (median $F_1 = 0.875$) outperformed the models trained on Predict rotation angle (rp) (median $F_1 = 0.863$). Similarly, Table 3 shows the average and standard deviation of downstream F1-score performance for all pretext and downstream data amounts. Again, models trained on the Jigsaw puzzle (j) ($\mu F_1 = 0.854 \pm 0.036$) outperformed models trained on Predict rotation angle (rp) ($\mu F_1 = 0.842 \pm 0.047$) on average downstream performance. The box plot showing all tasks, Figure 9, shows that both Jigsaw (j) and Predict rotation angle (rp) performed decent. And again, the Jigsaw (j) (ranked 11/38) clearly outperformed Predict rotation angle (rp) (ranked 26/38) in terms of ranking.

To summarize, results show that the Jigsaw puzzle (j) outperformed Predict rotation angle (rp). More importantly, the self-prediction strategy outperformed the innate relationship strategy on average downstream F1-score performance. Self-prediction had better high performing models, where innate relationship had worse low performing models. Despite both Jigsaw and Predict rotation angle models performing decent, they still had worse performance compared to most self-prediction models.

4.3 What effect does the amount of training data, for both self-supervised and supervised learning, have on the quality of organoid segmentation in relation to the complexity of the pretext task and the pretext task strategy type?

Both simple/complex self-prediction models and innate relationship models were trained on varying amounts of pretext and downstream training data. For each pretext task, pretext training data amount and downstream training data amount the downstream F1-score performance of the model was tested. Figure 12 shows downstream F1-score performance on the y-axis and downstream training data amount on the x-axis, further separated by pretext training data amount using color. These box plots show that models performed better with more downstream training data regardless of pretext training data. 10% downstream training data ($\mu F_1 = 0.825 \pm 0.129$), 50% downstream training data ($\mu F_1 = 0.872 \pm 0.109$) and 100% downstream training data ($\mu F_1 = 0.878 \pm 0.098$) have continuously increasing downstream F1-score performance. In terms of pretext training data, adding more data generally benefitted downstream performance on the condition that there was a decent amount of downstream training data available. Considering 50% and 100% downstream training data, 10% pretext training data ($50\% \mu F_1 = 0.869 \pm 0.109$, $100\% \mu F_1 = 0.875 \pm 0.097$), 50% pretext training data ($50\% \mu F_1 = 0.876 \pm 0.107$, $100\% \mu F_1 = 0.876 \pm 0.101$) and 50% pretext training data ($50\% \mu F_1 = 0.870 \pm 0.112$, $100\% \mu F_1 = 0.882 \pm 0.096$) have almost all increasing downstream F1 score performance with more pretext training data. With 10% downstream training data, 10% pretext training data ($\mu F_1 = 0.834 \pm 0.126$), 30% pretext training data ($\mu F_1 = 0.826 \pm 0.130$) and 50% pretext training data ($\mu F_1 = 0.815 \pm 0.132$) average downstream F1-score performance continuously decreased. This suggests that an increase in pretext training data, given too little downstream training data, is detrimental to downstream performance. This same effect is visualized in Figure 12.

Figure 13 shows self-prediction models with downstream F1-score performance on the y-axis and pretext training data amount on the x-axis, separated by downstream training data amount into multiple plots, and lastly, separated into simple and complex using color. This Figure shows the same effect previously discussed. A large amount of pretext training data is detrimental.
given little downstream training data, see Table 1. In addition, for all pretext and downstream training data combinations models trained on complex tasks outperformed simple tasks, again, see Table 1. This adds credibility to the claim that complex models outperformed simple models, as this effect is constant over all pretext and downstream training data combinations. Adding credibility to the claim that self-prediction is a better suited SSL strategy for organoid segmentation compared to the innate relationship strategy. Notably, models trained using the innate relationship strategy performed significantly worse than models trained using the self-prediction strategy when there is little downstream training data available. Table 3 shows that with 10% downstream training data the Jigsaw puzzle \((j)\) downstream performance increased with more pretext training data \((\mu F_1 = 0.800 \pm 0.153\) to \(0.816 \pm 0.125\)), but the Predict rotation angle \((rp)\) downstream performance sharply declined with more pretext training data \((\mu F_1 = 0.823 \pm 0.141\) to \(0.740 \pm 0.175\)).

To summarize, results show that complex self-prediction models outperformed simple self-prediction models for all training data amounts. The consistency of complex models outperforming simple models adds credibility to the claim that complex models are better suited for pretext learning with the intention of organoid segmentation compared to simple models. Results also show that self-prediction downstream performance mostly outperformed innate relationship downstream performance for all training data amounts. The consistency of the self-prediction strategy outperforming the innate relationship strategy adds credibility to the claim that the self-prediction strategy of pretext learning is better suited for organoid segmentation. Lastly, more pretext training data is beneficial to downstream performance given a minimum amount of downstream training data. For 10% downstream training data, more pretext training data negatively impacted downstream performance. Especially so for the Predict rotation angle \((rp)\) innate relationship pretext task.
ACKNOWLEDGMENTS

We thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high-performance computing cluster. We also thank the University Medical Center Groningen (UMCG) for sharing with us the data.

6 FUNDING

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 812968.

REFERENCES