Data Augmentations for Cell-Type Generalizable Instance Segmentation

Julie Chen
Stanford University
jchen80@stanford.edu

Abstract

Quantifying cell abundance through computer vision approaches in an instance segmentation task can greatly streamline cell-based assays and workflows for researchers in the biological sciences. Instance segmentation of cells in phase microscopy images, a relatively non-disruptive method of imaging live cells, is thus an important and high-demand task in computer vision. Recent work and the generation of a large-scale labeled dataset have greatly expedited progress on this task. However, given the highly heterogenous and diverse array of cell types and growth conditions, there still exists a need for generalizable models that can transfer and perform well on data from a cell type or condition not encountered in the training dataset. In this paper, I explore the usage of data augmentations to improve model performance on test data drawn from previously unseen cell types. In my experiments, I find that improvement in the transfer task following single data augmentation varies greatly depending on the transformation applied and the testing cell type. Pooling together transformations through stacking or random selection is able to yield augmentation strategies with much more generalizable improvements on the transfer task across all or nearly unseen cell types.

1. Introduction

Cell instance segmentation of phase contrast microscopy images through computer vision is a labor-efficient way of streamlining experiments in biological sciences research. Many experimental setups use cell abundance or survival as a functional readout following an induced stressor or treatment condition, for example when validating hits in a CRISPR genetic knockout screen. However, manual segmentation and/or counting of cell numbers is a time-consuming process. As such, various computer vision approaches have been applied to this task, often using models based on state of the art instance segmentation and techniques such as CNNs [13], Cascade Mask R-CNN [5] and U-Net [12].

Phase contrast microscopy is a method that allows live cells to be visualized with relatively minimal disruptions to their native state. However, unlike in fluorescence microscopy where cellular features are brightly demarcated by stains with fluorescent dyes or via expression of fluorescent marker proteins, phase contrast microscopy images are much lower in contrast, which increases the difficulty in performing the instance segmentation task. Another major challenge in instance segmentation of cells is the diversity in cell shapes, morphology and density. Certain cell types have highly globular, ordered shape while others take on an elongated, fibrotic appearance. Moreover, cell morphology is a highly dynamic quality that can undergo significant variation under different biological conditions such as growth, apoptosis or certain drug treatments. Given the labor intensive process of annotating cell images, there exists a need for highly generalizable instance segmentation performance which can accomodate for the wide diversity of cellular morphologies. This would allow for researchers to use pre-trained instance segmentation models with images taken from different cell lines or growth conditions without the need for costly hand-annotation.

Data augmentation has emerged as a promising method to enhance generalizability and robustness of deep learning models, particularly when datasets are relatively small in size. In computer vision tasks, this typically involves various geometric and color transformations being applied to the image to generate new training examples. In this paper, I explore the usage of data augmentation techniques to improve generalizability across cell types for the instance segmentation task.

2. Related Works

2.1. Cell instance segmentation approaches

Existing work to improve generalizability of cell instance segmentation performance relies heavily on the curation of large labeled datasets. For example, in CellPose, Stringer *et al.* proposed the usage of a community-sourced data approach that utilizes a large and highly heterogenous dataset of cell images with the goal of training a generalizable cell instance segmentation model [12]. However, the image dataset used largely consists of images obtained through fluorescence microscopy and contains relatively few phase contrast microscopy images.

A recent advance in improving instance segmentation of phase microscopy images was the curation of a large labeled dataset called LiveCell [5]. This dataset consists of over 1.6 million total examples drawn from 8 different cell types of diverse morphologies imaged at various cell densities [5]. The authors trained two different models based on Cascade R-CNN and CenterMask architectures respectively on an instance segmentation task. The model input consists of phase microscopy labeled images of cells and the output is bounding boxes and instance segmentation masks of predicted cell instances. Overall, both models performed well on the cell instance segmentation task with an average precision score of 47.8 and 47.9% with each respective underlying architecture, Cascade Mask R-CNN or CenterMask [5]. The authors also trained the models on subsets of the image dataset consisting of only a single cell type to assess how well the models were able to transfer knowledge learned to perform the segmentation task on a different cell type [5]. Unsurprisingly, transferability was highly variable with test performance on certain single cell types varying greatly depending on the training set cell type [5].

2.2. Data augmentations

Data augmentations are a powerful and widely used technique in computer vision tasks to prevent model overfitting and increase generalizability of learning by increasing the size of the dataset and providing new views of the the same training examples [11]. Commonly used methods to augment computer vision image datasets include both color and geometric transforms such as cropping, flipping, scaling, blurring and color jittering [11]. For the task of instance segmentation specifically, additional techniques have been proposed such as cut-and-paste strategies; Ghiasi *et al.* [6] demonstrate that a simple copy-and-paste approach, which operates in a context-independent manner, is able to achieve robust improvements on both COCO and LVIS object detection benchmarks.

Naturally, automated approaches for data augmentation are of great interest. One such approach is generative adversarial networks (GANs), where the objective is to minimize loss between the generated augmented images and the original image dataset through a paired generator and discriminator network module [7]. However, this technique is difficult to adapt for image segmentation since the new images generated would lack bounding boxes and segmentation masks. Although this could be countered through an unsupervised clustering approach to produce labels, like used in Jain *et al.* [9], this augmentation strategy is problematic because it ultimately does not make use of the ground truth label. Another approach is to try to learn the best data augmentation policy, i.e. what sequence or combinations of transformations will be most useful. Ratner *et al.* use a generative approach to learn policies that produce augmented images with a distribution that is similar to the training dataset [10]. In AutoAugment [4], the authors instead use a reinforcement learning approach to explore a search space of data augmentation policies, with the aim of finding a policy that generates augmented data to allow the model to achieve the best accuracy on the validation dataset. This same approach was also extended to object detection tasks by Zoph *et al.* [15].

3. Dataset

I used the publicly available LiveCell dataset collected by Sartorious which includes over 1.6 million labeled cell instances gathered across 8 cell types, available here: `https://github.com/sartorius-research/LIVECell`. Each cell instance was hand-annotated with a bounding box and segmentation mask label and an example from each cell type class with the mask annotation is shown in Figure 4.

4. Methods

4.1. Models

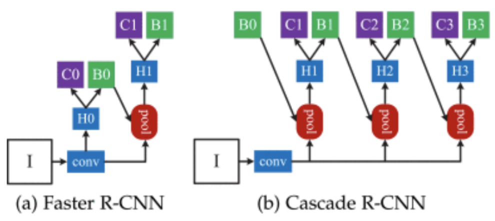

Figure 1. Comparison of Faster R-CNN and Cascade R-CNN architectures, reproduced from [2])

I used models from Edlund *et al.* [5] which employ a Cascade Mask R-CNN architecture [2] with a ResNeSt backbone [14] for image feature extraction. Although the performances of the two model architectures used by Edlund *et al.* were generally comparable, I chose to use the Cascade Mask R-CNN for the following experiments because it displayed better performance in the single-cell train and evaluate task. Briefly, Cascade Mask R-CNN is a variant on the Mask R-CNN approach, which uses an underlying Faster R-CNN network architecture [2]. Like Mask R-CNN, Cascade Mask R-CNN similarly uses an image feature extraction backbone network, which is then coupled with a region proposal network (RPN) to suggest regions of interest which are then fed into a fully connected network which outputs class labels, bounding boxes and mask predictions [2]. However, Cascade Mask R-CNN uses a series of RPN-FC networks with increasing IoU thresholds, where a bounding box or mask segmentation prediction is considered a ”positive” or ”correct” prediction if its IoU (intersection over union) with the ground truth label exceeds a given IoU threshold [2] (Figure 1).

4.2. Data Augmentations

I used the CLoDSA [3] and Albumentation [1] augmentation libraries and an open-source implementation of the simple copy paste transformation [6]. A brief description of each data augmentations used is written below. Figure 2 contains select examples of some of the naive spatially variant augmentation strategies.

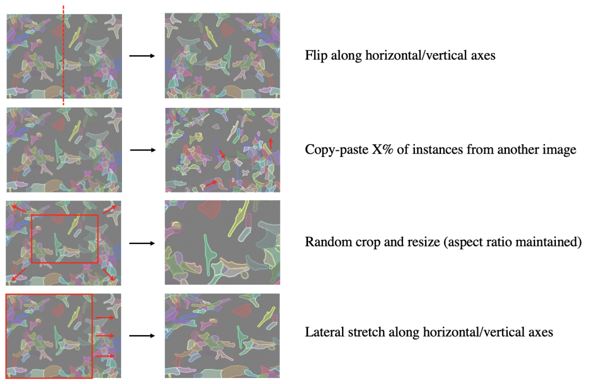
Flip along horizontal/vertical axes
Copy-paste X% of instances from another image
Random crop and resize (aspect ratio maintained)
Lateral stretch along horizontal/vertical axes

Figure 2. Examples of spatially variant data augmentation strategy methods (flip, simple copy paste, random crop resize, lateral stretch)

4.2.1 Flip

Flip image, bounding boxes and segmentation masks across either horizontal or vertical axes.

4.2.2 Lateral stretch

Crop along horizontal or vertical dimension of image by a random scaling factor, then resize to original starting dimensions, achieving a lateral stretch transformation.

4.2.3 Crop

Perform a random crop along both axes of image by a random scaling factor, then resize to original starting dimensions, maintaining original aspect ratio.

4.2.4 Color jitter

Randomly change brightness, contrast, and saturation of image. Bounding boxes and mask segmentation labels are unaffected by this transformation.

4.2.5 Simple copy paste

For each image i_j, choose an image i_k at random from the training dataset. Then, select cell instances i_{k_n} from i_k with a probability $p = 0.25$. For all instances i_{k_n}, copy each instance's image representation, bounding box and segmentation mask onto the image, bounding box and mask of the original image i_j.

4.2.6 Random scaled copy paste

All images in the dataset are randomly scaled prior to the simple copy paste procedure outlined in 4.2.5 to achieve cut-and-paste from variably sized source images.

4.2.7 Stacked augmentations

Each image was passed through a sequence of all naive data transformations with a tunable parameter p, where a given transformation is applied to the image with a probability p.

4.2.8 Random choice augmentations

Out of the set of all naive data transformations, one is chosen at random and applied to the data example.

4.2.9 Automated augmentation selection

I used a generative approach over the set of data transformations similar to that outlined in [10]. Here, the objective is to minimize the loss between augmented data produced by the proposed augmentation policy and the original training data. The exact implementation is based off of a compute-efficient approach called FasterAutoAugment implemented by Hataya *et al*. [8] and is available through the open source Albumentations library as AutoAlbument [1]. Similar to AutoAugment, FasterAutoAugment explores a search space of sub-policies, or image transformations, which have differentiable parameters such as the magnitude μ or probability of being applied p [8].

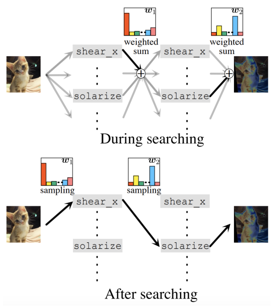

Figure 3. FasterAutoAugment search strategy

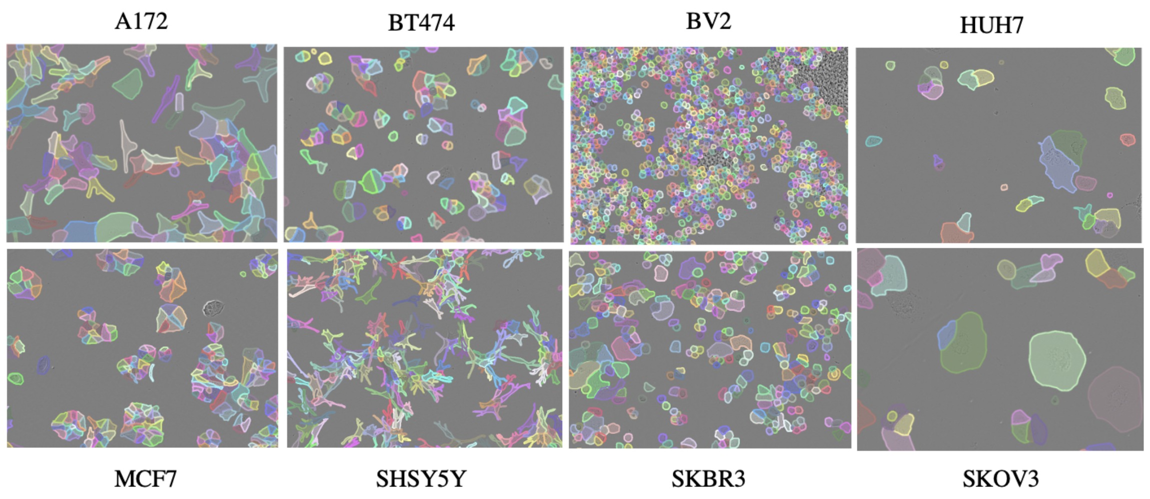

Figure 4. Example image with instance segmentation mask of each cell type

During training, each transformation is applied to the image with a learnable weight and the resulting augmented image is the weighted sum of all the transformed images [8]. Then, using the learned weights, the model can then ultimately choose the best policy of transformations to apply (Figure 3) [8]. The main advantage of FasterAutoAugment is that the authors make certain approximations to allow for backpropagation through image transformations that are not normally differentiable [8]. FasterAutoAugment applies a generated augmentation policy and then computes a loss against the distribution of produced images and ground truth images, which is then backpropagated through the model [8].

Since the library was only written for classification and semantic segmentation tasks, I decided that modifying my input data to a semantic segmentation mask format was the best compromise. Since I only have one object class, a classification-centered augmentation approach would be unlikely to be successful. On the other hand, I hypothesized that changing the instance segmentation masks to a binary semantic segmentation mask might still allow the generative model to learn augmentations that preserve some information about spatial segmentation of the image.

4.3. Image feature analysis

To supplement secondary analyses of the performance of different data augmentations, I generated size and convexity distributions of cell instances for each of the different cell types. I used the test dataset annotation labels to find the area of each mask and the number of vertices in the polygon segmentation mask for each instance and plotted distributions of each cell instance feature.

5. Results

For my experiments, I chose a model pre-trained on the dataset containing images of only the A172 cell line using the Cascade Mask R-CNN architecture described above. I chose this cell line for the pre-trained model because it demonstrated the best transfer performance when evaluated on the other cell type datasets. I used the same hyperparameters as those used during pre-training, froze the backbone parameters and fine tuned the pre-trained model on an augmented training dataset with single cell type images for 15 epochs. The A172 single cell type fine-tuned model is then evaluated at test time on unaugmented test datasets for each of the 8 cell types (Figure 5).

I report the average precision at an IoU threshold of 0.5 (AP50) for the mask segmentation as a metric for comparison between the various augmentation approaches. The baseline model is one that is pre-trained on the unaugmented training dataset alone on the A172 single cell type images. The post-augmentation training dataset size was kept constant at 3-fold the original training set size and conserved a full copy of unaugmented images.

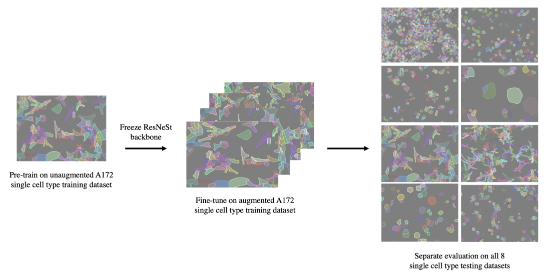
Pre-train on unaugmented A172 single cell type training dataset
Freeze ResNeSt backbone
Fine-tune on augmented A172 single cell type training dataset
Separate evaluation on all 8 single cell type testing datasets

Figure 5. Experimental pipeline

Average Precision @IoU=0.5	None	Lateral stretch	Flip	Simple copy paste	Scaled copy paste	Crop	Color jitter
A172	73.76	**74.09**	73.51	71.94	73.32	**73.86**	72.54
BT474	47.55	**49.9**	**53.3**	**54.52**	**50.41**	**49.35**	**55.3**
BV2	62.33	**62.46**	**66.70**	**66.33**	60.27	61.23	**62.35**
HUH7	72.32	**73.57**	**73.49**	72.21	**73.74**	**74.73**	71.66
MCF7	41.51	**48.04**	**47.49**	**47.38**	**45.36**	**45.32**	**50.19**
SHSY5Y	41.66	**45.11**	**47.03**	**45.78**	**43.79**	**43.74**	**45.92**
SKBR3	87.01	**87.15**	**88.13**	**88.01**	**87.65**	86.87	**88.17**
SKOV3	83.51	**84.73**	82.69	80.51	83.50	**84.24**	81.29

Table 1. AP50 segmentation mask results for single augmentation strategies compared to unaugmented baseline performance. A172 trained model evaluated across all 8 single cell types

Average Precision @IoU=0.5	None	Stacked p=0.5	Stacked p=0.8	Random choice	AutoAlbument
A172	73.76	**73.78**	71.77	73.25	71.61
BT474	47.55	**53.93**	**49.29**	**51.89**	**57.62**
BV2	62.33	**62.36**	49.83	**63.19**	60.25
HUH7	72.32	**75.61**	72.45	**73.33**	71.67
MCF7	41.51	**47.98**	**42.79**	**49.93**	**48.67**
SHSY5Y	41.66	**43.36**	39.77	**46.92**	41.47
SKBR3	87.01	**88.09**	86.54	**88.03**	**87.72**
SKOV3	83.51	**83.74**	81.07	82.95	79.96

Table 2. AP50 segmentation mask results for combined naive augmentation and autonomous augmentation search strategies compared to unaugmented baseline performance. A172 trained model evaluated across all 8 single cell types

5.1. Naive single augmentation strategies

Various geometric and color transformations were applied to the A172 training image dataset, with variable improvements in transfer performance onto previously unseen cell type image data. The one augmentation technique that seems to improve test performance across all of the single cell type tasks is the lateral stretch transformation. I would speculate that this distortion of original aspect ratio likely helps to generalize to different morphologies displayed by other cell types. Interestingly, many of the transformations did not improve the test performance on the training cell type (A172) despite significant improvements for transfer learning onto other cell types. As such, this suggests that the augmentation seems to mostly provide cell type generalizable information about segmenting cell instances and not just creating new examples of the same cell type class. However, this result is particularly interesting for the flip transformation since each individual image's instances are altered only positionally and not through any other geometric transformation. This suggests that simply increasing the number of training examples of a single cell type, in the absence of more aggressive distortions or cut-and-paste techniques, generally helps improve performance when the model is transferred to new cell types.

5.2. Naive combined augmentation strategies

Given that different transformation worked well for enhancing transfer onto different novel cell types, I hypothesized that stacking or combining the single augmentation approaches would provide better holistic enhancement coverage across all the cell types. I tried applying augmentations sequentially with a probability $p = 0.5$ and $p = 0.8$. The higher p value ablated the improved transfer performance across all cell types; possibly this may be due to repeated application of resizing transformations that might too severely alter magnify or distort cell instances. For example, BV2 cells have a relatively smaller and less convex shape compared to the A172 training cell type (Figure 6) which may explain why the stacked augmentation approach with higher p values actually performs much worse than the baseline unaugmented approach. To counter the potentially deleterious effects of applying multiple transformations to the same image while still pooling , I tried a random choice approach where a single transformation is selected at random and applied to the image.

5.3. Automated augmentation strategies

Given the growing body of literature exploring learned data augmentation policies, I hypothesized that using a similar approach would more thoroughly search the sample

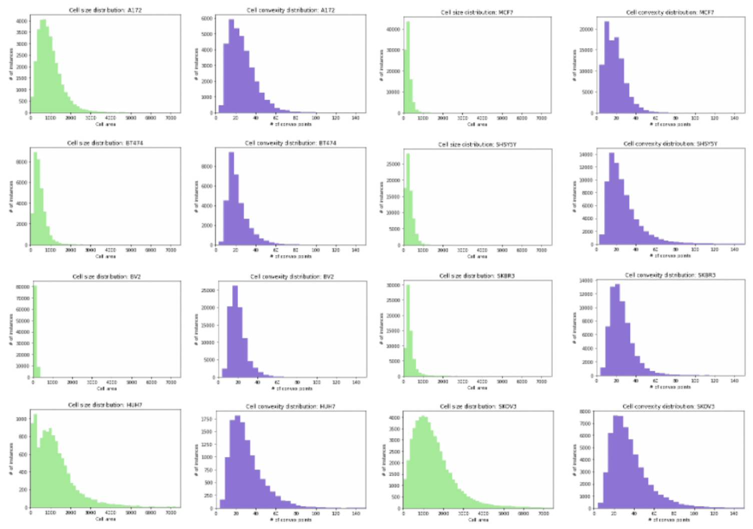

Figure 6. Distributions of cell size (green) and convexity (purple) across the test datasets of each of the 8 different cell types

space of possible combinations of augmentations. However, the augmentation policy found through AutoAlbument (an Albumentations library implementation of the FasterAutoAugment algorithm) demonstrated significant improvement in the transfer task for only two of the 7 novel cell types. Since the automated search approach was built for a semantic segmentation task and adapted for instance segmentation, the ground truth bounding box and separation of instances was ignored in the augmentation selection process. As such, the discriminator in the autonomous augmentation search algorithm would have the objective of minimizing the difference in the distributions of images and their semantic segmentation masks, losing the information about the distributions of instances, for example the sizes or shapes of individual cell instances. Ideally, to learn augmentations to best improve at the downstream transfer task, the discriminator should minimize the loss between distributions of generated images and instance segmentation bounding boxes/masks from the training dataset and the unseen cell type. The objective here would be to provide more guidance in learning transformations to bridge between different cell types. The major challenge is that these new cell type datasets would typically be unlabeled so would necessitate an unsupervised approach to generate weak labels.

5.4. Discussion

Interestingly, it seems like there are certain downstream task cell types that robustly improved with data augmentation on the training A172 dataset, for example BT474, MCF7 and SHSY5Y. I looked at the distribution of cell instance size and convexity across the test datasets for all 8 cell types to try to identify qualitative reasons why certain cell types would be consistently improved with upstream data augmentation. One possible explanation is that these three cell lines have slightly smaller size distributions than the training cell type A172, but not nearly as small as BV2 (Figure 6). Perhaps the size distortions are beneficial to help transfer between A172 and slightly smaller cell type instances. Why a similar reasoning does not apply to a cell type like SKBR3 may be due to the fact that SKBR3 segmentation performance is already quite good and cannot be that much improved by data augmentations, whereas model performance on BT474, MCF7 and SHSY5Y is much more modest.

6. Conclusions

Cell type generalizable instance segmentation models provide a time and labor efficient way to quantify cell abundance in live cell imaging experiments. In this paper, I propose the usage of data augmentations to help a model trained on a single cell type generalize to other cell types at test time. I apply various naive approaches consisting of both geometric and color-based image transformations, which achieve fairly variable downstream improvements when models are evaluated on new cell types. Combining these transformations in either a stacked or random choice approach helps to smooth out this variability, yielding a data augmentation strategy that can improve transfer learning to all or almost all of the new cell types.

Future work should consider an experimental setup where models are pre-trained on unaugmented and augmented training data drawn from multiple different cell types and then evaluated on a single unseen cell type. This would provide additional insight as to which transformations are broadly useful for generalizing across cell types, not only transformations that would help in the A172-

specific training set case. Another interesting experiment would be to examine the scaling effect of various data augmentation policies for this task particularly for the combined augmentation strategies. In this paper, my augmentations only expand the dataset size by 200% which may in it of itself limit the generalizability of the model, as the model may still overfit the augmented data. Ideally, later researchers should train on the augmented datasets and avoid the fine-tuning approach where the backbone parameters were frozen to allow the model's feature extraction network to also learn from the newly augmented data. Especially since the dataset is relatively small, compared to ImageNet for example, it is quite possible that the feature extraction module is also overfitted to the training cell type and would benefit from learning from the augmented data.

7. Contributions and Acknowledgements

For all modeling, I used the pre-trained models, images and annotations from the Sartorious LIVECell Github repository, available here: `https://github.com/sartorius-research/LIVECell`. For data augmentation, I used the frameworks from the CLoDSA library, available here: `https://github.com/joheras/CLoDSA` and the Albumentations library, available here: https://github.com/albumentations-team/albumentations. For the copy-paste data augmentation I used open-source code written by Ryan Conrad available on Github: `https://github.com/conradry/copy-paste-aug`. For automated augmentation search, I used the AutoAlbument code, available here on Github: `https://github.com/albumentations-team/autoalbument`.

References

[1] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[2] Z. Cai and N. Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1483–1498, 2021.

[3] Á. Casado-García, C. Domínguez, M. García-Domínguez, J. Heras, A. Inés, E. Mata, and V. Pascual. Clodsa: a tool for augmentation in classification, localization, detection, semantic segmentation and instance segmentation tasks. *BMC Bioinformatics*, 20(1):323, Jun 2019.

[4] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data, 2018.

[5] C. Edlund, T. R. Jackson, N. Khalid, N. Bevan, T. Dale, A. Dengel, S. Ahmed, J. Trygg, and R. Sjögren. Livecell—a large-scale dataset for label-free live cell segmentation. *Nature Methods*, 18(9):1038–1045, Sep 2021.

[6] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation, 2020.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[8] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation, 2019.

[9] M. Jain, C. Meegan, and S. Dev. Using gans to augment data for cloud image segmentation task, 2021.

[10] A. J. Ratner, H. R. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré. Learning to compose domain-specific transformations for data augmentation. 2017.

[11] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019.

[12] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods*, 18(1):100–106, Jan 2021.

[13] D. A. Van Valen, T. Kudo, K. M. Lane, D. N. Macklin, N. T. Quach, M. M. DeFelice, I. Maayan, Y. Tanouchi, E. A. Ashley, and M. W. Covert. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. *PLOS Computational Biology*, 12(11):1–24, 11 2016.

[14] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola. Resnest: Split-attention networks, 2020.

[15] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le. Learning data augmentation strategies for object detection, 2019.