Interactive Scribble Segmentation

Mathias Micheelsen Lowes*[1], Jakob Lønborg Christensen*[1], Bjørn Schreblowski Hansen*[1], Morten Rieger Hannemose[1], Anders Bjorholm Dahl[1], and Vedrana Andersen Dahl[1]

[1]Technical University of Denmark

Abstract

We present a deep learning model for image segmentation that uses weakly supervised inputs consisting of scribbles. A user can draw scribbles on an image with a brush tool corresponding to the labels they want segmented. The network can segment images in real time while scribbles are being drawn, giving instant feedback to the user. It is easy to correct mistakes made by the network, as more scribbles can be added. During training we use a similar psuedo-interactive and iterative setup to make sure that the network is optimized towards the human-in-the-loop inference setting. On the contrary, standard scribble segmentation methods do not consider the training of the algorithm as an interactive setting and thus are not suited for interactive inference. Our model is class-agnostic and we are able to generalize across many different data modalities. We compare our model with other weakly supervised methods such as bounding box and extrema point methods, and we show our model achieves a better mean DICE score.

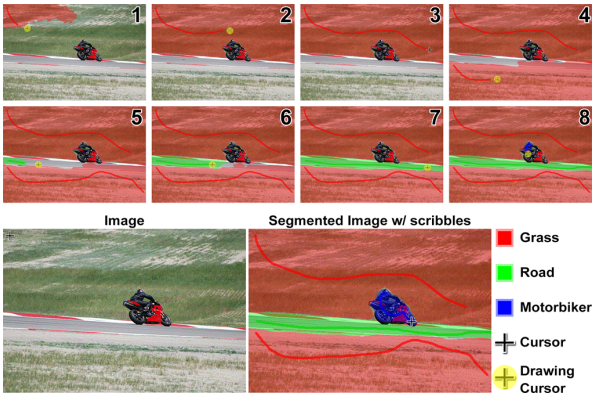

Figure 1: Selected frames from an example interaction with our segmentation tool.

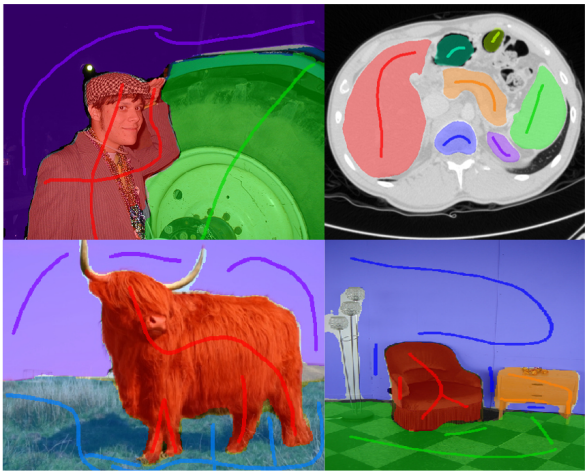

Figure 2: Segmentations from our model with hand-drawn scribbles, produced by our interactive tool.

1 Introduction

Many instance segmentation methods have been proposed that use user input (signals) along with the image. These methods are often referred to as *weakly-supervised semantic/instance segmentation*. Signals may vary from bounding boxes (bbox), extrema points [9] or background/foreground points [1] to more complex and commonly used signals such as scribbles. Scribbles are brush-like strokes used by the network to predict a label.

*Equal contribution

https://doi.org/10.7557/18.6823

Methods based on scribbles [7, 15, 16] usually train using the crowd-sourced PASCAL-Scribble Dataset from Lin et al. [7]. The scribbles in this dataset often consist of a single curved line in the central part of the corresponding label.

Additionally, these scribbles are used as a single-iteration signal. The fixed scribbles and the image are passed through the network and the output is used as-is. A similar single-iteration framework is also used in the bbox and extrema point methods. A consequence of a single-iteration model is that it is hard to make interactive adjustments to the input signals to get the desired segmentation; You cannot input a new bbox to get a better segmentation. Scribble methods have a larger input signal space but making adjustments to fix a segmentation is still hard. Drawing additional or very complex scribbles will rarely work since the model was only trained to receive a single fixed scribble that doesn't depend on the segmentation output. Our model is trained to receive additional signals where it fails, thus simulating an interactive segmentation setting that might arise during inference.

In real-life applications any model will inevitably make mistakes and the benefit of an interactive model should be to easily correct such mistakes. For example, GrabCut [11] allows the user to add more signals in an intuitive way. Our method aims at adjusting to the same level of interactive feedback. The network is directly optimized towards adjusting to feedback in the psuedo-interactive training loop.

We use scribbles as the weakly-supervised signal, but instead of using the PASCAL-Scribble Dataset [7], we create our own simulated scribbles as this provides more control. Specifically, we can let the scribbles depend on the network output, which allows us to simulate the human feedback interaction. During training we randomly do an additional forward pass. In the second forward pass the network is given additional scribbles in areas where the network prediction was incorrect.

In summary the main contributions of our paper are

- A scribble generating algorithm generating hand-drawn like scribbles.
- An iterative training scheme simulating human like behaviour.
- A segmentation network with real time human-in-the-loop interaction.

2 Methods

We outline our training data and network architecture, however these are fairly standard for segmentation problems. The novel part of our method is the algorithm we use to simulate user-provided scribbles, as well as the pseudo-interactive training framework.

2.1 Data

For our models to generalize on different image types, image qualities, and different data modalities, we use several datasets for training, testing, and validating. The used datasets are:

- COCO - Common Objects in Context [8], is a large dataset consisting of 118k images with instance segmentation labels in 91 different categories.
- Pascal Visual Object Classes [3], is also an image dataset with instance segmentation labeling between 20 different object classes. The dataset is smaller than the COCO dataset and consists of around 5k images, however, the quality of the segmentations is very high, with the majority of the images having multiple labels.
- CHAOS [5] is data originating from two different data sources, one from CT scans and one from MR scans. Each of these sub-datasets includes DICOM images from 40 different patients that on average have 40 slices from each patient. This gives around 3.3k images in the CHAOS dataset. The CT scans only have annotations of the liver, while the MR scans include segmentations for the liver, spleen, and kidneys.
- Decathlon [2], is a collection of 2.5k medical 3D volumetric images. The volumetric images are collected across multiple anatomies e.g. brain, heart, and liver. Since our models work on 2D images, we extract slices with segmentation from the volumetric images in the three different axis-parallel planes. Using this approach,

we have created 93k images with corresponding segmentations.

Each of the datasets is randomly split into a training, validation, and test set by the proportions $[0.8, 0.1, 0.1]$. Due to the different sizes and qualities of the datasets, we sample training data from the datasets with the probabilities $[0.6, 0.1, 0.1, 0.2]$ in the order COCO, Pascal, CHAOS and Decathlon.

2.2 Network and Interactive Tool

We use an implementation of Effecient-UNetV2 for our model. This is based on the results from Jahanifar et al. [4] who used Efficient-UNet for interactive segmentation, and on the improvement of Efficient-Net [12] to Efficient-NetV2 [13]. Hence, we apply the improvements from Efficient-NetV2 to a UNet [10] architecture. We utilize the proposed MB-conv and Fused MB-conv from Efficient-NetV2 along with their optimal parameters in both our encoder and decoder. For downscaling, we use average pool and for upscaling we use bilinear upsampling. The full network architecture is illustrated in Fig. 3.

The input to the network is an image with four channels: RGB and scribble rendering. The output of the model is a binary segmentation mask for the object containing the scribble. For segmentations with multiple classes a scribble input associated with each label and the same image is given to the model. E.g. an image with 3 classes is given as a batch of 3 identical images but where the associated scribble corresponds to one of the 3 different classes.

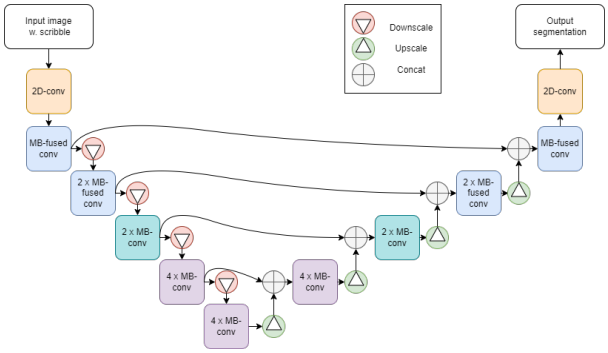

Figure 3: Visualization of the Efficient-UNetV2 architecture.

For the training of our network, we use Binary Cross Entropy loss and the Adam optimizer [6] with a learning rate of 0.0001. The binary labels of our models are background-foreground where the foreground is the label mask associated with the input scribble.

The network takes input images of size 128×128 pixels and has approximately 1 million parameters. An advantage of only having a fairly low number of parameters is a fast forward pass. Our model is able to do around 90 sequential forward passes in a second or a single forward pass in around 11 ms on a Nvidia-Volta-100 32 GB GPU. This corresponds to the refresh rate of modern computer screens, which makes the inference of our model feel instantaneous. For human-in-the-loop usage of our tool we have created a GUI[1], where the prediction of the network is overlaid on the provided image based on a scribble input from the user. The network prediction is updated for each new mouse movement in real time. Usage of this tool is shown in Fig. 1.

2.3 Scribble Simulation

For the network to accurately predict a label from an associated scribble, the network has to be trained on scribbles that resemble human drawings. Therefore, a significant contribution of our work is to simulate human-drawn scribbles with an algorithm fast enough to run during training.

In the context of this work, a scribble is formulated as a graph. This means the scribble is represented by a set of points and edges characterizing the shape of the label.

Our method utilizes the Skeletonize algorithm by Univ. et al. [14] for an initial shape representation of the label and builds upon it to obtain a set of evenly spaced points. The set of evenly spaced points is obtained by constructing a partitioning for each connected component in a label. The connected components in a label are partitioned into smaller parts by constructing a grid pattern, i.e. we partition the label both horizontally and vertically as seen in Fig. 4 (3). We denote the rectangular parts making up the original label as tiles. The set of tiles constitutes as candidate areas for

[1]See our GitHub - `https://github.com/MMLowes/Interactive-Scribble-Segmentation`.

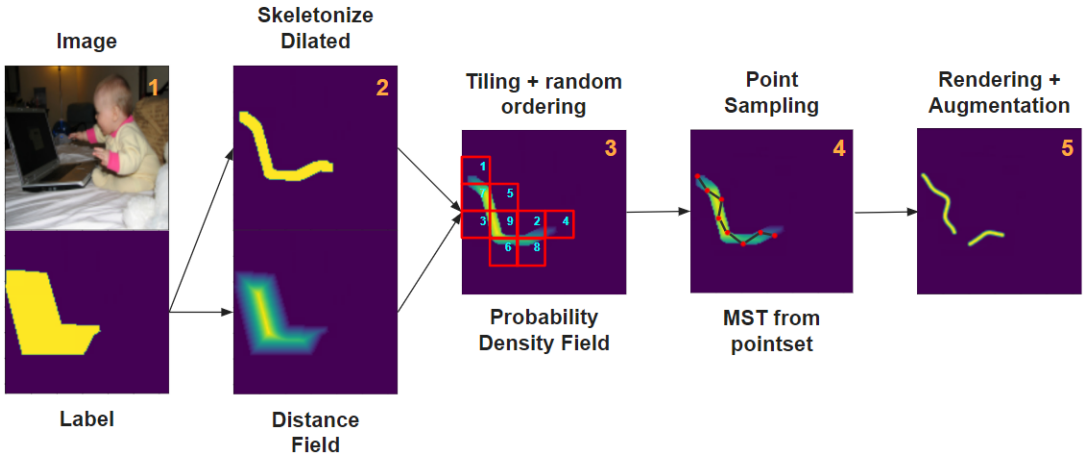

Figure 4: Visualization of the scribble simulation. (1) shows the image and label, (2) shows distance field and Skeletonize dilated, (3) shows the partitioning and ordering of the label, (4) shows the sampled points with minimum spanning tree (MST) connectivity and finally (5) the spline interpolation with graph augmentation.

points. The spacing of the horizontal and vertical lines making up the partitioning depends on the height and width of the respective connected component.

We iterate through a random sequence of all the tiles, and for each tile, we either sample a point or skip it. A random ordering of the partitioning can be seen in Fig. 4 (3) . We sample a point if the sum of the distance field within a tile exceeds a threshold based on the size of the total label. This is done to ensure that we only sample points from tiles containing a meaningful portion of the original label. When we sample a point, the position of the point is based on a probability density field in the tile given by the product of the distance field and the dilated output of Skeletonize. This sampling procedure for each tile ensures our approach is stochastic rather than deterministic. All pixels within radius r of the point we sample are set to background and we update the distance field for the label again. This is done to ensure points are not in close proximity of each other.

After iterating through each of the tiles, the connectivity of the sampled points is decided. The connectivity is based on the minimum spanning tree of the points given by a cost matrix. The cost between two points is given by the euclidean distance between them. If an edge connecting two points leaves the label, the distance outside the label is multiplied by 3 as a penalizer. This is done to punish generating scribbles outside the boundary of the label.

We augment 20% of our graphs by randomly removing either $1, 2$ or 3 edges with probability $\{1/2, 1/3, 1/6\}$ respectively from the connectivity. This is done to make the model more robust to varying user inputs. Only edges connecting two points both with valence ≥ 2 can be removed. This criteria is made to ensure points only connected by a single edge to the graph cannot be disconnected from all the other points.

With the points and their connectivity decided, we need to render the graph in pixel-space. This is done by creating a binary scribble-image, where we render the scribble. To achieve a human-like look the following steps are used to draw the scribbles:

1. Find the longest path (using breadth first search) in the graph, and remove its edges from the graph.

2. Use the $x-$ and $y-$coordinates of the points of the found path to fit a B-spline basis. This yields a smooth parameterized line that approximately follows the points.

3. Draw the path by setting all pixels in the scribble-image within a radius of 2 pixels of the parameterized spline-lines to 1.

4. Go to step 1. until no edges remain.

This way of sampling paths ensures we use the smallest number of paths and is similar to how a human would draw a scribble. The steps of simulating a scribble are visualized in Fig. 4. Examples of image and labels pairs with a generated scribble are visualized in Fig. 5.

Figure 5: Examples of the scribble (light blue) generation algorithm with image-label pairs.

2.4 Iterative Training

During training, we do an iterative forward pass 30% of the time. This means that we do two forward passes where the first is a normal forward pass but without gradients. For the second forward pass, we add an extra scribble where the prediction failed. If the prediction is too small we add a positive (foreground) scribble and if it is too large we add a negative (background) scribble. A negative scribble means that the pixel values of the scribble image are -1. The iterated scribble image is generated by the following procedure, illustrated in Fig. 6.

1. Look at the difference between the iteration 1 prediction and the ground truth. If there are more false positives than false negatives then use negative scribbles, otherwise positive. If the error is less than 1% relative to the segmentation area then no scribbles are added.

2. Create a temporary label that is used as the image for the additional scribble generation. First we threshold the label difference with values greater than 0.5. The binary image is then morphologically dilated by the approximate radius of the ground truth segmentation to make sure there is some overlap between the iteration 1 and 2 scribbles.

3. Use the scribble simulation algorithm on the temporary label to obtain the additional scribble.

4. Add the old and the additional scribbles to create a new scribble image.

During human-in-the-loop usage of our model for segmentation of multiple classes, we utilize negative scribbles. In the forward passes we segment one class at a time and here we render the scribbles for the other classes as negative scribbles.

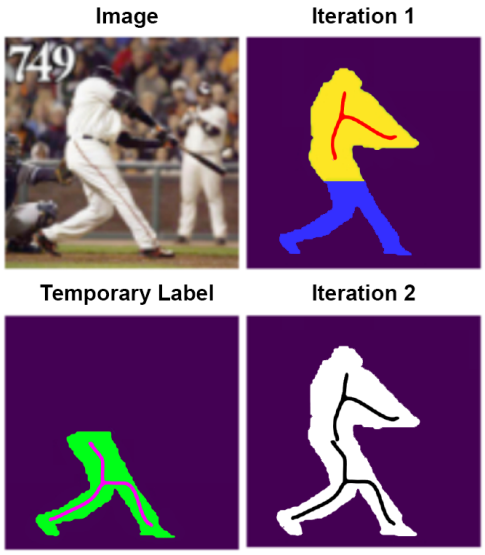

Figure 6: A conceptual visualization showing an image with a lackluster prediction (yellow) at iteration 1. The false negative pixels (blue) are dilated to get the temporary label (green), for which we generate an additional scribble (purple). A new prediction (white) is made using both scribbles (black).

3 Experiments

We choose to compare our model to the methods from Deep Extreme Cut [9] (extrema points) and the bounding box (bbox) method. We have used the same type of Effecient-UNetV2 for these methods as we have used in our model to better compare the methods instead of the networks. This entails that we render the bounding box and extrema points as the scribbles are rendered, i.e. in a fourth channel of the input image.

A manual test dataset is created as a subset of the Pascal test set where 10 persons have drawn a total of 197 scribbles for the labels on the test images. This dataset is made to test how well our

model extrapolates to human-drawn scribbles since it was only trained with simulated ones.

A quantitative comparison can be seen in Table 1, where we also compare the effect of using iterative scribble generation for our model. The table shows how our model outperforms both the extrema method and the bbox method. Furthermore, the iterative scribble generation gives the model a large boost in DICE-score. Our simulated scribbles emulate humans quite well, as the manual dataset DICE score of 0.904 still beats the other methods. Note that in Table 1 the bbox and extrema point inputs are perfectly positioned (not generated by humans). The iterative update for the manual dataset is not manual and still uses the simulated scribbles.

A visual comparison of the models is made in Fig. 7, where four different labels on different pictures are predicted. The input to our model is hand-drawn scribbles while the bbox and extrema models get the perfectly positioned bounding box and extrema points from the segmentation mask.

Dataset **Model**	Pascal test	Pascal vali.	COCO test	COCO vali.	Manual dataset
Our model	**0.933**	**0.936**	**0.924**	**0.919**	**0.904**
bbox	0.823	0.832	0.825	0.828	0.868
extrema	0.899	0.899	0.883	0.882	0.899
Our model (iterative)	0.962	0.963	0.951	0.949	0.938

Table 1: Mean DICE scores for different models and datasets. The manual dataset is a subset of Pascal test.

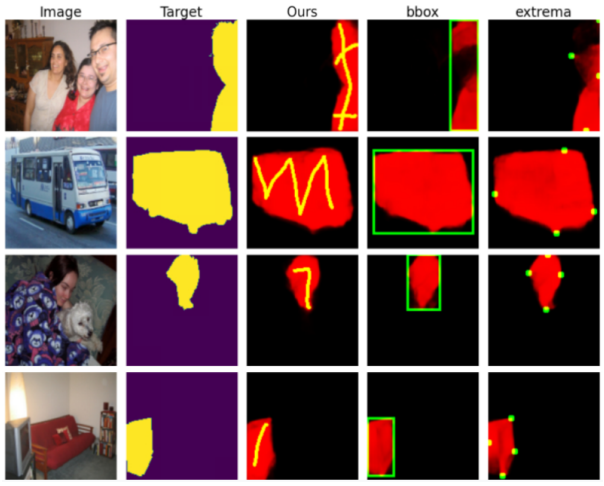

Figure 7: Examples comparing our method on manual hand-drawn scribbles with bbox and extrema methods.

Fig. 2 shows segmentations from our model on test images and Fig. 1 shows the interactive use of our model. The scribbles used for the segmentations are hand-drawn in our interactive segmentation tool, meaning they are iteratively drawn, as Fig. 1 illustrates. The figures show that with a few lines we can segment a whole image into different labels.

4 Discussion and Conclusion

An important benefit of our model is the ability to correct and iteratively draw on a label to achieve the desired segmentation. Methods like DEXTR [9] that use extrema points as network inputs are quite limited in how the model can be corrected, as there are no real adjustments you can make to the model input. For many simple labels, a scribble is also easier to draw than clicking four precise extrema points or a bounding box. Another downside to the bbox and extrema methods is the lack of unique correspondence between inputs and labels. For example, two different labels can have the exact same bounding box yet be completely different. We often observed this in the data when a label had a bounding box that was the size of the full image.

We used simulation to create scribbles instead of using the PASCAL-Scribble Dataset [7]. Direct comparison (such as mean DICE score) to most other scribble methods makes little sense as we do not have an identical and static input. Our approach is however similar to Jahanifar et al. [4] as they too create scribbles during training. However, they do not simulate a user correcting the initial prediction. Additionally, their code is not public and they are focused on a very specific data modality so comparing to their method would also be of little use.

Comparison to bbox and extrema points should also be taken with a grain of salt. In many cases where the static input is sufficient, there will be little to no difference between the resulting segmentation of our method, bbox, extrema points, or other scribble methods. The benefit of our model is that we can iteratively update the scribble to achieve the desired segmentation in the cases where the initial segmentation is not satisfactory.

To improve how our model performs on medical data, a more diverse medical dataset is required.

We tried to train models using only medical data (CHAOS and Decathlon) instead of all the data. We observed that models that were trained on a combination of all the datasets, and not only the medical data, were able to generalize better. The problem with the CHAOS and Decathlon datasets is that only certain organs and tissues are annotated. This means that the network never learns to segment bone, fat, or background as those areas are never annotated in the data. A more exhaustive search for segmentation datasets and especially medical datasets with more variation would be beneficial for future work.

The models described in the experiments section used around 1 million parameters, which makes them fairly small networks considering the level of generalization and abstraction required by the task we are trying to solve. Usually, more parameters improve model performance, but more parameters would likely overfit on the scribbles generated by our algorithm in the training process which defeats the motivation of our work - to use human-drawn scribbles.

We believe that using human-drawn scribbles for training could improve our model performance. It would be extremely slow to have an actual human react to network outputs during training. We could however have a dataset of initial scribbles which are drawn by humans while still using simulated second iteration scribbles.

Our work shows that deep learning models may be trained to simulate human-in-the-loop interactions. The network is still able to react to actual human inputs. Our model is fast due to the efficient architecture and reacts well to adjustments from humans, despite only being trained on a simulation of the human-network interaction.

References

[1] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the point: Semantic segmentation with point supervision. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 549–565, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46478-7. doi: 10.48550/ARXIV.1506.02106.

[2] M. A. et al. The medical segmentation decathlon. *Nature Communications*, 2021. doi: 10.48550/ARXIV.2106.05735. URL `https://arxiv.org/abs/2106.05735`.

[3] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. doi: 10.1007/s11263-009-0275-4.

[4] M. Jahanifar, N. Z. Tajeddin, N. A. Koohbanani, and N. Rajpoot. Robust interactive semantic segmentation of pathology images with minimal user input. 2021. doi: 10.48550/ARXIV.2108.13368. URL `https://arxiv.org/abs/2108.13368`.

[5] A. E. Kavur, N. S. Gezer, M. Barış, S. Aslan, P.-H. Conze, V. Groza, D. D. Pham, S. Chatterjee, P. Ernst, S. Özkan, B. Baydar, D. Lachinov, S. Han, J. Pauli, F. Isensee, M. Perkonigg, R. Sathish, R. Rajan, D. Sheet, G. Dovletov, O. Speck, A. Nürnberger, K. H. Maier-Hein, G. Bozdağı Akar, G. Ünal, O. Dicle, and M. A. Selver. Chaos challenge - combined (ct-mr) healthy abdominal organ segmentation. *Medical Image Analysis*, 69:101950, 2021. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2020.101950. URL `https://www.sciencedirect.com/science/article/pii/S1361841520303145`.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. doi: 10.48550/ARXIV.1412.6980.

[7] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. doi: 10.48550/ARXIV.1604.05144. URL `https://arxiv.org/abs/1604.05144`.

[8] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. doi: 10.48550/

ARXIV.1405.0312. URL http://arxiv.org/abs/1405.0312.

[9] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. doi: 10.48550/ARXIV.1711.09081. URL https://arxiv.org/abs/1711.09081.

[10] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. 2015. doi: 10.48550/ARXIV.1505.04597. URL https://arxiv.org/abs/1505.04597.

[11] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 2004. doi: 10.1145/1015706.1015720. URL http://dblp.uni-trier.de/db/journals/tog/tog23.html#RotherKB04.

[12] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *International conference on machine learning*, 2019. doi: 10.48550/ARXIV.1905.11946. URL https://arxiv.org/abs/1905.11946.

[13] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. *International Conference on Machine Learning*, 2021. doi: 10.48550/ARXIV.2104.00298. URL https://arxiv.org/abs/2104.00298.

[14] T. Y. Z. C. Univ., T. Y. Zhang, C. Univ., C. Y. S. C. Univ., C. Y. Suen, C. Univ., N. A. R. Center, and O. M. A. Metrics. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, Mar 1984. doi: /10.1145/357994.358023. URL https://dl.acm.org/doi/10.1145/357994.358023.

[15] G. Valvano, A. Leo, and S. A. Tsaftaris. Weakly supervised segmentation with multi-scale adversarial attention gates. *CoRR*, abs/2007.01152, 2020. doi: 10.1109/tmi.2021.3069634. URL https://arxiv.org/abs/2007.01152.

[16] J. Xu, C. Zhou, Z. Cui, C. Xu, Y. Huang, P. Shen, S. Li, and J. Yang. Scribble-supervised semantic segmentation inference. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. doi: 10.1109/ICCV48922.2021.01507. URL https://openaccess.thecvf.com/content/ICCV2021/papers/Xu_Scribble-Supervised_Semantic_Segmentation_Inference_ICCV_2021_paper.pdf.