[RE] UNSUPERVISED OBJECT SEGMENTATION BY REDRAWING, NEURIPS 2019 REPRODUCIBILITY CHALLENGE

Nabakumar Singh Khongbantabam
Autonomous Systems Master's student
KTH Royal Institute of Technology
Stockholm, Sweden
nskh@kth.se

Joel Fredriksson
Machine Learning Master's student
KTH Royal Institute of Technology
Stockholm, Sweden
joefre@kth.se

ABSTRACT

This report is part of the NeurIPS 2019 Reproducibility Challenge and attempts to replicate the results of the paper *Unsupervised Object Segmentation by Redrawing*, (4). We attempt to replicate the results by implementing the deep neural network architecture, named ReDo by the authors in the original paper[1], based on the specifications described in the paper. ReDo architecture is designed based on a Generative Adversarial Network (GAN) (3) to learn and segment object masks using unsupervised learning. [2]

1 INTRODUCTION

Object segmentation or image segmentation is the process of highlighting and identifying the objects borders' in an image. This effectively segments the image into regions of different objects in the scene. It is particularly useful in robotics and computer vision based processing, where objects in a scene need to be separately processed.

As a result, object segmentation is an important area of research in computer vision. There exists many traditional object segmentation algorithms, such as Watershed (1) and GraphCut (2). Lately, deep learning algorithms have also evolved to be a powerful contender in object segmentation. However, many of the deep learning based object segmentation networks rely on supervised learning, which require that the training images be labeled. Unfortunately, labeling images for segmentation can be quite laborious since it is done by drawing the segments manually over the images. As a result, the availability of datasets for training can be very limited or could be expensive to prepare. Please refer to the original paper (4) for more information.

2 RELATED WORK

The authors of the paper have identified several related work in domain of image segmentation (5) (6) (7). However, most of the methods use annotated images for training, which can be expensive to acquire.

3 METHOD

The purpose of this experiment is to reproduce the original results. We perform this by re-implementing the architecture from scratch, training the network on the same datasets and validating the results. We choose to implement from scratch the network using Keras/TensorFlow in order to keep the re-implementation substantially different from the author's original implementation in PyTorch.

[1] We based this report on the version from the 27 May 2019.

[2] The code, written in keras/tensor-flow, can be found on Gitlab (16)

4 ARCHITECTURE

The ReDo architecture is a fairly complex GAN architecture utilizing many new network architectures for its sub-networks. It consists of four major networks: Mask extractor network, Image generator network, Discriminator network and z-Reconstruction network. They are organized as shown in figure 1.

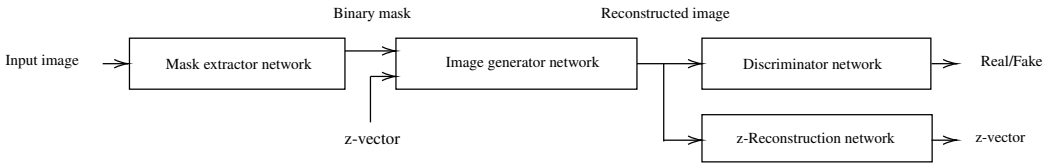

Figure 1: High-level architecture block diagram

The Mask Extractor network takes an input image and attempts to generate a binary mask corresponding to a segment in the image. The Image Generator network takes the generated mask and a latent vector, z, with a know prior that generator has learned to map to the distribution of the input samples. With these inputs, the generator attempts to recreate the content of the masked segment as a learned method from the training samples. The generated image segment is then combined with a background in the rest of the image to compose a final output image. The background is extracted from the original image using the complementary of the generated mask. The Discriminator network is a standard image binary classifier that differentiates between the generated images and real images. This process is illustrated in figure 2.

The z-Reconstruction network is an encoder network that attempts to encode the generated image into the original latent vector, z. This ensures that the generated image is truly representative of the latent vector and avoid converging to all-black masks.

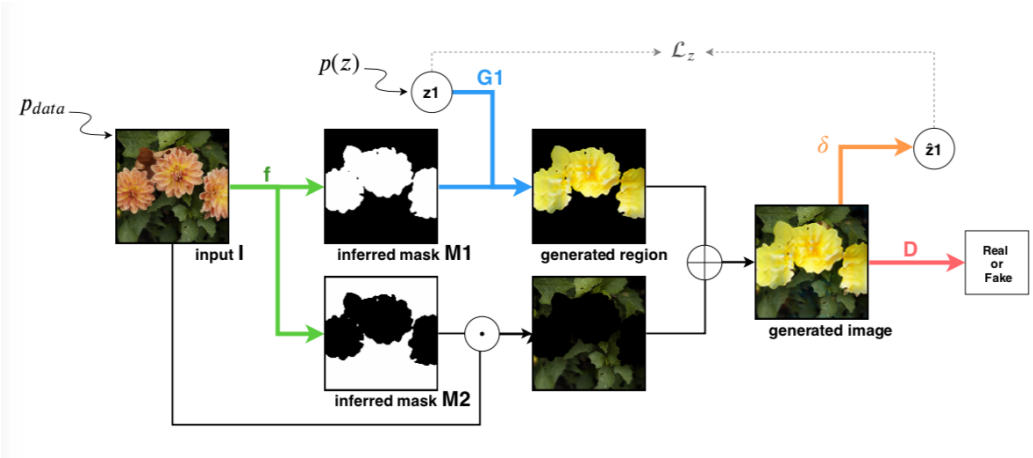

Figure 2: Mask extraction and image generation process, courtesy of (4)

4.1 MASK EXTRACTOR NETWORK

According to the original paper (4), the Mask Extractor network is loosely adapted from an image-to-image translation architecture (8), to which a pyramid pooling sub-network is added to capture information from different scales of the input image. The pyramid pooling architecture is inspired by the PSPNet architecture (10) and is shown in figure 3.

mask network f(I)	Non-linearities	Output size
Image I		3x128x128
Conv 7x7 (reflect. pad 3)	Instance Norm, ReLU	16x128x128
Conv 3x3 (stride 2, pad 1)	Instance Norm, ReLU	32x64x64
Conv 3x3 (stride 2, pad 1)	Instance Norm, ReLU	64x32x32
Residual Bloc (Instance Norm, ReLU)		64x32x32
Residual Bloc (Instance Norm, ReLU)		64x32x32
Residual Bloc (Instance Norm, ReLU)		64x32x32
Pyramid Pooling Module		68x32x32
Upsample		68x64x64
Conv 3x3 (pad 1)	Instance Norm, ReLU	34x64x64
Upsample		34x128x128
Conv 3x3 (pad 1)	Instance Norm, ReLU	17x128x128
Conv 3x3 (reflect. pad 3)	sigmoid (if n = 2) or softmax	nx128x128

Table 1: Layers in mask generator network

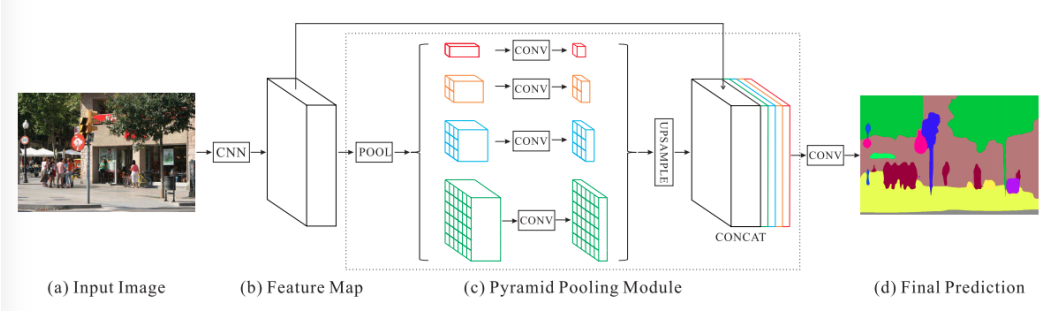

Figure 3: Pyramid pooling architecture for capturing information from different regions of the image, adapted from (10)

4.2 Image generator and discriminator networks

The Image generator, or region generator, as referred in the paper, and the Discriminator are based on an architecture known as Self-Attention generative adversarial network (SAGAN) (9). In addition, the image generator and discriminator utilizes spectral normalization (12) for weight regularization.

region generator network Gk(zk,Mk)	Output size
noise vector input zk	32
Linear, Conditional Batch Norm, ReLU	16ch.x4x4
Up Res Bloc (CBNorm, ReLU, concat 1x4x4 Mk)	16ch.x8x8
Up Res Bloc (CBNorm, ReLU, concat 1x8x8 Mk)	8ch.x16x16
Up Res Bloc (CBNorm, ReLU, concat 1x16x16 Mk)	4ch.x32x32
Up Res Bloc (CBNorm, ReLU, concat 1x32x32 Mk)	2ch.x64x64
Self-Attention Bloc	
Up Res Bloc (CBNorm, ReLU, concat 1x64x64 Mk)	ch.x128x128
Conditional Batch Norm, ReLU, concat 1x128x128 Mk	
Conv 3x3 (pad 1) tanh	3x128x128

Table 2: Layers in image/region generator network

Discriminator network D and z-Reconstruction network (encoder) δ	Output size
image input I	3x128x128
Down Res Bloc (ReLU)	64x64x64
Self-Attention Bloc	16ch.x8x8
Down Res Bloc (ReLU)	64x32x32
Down Res Bloc (ReLU)	128x16x16
Down Res Bloc (ReLU)	256x8x8
Down Res Bloc (ReLU)	512x4x4
Res Bloc (ReLU)	1024x4x4
Spatial sum pooling	1024x1x1
Linear	1 for D, 32 for δ

Table 3: Layers in discriminator network and encoder δ

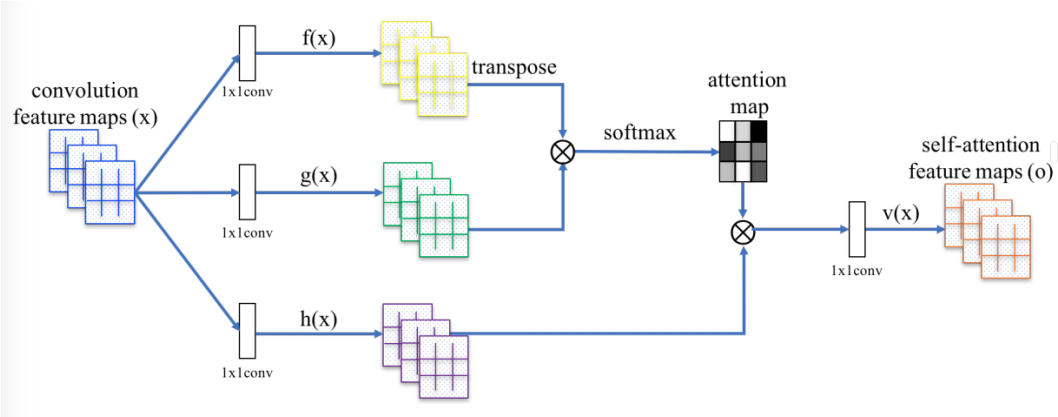

Figure 4: Self-Attention network used in image generator and discriminator, adapted from (9)

5 DATASETS

The same three datasets used by the authors in the original paper are used. They are split into training, validation and test sets in a similar manner performed by the authors:

- The flowers dataset (13) consisting of 8189 images, split into training set of 6150 images, validation set of 1020 images and test set of 1020 images. Some samples of the flowers dataset are shown in figure 5 for illustration.
- The *Labeled Faces in the Wild* dataset (14) consisting of 13233 faces, split into validation set of 1327 images and test set of 1600 images.
- The *Caltech UCSD Birds* dataset (15) consisting of 11788 images, split into training set of 10000 images, test set of 1000 images and the rest assigned to a validation set.

In the paper, the authors choose to pre-process the dataset during the loading time. In this reproduction, all the datasets are pre-processed prior to the training or validation procedure. This helps to slightly speed up the loading. The pre-processing consists of scaling the images and masks to 128x128 size and extracting the grey color binary masks from the original colored masks. The pre-processed datasets are saved and loaded during the training and validation executions. Some pre-processed samples of the flowers dataset, both images and masks, are shown in figure 5.

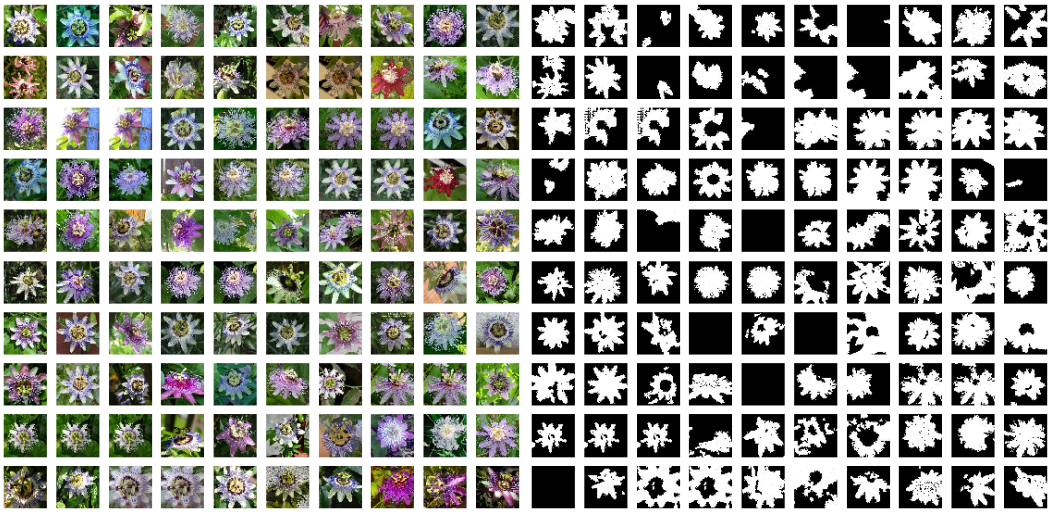

Figure 5: Sample images and corresponding masks after the pre-processing of flowers dataset (13)

6 Implementation

The architecture was implemented in google colab environment. All dataset images were uploaded and located in google drive. By mounting the drive remotely in the colab environment, the re-implementation code was able to execute the training.

The architecture utilizes network blocks that Keras do not yet support naively. As a result, we had to implement them from scratch. These blocks include: ResNet block, CNN with reflection-padding, Upsampling block and Pyramid Pooling block.

The dataset loading in keras using *ImageDataGenerator* lacked a suitable way to scale the training images to -1 and 1 range needed in the network inputs. As a result, a work around had to be implemented and dataset pre-processing was custom implemented for this project.

The GAN implementation Keras as a pipeline was a bit challenging, especially in getting the discriminator network to switch between trainable and non-trainable mode while sharing the same weights. This was eventually resolved by implementing two parallel models with shared networks.

The training loop has been implemented to save all the weights from the models constituting the network after a few epochs. This is done in order to recover the network after stopping the training.

Complete source code is available in gitlab (16). Figure 6 shows the complete timeline of the project, current status and remaining tasks to achieve the final goal.

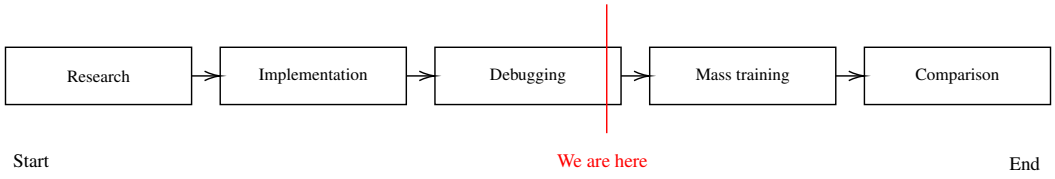

Figure 6: Project time-line and current status

7 Training & Experiments

When implementing the architecture directly from the paper, the training has not been successful. We referred to the authors' code several times during the implementation to make the training work

in our implementation. Besides, we attempted several other changes and tweaks in the code to make the training successful. However, they all led to different unsuccessful outcomes in the long run. Some notable changes we attempted are described below:

1. The authors used a discriminator network without activation function, perhaps because of the *hinge loss function* used. As we had difficulty reproducing the training in this setup, we instead tried a sigmoid activation function in the discriminator output, together with regular binary cross-entropy loss computation. Figure 7 shows the changes introduced. These changes brought some improvements in the early trainings, however, they still failed to converge. They would lead to either all black or all white masks output. Results are shown in figure 9.

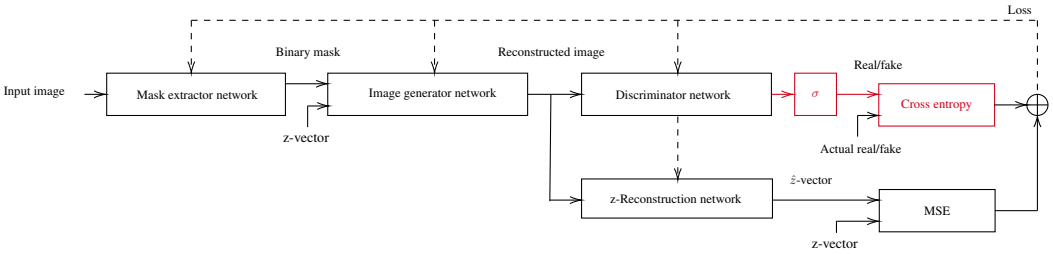

Figure 7: Original architecture was changed to incorporate sigmoid and binary cross-entropy at the discriminator output.

2. We attempted a different network where two generators were used, one for generating the foreground and another for the background. The discriminator would then take the combined image as input. The changes in the architecture are shown in figure 8. This avoided the divergence of the network to all black or white. However, it would still fail to converge. The mask generator would produce random patterns. Results are shown in figure 10.

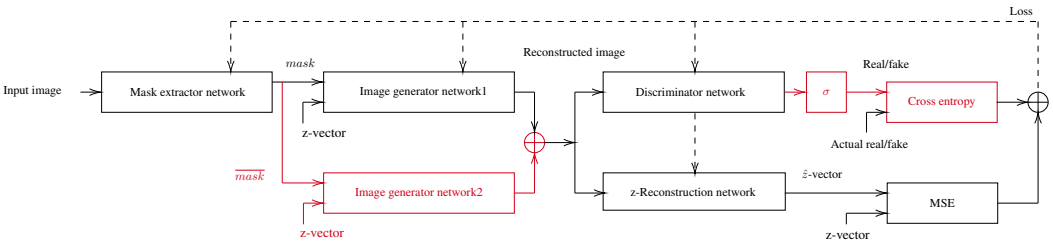

Figure 8: Original architecture further changed to incorporate additional generator that generated complementary image from the inverse of the mask.

3. We attempted different activation functions instead of tanh for the intermediate stages, used input images with 0 to 1 range instead of -1 to +1 range currently used by the paper, adjusted learning rates of the different components, and so on.

8 Results

Results mostly varied from promising mask generation in the early epochs, that gradually lose to all-white or all-black masks, to no seemingly obvious generation of the images. The most promising one was when using sigmod and cross-entropy for the discriminator output (figure 7. The early epoch results are shown in figure 9. It can be observed in the figure that the generator is attempting to learn to draw the flowers (first 3 rows), while some masks are shaping up from the mask extractor

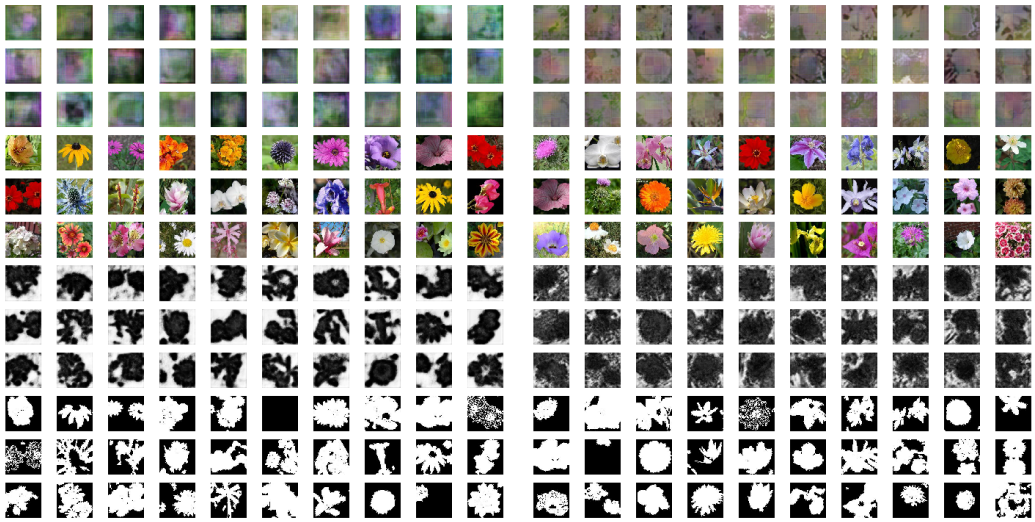

Figure 9: Result of early training using sigmoid and cross-entropy for discriminator at epochs 3 and 44

Figure 10:

network (rows 7 - 9). However, this did not improve over a longer training period and as a result the network did not manage to train successfully.

Figure 11 shows the generator and discriminator losses during the early training. It can be noted that the GAN network has not reached the 'adversarial' stage since the discriminator and generator losses are not competing. As noted previously already, this situation did not improve in further training epochs.

With the double generator architecture, the network could be seen to enter the adversarial stage as the discriminator and generator losses are seen competing. However, the network would output random patterns as masks and would again not converge successfully.

9 Conclusion

Due to the complex nature of the architecture and re-implementation from scratch in a different deeplearning framework (Keras/TensorFlow), the implementation has been challenging only by referring to the information in the paper. A continuous reference to the original ReDO code has to be made in order to progress the implementation in Keras/TensorFlow. Despite attempting to make it work in the past several months, the reproduction has not been possible. Furthermore, additional tweaks we introduced to the network in hope of making it work have not been fruitful either.

As a result, we conclude that there are some fundamental aspects missing or not clarified enough in the paper for reproduction of the results in a straightforward approach. We believe a further discussion and collaboration with the authors are needed into order to enable easier 3rd party reproduction

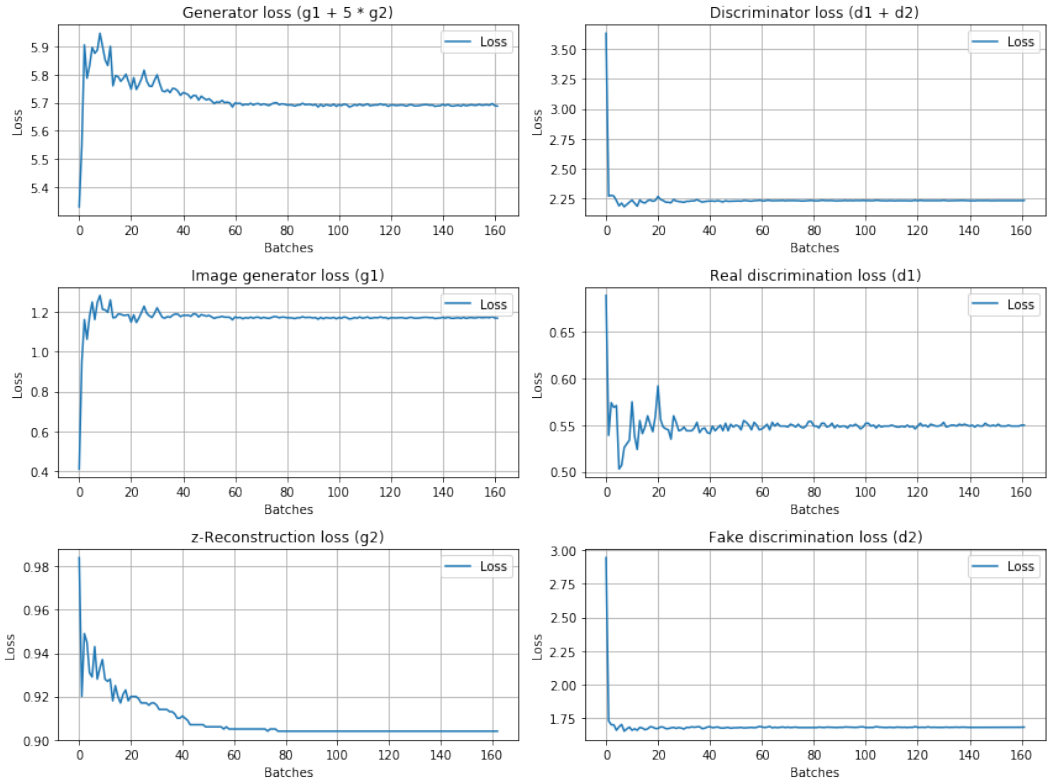

Figure 11: Generator and Discriminator losses progress during the training

of their work. We would be happy to continue to work with the authors to assist in improving the reproducibility.

REFERENCES

[1] Serge Beucher and Christian Lantuéj workshop on image processing, real-time edge and motion detection (1979). http://cmm.ensmp.fr/ beucher/publi/watershed.pdf

[2] D.M. Greig, B.T. Porteous and A.H. Seheult (1989), Exact maximum a posteriori estimation for binary images, Journal of the Royal Statistical Society, Series B, 51, 271–279.

[3] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). Generative Adversarial Networks (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[4] Chen, M., Artières, T., Denoyer, L., 2019. Unsupervised Object Segmentation by Redrawing. arXiv:1905.13539 [cs, stat].

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), pages 801–818, 2018.

[6] Mickael Chen, Ludovic Denoyer, and Thierry Artières. Multi-view data generation without view supervision. In International Conference on Learning Representations, 2018.

[7] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In Advances in Neural Information Processing Systems, pages 2610–2620, 2018.

[8] Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A., 2018. Image-to-Image Translation with Conditional Adversarial Networks. arXiv:1611.07004 [cs].

[9] Zhang, H., Goodfellow, I., Metaxas, D., Odena, A., 2019. Self-Attention Generative Adversarial Networks. arXiv:1805.08318 [cs, stat].

[10] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., 2017. Pyramid Scene Parsing Network. arXiv:1612.01105 [cs].

[11] Zhu, J.-Y., Park, T., Isola, P., Efros, A.A., 2018. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv:1703.10593 [cs].

[12] Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y., 2018. Spectral Normalization for Generative Adversarial Networks. arXiv:1802.05957 [cs, stat].

[13] Maria-Elena Nilsback and Andrew Zisserman. Delving into the whorl of flower segmentation. In BMVC, volume 2007, pages 1–10, 2007.

[14] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[15] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[16] https://gitlab.com/kh_naba/unsupervised_object_segmentation.git