Using Diffusion Models to Generate Synthetic Labelled Data for Medical Image Segmentation

Daniel Saragih
Department of Computer Science
University of Toronto
daniel.saragih@mail.utoronto.ca

Abstract

The analysis of medical image data is a costly and time consuming task for medical experts, yet it is a necessary diagnostic step. Recently, machine learning (ML) models have been used to automate this task, such as in the case of polyp segmentation. However, the quality and quantity of training data is often a limiting factor in the performance of these models due to the difficulty of obtaining annotated medical images. In this paper, we explore the use of diffusion models, alongside other generative techniques such as inpainting and styling, to synthesize data (both images and masks) for medical image segmentation. In doing so, we aim to improve the performance of segmentation models when fully or partially trained on synthetic data and help to alleviate the burden of annotation work by experts. Prior work has focused on GAN-generated data, however, the striking improvements of diffusion models in generating photorealistic images motivates their use in generating synthetic data for medical image segmentation. We find that diffusion models are able to generate more realistic images than GANs, and that styling with diffusion models improves the quality of the generated images and masks. Moreover, we find that the use of our generated data improves the performance of segmentation models when fully trained on synthetic data. In the case of augmentation, we find notable improvements over GAN-generated data when the real dataset is small. The GitHub repository is available at `https://github.com/dsaragih/diffuse-gen` and the synthetic datasets may be downloaded at `https://osf.io/ts6px/`.

1 Introduction

In recent decades, artificial intelligence (AI) has been growing in prominence, and its numerous uses have begun to be realized, including those in healthcare [1, 2]. For example, AI has been used to simulate molecular dynamics, discover drugs, and diagnose diseases [2]. Among these applications, medical image analysis has become a prominent area where AI has been applied [3, 4].

The ML models used in these applications learn from data. Of particular interest in this paper is segmentation models and the data used to train them. Segmentation models classify pixels in the image into regions, and in the case of medical imaging, this task often entails delineating malignancies, functional tissues, or organs of interest [5–7]. As with most ML models, the quality and quantity of the data is a significant factor in the model's performance. However, publicly available data is limited either due to patient privacy laws or because of the time and cost required for experts to annotate the images [3].

The addition of synthetically generated data is a possible approach to tackle the lack of training data as it would overcome the need for annotations by experts. Indeed, GANs have been used to generate training data and labels in the context of medical images to train detection and segmentation models

Preprint. Under review.

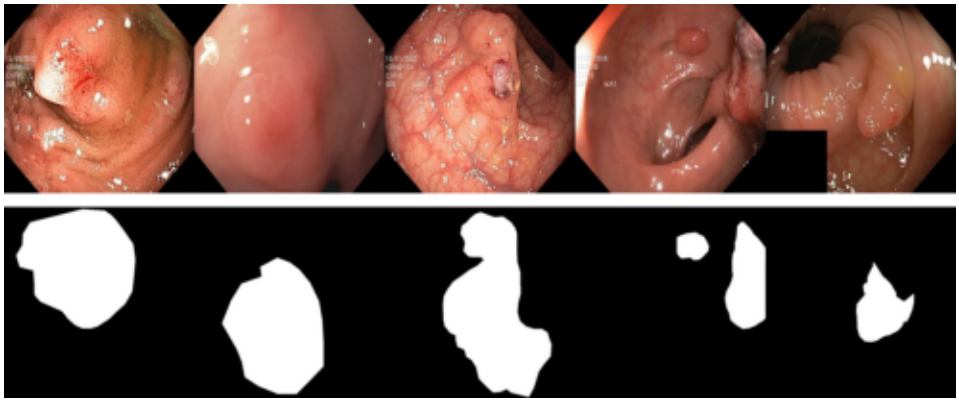

Figure 1: A few samples of the HyperKvasir dataset.

[8, 9]. Moreover, the advent of diffusion models [10] presents a new method of generating novel images which has shown remarkable results in creating photorealistic images [11].

The purpose of this paper is to explore the capabilities of diffusion models in generating synthetic data for medical image segmentation. Specifically, I aim to answer the following questions:

1. Can diffusion models be used to generate more realistic synthetic data for medical image segmentation as opposed to GANs?
2. Does styling with diffusion models improve the output images?
3. To what extent do the generated images improve performance on segmentation tasks?

We may look at some recent work to motivate our questions. Diffusion models have been used to successfully augment image datasets in [12, 13] for medical imaging and a general classification task. In particular, Trabucco et al. [13] showed an improvement in a few-shot classification task. Moreover, Thambawita et al. [4] used styling to improve segmentation performance on the HyperKvasir dataset [14].

2 Materials and Methods

In this paper, we focus on the task of polyp segmentation which employs ML techniques to detect and annotate polyps in images collected from examinations of the GI tract.

2.1 Dataset

The dataset we use is the HyperKvasir [14] dataset, which we found at `https://osf.io/mh9sj/`, from which we obtained 348 polyp images which have a corresponding segmentation mask annotated by medical experts. In this respect, we aim to use this dataset as a case study due to the time and resources required to train our pipeline. However, it's expected that our approach may be generalized to other segmentation datasets.

A few samples of the dataset are shown in Figure 1. The polyp images are RGB images, whereas the masks are single-channeled images. The mask colours the region with polyp white, while the surrounding regions are coloured black. This dataset will be used to train our generative diffusion models, and compare the performance of the segmentation models depending on the type of training data used.

2.2 Methods

We use the aforementioned dataset to study methods to improve automated segmentation accuracy given small datasets. In particular,, this generated data will consists of both images and masks. Originally, our approach was to train a diffusion model on the dataset, followed by styling of the images. However, we realized the need to narrow the scope of sampling and/or training to overcome

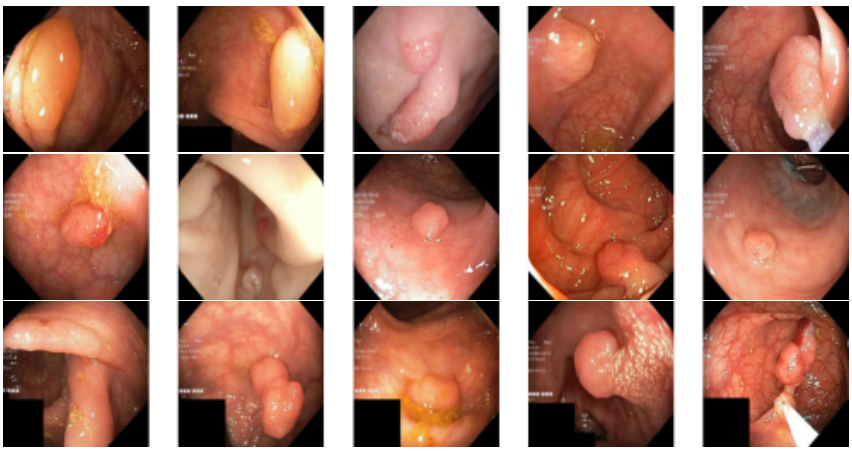

Figure 2: Example of clustering results.

the large variation between images. Thambawita et al. [4] resolved this issue by training a dedicated GAN model for each image of the dataset. In our approach, we wish to improve by adding more variation to the generated outputs. We do so by an approach consisting of several steps:

1. Cluster the images and masks in the dataset into groupss by similarity.
2. Train a diffusion model on each cluster, using a 4th channel input to generate the mask.
3. Use the image inpainting technique in [15] to modify the polyps in the images.
4. Style the resulting images using the methods of [16, 17].

Specifically, we first used an Inception-v3 model pretrained on ImageNet to obtain a feature set for all images (including the masks) in the dataset. Next, clustering is performed using scikit-learn's KMeans algorithm to obtain 20 clusters of images. See Figure 2 for examples of images in such clusters; each row depicts images in one cluster. Subsequently, we train a model on each cluster using the diffusion model implementation in [16]. The original implementation of the diffusion model involves a three-channel input corresponding to an RGB image, however, in order to generate the segmentation mask, we follow Thambawita et al. [4] and concatenate a fourth channel corresponding to the input mask. The image inpainting technique introduced by [15] was then used with the polyp mask serving as the input mask. Note that a number of the polyp masks fail to obscure the entire polyp, and thus, the model will recognize the polyp in its training set and return the original image. As a result, we first dilated the polyp mask by 20 pixels to reduce the number of such cases. Finally, we use a combination of the styling methods in [16, 17] to generate the final synthetic images.

Our method can generate multiple synthetic training images and labels from a single real image and label. We apply this procedure to all images in the source dataset from which we want to generate synthetic data.

To evaluate our samples, we had an expert review sample outputs, and we used Fréchet Inception Distance (FID) [18] and Multi-Scale Structural Similarity (MS-SSIM) [19] to respectively measure the distance between the output images and the dataset, as well as the variance of the output images. Subsequently, we trained a polyp segmentation model [20] with the original and synthetic dataset to compare their mean IoU with the target masks in order to evaluate their effect on model performance. Moreover, we compared the performance of the model when trained under different augmentation methods, namely, with cropping and rotation, with synthetic data generated by [4], and with our synthetic data. [1]

[1]We further split this case into two: one where the model was trained on each cluster, and the other where the model was trained on the entire dataset.

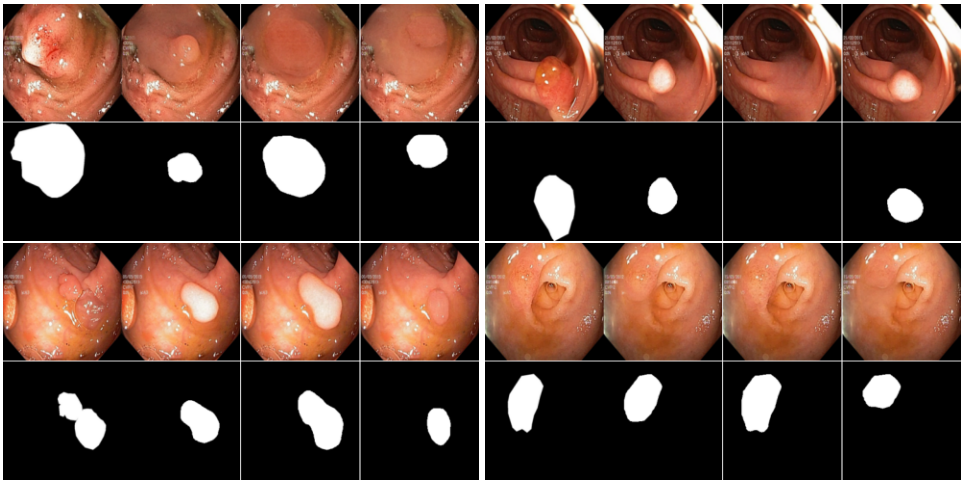

Figure 3: Example of inpainting results.

3 Results

3.1 Training diffusion models

To generate the synthetic data, we first trained our diffusion model one by one for each cluster we created; we use the 4th input channel to simultaneously generate the mask. Our method inherits from the diffusion model implementation of [16], which we trained for 110,000 iterations with hyperparameters given in Table A.1. Notably, we train the model on images of size 256×256, and we use a four channel model instead of the original three channels. The models were trained on Mist, a SciNet GPU cluster consisting of 54 IBM AC922 servers. Each node of the cluster has 32 IBM Power9 cores, 256GB RAM and 4 NVIDIA V100-SMX2-32GB GPU with NVLINKs in between, but only one GPU was used for training. The training time for each model was at most 16 hours, with variation caused by cluster size.

3.2 Image inpainting

After training the generative diffusion models, we generated 3 random samples for each image using the inpainting technique introduced in [15] with the parameters given in Table A.2. This procedure uses the polyp mask to obscure the region of the image coloured white in the mask. It is then the task of the diffusion model to fill in this gap. The upside of this procedure as compared to generating the full image is that we condition the diffusion model on the surrounding region. This significantly contributes to the realism of the image, while also injecting considerable variation to the image. Four real training images and three corresponding inpainted images are depicted in Figure 3. The first column of each collection represents the original image; the corresponding mask of each image is directly beneath it.

3.3 Style transfer

After sampling 348×3 inpainted images, the style transfer procedure [16, 17] was applied to each image. A NVIDIA RTX A6000 was used for this procedure, which was carried out by the previously trained diffusion model. Each image required about 1 minute to style. The hyperparameters of this procedure are given in Table A.3. Of note is the ratio between content and style losses. We visually compared different ratios, and found that a roughly $1 : 100$ content:style ratio works best. This ratio differs from the $1 : 1000$ given by Thambawita et al. [4], which we found to add extraneous elements to the image.

We combined the two methods by applying [17] for the content loss and [16] for the style loss. This takes advantage of the improvements in [17] while using a style loss that does not require a text

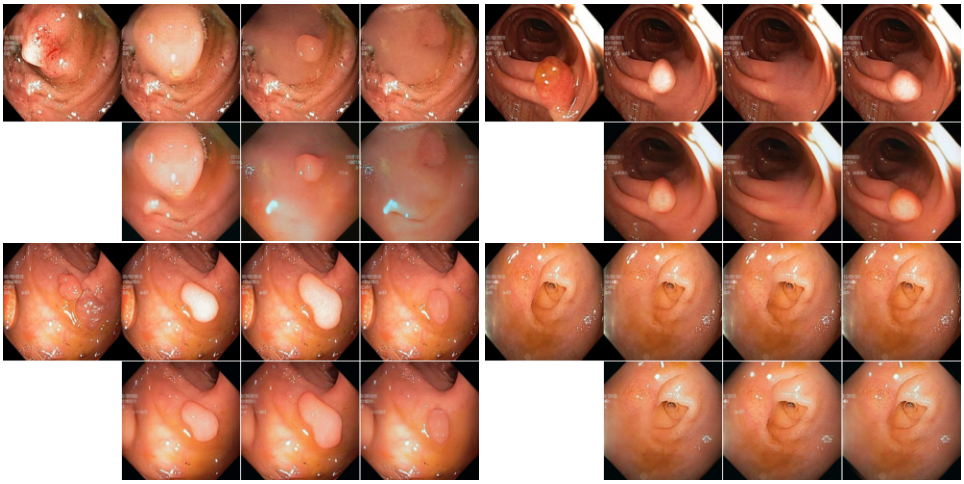

Figure 4: Example of styled results.

prompt. Therefore, we may summarize our loss functions as below:

$$\begin{aligned}
\mathcal{L}_{content} &= \lambda_{ZeCon}\ell_{ZeCon}(\hat{x}_{0,t}, x_0) + \lambda_{VGG}\ell_{VGG}(\hat{x}_{0,t}, x_0) + \lambda_{MSE}\ell_{MSE}(\hat{x}_{0,t}, x_0) \\
\mathcal{L}_{style} &= \lambda_{sty}\ell_{sty}(\hat{x}_{0,t}, x_{trg}) + \lambda_{L2}\ell_{L2}(\hat{x}_{0,t}, x_{trg}) + \lambda_{sem}\ell_{sem}(x_t; x_{t-1}) + \lambda_{rng}\ell_{rng} \\
\mathcal{L}_{total} &= \mathcal{L}_{content} + \mathcal{L}_{style}.
\end{aligned}$$

The results of this procedure are depicted in Figure 4. The first row of each collection corresponds to the first row of each collection in Figure 3, whereas the second row is the styled image.

3.4 Intrinsic evaluation

In addition to visual comparisons, we use FID [18] and MS-SSIM [19] to compare the real and synthetic images. Parmar et al. [21] showed that differences in image compression and resizing, among other factors may induce significant variation in the FID scores. Hence, we use the method introduced in [21] to compute the FID. These values are shown in Table 1 and Table 2 respectively.

We introduce the following convention when referring to each data type:

- **Real**: Real images from the training set.
- **GAN**: Synthetic images generated by SinGAN-Seg [4], downloaded at `https://osf.io/xrgz8/`.
- **Diff**: Images generated using the inpainting technique by a diffusion model trained on a single cluster. No styling applied.
- **Styled-Diff**: Images generated using the inpainting technique by a diffusion model trained on a single cluster. Style transfer applied.
- **Full-Diff**: Images generated using the inpainting technique by a diffusion model trained on the full dataset. No styling applied.
- **Full-Styled-Diff**: Images generated using the inpainting technique by a diffusion model trained on the full dataset. Style transfer applied.

In conclusion, the images generated through our diffusion pipeline has a significantly lower FID and slightly higher MS-SSIM as compared to the GAN images. As FID is a measure of distance between two sets of images, we may say that the diffusion-generated images are closer to the real images. However, we should be careful in interpreting this evaluation because our inpainting technique necessarily leaves parts of the original image intact. Our caution is supported by the fact that our pipeline without styling has a lower FID than the styled pipeline. However, as we see in Figure 4, the images without styling contains noticeable sudden changes in colour at the inpainting region, resulting in an unnatural appearance. Furthermore, for our purpose of training segmentation models, some deviation from the real images may be desirable.

Table 1: FID comparison between real and synthetic images. Images are shuffled, and the *Real* row is compared with itself.

Data type	FID		
	Set 1	Set 2	Set 3
Real	-4.412×10^{-5}		
GAN	118.73	119.45	116.92
Diff	36.01	37.90	37.78
Styled-Diff	66.17	65.48	66.32
Full-Diff	40.59	40.77	40.27
Full-Styled-Diff	60.15	59.98	59.95

Table 2: MS-SSIM Comparison

Data type	MS-SSIM		
	Set 1	Set 2	Set 3
Real		0.1813	
GAN	0.1986	0.2004	0.2037
Diff	0.2081	0.2126	0.2096
Styled-Diff	0.2304	0.2300	0.2433
Full-Diff	0.2030	0.2058	0.2078
Full-Styled-Diff	0.1971	0.1924	0.1969

On the other hand, there isn't much difference in the MS-SSIM scores – a measure of similarity between images in the set. This suggests that the variance of the synthetic datasets are similar to the real dataset. This is expected as the diffusion model is trained on the real dataset, and the inpainting technique may only add elements from other images in the dataset.

3.5 Experiments with segmentation models

Full Synthetic Training. In addition to the intrinsic evaluations above with the FID and MS-SSIM, we also evaluated the synthetic images by training a segmentation model with them. We used the same model as [20], which uses a UNet++ backbone. Specifically, the model was trained only using the generated data and tested using real data. We used the `se_resnext50_32x4d` network as the UNet++ encoder and `softmax2d` as the last layer activation function. PyTorch was used as the devlopment framework, and the data stream was handled by PYRA along with the Albumentations augmentation library. The standard training augmentations used were: `ShiftScaleRotate`, `HorizontalFlip`, and one of `CLAHE, RandomBrightness, RandomGamma`. The model was trained for 150 epochs, with a learning rate of 0.0001 for 50 epochs, which was reduced to 0.00001 for the remaining epochs. We ran three sets of experiments, each labelled "Type-N", where $N = 1, 2, 3$ and Type represents which of the synthetic sets were used. The experiment Diff-N, for example, means that $N \times 348$ images were used in the training set, all chosen from the diffusion model output without styling. These were then split into training (80%) and validation (20%) sets. For the test set, we found the HyperKvasir [14] dataset at `https://datasets.simula.no/hyper-kvasir/`, which contains 1000 images with corresponding masks. We selected images not used in the 348-size training set, and randomly chose 200 of these images as the test set. Tests were conducted only on the best checkpoint, as determined by the validation IOU score. The results are given in Table 3.

Augmented Synthetic Training. Subsequently, we experimented with the effect of augmentation on model performance. We follow the approach of Trabucco et al. [22] when performing augmentation. In particular, we fix $\alpha \in (0, 1)$, and sample indices i, j uniformly

$$i \sim U(1, \ldots, N), \quad j \sim U(1, \ldots, M)$$

where N is the number of images in the training set and M is the number of synthetic images per real image – in our case $N = 348, M = 3$. With probabilty α, a synthetic image $\tilde{X}_{ij}$ generated from real

Table 3: Metrics to compare real versus synthetic performance with segmentation model.

Data type	Scores	
	Dice Loss	IOU
Real	0.1320	0.8085
GAN-1	0.3447	0.5790
Diff-1	0.3434	0.5948
Styled-Diff-1	**0.2556**	**0.6840**
Full-Diff-1	0.3189	0.6149
Full-Styled-Diff-1	0.3486	0.5751
GAN-2	0.3447	0.5790
Diff-2	0.3434	0.5948
Styled-Diff-2	**0.2272**	**0.7027**
Full-Diff-2:	0.3403	0.5983
Full-Styled-Diff-2	0.3923	0.5534
GAN-3	0.3447	0.5790
Diff-3	0.3434	0.5948
Styled-Diff-3	**0.1958**	**0.7396**
Full-Diff-3	0.3915	0.5501
Full-Styled-Diff-3	0.4351	0.5213

Table 4: Metrics to compare synthetic augmentation performance ($\alpha = 0.5$) with segmentation model.

Data type	Scores	
	Dice Loss	IOU
Real	0.1320	0.8085
GAN	0.1873	0.7613
Diff	0.1768	0.7567
Styled-Diff	**0.1764**	0.7644
Full-Diff	0.2044	0.7568
Full-Styled-Diff	0.2006	**0.7716**

image X_i would be added to the training set. Our baseline augmentation experiments used $\alpha = 0.5$, as was used in [22].

Our experiments used the same parameters as those in the Full Synthetic Training experiments. Of note is the training augmentations mentioned above (henceforth referred to as "Control") which are used in one of the two runs. As can be seen in Table 4 and Table 5, the results of the synthetic images are very close to the scores of the real images. The augmented training performs slightly worse than the real dataset, however, the difference is quite small. In fact, the difference between the synthetic sets are also very small. This suggests that the augmentations do not mislead the model, and that the synthetic images are realistic enough to be used in training, however they do not add extra information to the training set that would improve model performance. Moreover, synthetic augmentation does not seem to complete the distribution of data in the training set, as the significant difference between Table 4 and Table 5 are caused by the addition of standard augmentation strategies.

Augmentation of Small Datasets. We also experimented with the effect of augmentation on small datasets. Because of resource and time limitations, we restrict our attention to **Styled-Diff** and **GAN**. The same parameters and augmentation methods were used as above, however, we experimented with real training sets of size $N \in \{16, 32, 64, 128\}$. Moreover, as it is typically the case that we have more synthetic data per real image, instead of simple addition with probabilty α, we add $r = 1, 2,$ or 3 synthetic images generated from X_i. The results are given in Table 6.

Table 5: Metrics to compare (Control + synthetic) augmentation performance ($\alpha = 0.5$) with segmentation model.

Data type	Scores	
	Dice Loss	IOU
Real	0.1320	0.8085
GAN	0.1394	0.8015
Diff	0.1606	0.7841
Styled-Diff	**0.1284**	**0.8084**
Full-Diff	0.1425	0.8013
Full-Styled-Diff	0.1450	0.7902

Table 6: Metrics to compare synthetic augmentation performance ($\alpha = 0.5$) on segmentation model with small N training sets. r is the augmented number of synthetic images per real image.

N	Data type	Scores, $r = 1$		Scores, $r = 2$		Scores, $r = 3$	
		Dice Loss	IOU	Dice Loss	IOU	Dice Loss	IOU
16	Real			0.6056	0.5263		
	Styled-Diff	**0.4838**	**0.5930**	**0.4886**	**0.5968**	0.5471	**0.6072**
	GAN	0.5165	0.5658	0.5003	0.5762	**0.5243**	0.5863
32	Real			0.5580	0.6201		
	Styled-Diff	0.4466	**0.6494**	**0.3392**	**0.6690**	0.2856	**0.6812**
	GAN	**0.4136**	0.6326	0.3413	0.6609	**0.2695**	0.6765
64	Real			0.4201	0.6930		
	Styled-Diff	0.2845	**0.7262**	**0.2851**	0.7133	**0.2187**	**0.7294**
	GAN	**0.2420**	0.7184	0.2976	**0.7142**	0.2493	0.7098
128	Real			**0.1768**	**0.7982**		
	Styled-Diff	0.1878	0.7511	0.2281	0.7525	0.2227	0.7494
	GAN	0.1770	0.7536	0.1900	0.7513	0.1927	0.7511

Expert Review Details For our expert review of the generated image, we had a radiologist review 30 images and their masks side by side. Out of these 30 images, 10 were real images, 10 were from **Diff**, and 10 were from **Styled-Diff**. The pairs of images were shuffled, and three assessments were made: image quality, segmentation quality, and whether the image was real or synthetic. The image and segmentation quality assessments had 3 options: Good, Moderate, and Poor. The results of the expert review are shown in Table 7.

Table 7: Expert Review Results

	Image Quality			Segmentation Quality			Real or Fake	
Data type	Good	Moderate	Poor	Good	Moderate	Poor	Real	Fake
Real	10	0	0	8	2	0	4	6
Diff	10	0	0	10	0	0	3	7
Styled-Diff	10	0	0	9	1	0	3	7

It is interesting to see how the real images were rated since it was the category with the highest number of fake images identified. This is likely caused by some bias in the assessment of the images. For example, as seen in Figure 5, the real images have a more jagged segmentation mask, whereas the synthetic images have a smoother mask. This is a well known artifact of segmentation software, that makes the masks appear unnatural. We can also perform a Cohen's Kappa test for binary classification (we combine the two synthetic sets into one) to compute a measure of agreement in the classfication

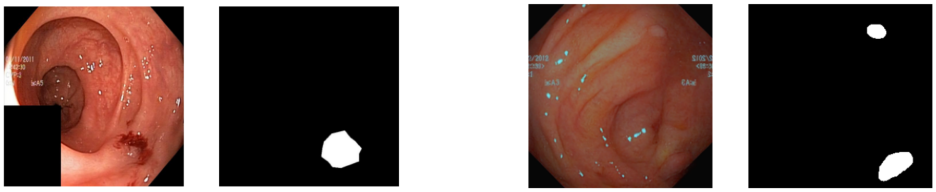

Figure 5: Example of survey images. Left pair is Real, whereas right pair is Fake.

of real vs. fake images:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$
$$p_o = \frac{TP + TN}{TP + TN + FP + FN}$$
$$p_e = \frac{(TP + FP)(TP + FN) + (TN + FP)(TN + FN)}{(TP + TN + FP + FN)^2}$$

We obtained a kappa value of $\kappa = 0.428$ which indicates moderate agreement, but leaves room for improvement especially for the real images.

4 Discussion

Our pipeline consists of multiple steps: first we clustered the images, then we generated inpainted images using a diffusion model, and finally we styled the images. We developed this pipeline to answer the questions we posed in the introduction.

4.1 As a Generation Procedure

We found that diffusion models are a viable method of generating synthetic data for medical image segmentation that enables further engineering efforts such as inpainting ans styling. Clustering the data also seems to have a positive effect on the quality of the synthetic data, as shown by our intrinsic evaluation and also when training a segmentation model. These techniques allow us to generate more realistic synthetic data, as determined by visual comparison and by the FID score. In conclusion,

On the other hand, there isn't much difference in the MS-SSIM scores – a measure of similarity between images in the set. This suggests that the variance of the synthetic datasets are similar to the real dataset. This is expected as the diffusion model is trained on the real dataset, and the inpainting technique may only add elements from other images in the dataset.

Moreover, by visual comparison in Figure 4, we see that styling improves the appearance of the synthetic images. This was done by smoothing out the inpainting region so as to remove the sudden change in colour. Moreover, it should be noted that the images generated through our diffusion pipeline has a significantly lower FID and slightly higher MS-SSIM as compared to the GAN images. As FID is a measure of distance between two sets of images, we may say that the diffusion-generated images are closer to the real images. However, we should be careful in interpreting this evaluation because our inpainting technique necessarily leaves parts of the original image intact. Our caution is supported by the fact that our pipeline without styling has a lower FID than the styled pipeline. For instance, the top left collection in Figure 4 shows a drastic change in the surroundings so as to better match the inpainted region. On one hand, this creates a more natural appearance, but on the other, it increases the distance between the synthetic and real images. However, as we see in Figure 4, the images without styling contains noticeable sudden changes in colour at the inpainting region, resulting in an unnatural appearance. Furthermore, for our purpose of training segmentation models, some deviation from the real images may be desirable. It should also be noted that despite this visual improvement, the FID score is higher for the styled images. This is likely because when smoothing the image, the added element from inpainting influences the surroundings.

4.2 As an Augmentation Procedure

In our first segmentation experiment, shown in Table 3, we trained the model only using generated data and tested using real data not used in the generation. We see in Table 3 that the diffusion-generated data performed better than the GAN-generated data. Moreover, styling resulted in significant improvement in the Dice loss and IOU scores, only behind the real data by about 0.12 in both measures. When we subsequently experimented with augmentation, we saw that the margin between the real and augmented sets are not far apart. However, we may hypothesize based on the difference between Table 4 and Table 5 that the synthetic data is not as effective as the Control strategy when augmenting the full dataset. This is likely because the synthetic data is generated from the same distribution as the real data, and thus does not add much new information. In contrast, when augmenting small datasets, as in Table 6, we see that the synthetic data performs better than the Control strategy. This is likely because the synthetic data incorporates elements from images not in the training set, and thus contributes new information. Indeed, this effect is most noticeable when the training set is starved for data, such as when $N = 16$, where we see the largest improvement in the Dice loss and IOU scores. Moreover, the effect converges as the training set size increases, as shown by the smaller improvement in the Dice loss and IOU scores when $N = 128$.

Our results therefore suggest that data generated by our pipeline may be used to either: replace the real data in the training set, or augment the real data in small training sets. In the former case, we see that the synthetic data performs better than the GAN-generated data (Table 3), and in the latter case, we see an improvement over the Control strategy and GANs (Table 6). As opposed to manual gathering and labelling of data, our pipeline is fully automated and thus can be used to generate large amounts of data with minimal human effort. In particular, our pipeline may increase the size and quality of small datasets by generating an arbitrary number of synthetic images from one real image.

5 Conclusion

In this paper, we explored the use of diffusion models to generate synthetic data for medical image segmentation. We found that diffusion models are a viable method of generating synthetic data, and that styling improves the appearance of the synthetic images. Moreover, we found that when a segmentation model is trained using synthetic data from our diffusion pipeline, the model performs better than when trained using GAN-generated data. Furthermore, styling the synthetic images resulted in a significant improvement in the Dice loss and IOU scores on the task of segmentation. This suggests that diffusion models are a promising method of generating synthetic data for medical image segmentation.

Further work is needed to fully explore the potential of diffusion models for synthetic data generation. For instance, application of our technique to other, small, low-similarity image segmentation datasets may be done to reinforce the results in this paper. Moreover, it would be interesting to explore the effect of using differentially-private diffusion models [23]. If such models do not compromise performance, then we may extend our method to a wider range of datasets and enable the use of our pipeline as an image-sharing technique, i.e. instead of the real images, we may instead share images generated from them. Finally, in this paper, we were constrained by time and computational resources to only explore 3 image samples per real image. It was shown by Fort et al. [24] that multiple augmentations per image can improve performance. Thus, it would be interesting to explore the effect of increasing the number of synthetic images per real image and augmenting with $N > 3$ images.

References

[1] Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. Artificial intelligence in healthcare: Past, present and future. *Stroke and Vascular Neurology*, 2(4), December 2017. ISSN 2059-8688, 2059-8696. doi: 10.1136/svn-2017-000101.

[2] Kun-Hsing Yu, Andrew L. Beam, and Isaac S. Kohane. Artificial intelligence in healthcare. *Nature Biomedical Engineering*, 2(10):719–731, October 2018. ISSN 2157-846X. doi: 10.1038/s41551-018-0305-z.

[3] Martin J. Willemink, Wojciech A. Koszek, Cailin Hardell, Jie Wu, Dominik Fleischmann, Hugh Harvey, Les R. Folio, Ronald M. Summers, Daniel L. Rubin, and Matthew P. Lungren. Preparing Medical

Imaging Data for Machine Learning. *Radiology*, 295(1):4–15, April 2020. ISSN 0033-8419. doi: 10.1148/radiol.2020192224.

[4] Vajira Thambawita, Pegah Salehi, Sajad Amouei Sheshkal, Steven A. Hicks, Hugo L. Hammer, Sravanthi Parasa, Thomas de Lange, Pål Halvorsen, and Michael A. Riegler. SinGAN-Seg: Synthetic training data generation for medical image segmentation. *PLOS ONE*, 17(5):e0267976, May 2022. ISSN 1932-6203. doi: 10.1371/journal.pone.0267976.

[5] Agus Pratondo, Chee-Kong Chui, and Sim-Heng Ong. Integrating machine learning with region-based active contour models in medical image segmentation. *Journal of Visual Communication and Image Representation*, 43:1–9, February 2017. ISSN 1047-3203. doi: 10.1016/j.jvcir.2016.11.019.

[6] Yuan Xu, Yuxin Wang, Jie Yuan, Qian Cheng, Xueding Wang, and Paul L. Carson. Medical breast ultrasound image segmentation by machine learning. *Ultrasonics*, 91:1–9, January 2019. ISSN 0041-624X. doi: 10.1016/j.ultras.2018.07.006.

[7] Fernando Navarro, Suprosanna Shit, Ivan Ezhov, Johannes Paetzold, Andrei Gafita, Jan C. Peeken, Stephanie E. Combs, and Bjoern H. Menze. Shape-Aware Complementary-Task Learning for Multi-organ Segmentation. In Heung-Il Suk, Mingxia Liu, Pingkun Yan, and Chunfeng Lian, editors, *Machine Learning in Medical Imaging*, Lecture Notes in Computer Science, pages 620–627, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32692-0. doi: 10.1007/978-3-030-32692-0_71.

[8] Younghak Shin, Hemin Ali Qadir, and Ilangko Balasingham. Abnormal Colon Polyp Image Synthesis Using Conditional Adversarial Networks for Improved Detection Performance. *IEEE Access*, 6:56007–56017, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2872717.

[9] Lydia Lindner, Dominik Narnhofer, Maximilian Weber, Christina Gsaxner, Malgorzata Kolodziej, and Jan Egger. Using Synthetic Training Data for Deep Learning-Based GBM Segmentation. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6724–6729, July 2019. doi: 10.1109/EMBC.2019.8856297.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[11] Ali Borji. Generated Faces in the Wild: Quantitative Comparison of Stable Diffusion, Midjourney and DALL-E 2, June 2023.

[12] Gustav Müller-Franzes, Jan Moritz Niehues, Firas Khader, Soroosh Tayebi Arasteh, Christoph Haarburger, Christiane Kuhl, Tianci Wang, Tianyu Han, Sven Nebelung, Jakob Nikolas Kather, and Daniel Truhn. Diffusion Probabilistic Models beat GANs on Medical Images. https://arxiv.org/abs/2212.07501v1, December 2022.

[13] Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective Data Augmentation With Diffusion Models, May 2023.

[14] Hanna Borgli, Vajira Thambawita, Pia H. Smedsrud, Steven Hicks, Debesh Jha, Sigrun L. Eskeland, Kristin Ranheim Randel, Konstantin Pogorelov, Mathias Lux, Duc Tien Dang Nguyen, Dag Johansen, Carsten Griwodz, Håkon K. Stensland, Enrique Garcia-Ceja, Peter T. Schmidt, Hugo L. Hammer, Michael A. Riegler, Pål Halvorsen, and Thomas de Lange. HyperKvasir, a comprehensive multi-class image and video dataset for gastrointestinal endoscopy. *Scientific Data*, 7(1):283, August 2020. ISSN 2052-4463. doi: 10.1038/s41597-020-00622-y.

[15] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11451–11461, June 2022. doi: 10.1109/CVPR52688.2022.01117.

[16] Gihyun Kwon and Jong Chul Ye. Diffusion-based Image Translation using Disentangled Style and Content Representation, February 2023.

[17] Serin Yang, Hyunmin Hwang, and Jong Chul Ye. Zero-Shot Contrastive Loss for Text-Guided Diffusion Image Style Transfer, April 2023.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[19] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402 Vol.2, November 2003. doi: 10.1109/ACSSC.2003.1292216.

[20] Vajira Thambawita, Steven A. Hicks, Pål Halvorsen, and Michael A. Riegler. DivergentNets: Medical Image Segmentation by Network Ensemble, July 2021.

[21] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On Aliased Resizing and Surprising Subtleties in GAN Evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11410–11420, 2022.

[22] Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective Data Augmentation With Diffusion Models, May 2023.

[23] Sahra Ghalebikesabi, Leonard Berrada, Sven Gowal, Ira Ktena, Robert Stanforth, Jamie Hayes, Soham De, Samuel L. Smith, Olivia Wiles, and Borja Balle. Differentially Private Diffusion Models Generate Useful Synthetic Images, February 2023.

[24] Stanislav Fort, Andrew Brock, Razvan Pascanu, Soham De, and Samuel L. Smith. Drawing Multiple Augmentation Samples Per Image During Training Efficiently Decreases Test Error, February 2022.

A Appendix

A.1 Hyperparameters

Table A.1: Diffusion model hyperparameters

Hyperparameters	Value
schedule_sampler	"uniform"
lr	1e-4
weight_decay	0.0
lr_anneal_steps	0
batch_size	1
microbatch	-1
attention_resolutions	"16, 8"
class_cond	False
diffusion_steps	1000
rescale_timesteps	True
timestep_respacing	200
skip_timesteps	80
image_size	256
learn_sigma	True
noise_schedule	"linear"
num_channels	256
num_head_channels	64
num_res_blocks	2
resblock_updown	True
use_fp16	True
use_scale_shift_norm	True
in_channels	4
num_heads	4
num_heads_upsample	-1
num_head_channels	-1
channel_mult	""
dropout	0.0
use_new_attention_order	False
use_kl	False
predict_xstart	False
rescale_learned_sigmas	False

Table A.2: Hyperparameters used in the inpainting procedure.

Hyperparameter	Value
n_sample	1
jump_length	10
jump_n_sample	5

Table A.3: Hyperparameters used in the style transfer procedure.

Hyperparameter	Value
λ_{ZeCon}	500
λ_{VGG}	100
λ_{MSE}	5000
λ_{sty}	10000
λ_{L2}	10000
λ_{sem}	40000
λ_{rng}	200