A European Cancer Image Platform Linked to Biological and Health Data for Next-Generation Artificial Intelligence and Precision Medicine in Oncology

Deliverable D4.4: Data enhancement through synthetic data generation

Reference	D4.4_ EuCanImage_LYN_v1
Lead Beneficiary	Lynkeus
Author(s)	Konstantinos Kechagias, Davide Zaccagnini, Richard Osuala
Dissemination level	PU
Type	R
Official Delivery Date	3rd October 2022
Date of validation of the WP leader	3rd October 2022
Date of validation by the Project Coordinator	3rd October 2022
Project Coordinator Signature	[illegible]

EuCanImage is funded by the European Union’s H2020 Framework
Under Grant Agreement No 952103

Version log

Issue Date	Version	Involved	Comments
18 Sep 2022	V0.1	Konstantinos Kechagias	First draft of the FL framework
27 Sep 2022	V0.2	Davide Zaccagnini	Second draft
28 Sep 2022	V0.3	Konstantinos Kechagias	Change Images, merge comments, fix typos
29 Sep 2022	V0.4	Richard Osuala	Modified and extended text in medigan section, included figures, comments, and further references (e.g. GAN models developed within EuCanImage)
03 Oct 2022	V0.5	Davide Zaccagnini	Final draft
03 Oct 2022	V1	Anais Emelie & Karim Lekadir	Revised and corrected final version.

Acronyms

Name	Abbreviation
Generative Adversarial Network	GAN
Conditional Generative Adversarial Network	CGAN
Federated Learning	FL
Magnetic Resonance Imaging	MRI
Computerized Tomography	CT

Executive Summary

Synthetic data are rapidly gaining recognition as a privacy-preserving methodology to share medical information between distrustful parties or publishing anonymous copies of data sets in public digital spaces without revealing personal or sensitive information. Generative algorithms, on the other hand, have shown to be able to preserve statistical properties of the underlying original distributions providing training sets for machine learning tool. These frameworks suffer from the issue of data scarcity, especially in regards to images carrying clinically relevant features which may be scarce in original datasets. For these reasons a key goal of this task was to implement generative pipelines which were, on one side, demonstrably guaranteeing privacy and on the other capable of producing synthetic data containing specific clinical features (attribute selection) even when such features were not sufficiently represented in original data. Progress made during the task in this direction demonstrated real potential.

The long term goal is to compose, for instance, mammograms positive for calcified nodules, by composing a "collage" of healthy tissues, nodules and calcifications in a clinically realistic image. While more work is needed on this front, Attention Mechanism, has shown to be a very effective method in this regard.

The other main area of this task has focused on the successful design and implementation of Medigan, a public, web-based environment for the management and distribution of synthetic data assets and of generative pipelines.

Parts of the task, namely the provision of enhanced data sets for the execution of other project's tasks are not included in the scope of this document. As data sources from participating centers become available, the tools developed during the task will be refined and applied to generation of target data assets.

1 Generating clinically relevant synthetic data

In order to create large samples of synthetic cancer images that accurately reflect clinical features (such as age, gender, lesion type, malignancy, location, and size), we created a user-friendly toolset leveraging Generative Adversarial Networks (GANs), which produce artificial cancer images for various imaging modalities (such as MRI and CT) under specific cancer circumstances (i.e. Conditional GANs: CGANs). This method controls the appearance of the cancer image generated by using important cancer variables (such as morphology, cancer stage, background tissue density, and scanner manufacturer) in addition to the features collected from the image. This enables, for instance, to create a specific breast cancer (triple negative breast cancer) of a specific size in a particular background region (fatty, scattered fibroglandular tissue, heterogeneously packed, highly dense), all while simulating a particular scanner (e.g. Hologic Selenia Dimensions). In future work, this model will enable the development of multi-parameter and multi-vendor cGAN models for cancer imaging datasets.

1.1 Synthetic Data and Privacy Protection

There was significant appeal to publishing data that represented "no genuine individual's" reactions when Rubin (1993) established the concept of totally synthetic data. Research after that has sufficiently shown the viability, but the fundamental query, "To what extent does the synthetic data process give protection?" remained largely unaddressed. Since then privacy in databases has evolved a well-defined framework that enables a synthesis of the methods used for disclosure restriction and privacy-preserving data mining also with the intent of creating synthetic copies of original data, allowing to create privacy-guaranteed copies of the underlying information, in order to measure the protection provided by synthetic data and the ensuing analytical validity of the release data.

Several frameworks developed within EuCanImage such us:

- High-resolution synthesis of high-density breast mammograms: Application to improved fairness in deep learning based mass detection (https://arxiv.org/abs/2209.09809)
- Sharing Generative Models Instead of Private Data: A Simulation Study on Mammography Patch Classification (https://arxiv.org/abs/2203.04961)
- nn-UNet Training on CycleGAN-Translated Images for Cross-modal Domain Adaptation in Biomedical Imaging (https://link.springer.com/chapter/10.1007/978-3-031-09002-8_47)

This is particularly relevant in medical data, and even more in medial images which are the most privacy sensitive type of health information.

In this view Lynkeus has developed and generative models which integrate Differential Privacy. Due to their elegant theoretical base and strong empirical performance as generative models, they have recently gained intense academic interest. In research when there is a lack of data, these techniques offer a promising direction. Due to the high model complexity of deep networks, one typical problem with GANs is that the density of the learnt generative distribution could focus on the training data points, making it easy for them to remember training samples. When GANs are used to analyze sensitive or private data, such as patient medical records, the concentration of distribution raises serious security concerns because it could reveal vital patient information. To solve this problem, Differentially Private GAN (DPGAN) model was introduced. In this model, differential privacy established in GANs by carefully constructing noise to be added to gradients during the learning process. DPGANs can produce high-quality data points with a respectable level of privacy. We developed DPGAN implementation which can be found in GitHub (https://github.com/athenarc/DPGANs). Also, relative work that our implementation was based on can be found in the above repository.

2 Technical Design

In terms of design choice, the most important factor is the resolution of generated images. A major constraint when resolution images generated is that current approaches failed to generate images with high resolution. We used taming transformers for high resolution image synthesis. Taming transformers bring transformer models to high-resolution picture synthesis up to the megapixel range and to take use of their highly promising learning capabilities. In order to produce locally realistic and globally consistent patterns, high-resolution image synthesis needs a model that comprehends the global composition of images. As a result, rather than describing an image with pixels, we instead compose it with codebook components that are perceptually rich. The description length of compositions can be drastically decreased by learning an efficient code, which enables us to effectively model their global interrelations within images using a transformer architecture. In both an unconditional and a conditional environment, this method may produce high quality images that are realistic and consistent. In the following diagram you can see more about the architecture that taming transformer followed:

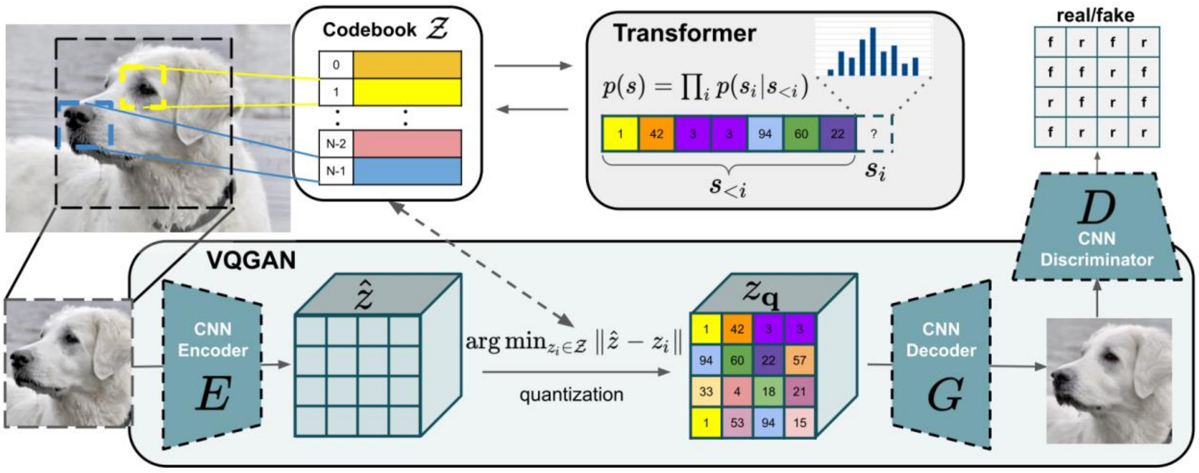


Figure 1: Overview of the transformer architecture

In the above figure, context-rich visual portions are learned as a codebook using a convolutional VQGAN, and their composition is then modeled using an autoregressive transformer architecture. The interface between these systems is a discrete codebook, and a patch-based discriminator allows for substantial compression while maintaining good perceptual quality. With this technique, high resolution picture synthesis based on transformers is made more effective.

We used the Azure Machine Learning Framework to get access into multiple GPU environments, due to high demand of GPU cores in order to train the model. Also, the model needed a specific GPU chipset to run and also required GPUs with more than 16GB of memory. So we created the environment on Azure, and then we tried different compute instances to find one that actually works. A compute instance is a fully managed cloud-based workstation optimized for your machine learning development environment.

2.1 Process

In order to synthesize images using the extremely expressive transformer architecture, we planned to express an image's component parts as a sequence. Complexity demands a method based on a discrete codebook of learnt representations rather than building on individual pixels.

To use the taming transformers, first we needed to learn an effective codebook of image constituents to use in transformers. In order to synthesize images using the extremely expressive transformer architecture, we needed to express an image's component parts as a sequence. Complexity demands a method based on a discrete codebook of learnt representations rather than building on individual pixels. Then, we needed to push the boundaries of compression and learn a comprehensive codebook in order to use transformers to represent images as a distribution over latent image elements. In order to achieve this, we used VQGAN, an alternative to the original VQVAE, which uses a discriminator and perceptual loss to maintain decent perceptual quality at higher compression rates. This is in contrast to earlier efforts that merely applied a shallow quantization model before applying pixel-based and transformer-based autoregressive models on top of it.

As a next step, we needed to learn the composition of images with transformers. When creation of a codebook is completed, we basically have to represent images in terms of the codebook-indices of their encodings. Then, we feeded this codebook to the transformer, in order to generate high resolution images.

3 Experiments

During the execution of the model, we made several attempts before tuning the model and finally got a high resolution image. We had to tune the VQGAN to produce perceptually rich codebooks.

3.1 Datasets

We tested our model in two datasets. Those datasets are Breast Cancer Digital Repository (Breast Cancer Digital Repository (bcdr.eu)) and INBreast data set (INbreast: toward a full-field digital mammographic database - PubMed (nih.gov)) both publicly available, extensive and well curated sets of mammographic images.

3.2 Results

Our first approach started generating images reaching a realistic overall shape but with very pixelated content as shown below:

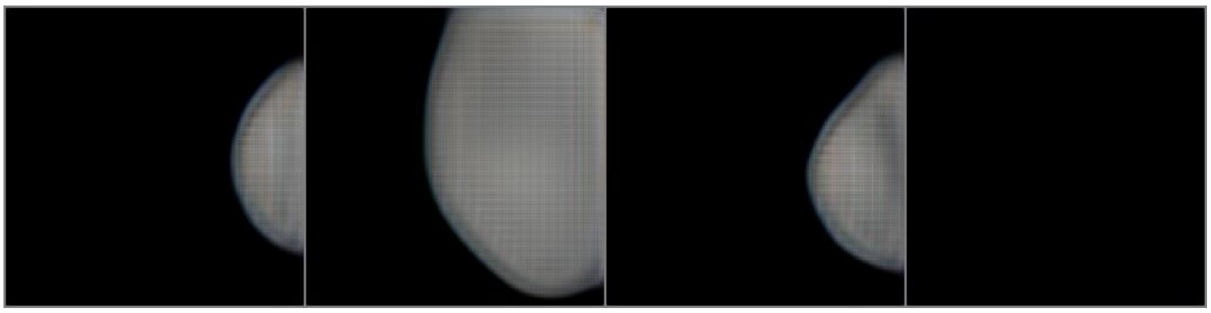


Figure 2: First approach of image generator

We needed to make several adjustments in the model, and also change the infrastructure we used. After those adjustments our approach seems to work a lot better and the produced image is shown below:

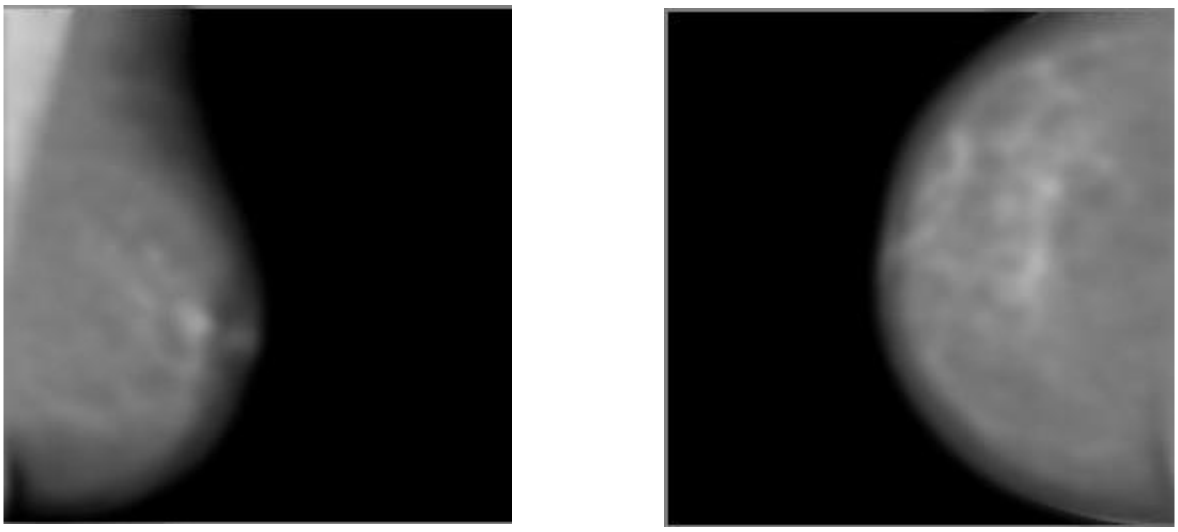


Figure 3: Examples of generated image

As shown above the model learned key and clinically realistic aspects of breast tissue (varying tissue densities and pattern of fibrosis, nipple location and rough anatomy) demonstrating actual potential in this methodology, while more data and further adjustment to the model will be required.

4 User-Friendly Data Synthesis Toolbox

We furthermore deliver medigan, a user-friendly toolbox for generating diverse sets of synthetic imaging data across modalities, organs, domains, and conditioning characteristics. The medigan toolbox is implemented as an open-source framework-agnostic Python library based on multiple pretrained generative models for automated and user-friendly synthetic data generation as shown in Figure 4.

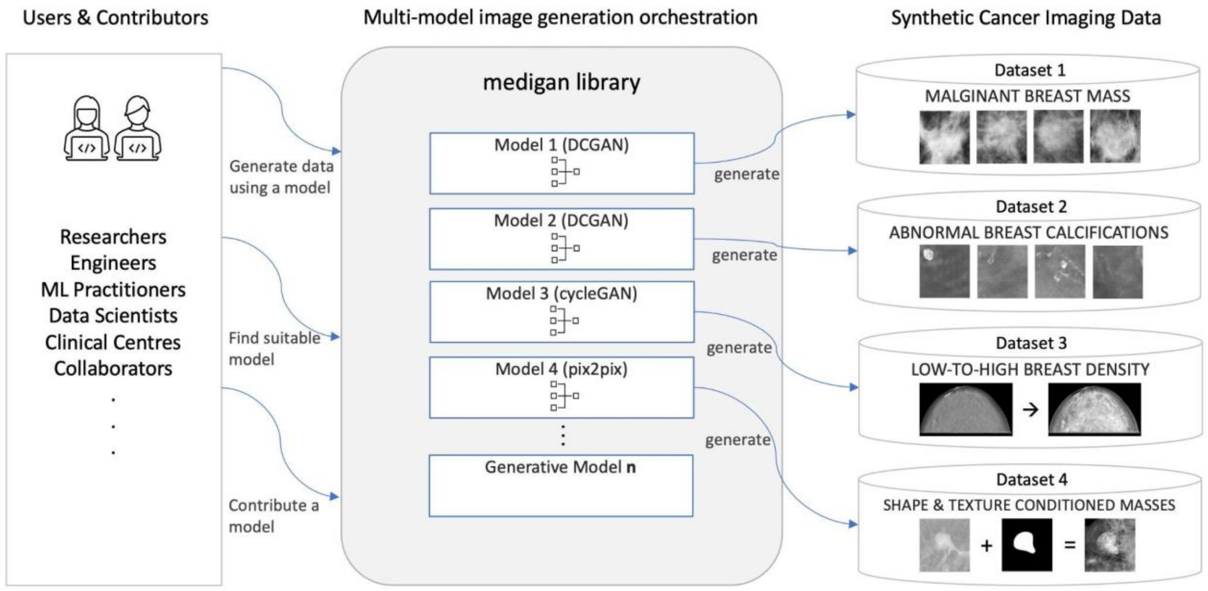


Figure 4: Overview of the open-source medigan toolbox for synthetic data generation

End-user requirement gathering is recommended for the development of trustworthy AI solutions in medical imaging. Therefore, we organised requirement gathering sessions with potential end-users, model contributors and stakeholders from the EuCanImage

Consortium. After gathering end-user requirements, design decisions based on usability, technical feasibility, and scalability are formulated as detailed below in Table 1.

Subsequently, we implement *medigan* based on modular components for generative model (i) execution, (ii) visualisation, (iii) search & ranking, and (iv) contribution. medigan is built with a focus on simplicity and usability. The integration of pretrained models is designed as internal Python package import and offers simultaneously (a) high flexibility to and (b) low code dependency on these generative models. The latter allows the reuse of the same orchestration functions in medigan for all model packages. From a user perspective, medigan allows researchers and developers to create, increase, and domain-adapt their training data in just a few lines of code.

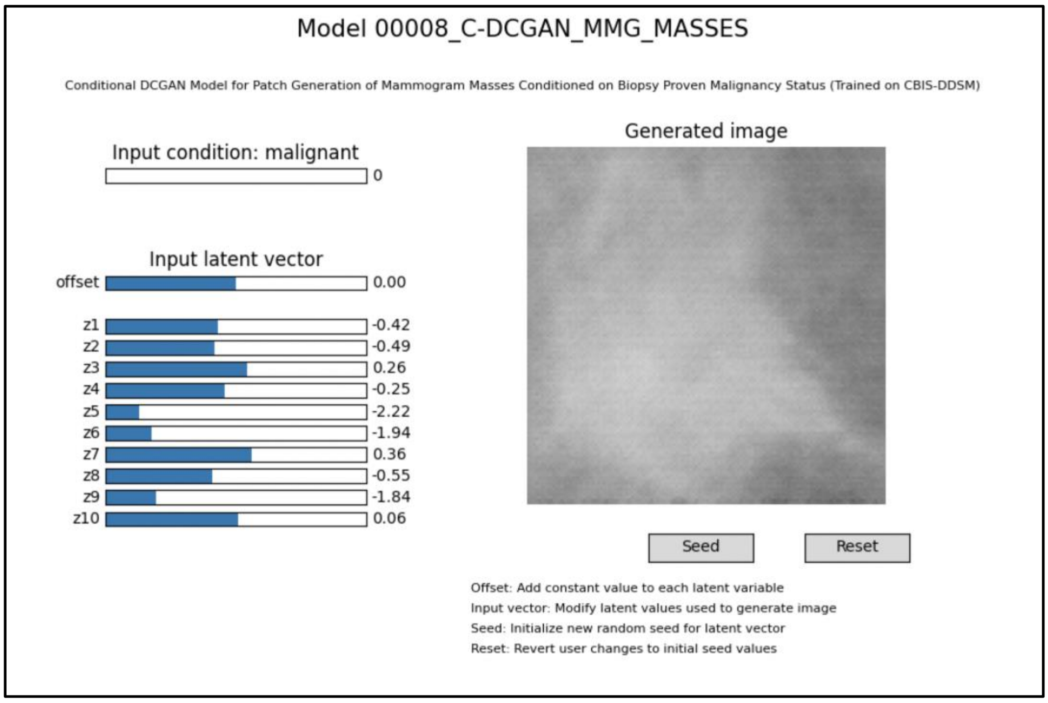
No	End-User Requirement	Respective Design Decision
1	Instead of a GUI tool, *medigan* should be implemented as a platform-independent library importable into users' code.	Implementation of *medigan* as publicly accessible Python package distributed via PyPI.
2	It should support common frameworks for building generative models, e.g., PyTorch,[67] TensorFlow,[74] Keras.[75]	*medigan* is built framework-agnostic treating each model as separate Python package with freedom of choice of framework and dependencies.
3	The library should allow different types of generative models and generation processes.	*medigan* supports any type of data generation model including GANs,[16] VAEs,[21] flow-based,[22–24] diffusion[25–27] and non-deep learning models.
4	The library should support different types of synthetic data.	*medigan* supports any type of synthetic data ranging from 2D and 3D images to image pairs, masks, and tabular data.
5	Sample generation functions should be easily integrable into diverse user code, pipelines and workflows.	*medigan*'s generate function can (i) return samples, (ii) generate folders with samples, or (iii) return a model's generate function as callable.
6	User should be able to integrate *medigan* data in AI training via a dataloader.	For each model, *medigan* supports returning a torch dataloader readily integrable in AI training pipelines, combinable with other dataloaders.
7	Despite using large deep learning models, the library should be as lightweight as possible.	Only the user-requested models are downloaded and locally imported. Thus, model dependencies are not part of *medigan*'s dependencies.
8	It should be possible to locally review and adjust a generative model of the library.	After download, a model's code and config is available for end-users to explore and adjust. *medigan* can also load models from local file systems.
9	The library should support both CPU and GPU usage depending on a user's hardware.	Contributed *medigan* models are reviewed and, if need be, enhanced to run on both GPU and CPU.
10	Version and source of the models that the library load should be transparent to the end-user.	Convention of storing *medigan* models on Zenodo, where each model's source code and version history is available.
11	There should be no need to update the version of the *medigan* package each time a new model is contributed.	*medigan* is designed independently of its model packages separately stored on Zenodo. Config updates do not require new *medigan* versions.
12	Following,[73] models are contributed in transparently and tracebly, allowing quality and reproducibility checks.	Model contribution is traceable via version control. Adding models to *medigan* requires a config change via pull request.
13	The risk that the library downloads models that contain malicious code should be minimised.	Zenodo model uploads receive static DOIs. After verification, unsolicited uploads/changes do not affect *medigan*, which points to specific DOI.
13	License and authorship of generative model contributors should be clearly stated and acknowledged.	Separation of models and library allows freedom of choice of model license and transparent authorship reported for each model.
14	Each generative model in the library should be documented.	Each available model is listed and described in *medigan*'s documentation, in the readme, and also separately in its Zenodo entry.
15	The library should have minimal dependencies on the user side and should run on common end-user systems.	*medigan* has a minimal set of Python dependencies, is OS-independent, and avoids system and third-party dependencies.
16	Contributing models should be simple and at least partially automated.	*medigan*'s contribution workflow automates local model configuration, testing, packaging, Zenodo upload, and issue creation on GitHub.
17	If different models have the same dependency but with different versions, this should not cause a conflict.	Model dependency versions are specified in the config. *medigan*'s generate method can install unsatisfied dependencies, avoiding conflicts.
19	Any model in the library should be automatically tested and results reported to make sure all models work as designed.	On each commit to main, a CI pipeline automatically builds, formats, and lints *medigan* before testing all models and core functions.
20	The library should make the results of the models visible with minimal code required by end-users.	*medigan*'s simple visualisation feature allows users to adjust a model's input latent vector for intuitive exploration of output diversity and fidelity.
21	The library should support large synthetic dataset generation on user machines with limited random-access memory.	For large synthetic dataset generation, *medigan* iteratively generates samples via small batches to avoid exceeding users' in-memory storage limits.
22	Users can specify model weights, model inputs, number and storage location of the synthetic samples.	Diverging from defaults, users can specify (i) weights, (ii) number of samples (iii) return or store, (iv) store location, (v) optional inputs.

Table 1: Overview of requirements gathered together with potential end-users next to the respective design decisions taken towards fulfilling these requirements in medigan.

The toolbox is easily integrable into other packages and tools, including commercial ones. Synthetic data can enhance the performance, capabilities, and robustness of data-hungry deep learning models as well as to mitigate common issues such as domain shift, data scarcity, class imbalance, and data privacy restrictions. Training one's own generative network can be complex and expensive since it requires a considerable amount of time, effort, specific dedicated hardware, carbon emissions, as well as knowledge and applied skills in generative AI. An alternative and complementary solution is the distribution of pretrained generative models to allow their reuse by AI researchers and engineers worldwide.

medigan can help to reduce the time to run synthetic data experiments and can readily be added as a component, e.g., as a torch dataloader, in AI training pipelines. As such, the generated data can be used to improve supervised learning models during training or fine-tuning, but can also serve as plug-and-play data source for self/semi-supervised learning, e.g., that pretrain models for a clinical downstream task.

Apart from that, medigan contains a generative model visualisation feature that allows users to explore how changes to the model input affect the synthetic data the model creates. As shown in Figure 5, users can adjust the input latent variables (alias random noise vector) that GANs use to generate images variations. Furthermore, users can adjust the conditional information that is input into conditional models (e.g. cGANs). In Figure 5, medigan's model 8 is visualised, which allows to adjust the malignancy (malignant/benign) of a generated synthetic mammogram-based tumour mass image.



Model 00008_C-DCGAN_MMG_MASSES

Conditional DCGAN Model for Patch Generation of Mammogram Masses Conditioned on Biopsy Proven Malignancy Status (Trained on CBIS-DDSM)

Input condition: malignant 0

Input latent vector
offset 0.00
z1 -0.42
z2 -0.49
z3 0.26
z4 -0.25
z5 -2.22
z6 -1.94
z7 0.36
z8 -0.55
z9 -1.84
z10 0.06

Offset: Add constant value to each latent variable
Input vector: Modify latent values used to generate image
Seed: Initialize new random seed for latent vector
Reset: Revert user changes to initial seed values

Figure 5: Example of model visualisation feature in the medigan toolbox: Malignancy conditioned tumour mass generation based on a conditional GAN.

The scalability and design of medigan is demonstrated by its growing number of integrated and readily-usable pretrained generative models, which include 21 models utilising 9 different Generative Adversarial Network architectures trained on 11 different cancer imaging datasets with the aim of solving specific cancer imaging challenges. Any type of generative model can be integrated into medigan with the ones to date already spanning a range of applications across modalities such as mammography, endoscopy, x-ray, and MRI. Among others, medigan contains 2 conditional DCGAN models (e.g., conditioned on tumour mass malignancy in mammograms), 6 domain translation CycleGAN models (e.g., conditioned on breast density in mammograms or MRI acquisition protocol) and 1 mask-to-image pix2pix model conditioned on tumour mass shapes in mammograms. Models allow to generate samples of different pixel resolutions ranging from regions-of-interest patches of size 128x128 and 256x256 to full images of 1024x1024 and 1332x800 pixels. The aforementioned VQGAN model we developed using taming-transformers is also planned to be integrated into and distributed via medigan. medigan enables the community to reuse, access, and build on the synthetic data generation efforts and achievements of EuCanImage. medigan is publicly available in its associated GitHub repository (https://github.com/RichardObi/medigan). The meta-information of the medigan library with links to documentation (https://medigan.readthedocs.io/) and distribution (via the python packaging index, https://pypi.org/project/medigan) are shown below in Table 2.

	Title	***medigan* metadata**
1	Code version	v1.0.0
2	Code license	MIT
3	Code version control system	git
4	Software languages	Python
5	Code repository	https://github.com/RichardObi/medigan
6	Software package repository	https://pypi.org/project/medigan/
7	Developer documentation	https://medigan.readthedocs.io
8	Tutorial	medigan quickstart (tutorial.ipynb)
9	Requirements for compilation	Python v3.6+
10	Operating system	OS independent. Tested on Linux, OSX, Windows.
11	Support email address	Richard.Osuala[at]gmail.com
12	Dependencies	tqdm, requests, torch, numpy, PyGithub, matplotlib (setup.py)

Table 2: Information overview of the medigan library

5 Conclusions

This task delivered both a mature framework to store, managed and distribute synthetic data assets and generative tool, and a promising albeit initial new framework for the creation of synthetic data sets with attribute selection. Despite time limitations, the teams will keep working on the evolution of generative framework aiming at providing synthetic data assets based on well defined data requirements from the AI teams once these will become available.

6 References

1. Patrick Esser, R. R. "Taming Transformers for High-Resolution Image Synthesis". in *arXiv*. 2021

2. Richard Osuala, et al. "medigan: A Python Library of Pretrained Generative Models for Enriched Data Access in Medical Imaging" in arXiv. 2022

3. Karim Lekadir, Richard Osuala, et al."FUTURE-AI: Guiding Principles and Consensus Recommendations for Trustworthy Artificial Intelligence in Medical Imaging" in arXiv. 2021

4. Lidia Garrucho, Kaisar Kushibar et al. "High-resolution synthesis of high-density breast mammograms: Application to improved fairness in deep learning based mass detection" in arXiv. 2022

5. Zuzanna Szafranowska, Richard Osuala et al, "Sharing Generative Models Instead of Private Data: A Simulation Study on Mammography Patch Classification" in arXiv. 2022

6. Joshi, S. et al. "nn-UNet Training on CycleGAN-Translated Images for Cross-modal Domain Adaptation in Biomedical Imaging". In: Crimi, A., Bakas, S. (eds) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. BrainLes 2021. Lecture Notes in Computer Science, vol 12963. Springer, Cham. (2022) https://doi.org/10.1007/978-3-031-09002-8_47

7. Kaisar Kushibar, & Richard Osuala. "PIX2PIX (BezierGAN) Model for Mammogram MASS with MASK Patch Generation (Trained on BCDR)". Zenodo. 2022. https://doi.org/10.5281/zenodo.7093760

8. Richard Osuala et al. "A Review of Generative Adversarial Networks in Cancer Imaging: New Applications, New Solutions" in arXiv. 2021

9. Abowd, J.M., Vilhuber, L. (2008). "How Protective Are Synthetic Data?".in Privacy in Statistical Databases. PSD 2008. Lecture Notes in Computer Science, vol 5262.https://doi.org/10.1007/978-3-540-87471-3_20

10. Xin Yia, Ekta Waliaa, Paul Babyn "Generative Adversarial Network in Medical Imaging: A Review" in *arXiv*. 2021

11. Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861