Generative Adversarial Networks for 3D Face Generation

Sebastian Oliver
University of Cape Town
Cape Town, South Africa
olvseb001@myuct.ac.za

Luc van den Handel
University of Cape Town
Cape Town, South Africa
vhnluc001@myuct.ac.za

Liam Watson
University of Cape Town
Cape Town, South Africa
wtslia001@myuct.ac.za

1 PROJECT DESCRIPTION

1.1 Introduction

Ever since generative adversarial networks(GANs) were introduced by Goodfellow et al. [6] in 2014 they have been able to produce some extraordinary results. Most of the applications in which they have excelled have been in the area of image manipulation and generation. One of the biggest advantages of GANs is their ability to generate novel samples which conform to the given training data. 3D facial data is limited and difficult to produce - for artistic purposes - or obtain - for research purposes. This is one of the areas where training a GAN for 3D face generation will come in handy. Not only will the GANs be able to produce more data for others to use for their own research but it also has the potential to produce photo realistic facial models for use in applications such as 3D computer graphics.

1.2 Current limitations

The current research has many limitations which can be broken into three categories, namely problem domain, quality and architecture.

1.2.1 Problem domain. The bulk of research has been focused on 3D kernel embeddings which reduce to 2D images of three dimensional faces which reduces the use cases of a model dramatically as 2D images have less applicability. Where fully three dimensional models are used the focus on a hybrid approach where a deep learning agent is used to manipulate a 3D Morphable Model (3DMM) rather than operating directly on the mesh structure [2, 14].

1.2.2 Quality. Recent approaches to mesh outputs have high quality macroscopic features such as face shape and expression. However, they tend to suffer from a lack in high frequency detail as well as unwanted high frequency aberrations when manipulating latent features [17].

1.2.3 Architecture. The vast majority of approaches have a highly entangled and complex architecture that involves many computational blocks for manipulations which limits the GANs knowledge of 3D faces. Rather the model learns an intermediate representation that is then computed into a 3D face using a 3DMM or various other techniques. Simplification through the removal of the intermediate representation, will result in a simpler architecture which more closely follows that of 2D GANs, which have been shown to posses a high degree of accuracy and quality, while remaining easier to implement and train [10].

2 PROBLEM STATEMENT

GANs have been extensively used for 2D face generation with astonishing results in papers like styleGAN[10], however, modern computational power has enabled highly detailed environments in three dimensions. Consumers in the Anthropocene era demand highly realistic 3D models for video games, movies and commercial engagement. In order to overcome the uncanny valley, a method to produce high fidelity, feature controlled human faces is needed [5] - in this paper we focus on expression manipulation as it is a clearly visable feature which has wide spread applications. Such a technique will enable many innovations in commercial applications such as virtual reality avatars, augmented reality, video game development, teleconferencing, virtual try-on, special effects in movies, with potentially many more applications as faces are ubiquitous.

2.1 Aims

The aims of this project are to evaluate and compare the effectiveness of the different types of GANs for 3D face generation - in this work we only focus on the facial structure and expression manipulation, textures may be explored in future studies. These different types include conditional, controllable and progressive. In order to assess each of the GANs fairly, an experimental platform will need to be developed. This experimental platform's purpose will be to easily interchange the type of GAN used while providing a common way of evaluating them effectively. Using the results from the experimental platform, a comparison will be made between the structures we used against the latest GANs used for 3D face generation.

2.2 Motivation

Face generation has a wide range of applications from augmented reality and virtual reality to movies and video games. In recent research, GANs have been making constant progress when it comes to face generation. Most of this research does involve 2D face generation, meaning the door is open for 3D face generation as it is less-explored, especially generating 3D faces using a 3D dataset. One of the main reasons that 3D face generation is the less explored option is the limited availability of 3D facial datasets. Recently more 3D datasets are becoming available, making 3D face generation using a 3D dataset a more viable option.

The three different types of GANs, conditional [13], controllable [11] and progressive [9], all have different strengths and weaknesses when it comes to face generation. Conditional and controllable GANs strengths lie in feature manipulation - such as making adjustments by manipulating facial expressions, age, gender, etc. - while progressive GANs focus on producing high-resolution outputs. When it comes to 3D face generation using 3D data, this area is still relatively unexplored compared to 2D face generation or 3D face generation using 2D data. This encourages the use of these architectures and compare the results with past research of GANs in 3D face generation.

The development of an experimental platform for testing these different architectures will be of great use, as it will allow easy interchange between the different architectures and provide a fair way of evaluating each type of GAN - both for this project and possibly for future work on 3D face generation.

2.3 Research questions

(1) Can GANs be used to generate arbitrary 3D mesh faces
(2) Can GANs be used to manipulate 3D facial expressions directly on 3D meshes
(3) Can GANs be used to generate high fidelity 3D mesh faces

3 PROCEDURE AND METHODS

3.1 Methodology

3.1.1 Datasets. There are numerous datasets which are available containing various types of facial data. For this work we will use the most commonly explored in the literature to enable comparison of results as well as ease of development.

(1) FaceWareHouse raw: 640 × 480 RGBD; processed: triangle mesh (11K vertices, consistent topology). 150 individuals × 20 expressions.
(2) CoMA triangle mesh (80K–140K vertices), texture images (avg resolution 3,700 × 3,200), six raw camera images (each 1,600 × 1,200), alignments in FLAME topology. full head including face, neck, ears. 12 individuals × 12 extreme expression sequences.
(3) BU-3DFE triangle mesh (20K–35K triangles), two texture images (1,300 × 900) face, neck, sometimes ears 100 individuals × 25 expressions.
(4) 4DFAB triangle mesh (60K–75K vertices), UV texture map face, neck, and ears 180 individuals × 4K–16K frames of dynamic sequences.

3.1.2 Testing Pipeline (explanation of pipeline). The pipeline can be broken up into seven discreet components.

(1) Dataset component which is the overall set of data that will be used for both training and testing model performance.
(2) In order to ensure that data is in the correct format and is ready for the model a pre-processing step is needed. Particularly important in this section is down sampling of highly detailed mesh scans of faces which is required as GANs have very high memory and computational requirements. Additionally steps will need to be taken to separate expression from identity such as subtraction, model registration and local smoothing.
(3) With data ready for the model we require a methodology that loads data into the model, taking care of batching and ensuring correct data delivery and format.
(4) The model will then fit into the pipeline in a plug and play fashion enabling agile testing and rapid investigation.
(5) In order to produce a high quality model a reliable training framework is needed where all the required steps for the specific GAN are completed to ensure model convergence. Included in the training framework should be a robust testing scheme to monitor the training progress.
(6) Given a trained model the team must have a robust and accurate evaluation framework that is model independent. The framework will include a suite of quantitative metrics as mentioned before as well as access to a suitable display method where qualitative tests can take place.
(7) Additionally the pipeline should include a model saving and loading fame work to ensure that well trained models can be retested at a later stage.

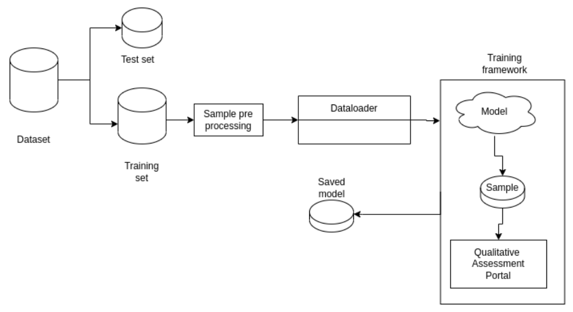

Figure 1: Diagram showing the pipeline required training the model under consideration

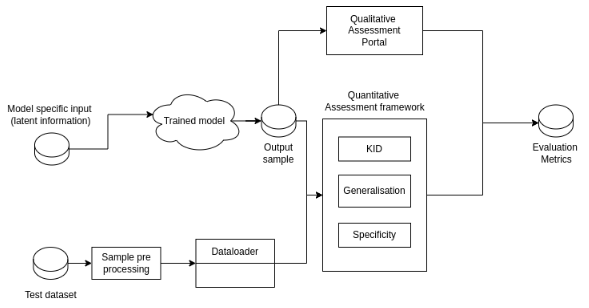

Figure 2: Diagram showing the pipeline required for evaluation of a trained model

3.2 Training and testing

3.2.1 Comparison methods and models. MeshGAN by Cheng et al. [2] will be used as the reference study for this project. Therefore it makes sense to use the CoMA autoencoder by Ranjan et al. [17] as the baseline for comparison, as this is the key benchmark they use in MeshGAN. The CoMA autoencoder will be run on the experimental platform, which will measure certain quantitative and qualitative characteristics.

3.2.2 Quantitative. Quantitative measurements are all about the accuracy of how well GANs are able to generate objects, in this case, faces. It is common to report back on the accuracy achieved for each of the different emotions(happy, sad, angry, etc.), the different features of the face(eyes, nose, mouth, etc.) and the age. **KID** (Kernel Inception Distance) is an unbiased metric that computes the squared maximum mean discrepancy between feature representations of the real and generated images. **Generalisation** measures the models ability to represent unseen face shapes that it did not encounter during training. Computation of generalisation error is done by per-vertex Euclidean distance between samples in the test set **x** and the generators reconstruction $\mathbf{x}^*$ which is expressed $\mathbf{x}^* = \text{argmin}_z |\mathbf{x} - G(\mathbf{z})|$. The overall metric for the model is found by averaging over all the vertices and test samples. **Specificity** evaluates the validity of faces generated by the model. An arbitrarily large number of generated faces are synthesised and proximity to real faces in the test dataset is measured. The proximity is calculated using the Euclidean distance described in generalisation. This metric can provide evidence that faces generated by a model are comparatively more realistic than other results.

3.2.3 Qualitative. While we have a preference for quantitative metrics, there is value in qualitative approaches. There are typically less qualitative evaluation metrics than quantitative, however, such metrics are important as generated faces must be of high quality when reviewed by humans in order to overcome human sensitivity to faces - the so called uncanny valley. Qualitative metrics can additionally be important for detecting training performance and failure such as over fitting or mode collapse. Simple preference judgement and a nearest neighbor visual analyses will be conducted. **Preference Judgement** asks participants to evaluate generated images in terms of quality. This metric attempts to determine if results are acceptable to a human audience. **Nearest neighbors** analysis has been used in the literature to detect over-fitting in a training set by showing synthetic samples to their nearest neighbors in the training set. Typically nearest neighbors is determined using Euclidean distance which is sensitive to small perturbations in perception. Models that store training images can circumvent this training trivially but this can be elevated by choosing nearest neighbors based on perceptual measures and selecting many nearest neighbors.

Mode drop and collapse are issues that are identified through qualitative evaluation. There have been many proposed quantitative measure for mode collapse, however, in certain small data size circumstances a nearest neighbor-like investigation can be useful to truly detect mode collapse.

3.3 Implementation

The following python libraries will be used in the implementation of each of the models and the training and evaluation pipelines:

(1) NumPy [7]
(2) PyTorch and Torchvision [16]
(3) Pandas [15]
(4) open3D [18]
(5) Matplot lib [8]

The CoMA autoencoder is also open-source and publicly available so it will be used as a baseline in the development of the training and evaluation pipelines [17].

4 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

There are two sources of ethical issues; qualitative analysis and output ramifications. A qualitative analysis will not involve a user study, therefore an ethical clearance will not be required. As we are working with facial data there are possible ramifications of our findings, and biases that may be found in one or more of the resulting models. These ramifications and biases are particularly dangerous when considering race. Such topics can be avoided - particularly with regards to feature control - in order to mitigate risk.

4.1 Datasets and software

Significant ethical issues may be encountered as a result of the misuse of facial data, however, this data is protected and its use requires accepting agreements not to abuse the data in any way. Therefore, the misuse of this data would incur an ethical issue and a significant legal issue. As facial scans can be a highly personal type of data, all of the datasets require you to agree to some kind of user agreement or license or both. When using the dataset special care needs to be taken to make sure all the requirements stated in the user agreement or license are adhered to. Such as no commercial use for the FaceWarehouse dataset and CoMa dataset, and non-profit use for the BU-3DFE dataset. The machine learning library of choice will be PyTorch, which is an open-source library that is under the Berkeley Software Distribution License. This license does not impose any strict restrictions when using this library.

5 RELATED WORK

5.1 Progressive GANs

Progressive GANs (ProGANs) are about producing high resolution output through progressively training the GAN, by increasing the resolution of the output layer by layer as training progresses. This has been used in great effect in the first progressive GAN paper by Karras et al. [9] for 2D face generation. In [9] they managed to produce photorealistic faces of people that do not actually exist. This paper also proved that training a ProGAN can be significantly shorter than any other type of GAN as most of the training epochs are done at a lower resolution.

The application of ProGANs using a 3D facial dataset for 3D face generation is yet to be done. Although this does not mean that ProGANs have not been used for 3D applications. Eklund et al. [3] implemented a 3D ProGAN based off of the original ProGAN by Karras et al. In [3] their focus was on synthesizing brain volumes so this does not quite relate to face generation, but it proves that a 3D ProGAN is possible. Plus the implementation of this 3D ProGAN is publicly available. Which is a promising sign in terms of implementing a 3D ProGAN for face generation.

5.2 Conditional GANs

Conditional GANs are a fundamental extension of the GAN which allows for user control over the output. They rely on labeled data to achieve this, an issue that is addressed by controllable GANs [13]. The input data is contained in the conditional information vector. This vector is then concatenated onto the input noise of the generator and provided as additional input to one of the discriminator layers. This user control is fundamental to expanding the possible applications of GANs and the general usability of GANs.

Conditional GANs have been used for image generation applications, where the models were trained using a collection of labelled image data allowing for the same model to generate images from a wide variety of categories. The primary example is images of handwritten digits; to be able to generate novel samples is effectively meaningless unless the user can specify which digit they wish to generate an image of. However, the most relevant example is the application of conditional GANs for face image generation[4]. In this case the conditional information vector was concatenated to the generator's input noise and provided as input to the final layer of the discriminator. Specifically, the vector was used to condition the output to a specified race, age and emotion. The extension of face generation to 3D outputs is still and area of exploration with regards to conditional GANs.

However conditional GANs have seen use in general 3D model generation - to generate models of objects other than faces[12]. In these cases the conditioning was used to specify the object, for example, objects would be given a label such as "chair", "desk", "couch", etc. These models were all trained to produce voxel output. Therefore, while their output samples were correct, they had a very low resolution, which is sub-optimal for face generation.

5.3 Controllable GANs

The main drawback of conditional GANs is that they require detailed labeling of training data, controllable GANs seek to circumvent this restraint while still retaining feature control. The strategy for feature control is to tweak the input latent vector to intelligently locate the desired feature set within the latent space. Typically a pre-trained classifier is used to check for the presence of desired features within the synthetic samples in order to refine changes to the latent vector. In this way a controllable GAN can be trained to disentangle identity which is controlled by the noise vector and features controlled by the latent code vector with in the latent space. Typically results have been comparable to conditional GANs, however, there inherently is low control on what the model learns. There have been attempts to specify the features that the model learns in papers such as holoGAN [14], however, the approaches typically add additional complexity and processing [1].

5.4 Mesh GAN

Cheng et al. proposed MeshGAN[2], a kind of GAN which implements several new strategies for training on 3D model input and producing a fully three dimensional output - it is the first intrinsically 3D GAN architecture. Although not in the normative form of controllable, conditional or progressive, MeshGAN still trains feature control as well as employing up scaling and down scaling layers. MeshGAN incorporates many new strategies in the model such as a novel input tensor strategy, Cheb convolutions and skip connections.
MeshGAN proposes the use of 3DMMs, which use Principal Component Analysis, where a model is used as an input to the network and the network proposes manipulations to change to model. It has been proven that MeshGAN outperforms the CoMA autoencoder[17] on both qualitative and quantitative tests.

6 ANTICIPATED OUTCOMES

6.1 Research contributions

The outcomes of this project include a number of research contributions. Should the project be completed, it should provide insights on the datasets used such as possible biases, label quality or methodologies for dealing with mesh data. Furthermore it will provide feedback on the applicability of evaluation techniques on mesh data. Lastly, it can serve as a basis for further research into the use of GANs for 3D face generation.

6.2 Systems

The completion of this project will result in a collection of systems. These systems include those which support the experimental process surrounding GANs and those which are used for the generation of novel 3D faces

6.2.1 *Experimental Platform.* The first resulting system is the experimental platform which is used to load the facial data into the models and evaluate the output samples. This platform can be repurposed into subsequent projects which involve the development of GANs for 3D face generation. It can also serve as a base from which to construct more complex platforms for more advanced GANs, or GANs for purposes other than face generation. The entire platform may serve to benefit the GAN research community while the data loader on its own may serve to benefit any individual who wishes to produce a machine learning model which takes 3D models as input.

6.2.2 *Models.* Secondary to the experimental platform, the models, or more specifically their generators, are resulting systems. The generators can be used to generate novel 3D faces. They have applications in further research which requires a dataset of novel 3D faces and in many industries from cyber-security to video game design.

6.3 Impact

6.3.1 *Computer graphics.* The field of computer graphics would be greatly impacted by an advancement where numerous, feature controlled 3D human faces can be arbitrarily generated. The cost of creating non-player characters in video games as well as characters in video content would be greatly reduced. Additionally the field of virtual and augmented reality are in rapid development, trivial 3D face generation could advance this industry dramatically with pre-generated facial expressions and user controlled features.

6.3.2 *Commercial.* The most prominent area of commerce where 3D face generation would be advantageous would be virtual try on where a customer can create an avatar with their likeness in three dimensions and could place accessories on this virtual avatar from their home.

7 PROJECT PLAN

7.1 Risks

The various risks that we could come across in this project have been compiled into a risk matrix, which can be found in Appendix A. Here the probability, impact, mitigation, monitoring and management of each risk are described.

7.2 Time line

The timeline of this project is broken into four phases (see 7.5 Milestones) and a final phase which will be used to finalise results and write the report, each of which will be completed by the following dates:

(1) Phase 1: 6 July 2022
(2) Phase 2: 20 July 2022
(3) Phase 3: 3 August 2022
(4) Phase 4: 17 August 2022
(5) Project Complete: 31 August 2022

The detailed Gantt chart can be seen in Appendix B

7.3 Resources

It is well known that machine learning can be a very computationally expensive task, specifically the training process. This is a result of the large number of floating point operations associated with the matrix multiplications used for each iteration. Similarly, such operations are required by computer graphics applications. Therefore, Graphical Processing Units (GPUs) are designed to perform these matrix multiplications with a high level of performance and efficiency. Certain packages such as PyTorch allow for training to be computed by the GPU, however in this case, only GPUs which are designed by NVIDIA and contain their CUDA cores are supported. Therefore we will require not only a system with a powerful GPU, but it also needs to be an NVIDIA GPU. As an example it was found that when training a DCGAN on the MNIST dataset of handwritten digits, training on the GPU showed a speedup of roughly 30 times, with CPU training taking about 15 hours complete and GPU training taking about 30 minutes to complete. As such there are a number of machines at our disposal which can provide this kind of performance:

(1) Personal Computer (Desktop): GTX 1660
(2) Personal Computer (Laptop): RTX 3050 Ti
(3) Personal Computer (Laptop): GTX 1050 Ti
(4) Private Server: 2x GTX 1080
(5) UCT HPC

That being said, the computing resources required largely depends on the model architecture; the number of layers, the size and format of the input and output data, etc. Therefore it is difficult to determine precisely the resources required.

Additionally, the experimental platform is another source of computational requirements as the data formatting and metric calculations require a similarly large number of floating point operations. Again, the exact requirements are unknown.

7.4 Deliverables

Below is a list of the deliverables required by this project:

(1) Project proposal
(2) Dataset Selection
(3) Data pre-processing framework
(4) Data loader
(5) Evaluation frameworks
(6) Testing pipeline
(7) Training pipeline
(8) Simple models
(9) Model integration
(10) Software feasibility demonstration
(11) Advanced models
(12) Draft academic paper
(13) Academic paper
(14) System source code with documentation
(15) Project poster
(16) Project website

7.5 Milestones

The project milestones correlate to our four phases of development; the series of iterative cycles which will be used to achieve the final product - the experimental platform and the models. Each milestone is the completion of a phase:

(1) Phase 1: Complete loading and testing pipeline with CoMA autoencoder
(2) Phase 2: Complete simple specific GAN model integration and feasibility
(3) Phase 3: Complete specific GAN model for simple face generation
(4) Phase 4: Complete advanced research on face manipulation and generation

The full breakdown of each phase is as follows.

7.5.1 Phase 1. The completion of phase 1 will result in full testing and training pipelines. It will start with dataset selection, which will also involve obtaining and understanding the format of said dataset, and an evaluation plan which will determine which metrics we will use to evaluate the model output. Once the input and output data is fully understood, we will then form a data standard for the loader and evaluator. From there the data loader and evaluator can be completed in parallel. The resulting systems should be able to take in the dataset, load it into the CoMA autoencoder[17] and have the output evaluated. Thus completing the two aforementioned pipelines; where the training pipeline will use the data loader and the testing pipeline will use both the data loader and the evaluator. The outputs of this phase will be:

(1) Data pre-processor
(2) Data loader
(3) Evaluation framework
(4) Training framework
(5) Qualitative assessment portal

7.5.2 Phase 2. Using the training and testing pipelines from phase 1 we can then start development on our respective GANs. Phase 2 will involve the creation of simpler versions of the final models with the purpose of ensuring integration with the pipelines. These simple GANs will work with 3D data that conforms to the established standard, however the data does not necessarily need to be facial data. Once phase 2 is completed we will have a progress and success assessment to determine if the project needs to be scaled back, and if so, to what degree.The outputs of this phase will be:

(1) Simple progressive GAN with pipeline integration
(2) Simple controllable GAN with pipeline integration
(3) Simple conditional GAN with pipeline integration

7.5.3 Phase 3. Once integration has been confirmed, the complexity of the GANs can be increased in order to deal with 3D facial data from the chosen dataset. This will most likely involve up-scaling the input/output layers of each of the GANs. The completion of this phase should confirm the feasibility of 3D face generation using GANs, and lay the groundwork for further complexity.The outputs of this phase will be:

(1) Progressive GAN for face generation
(2) Conditional GAN for face generation
(3) Controllable GAN for face generation

7.5.4 Phase 4. As the final phase of development, phase 4 will involve implementing advanced features with respect to the type of GAN being used; progressive, controllable and conditional. For controllable and conditional GANs, this will involve the incorporation of feature and/or expression manipulation - due to the nature of these kinds of GANs and how they allow for user input as a determining factor in the model output. As progressive GANs are slightly different in their intended function when compared to controllable and conditional GANs - with a focus on up-scaling the output as opposed to user control - the focus here will be in output resolution, aiming for extreme degrees of quality and detail in the model output.The outputs of this phase will be:

(1) Progressive GAN extended to produce high fidelity output
(2) Conditional GAN with expression feature control
(3) Controllable GAN with expression feature control

7.6 Work allocation

(1) Liam: Pipeline, Controllable GAN
(2) Sebastian: Pipeline, ProGAN
(3) Luc: Pipeline, cGAN

REFERENCES

[1] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems* 29 (2016).

[2] Shiyang Cheng, Michael Bronstein, Yuxiang Zhou, Irene Kotsia, Maja Pantic, and Stefanos Zafeiriou. 2019. Meshgan: Non-linear 3d morphable models of faces. *arXiv preprint arXiv:1903.10384* (2019).

[3] Anders Eklund. 2019. Feeding the zombies: Synthesizing brain volumes using a 3D progressive growing GAN. *arXiv preprint arXiv:1912.05357* (2019).

[4] Jon Gauthier. 2014. Conditional generative adversarial nets for convolutional face generation. *Class project for Stanford CS231N: convolutional neural networks for visual recognition, Winter semester* 2014, 5 (2014), 2.

[5] Tom Geller. 2008. Overcoming the uncanny valley. *IEEE computer graphics and applications* 28, 4 (2008), 11–17.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[8] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[9] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

[10] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 4401–4410.

[11] Minhyeok Lee and Junhee Seok. 2019. Controllable Generative Adversarial Network. *IEEE Access* 7 (2019), 28158–28169. https://doi.org/10.1109/ACCESS.2019.2899108

[12] Haisheng Li, Yanping Zheng, Xiaoqun Wu, and Qiang Cai. 2019. 3D model generation and reconstruction using conditional generative adversarial network. *International Journal of Computational Intelligence Systems* 12, 2 (2019), 697.

[13] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. https://doi.org/10.48550/ARXIV.1411.1784

[14] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. 2019. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 7588–7597.

[15] The pandas development team. 2020. *pandas-dev/pandas: Pandas.* https://doi.org/10.5281/zenodo.3509134

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR* abs/1912.01703 (2019). arXiv:1912.01703 http://arxiv.org/abs/1912.01703

[17] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. 2018. Generating 3D Faces using Convolutional Mesh Autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV).*

[18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

A RISK MATRIX

Table 1: showing the risks associated to the research project

Risk	Probability	Impact	Mitigation	Monitoring	Management
Loss or corruption of source code	Low	High	Use of a code repository system, like GitHub. Keep back-ups of the source code on multiple devices.	Ensure all team members all making regular commits to the repository.	Clone the most recent back-up of the source code.
Inability to gain access to chosen dataset	Medium	High	Apply for a license to dataset early. Strictly follow application instructions when applying.	Frequently check email inbox for license updates.	Apply for access to the second and third choice datasets simultaneously.
Lack of communication with supervisor	Low	Medium	Ensure regular meetings with supervisor to maintain frequent communication.	During meetings check with supervisor that the project is on the right track and schedule next meeting.	Complete as much work possible that does not require supervision.
Insufficient hardware for training each GAN	Medium	High	Look at using UCT's HPC and Google Colab. Ensure code is fully optimized.	Test source code to ensure there are not unexpected delays during execution.	Look at options for outsourcing execution. Apply/sign-up for UCT's HPC and Google Colab.
Insufficient experience with chosen libraries	Low	Medium	Allocate enough time to the chosen library tutorials and documentation.	Ensure each member is able to complete certain tutorials.	Keep the use of a library as simple as possible.
Unable to implement 3D GAN architecture for 3D face generation	Medium	High	Learn how to implement GANs starting from the bottom-up to gain a good understanding of how to implement certain types of GANs.	Ensure each member is making progress in each of specific GANs.	Change from implementing a 3D GAN architecture to a more common 2D GAN architecture.
Power outages interrupts training times	High	Medium	Allocate longer training times in each of the phases. Implement an auto saving feature in the training framework to ensure not all work is lost during a power outage.	Ensure the auto save feature is working. Frequently check load shedding news outlets for any possible updates.	Check the timestamp on the latest auto saved model. Continue training from there.

B GANTT CHART

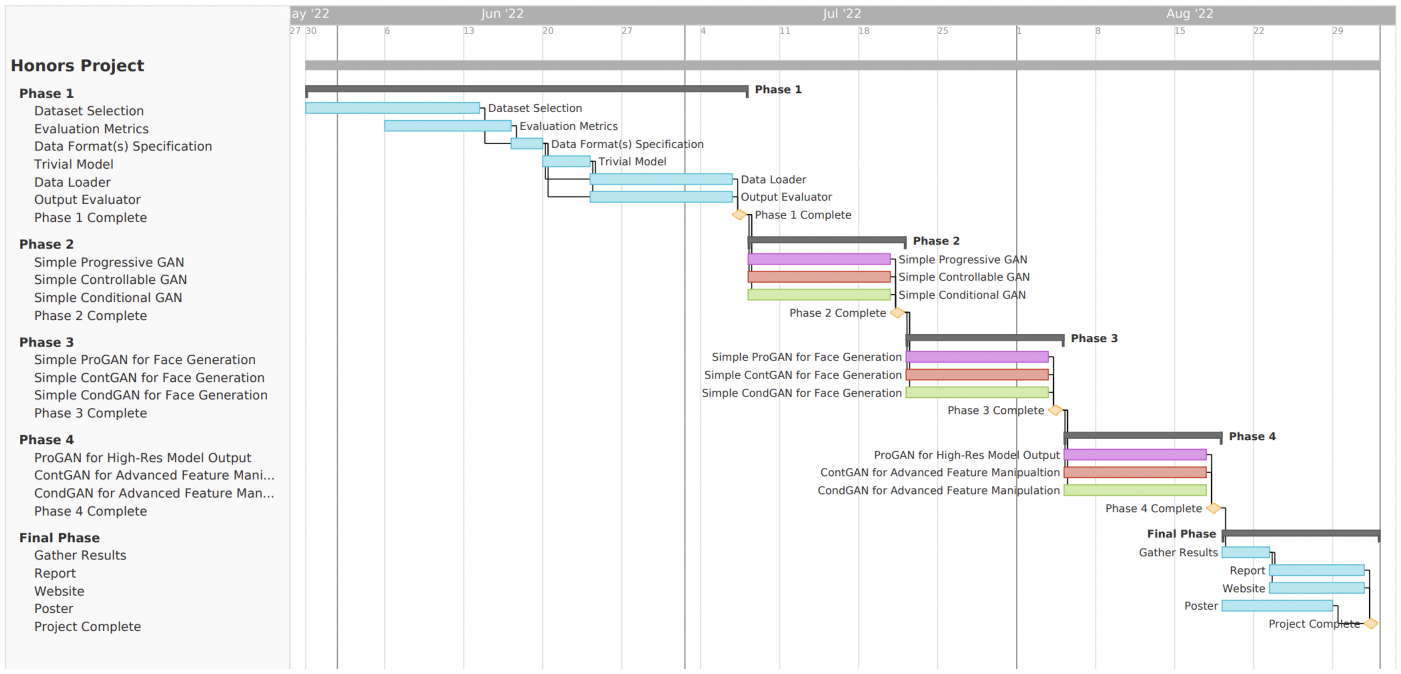

Figure 3: Gantt chart showing project timeline