Using generative adversarial networks to create graphical user Interfaces for video games

Christopher Acornley

Using Generative Adversarial Networks to Create Graphical User Interfaces for Video Games

Christopher Acornley
c.acornley@abertay.ac.uk
University of Abertay Dundee
Dundee, Scotland

ABSTRACT

Designing and creating a Graphical User Interface (GUI) is a difficult and slow process. It requires a number of professions to all contribute to its development and it can be heavily detrimental to a product if implemented poorly. This research aims to investigate a method of using Generative Adversarial Networks (GANs) to generate new and usable designs for GUIs. GANs are a relatively new architecture for adversarial learning and have been used to good effect in replicating instances of a real dataset. The primary aim is to develop a GAN that is capable of processing a collection of existing GUIs and learn how to replicate these to allow for creation of further designs. These GUI designs need to be formatted in a manner that enables modification, allowing for them to be used by a development team to enhance their production process. Completed work demonstrates numerous approaches at using GANs to create text files that contain the component elements of a GUI. Their results and the release of a similar research paper (GUIGAN) has led to a new approach focusing on more abstract data representation, with a quality control system for ensuring the output data is properly formatted. It is hypothesised that the approach will develop a model capable of creating new, editable GUI designs.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Human-centered computing → Systems and tools for interaction design**.

KEYWORDS

generative adversarial networks, graphical user interfaces, machine learning, datasets

ACM Reference Format:
Christopher Acornley. 2021. Using Generative Adversarial Networks to Create Graphical User Interfaces for Video Games. In *Proceedings of the 2021 International Conference on Multimodal Interaction (ICMI '21), October 18–22, 2021, Montréal, QC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3462244.3481273

1 INTRODUCTION

Most applications developed on a computer commonly makes use of Graphical User Interfaces (GUIs) to facilitate human-computer interaction [8]. Video games are a prime example as they are designed to be interactive, necessitating a consideration of user interface design, with a heavy majority being graphical in nature. The use of GUIs in games is incredibly varied, with unique approaches taken by almost each title available to allow the players to view information about the games current state and to perform actions within the games context.

The creation of these interfaces can be a very costly and time consuming process, being far more than just a visual layer to the user [14]. GUIs are the primary method that applications communicate to the user so it is necessary that it is clear and concise [21]. As such, designing an interface in a game requires the expertise and knowledge of a number of professions; programmers, artists, designers, quality assurance, etc. These interfaces are also subject to a number of other external influences such as the style of the game, guidelines by the company, or feedback from clients or publishers.

Due to the complexity of these systems and the large amount of development that must go into their production, methods that can streamline the process can be used to great effect. While assistance tools are widely available to allow designers and developers to retrieve templates to begin their work[1][2], none of these tools have the capacity to create the interfaces on their own. A model that could generate base designs that can be developed further could prove incredibly advantageous, not only in speeding up the process, but also giving smaller teams and companies access to 'good' designs without the need for specific user interface design expertise.

Within the field of Machine Learning, Generative Adversarial Networks (GANs) are a particularly unique architecture that is proficient at generating instances of unstructured patterned data; such as images[4]. This example of a GAN model can be reliably trained to mimic content of a training dataset, however the output of these models itself is not subject to any rules. Values in the output do not have to exactly match the training data for the model to consider the instance satisfactory nor does the data have to be presented in a specific order, such as individual values in predefined sequences. This allows GANs to be creative and produce new, unique output instances that are comparable to the original training data.

The intent of the study is to investigate the different methods and effectiveness of using GANs to produce GUI designs for use in the medium of video games. The research will be investigating how to encapsulate the components that constitute a GUI and train a model to replicate that data based on initial conditions, such as core elements required of the interface. The expected output of the model when properly trained will be a file format describing the structure of the interface, such as a list of the necessary components, or a file that can be directly loaded from the models output into visualisation software to test the integrity of the design.

The success of the model would allow for development of a tool where a designer or developer can input the specifics they require and the model will provide them with a repository of designs that can be developed further. The use of a machine learning model constitutes a great benefit should it properly learn how to replicate good design practices and heuristics [16] as it would further remove additional work from the production team.

2 RESEARCH QUESTION

Can a Generative Adversarial Network be used to generate usable Graphical User Interface Designs in Video Games?

The research question can be broken into three parts:

- Can a Generative Adversarial network be used to generate Graphical User Interface Designs?
- Would the design generated be usable?
- What is the best data representation that facilitates the model producing appealing and usable GUIs?

The initial and primary focus of the PhD is to establish that a GAN can reliably be used in this context. Machine learning algorithms have proven effective at learning how to operate with a large number of different sources of data, even proving to be more effective than initially thought possible [20]. It would be straightforward to assume that a GAN model should be able to work with creating interface designs, however this is not proven yet and therefore a focus of the study.

In terms of usability, the output of the model should ideally be in a form that allows for further development of the design. This will move the GAN away from processing the GUI like an image, despite the heavy graphical content of the data. An image would require additional steps for the design to be extracted for changes to be made, whereas a design based on metadata of a user interface, can be directly used to continue to build the design. The usability of the output can also be considered in terms of the user of the GUI. The designs provided by the model need to allow the user to access the individual elements; therefore the usability could be negatively affected if interactable elements are outside of the screen view, layered over one another, or cannot be selected due to their size and position. Training the model to output data that handles the limitations of being usable for both the designer and user needs to be considered.

The models training also requires that the output data is modifiable in some form. The model needs to produce an output that allows for further development of the design; this directly relates to the success of the research and the model itself. Designs produced that cannot be extended further limit practical applications of the research. This focuses the approach on generating metadata for a GUI as this format allows for further development once created. The exact structure of this metadata can be varied, with it being usable directly, or as meta-metadata with individual points representing entire UI elements. This output should be formatted in a way as to be loaded into a visualisation tool so it can be assessed and edited.

3 BACKGROUND

Generative Adversarial Networks *GANs* are a relatively new architecture of machine learning, proposed in 2014 [9]. The theory explained how two opposing networks could be used to train against each other using game theory. These two networks were labelled the Generator *G* and the Discriminator *D*. *G* would learn to replicate the data distribution within a pre-defined dataset, and *D* would attempt to determine if the sample it was processing was the output of *G* or not. The predictions of *D* are then used to train both models (See Figure 1). The study demonstrated the viability of the *GANs* model, which was trained on three datasets (MNIST, TFD & CIFAR-10). The results showed the GAN produced high log-likelihood figures when compared against other models; DBN[12], Stacked CAE[19], and Deep GSN[5].

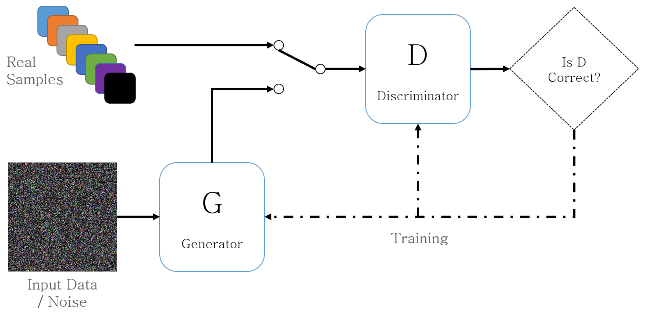

Figure 1: Basic structure of the GAN architecture

In its original form, both models are trained using the output of *D*, this output is being used as the label data that is traditional in most supervised model training. *G*'s loss is calculated by determining how many of the models output successfully fooled *D*, whereas *D*'s loss is based on how many out of both real and generated instances it correctly guessed. This form of adversarial training was stated to correspond with a minimax two-player game and follows the following formula:

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))] \tag{1}$$

The predictions of *D* are used to calculate the overall loss of the model, with $D(x)$ being the discriminators prediction of the real data instances, and $D(G(z))$ being the discriminators prediction of *G's* output based on input data z. The expected output of the real instance data E_x, all true, and the expected output of the *G's* input data E_z, all false, are compared against the predictions to calculate the final loss value. *D* attempts to maximize this formula as that would relate to its ability to accurately predict its input data. *G* attempts to minimize this, which to the model is equivalent to minimizing $log(1-D(G(z)))$ as this is the only part which relies on *G's* output. This half of the equation, when minimized, represents *D's* inability to correctly guess that *G's* output is fake.

The GAN itself can also be extended, with additional models added to the overall architecture to improve training and performance. Utilisation of these additional models can contribute towards more stabilised training, a problem inherent to GANs [18], or improve the quality of the models output [17].

4 RELATED WORK

A paper titled 'GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks'[22] focuses on the same problem as this study. It attempts to use GANs to create new GUI designs on an Android phone system, with positive results.

The selected method for generating GUIs is done by classifying a large dataset of images to create a repository of 'subtree' objects. Each subtree represents an aspect of the overall interface and can contain multiple UI widgets. The classification takes the overall GUI, breaks it down into these subtree objects using app metadata and adds them to a repository. The information in this repository can be embedded in a similar manner to text processing within neural networks, allowing each item to have a unique signature. Their generator is constructed as a Long Short-Term Memory(LSTM) model[13] and is trained to identify the pattern of subtrees that occur and attempts to replicate these patterns into new designs. The model breaks GUIs down into smaller building blocks and learns how to use these blocks to construct output instances.

Utilising Frechet Inception Distance (FID) and 1-Nearest Neighbor Accuracy (1-NNA) to evaluate the output of the model, they compared the output of a *GUIGAN* with two other models to determine performance; a modified version of a Wasserstein GAN[3] with an improved training method called **WGAN-GP**[10] and a web GUI designer **FaceOff**[23]. The results showed than the *GUIGAN* model outperformed both **WGAN-GP** and **FaceOff** in terms of FID and 1-NNA scores, proving that the approach was viable for GUI design.

While the research conducted by this paper is very similar to the PhD's focus, the approach has several limitations which can be addressed. Output from the model is currently limited to a vertically scrolling layout, such as a news feed App, due to the way the model sequences the subtrees. The model is also incapable of instantiating individual UI widgets within a design, such as a single button or image, as it can only create GUIs from the repository of building blocks it has access to. This rigidity prevents it from creating personalised GUI designs, rather, it can only create different sequences of existing designs.

5 RESEARCH PLAN

5.1 Generative Adversarial Network Design

The primary goal of the project is to develop a suitable GAN model with which to test and authenticate the research question. As this is a novel approach, there are no existing models that will handle the requirements of the project. This necessitates experimentation into several different approaches to determine a viable solution to the problem. The structure of GANs enables the internal *Generator* and *Discriminator* models to have any format, which gives this approach a great deal of flexibility when processing, managing and handling the input and output data.

Consideration is also required into how the data for the model, both the input and output for the *Generator* are to be represented. The data has to allow for the production of editable metadata of a GUI, either by the model directly creating the instance data itself, or by creating more abstract information that can be used to create this metadata. The structure of such a system could match the approach presented by GUIGAN[22] or formulate a new method to compensate for the limitations of that approach.

5.2 Dataset

As the structure and data representation of the GAN requires investigation, the format of the dataset also requires further work. There are collections of GUI data and images, such as the RICO dataset [6], which can provide a basis for the research, but depending on how the model requires the structure of its input and output may necessitate heavy manual processing to properly prepare the data.

5.3 Evaluation

The models effectiveness is to be evaluated in two parts. Firstly, the accuracy of the model needs to be determined, which is to be done by comparing the *Generators* output and items from the training dataset. The method of this comparison will be using FID[11] as it is widely used for measuring the quality of a GANs image output[22]. By supplying the *Generators* output to a visualisation program, we can create a representation of the generated GUIs allowing for this image comparison method. Secondly, the usability of the GUIs needs to be tested as the designs need to have created content in such a manner that it is all accessible. All interactive elements that the GUI design describes have to be positioned so they can be selected by a hypothetical end-user. This is to limit the possibility of the model producing designs that have overlapping elements, elements outside the confines of the users view and other obstructions that may reduce the ability of the GUI to function. The implementation for this is to be achieved through an automated test system which attempts to interact with each unique element.

6 COMPLETED WORK

Work conducted has included four experiments with different approaches to building and training the relevant GAN model.

(1) Direct-image synthesis. This aimed to create images of a GUI for initial proof of concept.
(2) Text generation through Deep Feed-Forward networks. Using a deep feed-forward network, the model was aiming to generate a text file that could be read by a JSON parser. The output text file would represent the direct text of the design.
(3) Text-to-image Convolutional Network. The model converted text files into images and trained a convolutional network to replicate the images. These can be converted back into text.
(4) Gated Recurrent Unit (GRU)[7] text generation using GAN as quality control. Text generation can be done with GRU's. Attempt to combine GRUs into the workflow of the GAN architecture.

The first set of experiments replicated images of the mobile video game Candy Crush Saga[15]. A standard GAN was tasked with recreating sections of the game based on screenshots of the levels. Focuses were on the top left hand corner, which contained a concentrated cluster of UI elements, a larger image of the screen, and a coloured image of the screen. These experiments aided with identifying and proving the necessity of creating the metadata for designs, rather than simply replicating images.

Further experiments focused on the generation of these metadata files, which were generally stored as Notepad text files. The images

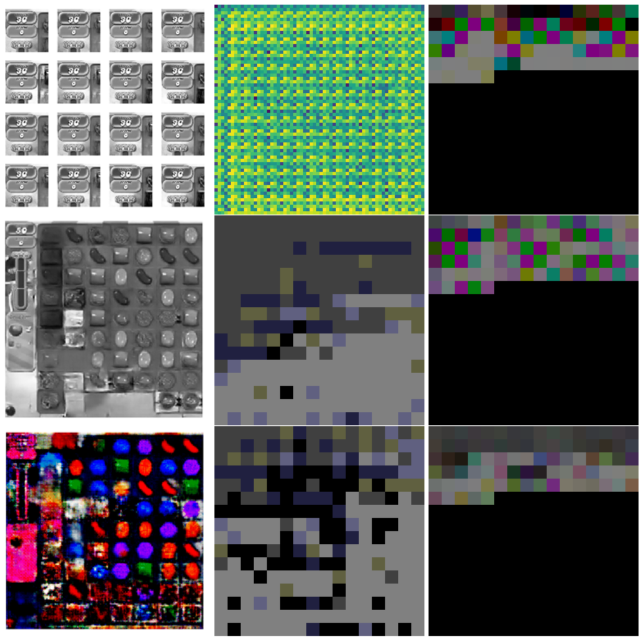
Figure 2: Outputs of experiments. Column 1: Direct Images Synthesis. Column 2: Text-To-Image Conversion. Column 3: GRU output, converted to image for visual representation.

shown in Figure 2 have the text files converted to images so the distribution of text data can be visualised. Each character in the text is converted to an ASCII value and embedded into the colour channel of an image, either a 3-colour channel image or a single colour channel grayscale image. This visual representation of the text allowed for the GAN to handle the training data like an image, a function that GANs are well suited for. Whilst initially promising, the primary flaw in this method was that while the output could achieve a very close similarity, converting the image back to text revealed that minor changes in colour distorted the correct sequence of text, rendering the output completely unreadable.

7 PLANNED WORK

Utilising the positive results of the GUIGAN paper and the knowledge gained from the described experiments, a new approach was devised and is to be conducted as the next stage of the research. A mix of structured requirement input data, representing the desired elements in the GUI, along with random noise will be input to the Generator, and a two stage Discriminator system along with a Quality Control module will be used.

Input to the model will consist of two series of data; a specification on the core elements that constitute the GUI, and a vector of random noise as seen in Figure 3. This aim is to train the GAN to match specific inputs that correspond to specific elements, with the random noise allowing for non-deterministic output. Input to the model is passed to the Generator which instantiates a vector representing the components of the overall design. The elements of the vector are directly related to the metadata of the design, such as an index to a UI widget, its position on the screen or its dimensions. This vector has to pass through a quality control module to ensure its validity as a design. The data in the vector has to be converted into a readable file that can be loaded by an external visualisation tool. This allows for two steps in the Discrimination process; an evaluation of the text itself to determine the probability of it being suitable to the Discriminator model, and by loading the created file into a visualiser, an image of the intended design can be used as input to a second Discriminator. This allows the GAN to evaluate the design on the suitability of the text and the structure it creates in a visual medium. Any error that occurs during the file generation or image generation, means the GAN can skip the Discrimination step entirely as the generated data would be unsuitable. The output of the two Discriminators are to be combined into a probability matrix that will determine if Generator was successful or not.

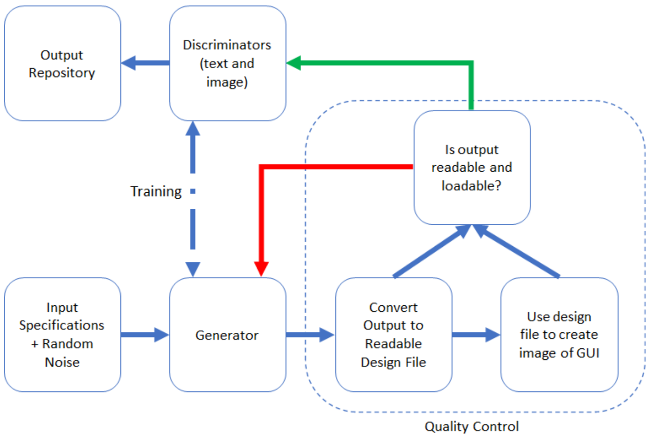
Output Repository
Discriminators (text and image)
Training
Is output readable and loadable?
Input Specifications + Random Noise
Generator
Convert Output to Readable Design File
Use design file to create image of GUI
Quality Control

Figure 3: Proposed structure of the GAN architecture

ACKNOWLEDGMENTS

I would like to thank my Supervisors, Dr. David King, Dr. Salma ElSayed, and Professor Ruth Falconer for all their support and guidance as I progress through my PhD Studies.

REFERENCES

[1] 2009. Dribbble: Discover the world's top designers & creatives. https://dribbble.com/
[2] 2021. UI Templates. https://themeforest.net/category/ui-templates
[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. arXiv:1701.07875 [stat.ML]
[4] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. 2017. CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training. *CoRR* abs/1703.10155 (2017). arXiv:1703.10155 http://arxiv.org/abs/1703.10155
[5] Yoshua Bengio, Eric Thibodeau-Laufer, and Jason Yosinski. 2013. Deep Generative Stochastic Networks Trainable by Backprop. *CoRR* abs/1306.1091 (2013). arXiv:1306.1091 http://arxiv.org/abs/1306.1091
[6] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 845–854. https://doi.org/10.1145/3126594.3126651
[7] Rahul Dey and Fathi M. Salem. 2017. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. *CoRR* abs/1701.05923 (2017). arXiv:1701.05923 http://arxiv.org/abs/1701.05923
[8] H. El-Bakry and N. Mastorakis. 2009. User interface for internet applications.

[9] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

[10] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved Training of Wasserstein GANs. *CoRR* abs/1704.00028 (2017). arXiv:1704.00028 http://arxiv.org/abs/1704.00028

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium. *CoRR* abs/1706.08500 (2017). arXiv:1706.08500 http://arxiv.org/abs/1706.08500

[12] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comput.* 18, 7 (July 2006), 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (15 Nov 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[14] Jason Hong. 2011. Matters of Design. *Commun. ACM* 54 (02 2011), 10–11. https://doi.org/10.1145/1897816.1897820

[15] King. April 2012. Candy Crush Saga. https://play.google.com/store/apps/details?id=com.king.candycrushsaga&hl=en_GB&gl=UK 1.204.0.2.

[16] Jakob Nielsen. 2020. 10 Usability Heuristics for User Interface Design. https://www.nngroup.com/articles/ten-usability-heuristics/ [Online; posted 24-April-1994: updated 15-November-2020].

[17] Augustus Odena. 2016. Semi-Supervised Learning with Generative Adversarial Networks. *arXiv e-prints*, Article arXiv:1606.01583 (June 2016), arXiv:1606.01583 pages. arXiv:1606.01583 [stat.ML]

[18] Teodora Pandeva and Matthias Schubert. 2019. MMGAN: Generative Adversarial Networks for Multi-Modal Distributions. *arXiv e-prints*, Article arXiv:1911.06663 (Nov. 2019), arXiv:1911.06663 pages. arXiv:1911.06663 [cs.LG]

[19] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive Auto-Encoders: Explicit Invariance during Feature Extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* (Bellevue, Washington, USA) *(ICML'11)*. Omnipress, Madison, WI, USA, 833–840.

[20] Terrence J. Sejnowski. 2020. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences* 117, 48 (2020), 30033–30038. https://doi.org/10.1073/pnas.1907373117 arXiv:https://www.pnas.org/content/117/48/30033.full.pdf

[21] Jessica Thornsby. 2016. *Android UI Design: Plan, design, and build engaging user interfaces for your Android applications*. Packt Publishing, Birmingham, UK.

[22] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. 2021. GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks. *CoRR* abs/2101.09978 (2021). arXiv:2101.09978 https://arxiv.org/abs/2101.09978

[23] Shuyu Zheng, Ziniu Hu, and Yun Ma. 2019. FaceOff: Assisting the Manifestation Design of Web Graphical User Interface. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM '19)*. Association for Computing Machinery, New York, NY, USA, 774–777. https://doi.org/10.1145/3289600.3290610