**Architexa project portfolio***Contributors*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Emmanuel Adegoke | Fahad Alsharaf | Matt Paver | Meshari Alshammari | Stephen Bromley | Wilf Morlidge |

# People and roles

## The teams

The team was split into 3 sub-teams for most of the project. With Matt taking the lead on the custom dataset, Fahad and Meshari composing the website design team and Wilf and Stephen on the neural network team. Emmanuel initially joined the neural network floated between teams giving support to the team with the most work to do at that time.

## Major Tasks and achievements

|  |  |  |
| --- | --- | --- |
| Task | Major Role | Minor Role |
| Generator Design | Emmanuel | Wilf, Stephen |
| Discriminator Design | Wilf | Emmanuel, Stephen |
| Training Implementation | Wilf, Stephen |  |
| Hyper-Tuning Implementation | Stephen | Wilf |
| Designing The Website | Fahad, Meshari | Matt |
| API Implementation | Fahad, Meshari |  |
| Gathering The Dataset | Matt | Emmanuel |
| Dataset CASE Tool Design | Matt |  |

Fahad and Meshari came into this project with moderate programming skills but little experience in web development. Through an immense amount of research and help from Matt who is a more experienced web developer they have made a fully functional website complete with a login feature and an API call.  
  
Wilf and Emmanuel started with some knowledge of neural networks but not much beyond that taught in COMP219. They led research and development into both architectures used with Wilf then going on to making the first training script which provided a great starting point for the later development of the hyper-tuning. Emmanuel showed great adaptability slotting into whichever team he was needed.

Matt oversaw collecting the dataset and creating the tools to augment it. This proved to be a quite a challenge due to having to scrape the internet for images. The solution was to use a combination of python and .yaml files which is an area of coding none of us had any experience with. The result was having a completely custom dataset of over 600 images with the tools to freely add classes if needed.

Stephen came into this project with some expertise in this area having previously worked on a speech-to-text model. He wrote most of the hyper tuning code and was able to utilise his past experience to interpret the model’s losses and outputs to help steer the training and tuning process. He was able to identify that the current training process from the CGAN paper was too unstable and cause artifacts and overfitting so spearheaded the switch to the more robust WGAN-GP framework.

# Application overview:

## Application domain:

Architexa is what is commonly known as an AI image generator (similar to products such as Craiyon, Midjourney or Imagen). This means that it takes textual prompts from a user and generates images corresponding to those prompts.

The purpose of such systems is to allow those without artistic or technical drawing skills to create images faster, and with a higher quality, than would be possible for a human.

Architexa differs from other applications in this domain, both in that our API is targeted specifically at architects and we hope our nascent model may eventually be able to generate detailed architectural diagrams in relation to highly specific prompts. Also, the project contains case tools designed to address the challenges involved with creating such a model in the first place.

Therefore, Architexa addresses both the noted lack of CGAN models for technical drawings, the difficulty and non-deterministic nature of hyperparameter optimisation, and the huge time taken to construct proprietary datasets (especially multi-label datasets) via conventional image dredging, farming, and hunting.

# Types of users

## Non-technical CGAN users

This user type refers to members of the general public with no technical background who may wish to use the current version of the CGAN model to generate images of buildings as a labor-saving device when creating artwork, or for entertainment purposes. (which are the current main use cases for non-technical users of CGAN models)

Users of this type may access the model by navigating to the main page of our website, submitting a simple prompt corresponding to one of our supported classes, and waiting roughly 75 seconds until being presented with four options of generated images corresponding to the submitted prompt.

## CASE tool users

This user type refers to other CGAN developers who wish to utilize either our Distributed optimization CASE tool to improve the performance of their models, or our database extraction CASE tool to reduce the time taken to create databases for their models.

Users of this type may access our case tool by navigating to our project's GitHub repository and installing the appropriate case tool by following the instructions found in the project README files.

## Database users #ASK WILF TO CLARIFY

This user type refers again to CGAN developers, but specifically to those who may be working on very similar products and may wish to bypass the database development process entirely by making use of our dataset. This is not unlikely given that no other multi-label datasets of external shots of buildings are readily found to exist.

Users of this type may access the database by once again navigating to our GitHub repository, and accessing the subdirectory of the database extraction CASE directory which contains our dataset (before downloading it).

## Technical CGAN users: (future development)

This user type refers to our initially targeted user type, and consists of architects, and other appropriate professionals, who may (once the model’s dataset is expanded and fuzzy logic processing is added) wish to use our model to generate technically accurate designs for buildings based upon detailed textual prompt input.

Users of this type would theoretically need to create a 2 or 3 paragraph English language description of the building they wish to design (remembering that this is still much less work than creating a blueprint) then feed it into our website in a similar manner to our non-technical users, before waiting several minutes to be greeted by an appropriate mockup design for their building. This unfortunately will need further refinement, and this is explained later in this document.

# Brief description of major components

## Neural networks

The core of the system consists of two neural networks, A generator which upsamples noise into an image using stabilized transpositional convolution, and a discriminator (sometimes referred to as a critic)

which down-samples generated images into feature maps to give each image a score based on how “real” it looks.

These work in tandem to form what is called an adversarial network, whereby the generator attempts to fool the discriminator into thinking the images it produces are real, and the discriminator tries to overcome the efforts of the generator. This struggle causes them both to learn to perform better, eventually improving the efficacy of the generator so that its eventual output looks good.

## Database and Database CASE tool:

To train the neural networks, it was necessary to produce a proprietary dataset of multi-labelled images of buildings (meaning that the images corresponded to multiple categories at once) this was done through a combination of conventional dredging (where appropriate images are extracted from the internet) and model autophagy (where autonomous tools are used to bulk out a dataset using data generated by an existing model).

In order to achieve this, we generated our own autophagy CASE tool to extract images from the mid journey model and were able to create a 6000-image dataset.

## Training function and optimisation CASE tool:

To run, our neural networks needed to be bound to a training function, which propagates 128 image sized noise matrices through the generator, then separately both the output of the generator and 128 real images from the dataset through the discriminator, before backpropagating the discriminator, and using the loss of the discriminator to backpropagation through the generator. (then does this again and again until every image in the dataset has passed though the discriminator 100 times)

However, in order to address the problem of hyperparameter optimisation non-determinism (meaning that the values of certain seemingly arbitrary values within the network have a huge effect on it and no algorithm exists to effectively pick the right ones) we also created an optimisation case tool which uses distributed computing to train the model multiple times simultaneously with different hyperparameter values, then use the results to estimate likely better hyperparameters, before training again, and so on until hyperparameter values converge.

## Website:

To make our model more accessible to non-technical users, and to fulfill the need for our project to have a GUI. We generated a simple website (hosted on the university servers) using hand coded HTML and CSS, which is bound together using XML and can use a JavaScript to send user generated textual prompts to a trained version of the model stored in a separate location.

## Extra features:

Originally, we assumed that the main components of the system would be the neural networks and their corresponding training loop. However, the difficulty in constructing an AI image generator turned out to lie less in the model construction and more in the acquisition of an appropriate dataset, and the choice of appropriate hyperparameters.

Therefore, we amended, and extended, our original specification to include a proprietary multi label dataset (as no appropriate dataset existed) and a system to autonomously optimize the models hyperparameters. Both of which deliverables were completed.

# Evaluation:

The original system objectives were as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Aim 1: | Underlying GAN can generate random, human-recognisable images without prompts. | Objective 1: Images thus generated are correctly classified by pre-existing classifiers as buildings 70% of the time.  Objective 2: The model can produce images in less than 15 seconds. | (primary aim) |
| Aim 2: | GAN is successfully updated into a CGAN and can generate recognisable images of specific building types in response to single-word prompts. | Objective 1: images thus generated are correctly classified by pre-existing classifiers as their building type 70% of the time.  Objective 2: The model can produce images in less than 20 seconds. | (primary aim) |
| Aim 3: | CGAN can be called through a web-based API. | Objective 1: The website can be accessed via Microsoft Edge and Google Chrome.  Objective 2: a remotely stored version of the model can be tasked through the website. | (secondary aim) |
| Aim 4: | CGAN has optimised hyperparameters and dataset. | Objective 1: images thus generated are correctly classified by pre-existing classifiers as their building type 90% of the time.  Objective 2: The model is re-trained in less than 8 minutes. | (secondary aim) |
| Aim 5: | CGAN gives four options in response to each prompt. | Objective 1: all four images are correctly classified.  Objective 2: human users report that images are diverse. | (secondary aim) |
| Aim 6: | CGAN can respond to multi-word prompts | Objective 1: images may be generated in response to 3-word prompts.  Objective 2: images thus generated are correctly labeled by existing classifiers 70% of the time. | (tertiary aim) |
| Aim 7: | database of login information. | Objective 1: the user may create accounts through the website.  Objective 2: users may log back into their accounts using stored login credentials. | (tertiary aim) |

And over the course of the project, we also added an aim to produce a proprietary dataset.

# Proof of met requirements:

We have absolutely completed the first aim since, as you will see from the testing documentation, the system can produce recognisable images.

(image screenshot)

We have also completed the second aim as the models are capable of being conditioned on labels and producing recognisable and distinct results.

(screenshot of images with different labels)

We have completed the fourth aim though the use of the optimisation case tool, and this can be seen though the difference in quality between images produced by a naively trained version of the model, and an optimized one:

(screenshots of images from different stages of optimisation)

The fifth aim turned out to be trivial, since it only involves putting a for loop in the code to generate four images every time the generation subroutine is called, however we have completed it.

(screenshot of new code)

The sixth aim is functionality completed, even though it was actually the dataset that had to be updated in order to make this possible, and technically the aim should read “model can respond to multi-label prompts”

(screenshots of images from the overlapping multi label classes (wood-house,wood-building,brick-building)

Our new aim of creating a proprietary dataset is also completed, as we have 8,000 images of our 8 classes stored in our GitHub repository.

(screenshot of subdirectory, and a subset of image files in the subdirectory)

System weaknesses:

One major weakness is that the system does not produce images as quickly as we had hoped, in fact it takes 60 seconds (instead of 20) to generate a single image, and up to 75 seconds via the website.

Another is that we were unable to implement a database of login information due to time constraints, however this aim was largely included to be completed in the event that the rest of the project was completed too quickly.

A third weakness is that the model may only respond to prompts with two overlapping labels, however this is due to the time and computational power required to generate images corresponding to the large number of well-defined classes necessary for triple overlap.

A fourth weakness is the fact that the model only accepts integers corresponding to embedded combinations of labels as input (instead of text), and we would want to later address this using a fuzzy logic analyzer as discussed in the future developments section.

And finally, there is the fact that the model only operates on a small number of classes, which we would again fix by expanding the dataset in the future.

Teamwork evaluation:

Future Developments:

Immediate fixes:

(essentially fixes for system weaknesses or unmet goals)

Long term fixes:

(explain how we would turn our proof-of-concept AI into a commercial product)

Professional Issues:

Bibliography: