COMP208 - Group Software Project - “Architexa”

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

Proposal summary:

This project, aimed at revolutionising architectural visualisation, proposes the development of a sophisticated AI-powered web application that transforms simple sketches into detailed architectural projections. It leverages the capabilities of conditional Generative Adversarial Networks (GANs), which aim to empower architects and designers with a versatile platform capable of adding depth, structural integrity, and material textures to basic drawings or generating detailed designs from straightforward prompts. Users can dictate specific architectural styles and material choices, making the design process more intuitive and efficient.

What We Will Do: Our project will craft an application utilising CGAN technology to transform basic sketches or textual prompts into detailed architectural renderings. These renderings will showcase enhanced structural soundness, feature realistic textures, and incorporate specific architectural elements tailored to user-defined styles.

How We Will Do It: The application will interpret sketches or textual prompts to produce intricate architectural visuals. It will include a user-friendly interface for uploading sketches or entering prompts, enabling users to specify desired features or styles effortlessly, thereby providing immediate visual feedback.

What It Will Look Like: ‘Architexa’ will provide a clean and efficient interface where users can upload sketches or input prompts, which the AI will then process to generate rich, detailed architectural images suitable for professional application or further refinement.

Experimental Process: The development will involve training the CGAN models with a vast collection of architectural data, fine-tuning the algorithm to accommodate various architectural elements and styles, and conducting iterative user testing to enhance functionality and user experience.

Project Description:

"Architexa" is envisioned as a revolutionary web application set to transform the landscape of architectural design visualisation. At its heart, the project utilises the advanced capabilities of CGANs to interpret simple sketches or textual prompts, converting them into detailed, rendered images that accurately depict structural integrity, material textures, and architectural styles. This application is set to be a vital tool for architects and designers. It offers a simple way to create and explore different architectural designs quickly. The unique feature of "Architexa" is its ability to take essential inputs and enhance them into detailed, visually appealing images that adhere to the specific requirements of structural integrity, material textures, and architectural styles. This adaptability ensures a highly customisable design process, catering to the diverse needs of its users. Please refer to the proposal summary to comprehensively explore "Architexa's" vision, objectives, and technological foundation.

Aims and Objectives:

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 labelled 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)

Key Literature & Background Reading:

"Architexa" is founded by the significant advancements in AI within creative and design fields, particularly the foundational work on Generative Adversarial Networks (GANs) by Goodfellow et al. (2014). This ground-breaking research introduced a new paradigm for generating complex images from more straightforward inputs, laying the groundwork for numerous applications in image generation, including architectural visualisation.

Building on the foundational GAN framework, the research community has delved into various adaptations, among which Conditional Generative Adversarial Networks (CGANs) stand out. CGANs introduce a conditional aspect to the generative process, allowing for the generation of images based on specific conditions or prompts, making them particularly suited for tasks that require a high degree of specificity, such as architectural design (Mirza & Osindero, 2014).

Mirza and Osindero's research on Conditional Generative Adversarial Networks (CGANs) demonstrates the potential of CGANs in generating images based on specific conditions or prompts, such as generating MNIST digits conditioned on class labels. Their work showcases the structure and training process of CGANs, highlighting their ability to produce diverse outputs based on given labels, which could be particularly relevant for tasks requiring specificity, like architectural design. "Architexa" aligns with this, leveraging the broad capabilities of CGANs and exploring their application in generating detailed architectural designs from simple inputs.

Development & Implementation Summary:

Planned design:

At its core, the program will consist of two neural networks: a generator (responsible for matching the distribution of features in an input image to a learned distribution within training data) and a discriminator (responsible for distinguishing training images from generated images and thereby producing a loss function used to update the generator). (Python, R, n.d)

Specifically, we must then use a CGAN(conditional Generative Adversarial Network) wherein the label for input images is concatenated to the image input for the discriminator and generator(yet the complete output for the generator is again linked to the label before it is input to the discriminator), this requires us to have higher dimensional input (and several extra partially connected layers) for both of our networks. (however, with the concatenation of labels, the discriminator's ability to identify a given image as false will be inversely proportional to the generator's ability to match its input image to the distribution of training images, where a particular label is known, which allows us to train the generator to produce output appropriate to each label). *(Brownlee, J,2020)*, hence allowing for prompts. The trained model will be quite a large file so the web API will be a thin client sending queries to a copy of the model on my MSI Trident tower.

Development tools:

The GAN will be implemented in procedural Python, the standard language for AI applications and includes many helpful libraries such as Pytorch.
The web API will be designed using a visual website builder such as Wickes, as this will allow for better visual design, and the website will be technically uncomplicated apart from the embedded Python program. Project documents shall be stored in Google Drive, as it is freely available and allows for collaborative editing. Project meetings will take place on WhatsApp, and team decisions will be logged in Canvas. The implementation stage shall be managed in Zobo to expedite debugging.

High-level project breakdown:

After completing this document, we will split into three teams of two: one working on CGAN design, one researching hyperparameter tuning and dataset refinement, and one building the web API. After this, we will spend some time pair programming the first iteration of the CGAN, and then we will spend time tuning in line with team 2's research. (Testing throughout) We will then split into two teams of 3 (with one member from each team of 2 on each team of 3) and create competing designs for CGANs capable of meeting aims 5 and 6. (Testing throughout)We will then implement and refine the more effective of the competing models. If all this is completed before week 9, we will work on aim seven and improve the training speed and accuracy of the model through further research. We will then spend the week before the presentation in two teams, one drawing up slides and the other developing a test suite which shows off the model's capabilities. Then, we will begin working on the portfolio and design document. Note that our design work will likely take longer than our implementation, as the mathematics underlying artificial intelligence is very complicated. Still, implementation may only require one or two hundred lines of code.

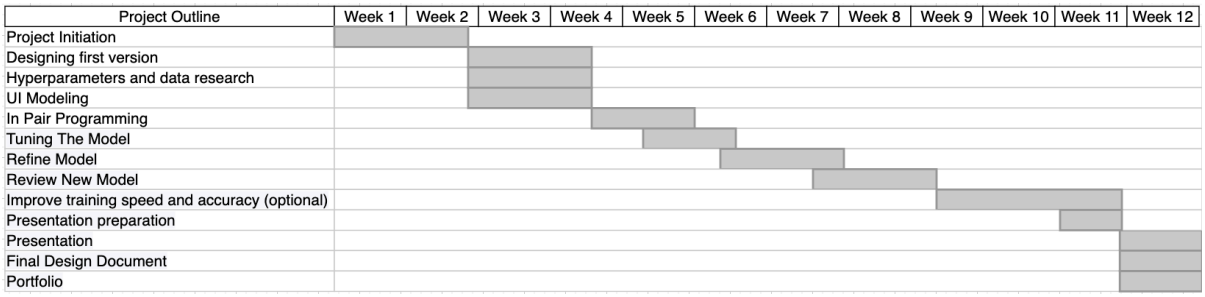


Figure 1: Gannt chart visualising the above

Data Sources:

No current dataset of purely labelled images fits our purpose. However, a large dataset called ImageNet includes some relevant image types, such as different forms of bridges. We can apply to use the dataset due to being associated with the university. Due to the need for publicly available datasets, we should be prepared to create our own by scraping images from the web. It will be labour-intensive to label all of these, so we propose to use online tools to speed up this process, such as Label Studio *(Labelstud.io, n.d)*

Testing & Evaluation:

Functional testing	Verify that Architexa accurately interprets uploaded sketches and transforms them into detailed architectural projections. Ensure that the application effectively enhances structural integrity, applies realistic textures, and incorporates style-specific architectural elements as per user prompts.

Performance testing	Evaluate the speed and efficiency of Architexa in processing sketches and generating architectural images. Assess the system's performance under various load conditions using methods including but not limited to stress testing to ensure it can handle simultaneous user requests without degradation.
Accuracy testing	Measure the accuracy of architectural projections generated by Architexa compared to original sketches and industry standards. Validate the fidelity of applied textures and style-specific architectural elements against user expectations and design standards.
Usability testing	Conduct usability testing to evaluate the intuitiveness and ease of use of Architexa's interface. Gather feedback from architects and designers to identify areas for improvement in workflow, navigation, and overall user experience.
Compatibility testing	Ensure that Architexa functions properly across web browsers, operating systems, and devices. Verify compatibility with various file formats for uploading sketches and exporting architectural projections.
Scalability testing	Evaluate Architexa's scalability to accommodate a growing user base and increased demand for processing architectural sketches. Test the system's ability to scale resources dynamically to meet workload demands while maintaining optimal performance.
Load testing	Simulate heavy user traffic on Architexa's website to assess its performance under peak load conditions. Measure response times and server resource utilisation to ensure Architexa can handle high concurrent requests effectively**.**
Error handling testing	Intentionally introduces errors, such as invalid prompts or network disruptions, to evaluate how Architexa handles and communicates errors to users. Ensure error messages are clear and informative and help users troubleshoot issues effectively.
Image quality testing	Evaluate the visual quality and resolution of images generated by Architexa to ensure they meet professional standards. Assess factors like sharpness, clarity, and colour accuracy across architectural styles and textures.
Prompt variability testing	Test Architexa's ability to generate diverse and visually compelling images in response to various prompts, including variations in architectural styles, materials, and spatial arrangements. Assess the robustness of the CGAN model in capturing and interpreting nuanced prompts effectively.
Cross-Device Compatibility Testing:	Test Architexa's website and API functionality across various devices, including desktop computers, laptops, tablets, and smartphones. Ensure consistent performance and user experience across different screen sizes and input methods.
Scalability and Resource Management:	Assess Architexa's scalability by gradually increasing the number of concurrent users and monitoring system performance. Implement efficient resource management strategies to optimise server resources and minimise operational costs.
User Feedback and Iteration:	Establish mechanisms for collecting user feedback within Architexa's interface, such as surveys, feedback forms or simple FAQs. Use feedback to identify areas for improvement and iterate on features, user interface design, and functionality.
Documentation and Support Testing:	Review the comprehensiveness and clarity of documentation provided to users, including user guides, FAQs, and tutorials. Evaluate the responsiveness and effectiveness of customer support channels, such as email support or online forums, in addressing user inquiries and issues.

Ethical Considerations:

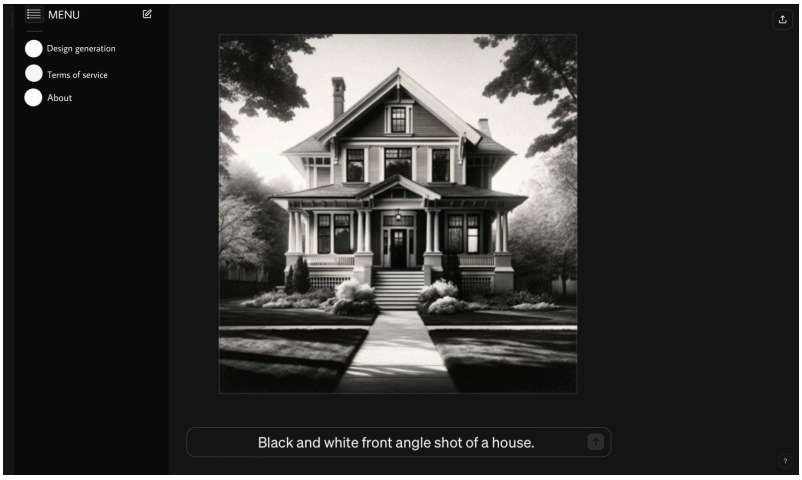
Data Integrity and Sourcing	"Architexa" commits to the highest data integrity standards, ensuring all utilised or generated data is authentic and derived from legitimate sources. Any data analysis conducted will be the original work of the project team, strictly adhering to academic and research ethics to prevent misconduct, including plagiarism.
Legal and Ethical Data Use	The project will legally obtain and ethically use data, particularly from external sources, ensuring compliance with copyright laws and avoiding unauthorised data acquisition methods, such as web scraping. Public domain data will be utilised responsibly, ensuring it is anonymised and does not infringe on privacy or intellectual property rights, with all necessary permissions verified.
Participant Engagement and Consent	Human participants involved in testing and evaluating "Architexa" will do so voluntarily, with informed consent obtained to ensure they are fully aware of the nature and scope of their participation. The project will respect participants' rights to withdraw at any time, ensuring their data is handled respectfully and ethically to align with the University of Liverpool's ethical guidelines.
Confidentiality and Data Protection	Participant data will be anonymised, securely stored, and not shared outside the project scope. It will be treated with the utmost confidentiality, and any data collected from participants who withdraw will be deleted. The project will ensure that

	participant data is used ethically, with no personal information disclosed, and all data will be deleted after the academic year or when no longer needed.
Ethical Compliance and Oversight	"Architexa" will undergo regular ethical reviews to ensure compliance with university ethical guidelines. The project team will consult with relevant university bodies for ethical guidance and report any concerns or deviations from approved ethical practices for investigation and resolution. By adhering to these ethical considerations, "Architexa" aims to uphold the University of Liverpool's commitment to research rigour, integrity, and ethical conduct, ensuring responsible data use, participant respect, and compliance with all relevant laws and regulations.

BCS Project Criteria:

1) This will involve programming skills from COMP 101, AI mathematics from COMP 219, and software project organisation skills from COMP 107 and COMP 201.
2) This idea addresses a GAP in the technically accurate image generation market.
3) This project will involve large amounts of academic research into GAN architecture and was decided upon based on market research, listed in the documents section of our canvas page (and referred to in the background reading section)
4) Similar projects are in everyday use worldwide. This technical implementation will save time and effort for architects who cannot use existing implementations practically.
5) Combining academic research with complex neural network construction (not directly taught in COMP 219) constitutes significant work.
6) We will have weekly / twice weekly meetings to discuss the outcome of our members' work in the interim and will practise pair programming during implementation.

UI/UX Mockup: UI: User interface UX: User experience:



Novel Features	**Design generation**:The option to generate architectural designs from user-written prompts. Editing generated design:The option to edit a preferred design by running it in the design generator to change aspects of the design, such as lighting, background, or colours. **Sharing & saving:**The option to share and save generated designs**.** **Terms of services page:**A section that informs the user of the rules and guidelines. **About page:**A section that informs the user of the history, mission, and who the people behind the work are.
User Demographic	**Primary User Group Profession:** Architects (students, early career architects, and senior architects) **Age Range:** 20 to 60, catering to architects in academic settings and those deeply involved in the industry. **Tech Savviness:** Moderate to high. Assuming a foundational level of comfort with architectural design software and a willingness to adopt new tools for efficiency and enhanced presentation. **Secondary User Group Profession:** Architectural designers, landscape architects, interior designers, and urban planners who participate in the architectural design process and require visualization tools. **Age Range:** 20 to 60, encompassing professionals at various stages of their careers who add to the architectural project lifecycle. **Tech Savviness:** Moderate to high. They are interested in software that offers specialized features for their domain, such as urban context integration and sustainable design visualization.
User Needs	**Swift Visualization:** Allow users to quickly bring ideas to life through intuitive interfaces and tools, increasing productivity and reducing the learning curve. **Versatile Presentation Tools:** A range of presentation features that enable users to create visualizations tailored to various stakeholder needs, from clients to regulatory authorities. **Time Efficiency:** Provide features that optimise design workflows, such as effortless integration with other design software,

	cloud-based collaboration, and rapid rendering capabilities.
User Benefits	**Enhanced Communication:** Clear, visually engaging presentations will help convey complex architectural concepts to non-specialists, improving stakeholder understanding and engagement. **Increased Productivity:** By minimising the time for visualization tasks, users have more time for design exploration.

Risk assessment:

	Contingencies	Likelihood	Impact
omplexity: The complexity of the AI image model may lead to challenges in understanding, , and updating the code.	Document the model architecture thoroughly. Use modular and well-documented code. Consider code reviews and collaboration to enhance code clarity.	Medium	Might impact updating the model and ma the model.
calability: The software may struggle to scale especially when handling image generation	Design the software with scalability in mind. Consider distributed computing and load-balancing strategies. Monitor and optimise resource usage.	Medium	The run time might be slower because of load.
t Hardware Utilisation: inefficient use of esources may lead to suboptimal performance or osts.	Optimise hardware utilisation through parallelisation, load balancing, and efficient resource allocation. Regularly monitor and adjust resource usage based on project demands.	Low	The hardware components might not be c the project's description.
nt Computational Resources: e processing power, memory, or GPU resources slow model training or inference.	Assess hardware requirements carefully. Invest in powerful GPUs or TPUs based on the model's complexity. Consider cloud-based solutions for scalable resources.	Medium	The participants' devices might not be abl the amount of data that will be possessed
Model Training Time: Training deep dels for image generation can be ming, especially for large datasets or complex es.	Optimise model architecture, utilise pre-trained models and consider distributed training to reduce training time. Set realistic expectations for the training phase.	High	The model training time might affect the r our project after implementation
ection Challenges: Collecting and ng high-quality training data may take longer y estimated.	Plan for data collection well in advance. Automate data preprocessing where possible. Consider using existing datasets to speed up the process.	Low	Might need more time to get more resour
nplexity: The complexity of the codebase may rstanding, debugging, and maintenance.	Prioritise clean and modular code. Use design patterns and follow coding standards. Document complex sections and provide explicit comments.	High	This will affect the time to train the mode coding it
n Selection: Choosing an inappropriate or gorithm may result in suboptimal performance eneration.	Stay updated on the latest research. Conduct thorough evaluations of different algorithms before selecting one. Consider using pre-trained models for a head start.	Low	The current algorithm, if implemented inc will be hard to fix.

References

- Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative Adversarial Networks. [online] arXiv.org. Available at: https://arxiv.org/abs/1406.2661 [Accessed 7 Feb. 2024].
- Mirza, M. and Osindero, S. (2014). Conditional Generative Adversarial Nets. [online] arXiv.org. Available at: https://arxiv.org/abs/1411.1784 [Accessed 7 Feb. 2024].
- Labelstud.io. (n.d.). Label Studio Documentation — Overview of Label Studio. [online] Available at: https://labelstud.io/guide/get_started.html#Quick-start [Accessed 10 Feb. 2024].
- 1 Python, R. (n.d.). Generative Adversarial Networks: Build Your First Models – Real Python. [online] realpython.com. Available at: https://realpython.com/generative-adversarial-networks/. [Accessed 12 February 2024].
- Brownlee, J. (2020). How to Develop a Conditional GAN (cGAN) From Scratch. MachineLearningMastery.com. Available at: https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/ [Accessed 12 February 2024]