Abstract

Visual representation in architecture, urban-design and planning is critical for both the design and decision-making processes. Despite major advancements in the field of computer graphics, crafting visual representations is still a complex and costly task, usually carried out by highly-trained professionals. This is particularly true during preliminary design stages - such as zoning exercises or schematic design - in which key decisions are made yet partial information about final design details is available. This work attempts to substitute common practices of urban-design visualization with a machine-learnt, generative approach. By implementing a Deep Convolutional Generative Adversarial Networks (DCGAN) and a Tangible User Interface (TUI), this work aims to allow for real-time urban prototyping and visualizations. The DCGAN model was trained on Cityscapes, a semantic street-view dataset. A version of CityScope (CS), a rapid urban-prototyping platform is used as tangible design interface. Following each design iteration on CS, the DCGAN model generates a rendering associated with the selected street-view in the design space. A light-weight, web-based and platform-agnostic tool was also created. Unlike traditional rendering techniques, this tool could help designers focus on spatial organization, urban programming and massing exercises without the need for detailed design, complex visualization processes and costly setups. This approach could support rudimentary urban design processes that are enhanced by the visual atmosphere, impression and discussion around 'The Image of the City'.

Keywords

Full paper, Machine learning and artificial intelligence (AI), AI + Machine Learning, Generative Design, Collaboration, Urban Modelling, Tangible User Interfaces

1 Introduction: The Image of the City

“To understand the role of environmental images in our own urban lives (...) we needed to develop and test the idea of imageability (...) and thus to suggest some principles for urban design.” (Lynch 1960, 11:14)

In his seminal 1960 book, MIT Prof. Kevin Lynch introduced a novel approach for analyzing the urban realm. Lynch offered a toolset for experiential classification of the city in which nodes, landmarks, paths, edges and districts are used to reflect the sensation of transitioning in the urban scape. Later, in his 1964 “The View from the Road” study, Lynch’s ‘imageability’ paradigm was manifested using a novel medium: He mounted an early version of a dashcam to a car and went on several daytime rides around Boston and other US metros (Lynch 1960, Andrews 2007). These films were later sped to reflect the
overall ‘feel’ of the road trip, surpassing the fine-details of the city. In this study, Lynch proposed to overlook fine-grained street elements or architectural details and instead focus on the ‘imageability’ of the urban outline: What composes the built mass? What shapes the street-section? Are there any noticeable landmarks, shapes or gaps, near or far? In the following years, Lynch’s novel documentation techniques became common practice in modern city-planning and urban-design (Carr and Schissler 1969; Pearce and Fagence 1996). This paper attempts to incorporate Lynch’s emphasis on visual coherence, continuity and user-experience of the built environment and utilize it on iterative, virtually created and machine-learnt environments.

Figure 1. The Five Elements of ‘Imageability’

Lynch asked his readers to avoid breaking down the above image into the common scale-based hierarchies of the built environment and instead asked to look at the overall mesh of the built environment.

1.1 The Issue with Rendering

Visual representation of early design studies is critical for both designers and stakeholders. The rapid development of computer graphics and the abundance of 3D tools made high-end visualizations a common deliverable in design practices (Batty et al. 2000). This demand grew cultures and styles of design renderings, in which the visualization of design - whether it is a city, a building or a doorknob - has to be seemingly realistic, marketable and appealing (Mekni and Lemieux 2014). Most design tools and CAD software followed these trends and streamlined the transition from rudimentary design to photo-realistic renderings.

Deep Image of the City: A generative approach to urban-design visualization
Nevertheless, the art of rendering still requires complex processes and diverse sets of expertise: High-end visualizations commonly involve tools and techniques with steep learning curves and often require costly software and hardware (Yan 2014). For example, most rendering tools require designers to construct virtual spaces in which multiple variables control the outcome, such as cameras, lights, materials and numerous other parameters. This process can quickly become tedious and laborious, especially for large and complex designs; occasionally, minor changes to certain parameters can gravely affect the output and duration of the rendering process (Lovett et al. 2015).

Moreover, early urban-design, planning or zoning stages tend to lack the details needed for proper representation of the design proposal. As a result, two common practices are used to visualize preliminary studies: (1) under-detailed and schematic representation, or (2) highly-detailed and realistically looking renders. Both approaches tend to not communicate the true impression and ‘feel’ of the urban scape, either due to lack of detail or as a result of overly-detailed yet unrealistic representation (Brusaporci 2017).

Lastly, the usage of design renders for marketing brought many experts to establish trends and fashions as to what is considered ‘good’ rendering outcomes. These norms favor the designed object and downplay its surrounding reality so that the design would appear at its best (Lovett et al. 2015). For example, a certain light angle might portray an office-tower as shiny and bright; saturation control can accent the design of a park over its gloomy surroundings; lavish storefronts branded with popular trademarks can help market an urban design scene. Although the outputs of these renderings might share visual realism, it is - in many cases - no more than a visual utopia.

---

**Figure 2. Preliminary Urban-Design Renderings**

These two visuals represent the same area in Cambridge, MA which is undergoing a large urban redevelopment process. The figure on the left shows volumetric depiction of the plan [From Cambridge, MA Citywide Model] and the one on the right is an artistic impression of the same plan [From MITIMCO]. The lack of data on the left figure is matched with highly-vivid representation of the proposal; in reality, the final design is significantly different from the shown here in the right.

This paper aims to explore a novel methodology for real-time, minimally-biased and setup-less urban-design rendering. The tool presented here features two interconnected systems: (1) a Deep Conditional Generative Adversarial Networks (DCGAN) which is trained on a
pre-classified street-view dataset and (2) CityScope (CS), a rapid prototyping platform that acts as a tangible user interface (TUI). CS TUI is used as the design space on which iterations are experimented by users. Following each design step, a 3d vector representation of the CS TUI is converted into an input vector raster. the DCGAN model creates a rendering based on this raster, associated with a selected street-view in the design space.

The next sections will explore the DCGAN model training, its dataset and the CS TUI platform architecture.

![Figure 3. CS+DCGAN: System’s Architecture](image)

Upper row: model is trained on Cityscapes dataset, converted and deployed as real-time Node.js app. Bottom row: CS instance collects user interaction and triggers the DCGAN renderings.

2 Model and Data

The recent popularity of Neural Networks (NN) disrupted many industries and research communities. From medical imaging to the stock-market, NN introduced new ways to observe large amounts of seemingly unrelated data and use it to perform state-of-the-art predictions. This work implements a Generative Adversarial Network, a variant of NN which uses large datasets to generate new data samples. This section explores the dataset, the model and its training process.

2.1 Deep Conditional Generative Adversarial Networks (DCGAN)

High-accuracy pattern recognition in large datasets is a key advantage of NN, with successful results as early as the late 80’s (LeCun et al. 1989). Nevertheless, generating new data points that well correspond to the original dataset is still considered a complex
challenge. The act of generation requires the model to not only recognize whether testing data corresponds to a training dataset but to also deeply understand the underlying structure of the data it creates (Creswell et al. 2018).

Generating images using NN was boosted with the Generative Adversarial Network (GAN) process introduced by Ian Goodfellow (Goodfellow et al. 2013). This process uses two competing NN to train and ‘adverse’ each other: the first network (the Generator) has the goal to generate a data sample indistinguishable from ground-truth data. The second network (Discriminator) is asked to decide whether the generated sample is close to the ground-truth data or otherwise discriminate it. During the course of training, the Generator improves its output so that the Discriminator would falsely assume it as the ground truth. The Discriminator is also improving its skill to decipher between a product of the Generator and the ground truth. Training may end in several ways, but commonly a certain proximity threshold between the generated and ground-truth data is used to conclude training.

A subset of GAN is the DC-GAN (deep conditional) algorithm (Isola et al. 2017). This type of networks trains a conditional model in which both the Discriminator and the Generator are fed with additional label-data and are trained on an objective function that is conditioned on these class labels (Mirza and Osindero 2014). Therefore, the Discriminator does not only falsify data points that are far from ground-truth, but also those who do not follow their associated class. For example, in a dataset of labeled dogs and cats’ images, the Generator might be trained well enough to produce a visually similar image of cat, yet since the label presented to the Generator is of a ’dog’, the Discriminator will falsify this product (Salimans et al. 2016).

A common usage of DCGAN architecture is translation form a class to data-point and vice-versa (i.e., denoting the label ’dog’ to generate a dog image); the rest of this section discuss this process, also known as pix2pix.

2.2 Image-to-Image Translation

GAN architecture is widely used to generate data samples such as images, videos or sounds (Donahue et al. 2019). One of the most notable usages of DCGAN is ’pix2pix’ (Isola et al. 2016) in which a label can trigger the generation of its parallel data-point and vice-versa. This method is specifically useful for generating highly-realistic yet new images from nothing but a label. Since its introduction, pix2pix methods were constantly improved to overcome false-positive effects such as ’mode collapse’ and the need for well-labeled datasets (Arjovsky et al. 2017; Zhu et al. 2017). Others looked into improving the trainable resolution and output of DCGAN. Nvidia’s pix2pixHD managed to train up to 2048px images while maintaining quality across their results (T.C. Wang et al. 2018). As discussed in ‘Methodology’ chapter, most of recent techniques were deemed less suitable for this work due to challenges in portability, model size and rendering speed.
3 Methodology and System Architecture

The purpose of this work was to deploy a prototype of an urban modelling platform that responds in real-time to design iterations made on a TUI and creates a corresponding street-view rendering. This system consists of a CityScope instance, a TUI dedicated for rapid urban-prototyping and visual feedback. The software architecture consists of three parts: (1) CS TUI acquisition, (2) a trained DCGAN model converted to web format and (3) front-end UI for visualizations. This section describes the different system components, including the DCGAN model, its training and dataset, the CS TUI for design iterations, the web-based frontend and the backend service connecting the different components.

3.1 Cityscapes Dataset

For the purpose of this work, a GAN model was trained on the Cityscapes dataset (Cordts et al. 2016). This dataset is composed of pairs of street-view images taken from a dashcam of car driving in over 50 German cities over several months, seasons (spring, summer, fall), daytime and weather conditions. For each street-view image, a corresponding image with a set of semantic labels is created. The labels are of 30 different color values, each of which represents a different class of street element, ranging from large scale surfaces (as buildings or roads) to small details (as license-plates or a road sign). The dataset contains 5000 pairs with fine class annotations, and 20k with coarse annotations. Table 1 includes the different labels of the Cityscapes dataset and the ones used for this project.

Table 1: Cityscapes dataset classes

<table>
<thead>
<tr>
<th>Group</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>flat</td>
<td>road*+ · sidewalk*+ · parking*+ · rail track</td>
</tr>
<tr>
<td>human</td>
<td>person* · rider*</td>
</tr>
<tr>
<td>vehicle</td>
<td>car* · truck* · bus* · on rails · motorcycle · bicycle* · caravan · trailer</td>
</tr>
<tr>
<td>construction</td>
<td>building · wall* · fence · guard rail · bridge · tunnel</td>
</tr>
<tr>
<td>object</td>
<td>pole* · pole group · traffic sign* · traffic light*</td>
</tr>
<tr>
<td>nature</td>
<td>vegetation*+ · terrain</td>
</tr>
<tr>
<td>sky</td>
<td>sky*</td>
</tr>
<tr>
<td>void</td>
<td>ground · dynamic · static</td>
</tr>
</tbody>
</table>

The different classes and their groups in Cityscape dataset. Marked with ‘+’ are labels which can be altered dynamically using CS TUI. ‘*’ are labels that are generated dynamically in the 3d model.

3.2 Dataset Tuning

For the purpose of this study, a subset of 2975 fine-annotated images were used. This set was deemed sufficient for image translations by Isola et al. (Isola et al. 2017). Each pair is concatenated into one 512x256px image composed of street-view and labels. Upon
observing the data, several image quality issues were found: (1) Since images are captured on a moving vehicle, motion-blur and lack of sharpness are apparent. Additionally, (2) given the dataset was created in Germany, the light and contrast conditions are overall gloomy and contain color cast of dark blues and greens. To address some of these issues, a pre-processing Open-CV algorithm was designed to increase sharpness, saturation and remove color-casting using (Bradski 2000). This algorithm was applied on significantly low-quality samples and only on the street-view half of the pair.

![Cityscapes Labels](image)

**Figure 4. Cityscapes Data Samples and Edits**

Left: dataset labels. Middle: original dashcam footage. Apparent color-cast, blur and color desaturation. Left: edited samples using kernel sharpening and channel enhancements

3.3 DCGAN Model Architecture and Training

Pix2pix popularity was manifested by manifold of clones and variations of the original algorithm. As described in section 2.2, the model used for this paper carries a mainstream pix2pix architecture with minimal alterations to allow for faster prediction rate and a lightweight model. To allow for real-time output on multiple platforms without cumbersome setup, a modified pix2pix port of TensorFlow 1.12 in python 3.6 was selected (Abadi et al. 2015). This port allows to export the model’s weights to a web-ready format and load it onto a Node.js service through Tensorflow.js library, so it can be deployed to live websites without the need for remote server processing.
Figure 5. DCGAN Training Phases

These samples are of the training session with 64 filters with 450 epochs. Upper raw depicts first, 75 and 150 epochs results. Clear improvement is apparent in the DCGAN predictions for shape, colors and sharpness. Bottom raw shows the continuation of the same session (epochs 200-350) in which degradation from prior epochs and potential ‘mode collapse’ are apparent. A reason for the bottom results could be related to the uniqueness of this sample: rarely in this dataset, skyline or building shapes are missing. Therefore, the model is well predicating the right-side façade perspectival orientation but fails in applying prediction to the vanishing point area which is labeled as s building in this case.

3.4 Network Layers Architecture

As described in section 2.1, the model is composed of two competing networks: [D]iscriminator and [G]enerator. The G network contains 16 layers with a U-Net encoder-decoder structure (Ronneberger, Fischer, and Brox 2015). The first 8 layer compress the input vector into a compact latent space. The other half of the U-Net is a decoder that up-samples the compact latent vector. The architecture of the D network has 5 layers and is using Leaky ReLU as activation function that has been shown to improve stability in GAN training (Radford, Metz, and Chintala 2015). During training, the model creates two identical copies of the D network, one to learn ground-truth pairs and the other for generated pairs.

3.5 Model Size and Performance

Both the D and G networks can benefit from high number of filters applied to detect details of the input latent vector. However, with more filters the model size grows, which can gravely impact real-time performance and usability in most devices. As discussed in ‘Results’, a shallow network with low number of filters was implemented. Due to this number, re-scaling the input vector was used to improve the attention to fine image details. As explored by Isola et al. (Isola et al. 2016), up-scaling the latent input vector improves both D and G attention to fine details. Therefore, a random up-sampling of up to 150% was
applied during training.

4 CityScope: Rapid Urban Prototyping

The second part of the system is composed of a CityScope (CS) instance. CS TUI is used for triggering the model generation of street-view renderings and displaying them as they emerge. This section presents CS design and interaction.

![CityScope Platform](image)

**Figure 6. The CityScope Platform**

Multiple user (up to 4 in this case) can together interact, discuss and design the urban space using CS TUI. The horizontal table-top is used as both the design space and a schematic plan view of the design. The vertical screen visualizes the DCGAN results.

4.1 HCI for Urban Design

Since the emergence of Computational Aided Design (CAD), designers and engineers looked for new ways to collaborate in design processes (Sutherland 1964). Nevertheless, traditional CAD tools were commonly designed around the working environment of a single person with limited input (mouse, keyboard) and output devices (monitor, printer). Moreover, most CAD and design tools carry steep learning curves, require specialized skills and feature limited capabilities for interaction (Noyman, Sakai, and Larson 2018). These tools challenges both experts and non-experts to properly evaluate designs and to rapidly iterate on design scenarios.

To date, few Tangible User Interfaces (TUI) have been developed to facilitate a more collaborative design processes, augmented by computation and data analytics. As such, the Augmented Urban Planning Workbench (Ishii et al. 2002), URP (Underkoffler 1999) and
The Clay Table (Ishii et al. 2004) were designed to facilitate a collaborative, computer-aided urban-design process.

Since 2013, researchers are developing CityScope (CS): a human-centered, urban modeling, simulation and decision-making platform. CS seats in the intersection of Urban-Planning, Human-Computer Interaction and Social Sciences with the goal of prioritizing an evidence-based discourse around the nature of the built environments. Through a series of lab experiments and real-world deployments, CS has been successful in providing insights, predictions and consensus to various urban-design questions (Noyman 2015; Noyman et al. 2017). For the purpose this project, a CS instance was designed and constructed in an active demo area situated at ________________.

4.2 CityScope User Interaction

CS is designed to allow for a playful, unrestricted interaction with a tangible environment, which is augmented by real-time analysis, simulations and predictions. This differentiates CS from other design tools, in which basic interaction requires experience and feedback is rarely produced in real-time.

A CS platform features several key components: (1) a tangible urban model, (2) an acquisition unit and (3) a feedback module. The tangible urban model includes a gridded

Figure 7. CityScope, DCGAN and cityIO data flow

User interactions are scanned on the CS scanner and sent to cityIO for storage, logging and distribution. In this case, the DCGAN model is designed to listen to cityIO changes (through Node.JS app), so that new visualization is produced with each new cityIO data.

Deep Image of the City: A generative approach to urban-design visualization
table-top of pre-tagged bricks acting as intractable building-blocks or massing elements. This allows users to rapidly modify different land-uses and streetscape elements by manipulating the tiles locations, adding or removing them. The acquisition unit is in-charge of scanning the scene: Each user interaction is detected via a scanning hardware and is transmitted to cityIO, a distributed cloud service. An openCV tool (Bradski 2000) was designed to recognize these interactions. Lastly, a feedback module which contains display screens and projectors communicates the outcomes of the CS interaction and the DCGAN model perdictions.

![CityScope Schematics](image)

Figure 8. Schematics of CityScope TUI and Feedback

For this case study, the CS platform was designed with a 16x16 grid of 4x4 Lego tiles that are randomly populated by clones of five patterns. Each pattern is a unique 16 bits matrix of black and white studs facing downwards to be scanned via the CS scanner. Each tile and pattern represent a different land-use or streetscape element: roads, buildings, green-spaces, parking and sidewalks. In addition, a single 'observer' pattern was designed to mimic a virtual pedestrian and to set its angle of view. In addition, each tile is associated with virtual parameters (such as z-axis height, rotation, density, etc.) at a scale of 10x10 meters. This scale was found suitable to allow rapid street-form and building-mass editing while maintaining a decent level of granularity for urban design exercise (Noyman 2015). Table 1 specifies the dataset classes used as patterns.

When users interact with the CS grid, the scanner decodes the patterns of each grid-cell and triggers a change to the data-structure of the CS table. These changes are then sent to
cityIO server, a cloud service holding the CS data and is responsible to distribute it among all CS instances. CityIO exposes its data as an HTTP API with JSON structure which is later used by the frontend for DCGAN model renderings.

4.3 Model Output and Frontend

As users manipulate the tiles via the TUI and design the streetscape, their interactions are visualized on the CS table-top surface and on the vertically mounted display. Each design iteration creates a new array of digital patterns. This array is then translated into a 3D environment in which each grid-cell is represented via its label and predefined parameters. For example, a vegetation pattern yields a flat green rectangle colored in RGB values that correspond to the Cityscapes label-color.

In addition, this pattern is triggering an algorithm that proliferates small-scale streetscape objects associated with that label, so that in the case of the vegetation label, trees, bushes and live-fences will emerge in the virtual space. This algorithm is also controlling the position, rotation, shape and height of these generated objects. As such, a sidewalk pattern will yield pedestrians or street-signs and a parking-lot pattern will be proliferated with parked vehicles. As users design the overall urban structure, the environment is autonomously filled with small-scale street elements.

The UI for this system was built as an online Node.js application using the TFjs library (Smilkov et al. 2019) for handling input and predictions. The 3D scene is built using WebGL and THREE.js (Mrdoob 2019) library.

4.4 Observer and Rendering

This composed 3D environment is captured via the ‘observer’ pattern. This one-off plate is designed to control the 3D camera position so that each move or rotation will create a new perspective viewport. The camera baseline parameters, such as FOV and height were approximated to the Cityscapes camera calibration appendix, so that the Observer would yield a matching input vector to the trained dataset (Cordts et al. 2016). Nevertheless, additional camera controls were implemented to allow CS users to rotate, pan and zoom the ‘observer’ via a custom TUI built of a game pad joystick.

With every change to the CS grid array, the observer current viewpoint is captured as a raster image. This image is then converted into an input vector which is fed to the DCGAN Tensorflow.js converted model. The model then predicts an image that corresponds to the input labeled image and its output is then drawn onto the CS canvas and displayed on a vertical monitor.

4.5 CityScope Table-Top

The CS platform horizontal table-top is used as both the design space as well as a canvas for visualization. Following each design move, a schematics diagram of the Cityscapes labels is projected onto the canvas to reference the design. The observer position is
displayed using a colored tile with a perspective cone that indicates the direction of its view.

Together, the HCI components of this system allow users to effortlessly design and amend the urban streetscape environment and observe the effects of different scenarios in real-time.

Figure 9. CityScope Design Space

[Left figure] The CS table top is projected with matching hues from the Cityscapes dataset labels, so that users could easily match that with the design of preferred streetscape objects. In this case, the Observer (marked by a Lego mini-figure) with view cone situated in the center of the array. [Right figure] an Arduino joystick module was designed to allow for additional Observer control, such as rotate, pan, zoom and render on click.

5 Results

5.1 DCGAN Training and Results

As described in section 3.5, the model size and portability were significant factors for choosing training methodology. Overall, nine training sessions were carried out to test various initial filter sizes (16, 32, 64 and 128 filters), different epochs (between 200 to 2000) and various degrees of dataset manipulation as described in section 3.4. The resulting models were converted to a web-ready format and tested for response time in browser environments using Chrome and Firefox on (1) a 2016 Macbook-Pro, (2) a 2015 Windows 10 desktop with dedicated GPU and (3) an Android Pixel phone. When evaluating model size, real-time performance and generated image quality, the model trained using 64 filters with 455 epochs showed the best overall results. Models with lower filters produced low-quality images; Models with similar or more filters, that were trained on 800-2000 epochs displayed significant ‘mode collapse’ (Arjovsky, Chintala, and Bottou 2017) and inconstancy of results.

Deep Image of the City: A generative approach to urban-design visualization
5.2 CityScope Platform and User Interaction

The CS instance and UI components were constructed in a highly-active demo area at _________ for testing and evaluation. The system has two asynchronous processes set to avoid interaction latency: (1) DCGAN prediction process and (2) CS TUI interaction response. The DCGAN model performs at ~0.66 sec/prediction and the TUI has a fixed cycle of 100ms. Although the model slightly trails the TUI, users tend to focus their attention on the ‘observer’ or the design space before expecting the DCGAN output. In that sense, the overall experience can be considered real-time, allowing a continuous and seamless design-and-feedback loop.

6 Discussion and Conclusion

This paper has described the design and deployment of a tangible urban-design platform that utilizes Deep Neural Networks (NN) to perform real-time street-view rendering. This platform was built using a Deep Conditional Generative Adversarial Networks (DCGAN), a NN architecture that is trained on a classified street-view dataset. An instance of CityScope, a tangible user interface for rapid urban-prototyping was used to allow design iterations and feedback. The rest of this section will discuss the strengths, weaknesses, threats and opportunities of this work.

6.1 Strengths

Insightful visualizations during design processes are crucial for designer as well as the wide-range of stakeholders and decision-makers. While there already exist a slew of tools for 3d modelling and rendering, few efforts have been made to integrate these tools into a real-time, non-expert and tangible platform that is capable of producing seemingly-realistic outputs. Moreover, urban-modeling and visualization typically requires laborious setup, costly hardware, software and steep learning curves. The tool presented here is the first to provide a web-based and intuitive user-interface, allowing multiple people with varying levels of expertise to collaboratively experiment with urban-design scenarios and real-time feedback. Unlike traditional design tools, significant part of the effort to construct a render-able 3d scene (including setting up lights, cameras, materials, textures, shaders and parameters) is carried out by the trained NN.

In the context of urban-design processes, this tool can augment early stages of regulatory design with vivid street-view insights. This is especially important when early massing and zoning exercises are preformed; These stages have a major impact on the spatial organization of the city, but commonly tend to lack visual representation of the design (Noyman 2015). This tool offers designer and regulators the ability to bypass the need for detailed-design and instead focus on the 'Image of the City' (Lynch 1960).

6.2 Weaknesses

Despite the promise of generative NN, utilizing DCGANs has several drawbacks: (1) These
models require large and properly labeled datasets. In this case study, creating a new Cityscapes dataset (for example, for non-German cities) involves driving around multiple cities for several months, capturing thousands of images, segmenting and classifying them. Several emerging methods suggest decoupled (Zhu et al. 2017) and label-less learning (Lucic et al. 2019), which carry promise to simplify the labeling effort; Nevertheless, aggregation of the dataset and partial labeling will still be required. Additionally, (2) GAN models tend to be inconstant during the learning process and in their outputs, as explored in section 3.4.

Another concern is the model’s inability to generalize to non-street view angles. Since the entire dataset was created using a vehicle dashcam, only street-view angles produce reasonable predictions. This issue is true to any NN architecture that is based on labeled data: as soon as predictions are required outside the bounds of the expected labels, the network will falsely converge into an unpredicted latent space resulting with poor or irrelevant outputs (Salimans et al. 2016).

6.3 Threats

The seemingly autonomous nature by which GAN can generate or ‘create’ is compelling to many and was a strong motivator to NN rising popularity. Nevertheless, GAN tend to be highly unpredicted both in training and usage (Shin et al. 2017). When it comes to the design practice, certain degree of ‘creative freedom’ is indeed desired, yet unpredicted tools might create resentment or misleading impressions. In this case study, when rendering twice from the same exact street-view angle and the same urban-design setup, different results might occur. While the authors of this paper perceive that a positive manifestation of Lynch’s ‘Imageability’, where the overall urban-form is surpassing details, others might observe this as an untamed technology.

Another threat is the degree of bias associated with the model’s results. Although this tool might appear to yield seemingly unbiased outputs, it is important to emphasize that NN are strictly bounded by their architecture and their trainable data. Thus, tempered datasets and certain NN designs might greatly affect the outcomes of the model. With more machine-learning tools offered to the design industry, these concerns should be addressed by testing, validating and questioning their impact on design outcomes.

6.4 Opportunities

The tool presented in this paper is a prototype for real-time design visualization. This work can be greatly improved in several aspects: First, the DCGAN results can excel using better NN architecture and better-tuned training parameters. Advanced GAN architectures or other generative methods such as VAE can bring finer results with greater control over data generation. Improvements to both mobile-devices hardware and web-based NN software could bring such tools to the masses with minimal cost and learning-curve. As well, extending the model’s training to datasets of different urban environments and geographies could allow for versatile representation of urban-design scenarios in different contexts.
Lastly, the CS TUI can be extended to include more granular interaction so that design will not be limited to at scale but will concatenate interactions in multi-scale environments.

More broadly, the approach presented in this paper might approximate design-computation to offer insightful assistantship spanning beyond toolbars and rulers. In that sense, CAD tools would not only expedite tedious drafting and prototyping tasks, but might be able to leverage the power of advanced computation to become ‘companions’ who offer valuable insights during the design processes.
References


Donahue, Chris, Julian McAuley, and Miller Puckette. 2019. “Adversarial Audio Synthesis.” In ICLR.


Deep Image of the City: A generative approach to urban-design visualization


