Journal of Building Engineering 114 (2025) 114211

Contents lists available at ScienceDirect

Journal of Building Engineering

journal homepage: www.elsevier.com/locate/jobe

Full length article

Low-resource GAN-stack for high-resolution floor plan generation with enhanced evaluation and contextual validation

Michael Sahl Lystbæk [a],*, Michail J. Beliatis [a], Archontis Giannakidis [b,c]

[a] Department of Business Development and Technology, Aarhus University, Birk Centerpark 15, Herning, 7400, Denmark
[b] School of Science and Technology, Nottingham Trent University, Nottingham, NG11 8NS, UK
[c] Archimedes Unit in Artificial Intelligence, Data Science and Algorithms, Athena Research Center, Marousi, 15125, Greece

ARTICLE INFO

Keywords:
Architectural design automation
Generative machine learning
Generative adversarial network
Artificial intelligence
Resource efficiency
Deep learning
Floor plan layout

ABSTRACT

The fast and reliable prototyping of floor plan layouts is a crucial element in the early stage of the building construction cycle. The purpose of this study is to ameliorate the major difficulties associated with the use of generative adversarial networks (GANs) for automating high-resolution floor plan generation: vast computational and data requirements, training instability, and problematic result evaluation. A stable resource-efficient multi-module GAN-stack framework is proposed comprising pre-processing (denoising and 4× down-sampling), floor plan image generation, and up-sampling modules. Each module is individually optimized. An innovative holistic evaluation framework of the generated building floor plans is presented covering image quality, diversity, truthfulness, overall training time, energy spent during training and associated carbon emissions. A novel validation framework is introduced, involving contextual building functionality, data privacy, capability to manipulate design generation to suit the designer's desires, usability in BIM downstream tasks, and inference speed. Results demonstrate that the apropos network architecture choices allow for significantly cutting down the wall clock training time (<60 h) while maintaining superior generated image quality and contextual meaningfulness, given only sub-thousand 1024 × 1024 training images of single-story residential Danish homes and a limited computing resource (a single RTX-3090 GPU). The proposed computer-aided pipeline may support decisions between architects and their clients. It may broaden access to the GAN-based research on the automation of building design. The presented automated tools could find application in other industries which have the same driving needs and resource constraints for adopting GANs, and that also lack ways of validating their end product.

1. Introduction

1.1. Background

A floor plan is a horizontal cross-section representing the spatial architecture of a building and its functionalities. The floor plan design is a crucial early phase in the architectural design process, since many fundamental decisions need to be made at this stage; decisions which are both difficult and expensive to change later on [1]. It is also a complex iterative task, which largely depends on the designer's experience, skills, intuition, and creativity [1]. In order for the architectural consultants to properly

* Corresponding author.
E-mail address: msl@btech.au.dk (M.S. Lystbæk).

https://doi.org/10.1016/j.jobe.2025.114211
Received 17 February 2025; Received in revised form 8 August 2025; Accepted 26 September 2025
Available online 6 October 2025

communicate the design solutions with clients and reach final agreement, the fast iterative prototyping of architectural building layouts is of vital importance. However, the automated generation of floor plans is challenging since it requires expertise due to building functionalities [2]. In addition, floor plan drawings are significantly different from other natural images [3].

Generative machine learning (GML) is a widely discussed branch of artificial intelligence (AI), mainly because of its capacity for performing creative tasks, which had always been associated with a unique ability possessed by humans [4–11]. NVIDIA impressed the world in 2018 by introducing StyleGAN, which is a style-based generative adversarial network (GAN) [12] with the ability to generate high-quality images; for instance, artificial human faces which have been introduced in a variety of different cases (i.e., "thispersondoesnotexist.com"). StyleGAN has been further improved ever since, lastly as the StyleGAN3 implementation [13]. Other variants such as StyleGAN-XL, which has a deeper artificial neural network (ANN) structure, do exist [14]. The research community finds the use of GANs relevant because of their relatively fast image generation process compared to the highly credited diffusion models. The latter have recently been the focus of attention due to their ability to generate high-quality images by relying on large datasets and training setups [15]. Tests by Sauer et al. [15] showed that GANs have the power to perform image generations much faster than other GML techniques and still maintain a high image quality [15,16]. Utilizing StyleGAN with adaptive discriminator augmentation (ADA) has been shown to improve a model's performance on smaller datasets [17]. Besides, the integration of the CLIP dual text and image encoder into the pipeline has pointed out GANs' capability to perform text to image generation [18]. Recent research has focused on improving the training process by modifying the StyleGAN discriminator and generator architecture and by adapting weights from pre-trained models [19–21]. These techniques have shown superior results in comparison to the original model proposed by NVIDIA, both in the sense of faster training and training on smaller datasets [19–21]. GML is expected to have a growing impact on the development of construction applications.

1.2. *The challenge*

In a recent review study, Chai et al. [22] identified the four main challenges concerning the use of GANs in construction applications as:

1. **Questionable applicability due to problematic evaluation of results:** As GANs are implicit models and do not return likelihood values, it is inherently difficult to evaluate these models using general quality metrics [23]. Fréchet inception distance (FID) is a popular quality evaluation metric, having dominated image quality evaluation also in construction applications. However, it has been criticized for being too unilateral due to its single-dimensional assessment values and also for providing some unrealistically low quality scores owing to leftover bias [16]. In addition, precision and recall have been found to be too sensitive to outliers [24]. The above factors call for a new and more exhaustive evaluation framework for GAN applications to construction.
2. **Requirement for substantial computational resources:** GAN training requires vast amounts of compute power [21]. Tech giants such as OpenAI, Alphabet, MidJourney or NVIDIA have been keen to enhance their computing infrastructure to meet the demands for training large generic GML models [9]. However, access to such large-scale GPU clusters is a limitation in the technology's applicability to the construction industry.
3. **Provision of a large training dataset:** The dataset quantity, as well as quality, are of paramount importance for developing and training GAN models [17,25,26]. The proper amount of data needed for training GANs can be difficult to define. Previous studies indicated several thousand, or hundreds of thousands, of images to be a cohesive dataset [17,27]. However, the collection of a large training dataset in the range of a hundred thousand or millions of images is costly or even not feasible [17,22]. It is worth noting that the investigation of limited datasets starts from 2k images [17]. The small datasets can be supported with data augmentation techniques which can have an important impact on the training [17,28].
4. **Training instability:** This challenge relates to the generated image quality. It refers to vanishing gradient problems, mode collapse in the generator (low diversity), and other discrepancies [22,26].

The above difficulties in generating high quality floor plan images with GANs have also been described by other studies [29].

1.3. *Related work*

In the GML-based synthetic floor plan generation domain, several studies have been performed recently, focusing on various aspects of the problem. A variety of methodologies have been presented, reflecting the growing interest in using advanced computational techniques to automate and enhance architectural design processes.

Graph-based GAN methods are a prominent approach for the conditional generation of 2D architectural space layouts. The conditions are user-defined, offering flexibility and control over the floor plan design output. Recent works have highlighted how these graph-based techniques can be employed to model the spatial relationships and architectural constraints in a way that aligns with traditional design principles while also pushing the boundaries of automated design [30–32].

The pixel-based generation of architectural floor plan layouts with GAN techniques has also been widely explored due to its ability to produce detailed and visually coherent layouts [33–41]. Image-to-image conditional GANs (cGANs) such as CycleGAN [33] offer flexibility in working with data classes. On the other hand, pix2pix [34–36] needs to be trained on data pairs. Diffusion-based image inpainting is proposed for conditional floor plan analysis and generation tasks [42,43]. However, diffusion models are notoriously data-hungry and computationally expensive to train. A coupled generative adversarial network (CoGAN)-based framework [41] for the rapid generation of an unlimited number of floor plan design solutions has also been proposed.

For construction applications in particular, GAN-chains have also been proposed in several studies so that the multi-module collaborative system can benefit the processing of multi-property tasks [38,44,45].

The studies discussed above highlight the promise of employing GAN techniques in construction applications. However, to the best of our knowledge, no literature exists to date that has attempted to collectively address all four challenges identified in the review study [22].

1.4. Aim and contribution

The aim of this study is to make important headway in dealing with the challenges discussed in Section 1.2. The contributions are summarized as:

- A novel resource-efficient multi-module GAN-stack framework is proposed (Fig. 1) for the autonomous generation of unique and contextually meaningful high-resolution (HR) (1024 × 1024) floor plan layouts aimed at assisting in the architectural design workflow. The framework comprises pre-processing (i.e., denoising and 4× image down-sampling), floor plan image generation, and image up-sampling modules. The proposed implementations are the AREA down-sampling method [46] and a transformer-based denoising approach [47] (pre-processing module), the Projected GAN [21] with FastGAN generator [19] (module 1), and the Real-ESRGAN [48] (module 2). The configuration of each module is individually optimized.
- To address gaps in previous related studies, an innovative holistic evaluation framework of the generated floor plan images is presented that covers (apart from generated image quality) also fidelity, diversity, and truthfulness. In addition, the framework involves assessment of the overall training time, energy spent during training, and associated carbon emissions. The latter are important due to deep learning's terrible environmental cost [49].
- A novel validation framework is presented to ensure that the computerized framework performs the way it was intended and it can be used to achieve business objectives. The framework includes elements such as contextual functionality, data privacy, capability to manipulate design generations to suit the designer's desires, usability in a downstream task, namely a building information modeling (BIM) software-based implementation task, and inference speed.
- Results demonstrate that the apropos framework architecture choices allow for advanced computational resource and data efficiency and, at the same time, mitigate the training instability in the GML-based floor plan generation. Notably, the proposed framework is shown to significantly cut down the wall clock training time (<60 h) while maintaining superior generated image quality, given only sub-thousand high-fidelity training images of single-story residential Danish homes and a limited computing resource (i.e., a single RTX-3090 GPU).
- The performed experiments provide insight that could form a knowledge base about the future applicability of the proposed tools (for image generation, evaluation, and validation) to other industries for which reliability and fast iterative prototyping are some of the driving needs for adopting GML, but they do not have access to large-scale data and computational resources and that also lack ways of ensuring that their end product is consistent with the intended application.

This study introduces down-sampling (pre-processing) and up-sampling (post-processing) in the floor plan generation process (in contrast to training models on HR datasets) as a means to boost computational resource efficiency. This decision was driven by the fact that image resolution has been shown to have a large impact on the training performance of GANs [19,50]. The proposed multi-module GAN-stack framework for generating HR architectural floor plan layouts is therefore in a sense analogous to mobile telecom voice/video data processing methodologies, where the original data signals are down-sampled and compressed with multi-rate signal processing so that the time and cost required to transmit the data across the channel is reduced, and then the opposite process is applied on the receiver side [51]. It has been found that down-sampling the data resolution from 1024 × 1024 pixels to 256 × 256 pixels reduces the training processing time (s/kimg) by threefold and it also lowers the required VRAM [19,50].

2. Methodology and implementation

A novel multi-module GAN-based framework (Fig. 1) is proposed for generating HR synthetic floor plans with low data and computational resources. In the pre-processing module, the floor plan images are rendered without noise (i.e., unwanted text content, dimensions, and logos) and downsized for the streamlined training of the subsequent floor plan generator. The purpose of the next module is to train a resource-efficient style-based GAN model to generate new synthetic floor plan designs. In the final module, a GAN-based up-sampler, trained on HR and low-resolution (LR) dataset pairs provided by the pre-processing module, increases the image resolution and quality of the floor plan generated in the previous module. To boost the overall framework performance, each module configuration is optimized. The framework performance is evaluated and validated using a combination of established and innovative metrics. A definition of technical terms is presented in Table 1.

2.1. Dataset

The experiments are based on an initial dataset of 1233 HR architectural floor plan images of one-story, single-family homes in Denmark, scraped from a real estate web database [52].

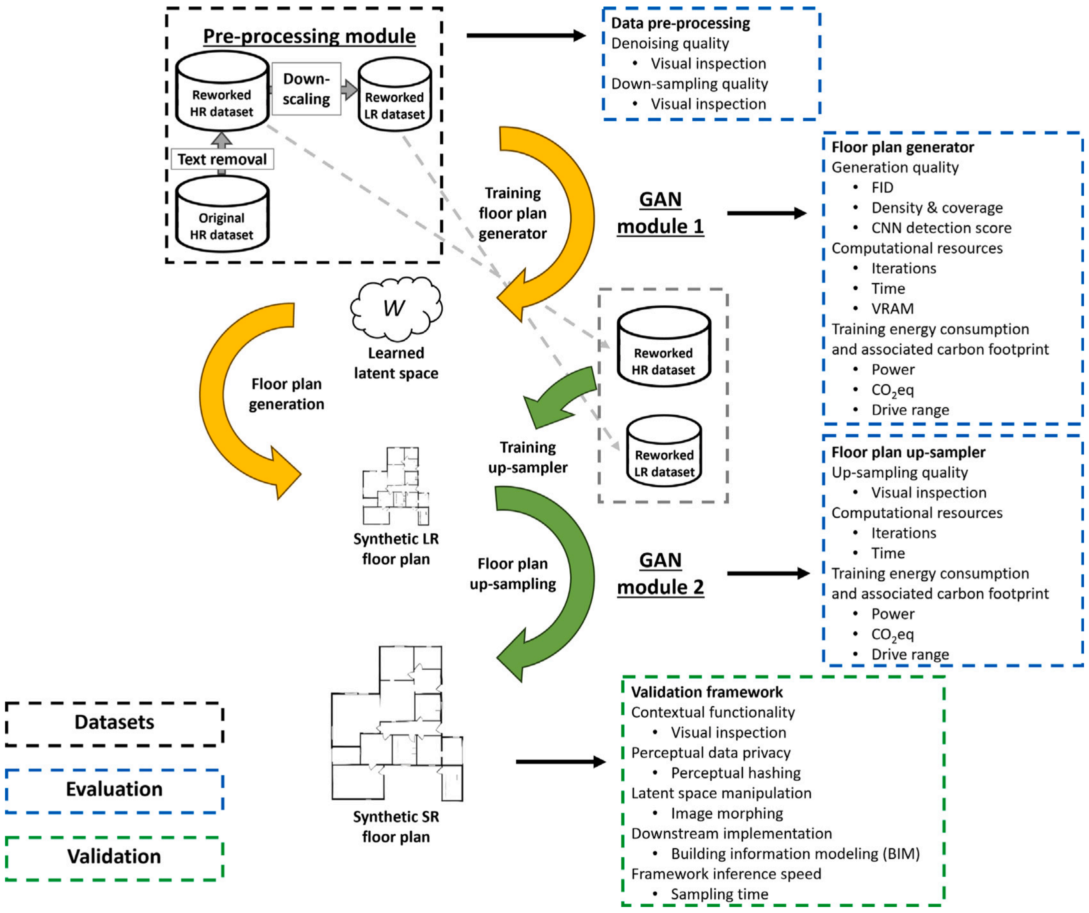

Fig. 1. Multi-module GAN-stack framework for generating high resolution architectural floor plans with limited computational and data resources. Pre-processing module involves denoising (text removal) and image down-sampling (processes indicated by gray arrows). Module 1 refers to floor plan image generation (processes indicated by yellow arrows). Module 2 refers to image up-sampling (processes indicated by green arrows). Black dashed line indicates datasets, blue dashed line indicates evaluation metrics, and green dashed line indicates validation metrics. BIM stands for building information modeling, CNN stands for convolutional neural network, FID stands for Fréchet inception distance, HR stands for high resolution, LR stands for low resolution, and SR stands for super resolved.

Table 1
Definition of technical terms.

Term	Definition
GAN	A GML model consisting of a generator and a discriminator architecture, trained adversarially to produce realistic synthetic data.
SRGAN	A GAN-based model for upsampling low-resolution image inputs to high-resolution outputs.
StyleGAN	This GAN architecture introduces style-based modulation at different layers of the generator, enabling fine control over image features and structure to produce highly realistic images.
Latent space	A compressed numerical representation space used by generative models to encode features.
FID	A metric used to evaluate the quality of images generated by generative models by measuring the distance between the feature distributions of real and generated images.
Density & coverage	Metrics that assess how well generated data matches or covers the real data distribution.
CNN classifier	A CNN-model that can assess image content, including whether generated images can be distinguished from real images based on visual features.
TL	Transfer learning enables model training on one task to improve performance on a related task with limited training data.
End-to-end training	This training setup enables the full training of a model, where all components are optimized from input to output.
Carbontracker	A tool for estimating energy use and CO_2 emissions in ML experiments.
Image hashing	A technique that maps data to fixed-size representations, often used for fast similarity search and retrieval in large datasets.
Morphing	A technique used to smoothly transform one image into another by interpolating shapes and appearances, often applied in data augmentation or style blending.

2.2. Pre-processing module

The original floor plan image dataset contains noise in the form of logos, text descriptions, and dimensions. To suppress this noise, a multi-head attention method [47] is proposed in the pre-processing module. The denoised HR dataset is further down-sampled to an LR dataset to achieve a cost-effective and faster training process with low computational resources in the subsequent GAN modeling. It is proposed to optimally downscale the clean dataset using the AREA method (OpenCV INTER_AREA) [46,53]. The impact of removing unwanted text content from the initial training dataset is evaluated on a downstream StyleGAN3-based floor plan image generation task. In particular, the quality of images generated by the GAN model trained on a LR dataset, with and without text content, is compared. A qualitative visual assessment is found appropriate for this comparison. A visual comparison is then conducted between the AREA down-sampling method and three baselines: bilinear, bicubic, and nearest neighbor.

2.2.1. Dataset denoising

The dataset is pre-processed with the image text remover and label extractor (ITRLE) model [47] in an automated process based on the text spotting transformer (TESTR) architecture [54]. In the encoder, the model relies on ResNet-50 [55] to extract features from the input images [47,54], whereas the detection of small text instances is enhanced by applying multi-scale deformable attention to the feature maps. To train the dual (i.e., character recognition and location) decoder, the following loss function (L_{dec}) is used:

$$L_{\text{dec}} = \sum_{j} \left(\lambda_{\text{cls}} L_{\text{cls}}^{(j)} + \lambda_{\text{coord}} L_{\text{coord}}^{(j)} + \lambda_{\text{char}} L_{\text{char}}^{(j)} \right) \tag{1}$$

where λ_{cls}, L_{cls} represent the classification loss weight and classification loss of the text instances, respectively, λ_{coord} is the control point coordinate regression loss weight, L_{coord} represents the control point loss, λ_{char} and L_{char} are the classification loss weight and character classification loss, respectively, and j represents the query number.

2.2.2. Dataset down-sampling

The denoised images are resized from the resolution of 1024 × 1024 pixels to 256 × 256 pixels. The AREA method (OpenCV INTER_AREA) [53], which relies on pixel area relations and resizes images using adaptive average pooling, is utilized. It is a highly endorsed method to shrink an image because the results are free from aliasing-based Moirė patterns.

2.3. Floor plan generation module

The floor plan generation module is applied to the LR dataset, obtained after the pre-processing described in Section 2.2. To achieve the objective of a resource-efficient and stable framework, this module is implemented using Projected GAN with FastGAN architecture for the generator [19,21]. A holistic comparison of the method is conducted with two state-of-the-art (SOTA) approaches, namely end-to-end StyleGAN3 training and StyleGAN3 transfer learning (TL), in terms of generated floor plan image quality, fidelity, diversity, and truthfulness, as well as VRAM requirements, overall training time, energy consumption and the associated environmental footprint. The majority of these criteria have not been used before to assess GAN-based generation in construction applications.

2.3.1. StyleGAN3 end-to-end training

The end-to-end trained floor plan generator is based on the StyleGAN3 architecture, which is the latest release of the iconic style-based GAN by NVIDIA [13]. While the discriminator of StyleGAN2 is left unchanged, the StyleGAN3 generator has an improved architecture by upgrading the quality of the up-sampling filters to reduce aliasing issues that have been one of the main challenges that StyleGAN2 faces. As opposed to StyleGAN2, StyleGAN3 relies on Fourier features, the per-pixel noise inputs are discarded, the mapping network depth is reduced, the mixing and path length regularizations are disabled, the output skip connections are disposed of, the boundaries and up-sampling are handled with enhanced filtering, the nonlinearities are treated with a custom CUDA kernel leading to a tenfold decrease in training time, oversampling is used in all layers except for the highest-resolution ones, flexible layer specifications are allowed, 1 × 1 convolutions are used, an improved jinc-based down-sampling filter is used in all layers except for the two critically sampled ones, and Gaussian blurring is applied early in the training. Even though StyleGAN3 matches FID-wise the StyleGAN2 generation quality, their internal representations are noticeably different. StyleGAN3 is developed to perform a natural transformation hierarchy, ensuring that the sub-pixel positioning of each feature is accurately inherited from the underlying coarse features, as described by [13].

The objective function (Eq. (2)), $V(D,G)$, for GANs can in general be expressed as [56]:

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2}$$

G represents the generator, in which V is aimed to be minimized, and D represents the discriminator, where V is aimed to be maximized. Hence, there is a competition between G and D driven by V, with G trying to generate images that D is not able to classify as fake. $\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)]$ encourages D to rightly predict high probabilities when classifying real data samples from the data distribution. $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$ helps D to correctly classify fake data generated by G.

The above adversarial training process for GANs iterates for a large number of epochs. This iterative backpropagation process requires high computational resources. Over time, G becomes better at creating realistic images, while D improves its capability to identify fakes over real images. Convergence is achieved when G generates realistic-looking data.

The learning performed by GAN models is captured in a vector space denoted as the latent space. StyleGAN was constructed with three latent space levels to gradually develop the style features [57]. Z is the first latent space, which is the random (typically Gaussian) input noise vectors that provide the starting point for the image generation process [13,57]. Next, the intermediate latent space W is formed by the learned, fully-connected layer output mapping of the Z-noise vectors. W is a higher-level representation of the images [13,57]. $W+$ is an improved (extended) intermediate latent space, in which a distinct 512-dimensional w vector is assigned to each layer of the generator [57,58]. The intermediate latent space W ($W+$) contains information about **"style" attributes** of the generated image such as texture, color, and object shape [58]. Last, the Style Space S represents the high-dimensional space of channel-wise style parameters, derived through distinct affine transformations applied to each layer of the generator [57,59,60]. Compared to the other intermediate latent spaces, S demonstrates a significantly higher degree of disentanglement [57]. Therefore, it is the most suited latent space to represent styles. After training is completed, the model is then capable of generating data in a sampling process; that is, by mapping into the learned latent space [57]. Style vectors within this space can manipulate aspects of the generated images, such as lighting, color schemes, textures, and object appearance, while maintaining the overall structure defined by the W-latent space.

ADA is a mechanism which allows StyleGAN to make adjusted data augmentation "on the fly" during training. This technique helps to improve the quality and diversity of the generated images by avoiding overfitting of the discriminator when training on limited datasets [17].

2.3.2. StyleGAN3 transfer learning

Former research has shown that TL can have a positive effect on the generative models' ability to adapt features from the real image to the generated outputs earlier in the training process, especially when the subject of the data used to train the previous model and the new data subject share semantic similarities [61]. Thus, TL has been the go-to method for cases with limited availability of training data. However, it might be difficult to find highly relatable pre-trained models for domain-specific tasks. For the TL experiments, the StyleGAN3 model, pre-trained on the LHQ-256 dataset [62], is employed.

2.3.3. Projected GAN with FastGAN generator

Projected GAN utilizes pre-trained representations to expedite and stabilize training [21]. Following guidance from [21], EfficientNet-Lite1 [63] is employed for the pre-trained feature network, an image classification network trained on ImageNet [64], which has been shown to produce compact representations that achieve lower FIDs and at the same time reduce computational cost and training time. In order for the discriminator to fully exploit features from deeper layers of the pre-trained model, Projected GAN mixes features across channels and scales using differentiable random projections that are fixed [21]. The use of multiple discriminators allows for the provision of multi-scale feedback. Differentiable augmentation methods [65] are leveraged to prevent overfitting in the discriminator. The Projected GAN training objective is expressed by [21]:

$$\min_{\{G\}} \max_{\{D_l\}} \sum_{l \in L} \left(\mathbb{E}_x[\log D_l(P_l(x))] + \mathbb{E}_z[\log(1 - D_l(P_l(G(z))))] \right) \tag{3}$$

where $\{D_l\}$ is a set of independent discriminators that operate on different feature projections. D_l is associated with features from layer l of the pre-trained feature network where L is the total number of layers. The feature projectors P_l, introduced for mapping real and generated images to the input space of the discriminators, are fixed and only the parameters of $\{G\}$ and $\{D_l\}$ are optimized [21]. The consistency theorem of [66] ensures that G matches the generated and true feature space distributions at convergence (i.e. after solving the optimization problem described by Eq. (3)).

For the generator, Projected GAN uses the minimalistic FastGAN architecture [19], comprising several up-sampling and skip-layer-excitation (SLE) blocks. The SLE block, inspired by residual blocks [55], provides a shortcut to gradient flow by applying channel-wise multiplications between the activations and by performing skip connections between resolutions of much longer range [19]. The SLE module is formally defined as:

$$y = F(x_{\text{low}}, \{W_i\}) \cdot x_{\text{high}} \tag{4}$$

where x_{low}, x_{high} are the 8×8 and 128×128 input feature maps, respectively, y is the output feature map, F are the operations on x_{low} and W_i are the weights to be learnt. The structure of an SLE module and the overall generator can be seen in Fig. 2.

The Projected GAN with FastGAN generator has been shown to outperform StyleGAN2-ADA, FastGAN, and Projected StyleGAN2-ADA in a variety of domains in terms of convergence speed, data efficiency, and image quality [21]. It has further shown promising results for training on small datasets and achieving low computational cost [19,21].

2.4. Up-sampling module

The up-sampling module is applied to the images obtained by the floor plan generation module, described in Section 2.3. The up-sampling module is implemented using the Real-ESRGAN architecture [48]. Comparisons are drawn between the proposed approach and a baseline method, namely SRGAN [67], in terms of up-sampled image quality. Given that the framework proposes the up-sampling of generated (synthetic) floor plan samples for which it is inherently impossible to have the ground through (GT) HR images, only a qualitative evaluation of the investigated SR models is conducted. All SR models are fed with synthetic images obtained from the best performing floor plan generation model.

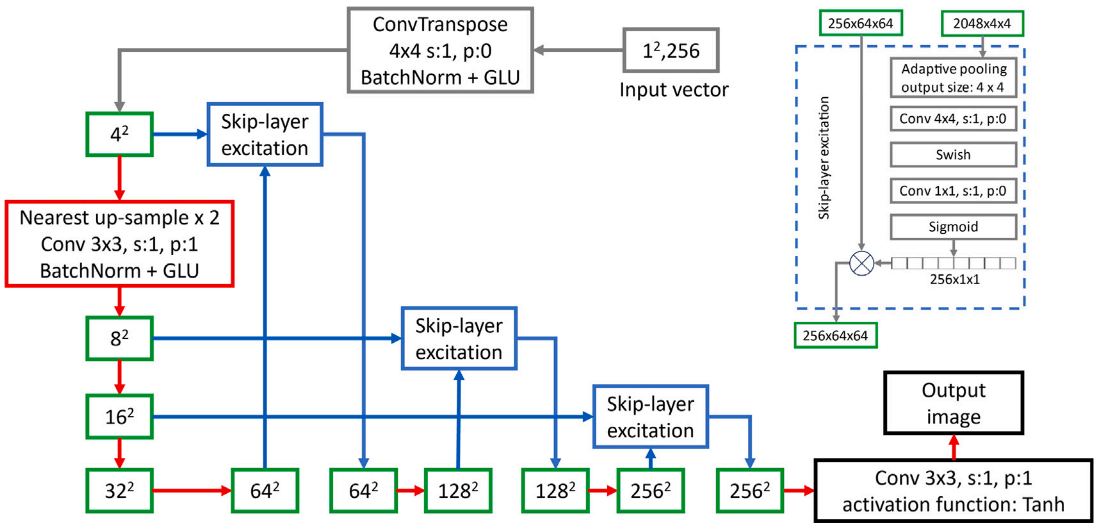

Fig. 2. The generator network, inspired by the original FastGAN generator [19]. Gray represents the initial mapping structure, green boxes indicate the spatial size of the feature maps, red boxes and arrows illustrate the up-sampling structure, and blue boxes contain the skip-layer excitation (SLE) modules. Dashed blue line illustrates the detailed structure on an SLE module.

2.4.1. SRGAN

SRGAN [67] is the first single image super-resolution (SISR) framework, proposed for high (4×) upscaling factors, that is not solely based on the mean squared reconstruction error (MSE) content loss to obtain the solution [67,68]. The generator comprises 16 identical residual blocks, whereas the differentiable discriminator, trained to distinguish real HR images from generated HR samples, consists of eight convolutional layers followed by two dense layers and a sigmoid activation function [67].

To optimize the SR image quality, the perceptual loss, L^{SR}, was proposed. L^{SR} minimizes errors in perceptually relevant characteristics and is given by the sum of two terms: the content loss, L_X^{SR}, and adversarial loss, L_{Gen}^{SR} [67]:

$$L^{SR} = \underbrace{L_X^{SR} + 10^{-3} L_{\text{Gen}}^{SR}}_{\text{perceptual loss}}. \tag{5}$$

The content loss is defined as the Euclidean distance between the high-level visual geometry group (VGG19) [69] feature maps of the reconstructed and reference images, whereas the adversarial loss component pushes solutions towards the natural photo-realistic image manifold through the discriminator network.

2.4.2. Real-ESRGAN

The Real-ESRGAN [48] is built on the enhanced SRGAN (ESRGAN) [70]. ESRGAN has shown great results by adopting a more complex architecture, which generates more realistic and natural textures in the up-sampling process. The three main improvements of the ESRGAN architecture over SRGAN are:

1. Higher capacity and easier training by replacing the original basic block structure of the generator with 23 residual-in-residual dense blocks (RRDBs). Furthermore, all batch normalization (BN) layers have been removed to enhance performance and generalization ability, stabilize training, and lower the memory usage and computational complexity for both SR and deblurring tasks. In addition, residual scaling and smaller initialization are leveraged to improve training.

2. The discriminator has also been modified to a relativistic discriminator utilizing the relativistic average GAN (RaGAN) structure [71], enabling it to predict relative realism between real and generated images rather than the absolute realism.

3. The perceptual loss (L_{percep}) is improved by using the VGG19 [69] network feature space before the activation layers, instead of after as employed in the SRGAN architecture [67], which strengthens the supervision for brightness consistency and texture recovery [70]. The total loss for the ESRGAN generator (total loss$_{G_{\text{ESRGAN}}}$) is expressed as:

$$\text{total loss}_{G_{\text{ESRGAN}}} = L_{\text{percep}} + \lambda_{bal} L_G^{\text{Ra}} + \eta_{bal} L_1 \tag{6}$$

where L_{percep} denotes the VGG loss, L_G^{Ra} denotes the adversarial loss, $L_1 = \mathbb{E}_{x_i} \|G(x_i) - y\|_1$ is the content loss, and λ_{bal}, η_{bal} are the balancing coefficients.

The Real-ESRGAN [48] further extends the above ESRGAN [70] capabilities, by enabling blind SR capable of reconstructing from unknown and complex real-world degradations. While the generator stays the same, the VGG-style discriminator has been modified to a U-Net design with skip connections for handling a significantly larger degradation space than the one that the ESRGAN is capable of [48]. Spectral normalization (SN) is employed as a regularization technique to improve restored textures and stabilize the training of the discriminator [48].

For ESRGAN, it is assumed that degradation noises are applied to images once, while real-world images can, in fact, undergo a series of degradations. To model more practical real-world degradations (such as blur, general noise (Gaussian, Poisson, Color, Gray), and JPEG compression), a high-order degradation process is proposed for Real-ESRGAN [48]. Also, a 2D-sinc filter is applied to represent ringing and overshoot artifacts. Training Real-ESRGAN on pure synthetic data has resulted in better visual performance in restoring real-world images, making it more practical than ESRGAN [48].

2.5. Floor plan generation evaluation framework

2.5.1. Fréchet inception distance

The FID score is commonly used as an evaluation metric for the quantitative assessment of the quality of the images generated by a model [16,23]. It computes the distance between the distributions of real and generated images by embedding them to a vision-relevant feature space [16,72]. FID is typically based on an ImageNet [64] pre-trained Inception-V3 classifier network [16,73]. The FID metric is expressed as:

$$\mathrm{FID} = \|\mu_X - \mu_Y\|_2^2 + \mathrm{Tr}\left(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y}\right) \tag{7}$$

where X, Y represent the real and synthetic image feature vectors, respectively, $\|\mu_X - \mu_Y\|_2^2$ denotes the squared Euclidean distance between the mean vectors, and $\mathrm{Tr}(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y})$ represents the trace of the sum of the two covariance matrices minus twice the matrix square root of their product. A low FID score indicates high quality between the 50k generated images and the real image dataset [13].

2.5.2. Density and coverage metrics

The FID score is a one-value metric encompassing both the fidelity and diversity aspects of the generated image quality. Therefore, previous studies have focused on separating FID into two distinct metrics, namely precision and recall, that are representative of these two aspects of generative model performance [74–76].

However, precision and recall fail when the real and fake distributions are identical, and also in the presence of outliers and certain mode droppings. They also do not allow for systematic selection of hyper-parameter values [24]. In this study, the density and coverage metrics are employed as a more reliable method to tackle the above matters [3]. Density is representative of fidelity, defining how closely the model resembles the real data. Coverage, on the other hand, represents diversity, measuring how well the generated samples cover the variety of the real dataset [24]. On top of this, the density and coverage metrics tackle the introduced bias by the ImageNet pre-trained VGG16 model [76], which may limit fair evaluation. In cases where the dataset distribution is distinct from ImageNet, random embedding is introduced as a better choice [24].

The density function is an improvement of the precision metric [76], where the overestimation of the manifold around real outliers has been fixed by counting how many real-sample neighborhood spheres contain fake samples Y_f. The density is calculated as:

$$\mathrm{density} = \frac{1}{kM_{fake}} \sum_{f=1}^{M_{fake}} \sum_{r=1}^{M_{real}} I_{Y_f \in B(X_r, \mathrm{NND}_k(X_r))} \tag{8}$$

where X_r are the real samples, k denotes the number of nearest neighbors, $NND_k(X_r)$ denotes the distance from X_r to the kth nearest neighbor of the remaining X_r, $B(X_r, \mathrm{NND}_k(X_r))$ represents the neighborhood spheres around each real sample X_r, M_{fake} and M_{real} are the numbers of fake and real samples, respectively, and I is the indicator function.

Coverage enhances the recall metric [76] by computing the nearest neighbor manifolds around real samples, which are expected to have fewer outliers than fake samples. It is defined as [24]:

$$\mathrm{coverage} = \frac{1}{M_{real}} \sum_{r=1}^{M_{real}} I_{\exists f \text{ where } Y_f \in B(X_r, \mathrm{NND}_k(X_r))} \tag{9}$$

where the function $I_{\exists f}$ returns 1 if there exists at least one fake sample Y_f within the neighborhood sphere of real sample X_r.

2.5.3. CNN detection score

The generated floor plan images should be indistinguishable from the real data. A convolutional neural network (CNN)-based detection score is employed for assess the floor plan generation module's ability to fool a SOTA classifier from a visual perception perspective. Previous literature has also relied on CNN classification for distinguishing synthetic samples generated by GANs from real images [77–79].

In this study, a binary CNN classifier based on Inception V3 [73] is employed. The GoogLeNet [80] Inception V3 [73]-based CNN architecture is widely used for visual machine learning tasks [73]. The model is trained and tested on equal proportions of synthetically-generated (fake) and real data. The accuracy is reported, calculated as follows:

$$\mathrm{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}. \tag{10}$$

Low accuracy values would indicate that the classification model is confused about the truthfulness of fake data, and thereby the GAN floor plan generator performs well.

2.5.4. Carbontracker

The Carbontracker tool is a method for evaluating the energy consumption and the associated carbon dioxide emissions in deep learning experiments [81]. It has been utilized by previous work on developing low emission deep learning models [13,82,83]. The total energy consumption (EC) is calculated based on the formula:

$$\mathrm{EC} = \mathrm{PUE} \sum_{e\in E} \sum_{c\in C} P_{\mathrm{avg},de} T_e \tag{11}$$

where PUE stands for power usage effectiveness and is a ratio parameter characterizing a data center's efficiency and the energy overhead of computing equipment, $P_{avg,de}$ denotes the average power consumed by device $c \in C$ in epoch $e \in E$, and T_e is the duration of epoch e.

The carbon footprint (CF) for training a model is then calculated by converting the monitored EC using the formula:

$$\mathrm{CF} = \mathrm{EC} \times \mathrm{CI} \tag{12}$$

where CI is the carbon intensity. There is some variation in the CI calculation depending on the source or region where the electricity comes from. To better appreciate the CF generated during training, the equivalent distance traveled by an average car is calculated and reported, using a rate of 107.5 g of $\mathrm{CO_2}$eq/km [81].

2.6. Validation framework

After assessing the HR synthetic floor plan designs produced by the up-sampling module, a validation of the overall proposed framework would be beneficial. To ensure that the computerized framework performs as intended and can be used to achieve business objectives, validation is performed in terms of contextual functionality and data privacy. To showcase the capability of the framework to be manipulated to suit the designer's desires, and its usability in a downstream task, namely a software-based building information modeling (BIM) implementation task is conducted. Lastly, driven by the intended use of the proposed framework, a validation test is performed to assess the fast delivery of the generated HR floor plan images. Another goal of the proposed validation framework is to identify potential errors. The validation is retrospective and is applied to floor plan images produced by the best-performing floor plan generation and image up-sampling GAN model stack. It is also independent, meaning that the input images for validation are different from the ones that had been used for model evaluation.

2.6.1. Contextual functionality

The main objective of this study is to generate floor plan images for architectural building design processes where the generated content is intended to support: (i) the early design stage, and (ii) design identification between clients and architects. Hence, quality standards should reflect the "diversity of human demands" in a context-based manner. Driven by the above objective and the intended use of the framework, a manual validation test is performed on the generated samples concerning contextual functionality.

Such a validation test also reflects the proposed framework's practical usability in the context of architectural floor plan design, allowing one to assess how well each generated sample has adopted good design practices from the input data. These practices are influenced by local building codes and regulations, as well as structural, energy, and sun- and wind-exposure considerations. To make this contextual functionality validation possible, the authors conducted semi-structured interviews with experts in Danish building architecture.

Inspired by the work of Park et al. [2], where over 80,000 real and synthetic floor plans were qualitatively assessed based on architectural rules-of-thumb, the key elements of that method are adapted for the contextual functionality validation of the generated floor plans. The contextual functionality and applicability of each analyzed floor plan image were gauged by all authors of this study, taking into account the holistic synthetic floor plan assessment guidance by Weber et al. [29], national guidance [84], and domain-specific knowledge obtained from interviews with experts in Danish architectural building design. The contextual functionality criteria that are considered for validation are presented in Table 2. The number of misplaced or missing doors and the number of windows with defects in each analyzed floor plan were also counted. Fig. 3 shows examples of good and bad door and window placement, with reference to a real floor plan.

Although the dataset does not distinguish between building age, the layout planning in a Danish context can generally be divided into three design paradigms from the period of 1950 to today. Houses in the period from 1950–1960 were designed with rooms centered around the building's chimney, which was the dominant heat source of that time, in a cruciform deformation to exploit the heat in all the rooms of the building. Later, in 1960–1980, the floor plan layouts became larger and rooms were designed to benefit from the central heating system, where radiators were placed under the windows in almost every room. The rooms were typically connected through a pistol-shaped corridor, a characteristic of the building plans at the time. Contemporary plan layouts are characterized by an open area kitchen and dining room, with minimized corridor areas to optimize space, and the floor heating systems (which most modern houses are equipped with) offer more freedom in the room design shapes. However, the size of buildings has been constantly increasing over time [85].

Since the proposed framework aims to support the early design stage with automatically-generated HR floor plan layouts, on which further design development can be based, it is important that the generated layouts support the floor plan development in the right direction to be useful in that process. The average contextual functionality quality (Q_{context}) for each validation criterion is calculated by visual human evaluation of a generated sample of plan layouts using the formula:

$$Q_{\mathrm{context}} = \frac{1}{N_{Gen}} \sum_{i=1}^{N_{Gen}} Q_i \tag{13}$$

Table 2
Contextual functionality validation criteria.

Structural design
• Minimization of the open area spans to reduce structural design issues.
• Presence of load-bearing and shear walls to bring stability to the construction.
Energy
• Minimization of the building's energy consumption by reducing outer walls and placing fewer conditioned zones with buffer zones in the conditioned rooms.
• Balancing the access to daylight throughout the building layout with optimal space allocation.
Space optimization
• Minimization of the wasted space in layout planning, which is compatible with adaptable and modular layouts.
Layout relations
• Ensuring proper room connections.

where N_{Gen} is the number of generated images, and Q_i is the quality assessment of an individual floor plan image for the specific validation aspect (with permissible Q_i values: 2 = Excellent, 1 = Good, 0 = Insufficient).

2.6.2. *Data privacy with perceptual image hashing*

The existing GAN-based autonomous design generation literature is not aimed at detecting overfitting that could lead to violations of intellectual properties (IPs). However, copying data from the training set is a highly important challenge for generative models. It is necessary to ensure that these models do not obtain high-quality scores by merely outputting data similar to the input data [86]. To demonstrate the proposed framework's capability to preserve privacy in floor plan generation, robust image hashing is utilized, where the generated floor plans are checked against the input data [87]. Image hashing is the most common data-copy detection method [88–90] and has been previously used to investigate if the outputs generated by GANs suffer from overfitting [91].

Image hashing allows the mapping of the perceptual image content into bit strings for content-based indexing [90,92] by extracting content-based features. For perceptual hashing, a popular discrete cosine transform (DCT)-based technique (pHash) [87,89] is utilized to transform images into their frequency components [93].

For evaluating image similarities, the Hamming distance (HD) is employed, which has been shown to be efficient [90,91,94]. The normalized HD (NHD) [92] enables the use of different hash types. After hashing reduces the images to bitstrings, HD can be computed as follows [90]:

$$\mathrm{HD}(h, h') = \sum_{i=1}^{S} I(h_i - h'_i) \tag{14}$$

where h_i, h'_i are the real and generated image bitstrings, respectively,

$$I(h_i - h'_i) = \begin{cases} 0, & h_i = h'_i \\ 1, & h_i \neq h'_i \end{cases}$$

denotes the indicator function, and S represents the total number of samples.

2.6.3. *Latent space manipulation with image morphing*

Another aim is to confirm that the framework can be manipulated to suit the designer's needs. To this end, the approach leant on image morphing. Image morphing is an established technique within computer graphics and computer vision for image manipulation [95]. To drift the style, appearance, and size of floor plans, latent code interpolation is performed between the respective latent vectors of embedded images [95,96]. By adding a linear combination of the latent vectors w_1 and w_2 of two embedded images, a subsequent image can be generated as a new outcome w. Cross-domain image morphing is also possible [20].

2.6.4. *Downstream BIM implementation*

The framework's usability is showcased in a downstream task, namely a BIM software-based implementation task. Resketching the generated HR floor plan using a BIM software environment offers the opportunity to make the necessary design adjustments and to launch the layout planning of room placement. The BIM design could benefit from the automated generated floor plan with a fast and straightforward manual transformation of the raster image into the spatial BIM format.

2.6.5. *Framework inference speed*

The intended use of the proposed framework is to be able to provide high-quality products in a rapid way even in low-resource regimes. Therefore, the average latent space sampling time is calculated for the two GAN-based modules of the framework for a sequence of 10,000 generated floor plan images.

2.7. *Implementation*

In this study, all implementations were run on a commercial NVIDIA RXT-3090 GPU in a Linux-based Anaconda operating system.

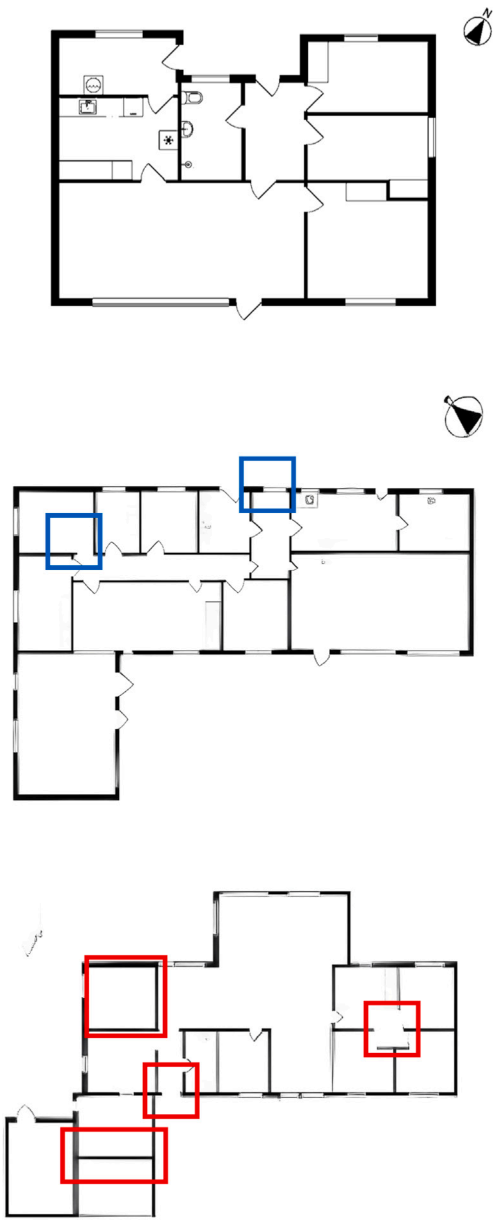

Fig. 3. Top: A real floor plan image. Center: An example of a relatively good, generated floor plan with very few door and window misalignments, marked with blue boxes. Bottom: An example of a bad generation with several door, window (and even wall) missing details, marked with red boxes.

2.7.1. Pre-processing module

The denoising ITRLE model is a local running Python script for processing the full dataset automatically. To optimize the trade-off between intended and unintended content removal from the floor plan dataset, an inference threshold value of 0.4 was chosen [36]. After the denoising process, the data were manually examined for errors. Any damaged data, in which the automated pre-processing had produced either text leftovers or missing image content, was scraped or reworked.

Next, any non-square images were squared up (1440 × 1440 pixels) by centering them into a square white background. These were then converted to black and white images using the OpenCV library's "COLOR_BGR2GRAY" function [97], where a threshold value of 180 was set using "cv2.threshold" and the type "cv2.THRESH_BINARY", with 255 assigned to pixels exceeding the threshold. The next step was to resize the images to 1024 × 1024 pixels for processing purposes. Finally, all (4×) down-sampling methods were implemented by running a local Python script based on OpenCV libraries [53].

Table 3
Hyperparameters for the end-to-end StyleGAN3 training experimental setup.

Type	Hyperparameters
Fixed	`--cfg=stylegan3-t, --gpus=1, --aug=ada (0.6), --cmax=256, --kimg=10000`
Variable	`--batch, --batch-gpu, --gamma`

Table 4
Tunable hyperparameter values for the six end-to-end trained models.

Model #	1	2	3	4	5	6
--gamma	2	2	32	32	32	32
--batch size	32	32	32	32	12	16
--batch gpu	16	4	16	4	4	16

Table 5
Real-ESRGAN fine-tuning hyper-parameter values.

Parameter	Value
Optimizer	Adam
Batch size	8
Lr	1e−4
Scale	4
Network_g	RRDBNet
Beta 1, 2	[0.9, 0.99]
Network_d	UNetDiscriminatorSN
GAN loss	Vanilla
Use_shuffle	True
Patch_size	24

2.7.2. Floor plan generation module

The StyleGAN3 implementation was based on the official PyTorch GitHub release [13]. The dataset preparation was carried out using the “dataset_tool.py” module [13]. The floor plan generation was processed by the built-in StyleGAN3 script “gen_images.py”, which was customized to generate batches of randomly generated floor plan layouts and to plot the sampling time. Different hyperparameter value combinations were explored with reference to the proposed StyleGAN3 training configuration values [50]. A series of six experiments was conducted for the end-to-end training of the StyleGAN3 floor plan generator on a set of different fixed and variable hyperparameters (Tables 3 and 4). For training of models 1–6, ADA was activated with the default target value of 0.6 [17]. Every model was trained for 10,000 kimg of iterations, resulting in approximately one week of training per model. This training duration was chosen to ensure a progressive and comprehensive flow of the experiments. All six models were evaluated based on the FID score and training time. The two best performing models were selected for a more comprehensive evaluation covering generated floor plan image fidelity, diversity, and truthfulness, as well as resource consumption and environmental impact.

To assess the benefits of training with TL on pre-trained models for the floor plan generator, a further experiment (model 7) was run, based on the hyperparameter set of one of the best-performing StyleGAN3 end-to-end models. The StyleGAN3 LHQ-256 model was trained with the first four layers of the discriminator frozen. The generated floor plan image outputs of the TL model were comprehensively assessed as described above.

To evaluate the impact of using Projected GAN with the FastGAN generator for floor plan synthesis, three additional models were trained. The first model (model 8) was trained using the proposed hyperparameter settings from the original implementation (batch = 64, batch-GPU = 8) [27]. In the second model (model 9), both the batch and batch-GPU sizes were halved (batch = 32, batch-GPU = 4). The third model (model 10) used the hyperparameter values of the best-performing StyleGAN3 end-to-end training experiment. The performance of all three models was thoroughly evaluated.

2.7.3. Up-sampling module

The Real-ESRGAN was fine-tuned on the domain-specific paired datasets comprising 984 LR and HR image pairs. The former comprises the down-sampled images by using the AREA method (pre-processing module), whereas the latter are the pre-processed images just before the down-sampling occurred. One model was trained for 30.5k iterations and another for 372.7k iterations. The implementation parameters are given in Table 5.

An initial comparison was made between the pre-trained Real-ESRGAN model, proposed in the original paper [48], and the custom-trained (fine-tuned) Real-ESRGAN. The pre-trained Real-ESRGAN model adapted many of ESRGAN’s [70] settings. It had been trained with four NVIDIA V100 GPUs on data from the DIV2K, Flickr2K, and Outdoor Scene Training (OST) datasets for 400K iterations with $lr = 1 \times 10^{-4}$, using a combination of L1 loss, perceptual loss [70,98] and GAN loss [48]. The Adam optimizer with the settings $\beta_1 = 0.9$, $\beta_2 = 0.999$ had been used.

The baseline SRGAN up-sampling model was fine-tuned on the same custom data as Real-ESRGAN. A 4× upscaling factor was attempted to output a final resolution of 1024 × 1024 pixels for the generated architectural floor plans. The pre-trained SRGAN

Table 6
SRGAN fine-tuning hyper-parameter values.

Parameter	Value
Optimizer	Adam
Batch size	16
lr	1e−3
Scale	4
Patch size	24

had been trained with an NVIDIA Tesla M40 GPU, on a sample of 350 thousand random images from the ImageNet database [64]. The LR dataset had been obtained by using the bicubic kernel for down-sampling on the HR dataset with a factor of 4. The model had been further trained with an $lr = 10^{-4}$ for 10^5 update iterations and an $lr = 10^{-5}$ for another 10^5 iterations, where the Adam optimizer had been utilized with $\beta_1 = 0.9$ [67]. The training hyper-parameter values are shown in Table 6.

2.7.4. Floor plan generation evaluation framework

For reporting FID values, the FID50k metric is utilized; that is, 50k generated test images are compared with the real dataset. To realize the metric, the "calc_metrics.py" module was used, which is available in both the StyleGAN3 and the Projected GAN with FastGAN generator implementations.

The density and coverage metrics were implemented using code from [99], which builds on the original implementation by [24]. For the calculations, the real LR dataset was compared to a dataset of 10k synthetic images generated for each evaluated model. The nearest neighbor parameter values $k = 2$ for the density metric and $k = 1$ for the coverage metric were utilized.

To estimate the detection score of each generative model, the Inception-v3 [73] CNN was trained for 50 epochs utilizing early stopping. For the training, a custom dataset containing 1000 generated images and 988 real images is utilized. An 80/20 data split was made for the training/testing dataset, and accuracy was reported on the testing set.

Carbontracker was implemented using code from the GitHub repository [100]. The tracker was embedded to the training code of both the floor plan generation and up-sampling networks. Ten iterations were measured and the total energy consumption for the full training time was calculated based on the average CPU and GPU power values. Following a CI value of 149.75 CO_2/kWh was employed, corresponding to Danish data from 2020 [81].

2.7.5. Validation framework

For the contextual functionality validation, a sample of 25 out of 10,000 randomly generated and up-sampled floor plans by the best performing GAN-stack pipeline was selected. The reported values for all criteria are the mean opinion scores assigned by the list of authors.

For computing and comparing the image-hashes, the method by [87,101] was employed. To detect any similarities between the two datasets (i.e., HR-real and SR-fake), 984 real HR floor plans and 25 generated floor plans were used. Additionally, a function was applied to obtain 12 positions in each real image by rotating the images by 90°, 180°, and 270°, and by flipping the image positions horizontally and vertically to make the implementation more robust. This process provided 295,200 data rows for analyzing the diversity in real and fake images, aiming to verify the framework's capability to generate unique floor plan designs. The pHash relied on the DCT technique for generating 6-bit hash strings for each real and fake image, which were then compared using the HD metric [101]. It was implemented using code from the Imagehashing library [101].

3. Results

3.1. Pre-processing module

3.1.1. Dataset denoising

After scraping the damaged data (i.e., images with text leftovers or missing image content) of the denoising process, the dataset size was reduced from 1233 to 984. Fig. 4 illustrates the LR generated floor plans when training the StyleGAN3 generator with and without text denoising. A visual assessment of Fig. 4 clearly emphasizes the need for removing text content, since the generated image text in the floor plan (A) is unrecognizable and appears as noise that will most likely impact further post-processing through the framework. Thereby this result strongly advocates for removing text from the input dataset due to the GAN model's poor ability to imitate text.

3.1.2. Dataset down-sampling

Fig. 5 shows the results of the four tested four-fold down-sampling methods, as well as the HR input image with a resolution of 1024 × 1024 pixels. Down-sampling the images to 256 × 256 pixels with the AREA module minimized the loss of details compared to the other three methods, and the results appear less blurred and have a better representation of the geometry compared to the other methods. The bilinear and bicubic down-sampling methods tend to have a relatively high loss of details such that the image details appear very crude.

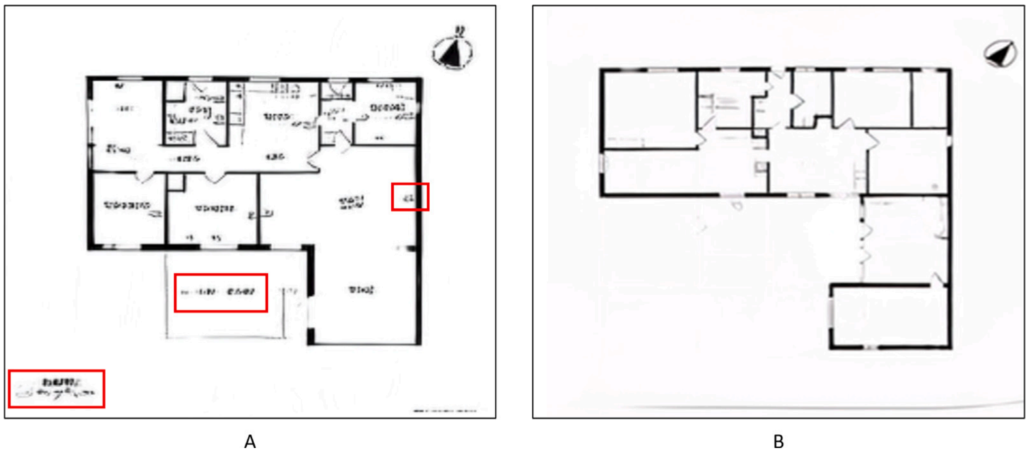

Fig. 4. Generated floor plans when StyleGAN3 was trained (A) on the LR version of the original dataset at batch size 12, and (B) on the denoised and downscaled dataset at batch size 12. Red boxes indicate locations of generated unrecognizable noise.

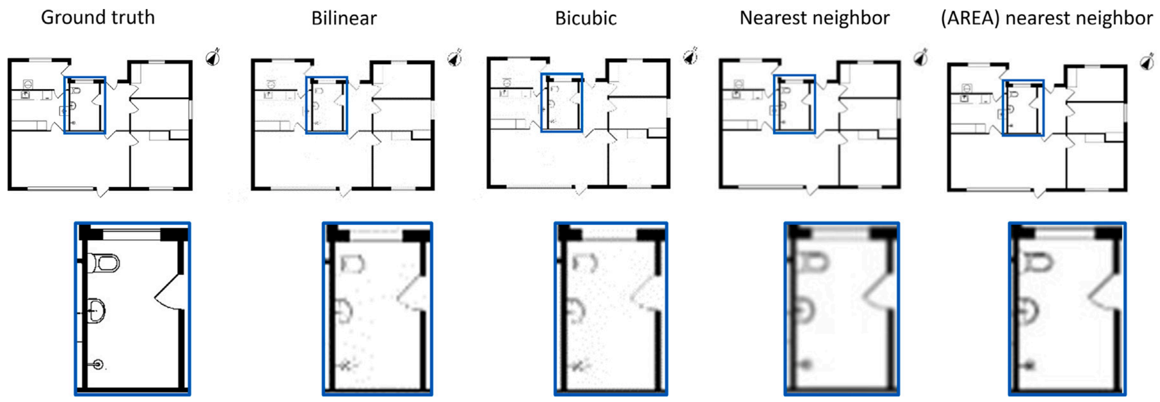

Fig. 5. Results of the down-sampling methods in the dataset pre-processing. The images are resized from 1024 × 1024 pixels to 256 × 256 pixels. Top: The complete floor plans. Bottom: Zoomed-in view of the areas inside the blue boxes.

Table 7
Overview of results (FID, training time, iterations) for end-to-end StyleGAN3 training. Boldface indicates best performance.

Model #	FID	Time [h/m]	Iterations [kimg]
1	88.9	146 h 58 m	10,000
2	101.7	112 h 14 m	7,008
3	30.4	**103 h 40 m**	7,008
4	55.4	160 h 00 m	10,000
5	**19.1**	112 h 13 m	**7,005**
6	19.3	147 h 53 m	10,000

3.2. Floor plan generation module

3.2.1. StyleGAN3 end-to-end training

The results of the end-to-end training are shown in Table 7. Model 5 achieved the lowest FID score (19.1) with the shortest training time (≈112 h). Model 6 ranked second, with an FID score of 19.3 after ≈148 h of training. Although the lower batch size and batch GPU in experiment 5 resulted in a longer training time per kimg compared to model 6, the image quality generated by model 5 improved much faster than that of model 6.

Fig. 6 shows a plot of the model performance (calculated FID score) as a function of kimg (total number of training iterations) for the end-to-end StyleGAN3 training experiments. The graph is based on four FID data points for each experiment, computed at 1k, 4k, 7k, and 10k of kimg. It can be seen that increasing the value of the regularization weight gamma from 2.0 (models 1 and 2) to 32.0 (models 3–6) made the performance improve rapidly. In addition, using a batch size smaller than 32 exhibited a lower

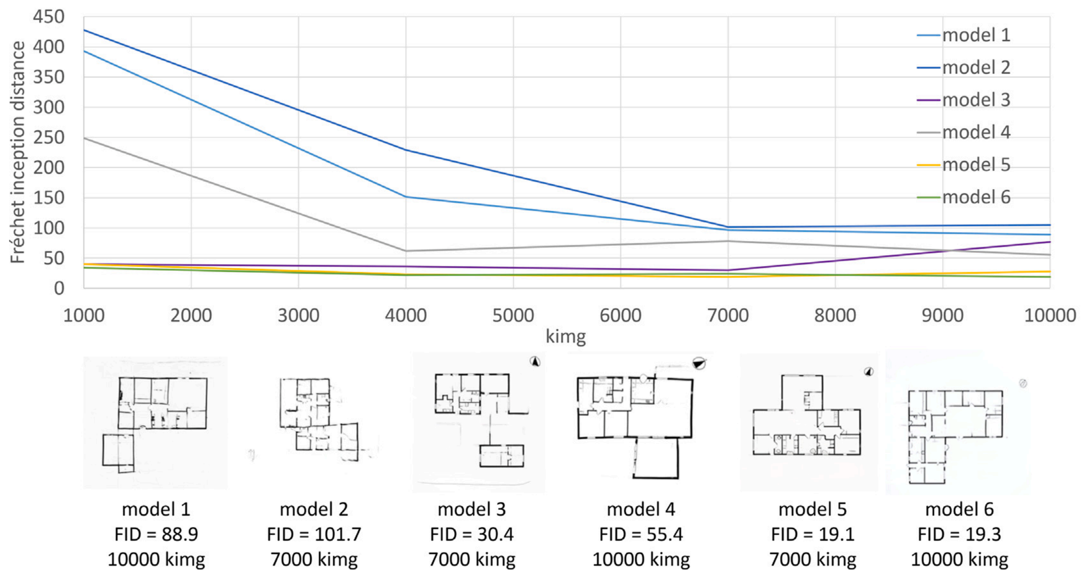

Fig. 6. End-to-end StyleGAN3 training results (left), and representative floor plan images generated at the minimum FID timepoint (right) for the six models.

Table 8
Overview of results (FID, training time) for StyleGAN3 transfer learning (model 7).

Kimg	0.2k	1k	4k	7k	10k
FID	22.2	20.8	56.8	62.5	75.1
Time	4 h 30 m	21 h 44 m	86 h 13 m	151 h 03 m	215 h 32 m

FID score after 1k kimg and showed a relatively flat learning curve over the 10k kimg iterations. Also shown in Fig. 6 (right) are representative floor plan images generated at the minimum FID timepoint for the six models. The visual assessment is in agreement with the calculated FID scores. Models with higher gamma combined with a lower batch size tend to generate output data that is more representative of the input data. In summary, models 5 and 6 have minor differences in performance. Model 5 has a higher processing time per iteration, but it achieves a lower FID score much earlier than model 6. Both models 5 and 6 appear to have reached their optimum level, and they would most likely have started collapsing like model 3 if the training process had continued.

3.2.2. *StyleGAN3 transfer learning*

The results in Table 8 show that utilizing TL for StyleGAN3 (with hyper-parameter values: batch size = 16, batch GPU = 16, gamma = 32) helped the model achieve faster training progress by adapting the weights from the pre-trained model. TL based on LHQ-256 reached an FID level of 20.8 after 1k kimg iterations (22 h of training). However, even though TL provided faster FID development compared to the end-to-end training of the StyleGAN model, a model collapse was observed, resulting in a decrease in model performance. The training adaptation process of the StyleGAN3 LHQ-256 pre-trained model is shown in Fig. 7, where it can be clearly seen how the model transitions from the pre-trained landscape domain to the style domain of floor plans.

3.2.3. *Projected GAN with FastGAN generator*

The results of the Projected GAN with FastGAN generator training are outlined in Table 9. It was observed that reducing the batch and batch-GPU sizes by half slightly improved the FID performance. Model 10, which used the hyperparameter values of model 5, achieved an FID of 9.3 after 106 h 54 m of training time. Fig. 8 displays both the FID vs. kimg graph (left) and the generated images in the best performing iterations (right). Even though the generated image quality performance of the three implemented models is quite similar, model 9 converged faster in relation to FID. Both models 9 and 10 have a significantly faster learning curve compared to model 8.

3.2.4. *Comprehensive comparative analysis of floor plan image generation models*

The graph in Fig. 9 (top) aggregates the best-performing models (based on FID vs. training time) of the three utilized training approaches. It can be seen that the Projected GAN with FastGAN generator-based models have superior performance compared to the other models. In comparison to the StyleGAN3 end-to-end trained models, the FID score was improved from 19.1 in model 5 after 112 h of training to 9.4 after 59 h of training in model 9.

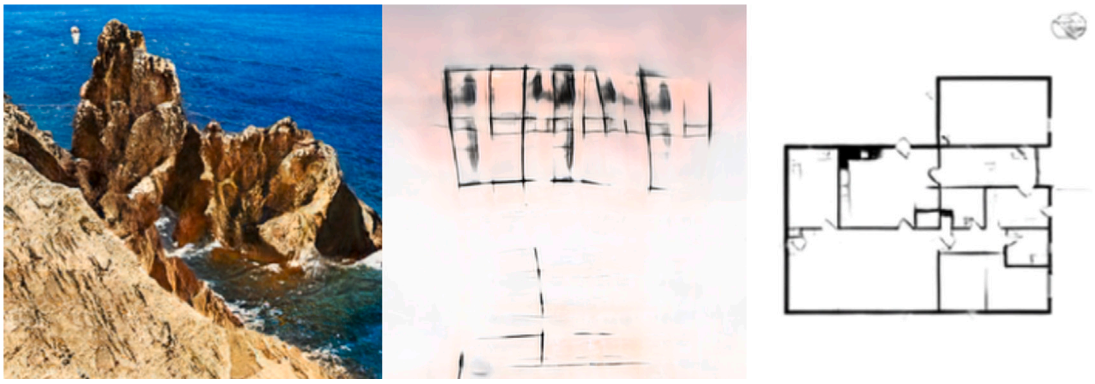

Fig. 7. A representative example of a generated floor plan image with StyleGAN3 transfer learning. Left is the initial result of the LHQ-256 model, center is the result after 16 kimg, and right is the result after 1000 kimg (FID = 20.8).

Table 9
Overview of results (FID, training time, iterations) for Projected GAN with FastGAN generator training (models 8–10). Boldface indicates best performance.

Model #	FID	Time [h/m]	Iterations [kimg]
8	9.8	85 h 59 m	9,475
9	9.4	**58 h 30 m**	**5,400**
10	**9.3**	106 h 54 m	8,416

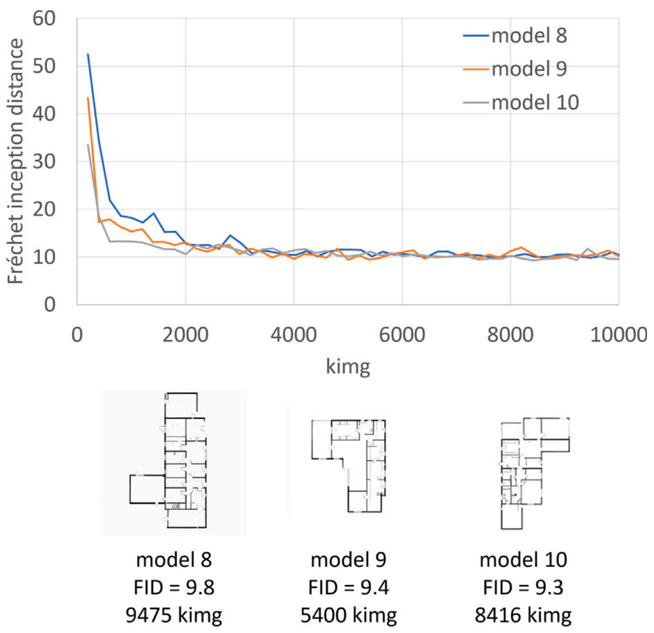

Fig. 8. Projected GAN with FastGAN generator training results (left), and representative floor plan images generated at the minimum FID timepoint (right) for the three models.

In reference to the StyleGAN TL training, the TL model achieved its minimum FID score of 21 after 22 h of training, whereas model 9 achieved an FID score of 17 after only 4 h of training. Hence, a primary strength of the Projected GAN with FastGAN generator is the low FID score, which is not attained by other methods, even when extending their training time.

From a visual inspection point of view, the anticipated quality gap in the generated floor plan images by the various models (Fig. 9, bottom), as indicated by the FID differences, was not observed.

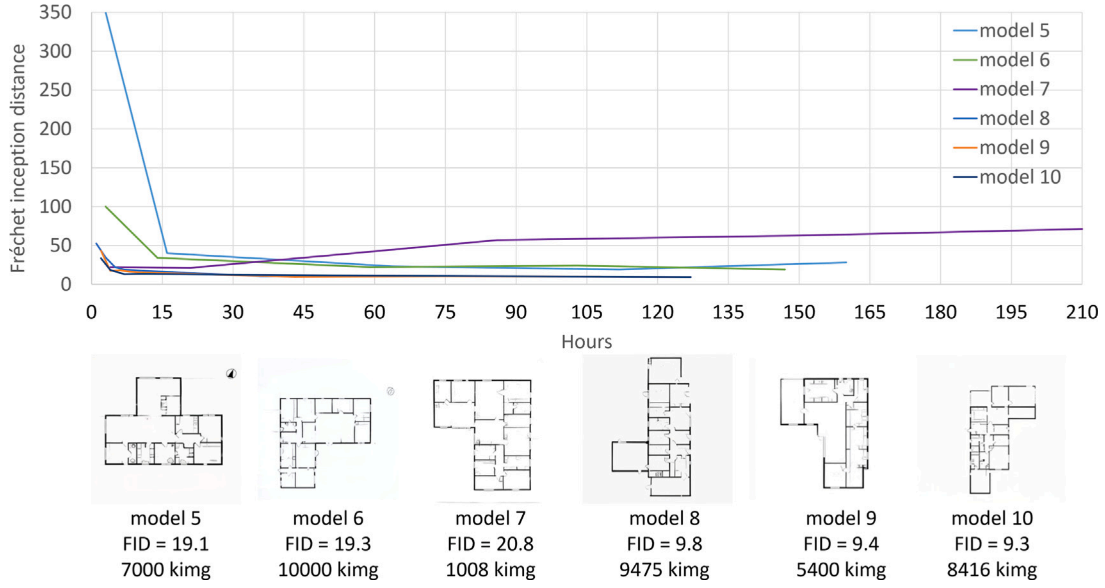

Fig. 9. Comparative analysis of training performance (top), and representative floor plan images generated at the minimum FID timepoint (bottom) for all explored models.

Table 10
Holistic floor plan generation model performance evaluation. Boldface indicates best performance.

Model	FID	Density/Coverage (ImageNet)		CNN detection score (GoogLeNet)			Training resource consumption			Energy consumption		
							Iterations	Time	VRAM	Energy	CO_2eq	Travel
#		Density	Coverage	TP/FP	FN/TN	Acc.	[kimg]	[h/m]	[GB]	[kWh]	[g]	[km]
5	19.1	0.75	0.88	170/22	30/178	0.87	7,005	112 h 13 m	**1.9**	48.31	7,235	67.3
6	19.3	0.92	0.95	174/28	26/172	0.87	10,000	147 h 53 m	7.0	63.54	9,516	88.5
7	20.8	0.69	0.90	164/29	36/171	0.84	**1,008**	**21 h 44 m**	10.4	**9.12**	**1,365**	**12.7**
8	9.8	0.94	0.95	146/48	54/152	0.75	9,475	85 h 59 m	7.3	35.67	5,341	49.7
9	9.4	**1.01**	**0.97**	**137/34**	**63/166**	0.76	5,400	58 h 30 m	5.2	23.45	3,512	32.7
10	**9.3**	0.92	0.96	150/54	50/146	**0.74**	8,416	106 h 54 m	5.3	38.18	6,491	60.4

Table 10 gives the holistic quantitative comparison results of the investigated floor plan image generation methods. From the results, a connection between GPU-batch size and duration time per training iteration is seen, where the models with a high GPU-batch seem to achieve a lower FID earlier in the training process. There is also a relatively high difference in the GPU memory requirements during training among the various models. However, all those GPU memory values are nowadays easily available.

The CNN classifier detection scores show that the Projected GAN with FastGAN implementations (models 8–10) achieved by far the lowest accuracy scores, indicating that these models are the best at generating fake outputs with high truthfulness.

The density and coverage results show that the generated floor plan images by the Projected GAN with FastGAN generator-based model 9 achieved the best image fidelity and diversity performance. Moreover, the performance of the CNN classifier shows that model 9 has the highest number of false negative (false) instances and the lowest number of true positives (real) image instances. This demonstrates that the generated floor plan images by model 9 are relatively indistinguishable from the real image data. It also achieved the second-best FID score. The density and coverage metrics have nevertheless been shown to provide a more exhaustive and itemized image quality performance evaluation than the single-number FID, which has received criticism [24].

Furthermore, model 9 notably emitted the lowest carbon dioxide emissions during training in comparison to the other models that were in the same FID range, causing pollution that is equivalent to traveling a distance of 32.7 km by car [81]. This was nearly half the emissions of model 10, which achieved a slightly better FID score, and 63% lower than the emissions of the worst polluter (model 6). Model 9 also required significantly less training time than the other models that achieved an FID score below 10. Therefore, it is concluded that the Projected GAN with FastGAN generator-based Model 9 is the most effective model for floor plan image generation.

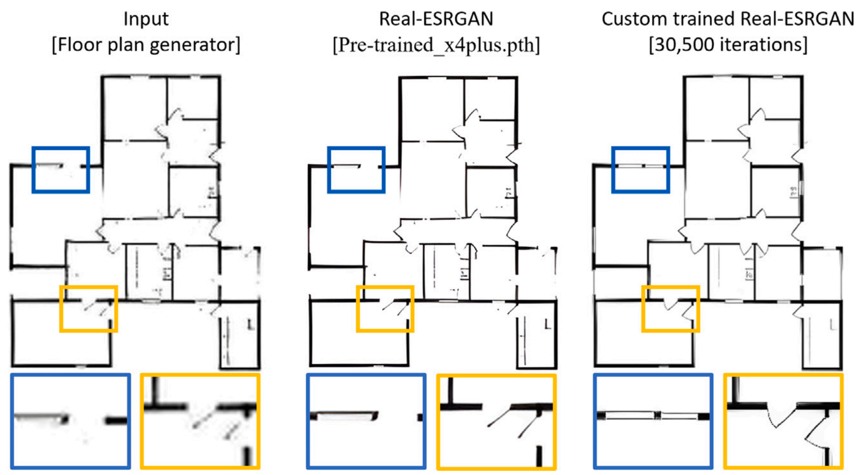

Fig. 10. Comparison of the 4× upscaling process between the pre-trained Real-ESRGAN (2nd column) and the custom fine-tuned Real-ESRGAN (3rd column), based on a generated input image of 256 × 256 pixels (1st column). Top: The complete floor plans. Bottom: Zoomed-in view of the areas inside the blue and yellow boxes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 11
Comparison of SRGAN and Real-ESRGAN (both custom-trained) in terms of training duration, energy consumption and associated carbon dioxide emissions. Boldface indicates best performance.

Model	Iterations [kimg]	Training hours [h]	Total energy [kWh]	Total CO_2eq [g]	Travel [km]
Real-ESRGAN	**30.5**	4.96	2.05	307.64	2.9
SRGAN	31.2	**0.78**	**0.19**	**27.86**	**0.3**

3.3. Up-sampling module

3.3.1. Real-ESRGAN

Fig. 10 shows the up-sampling comparison of the generated (synthetic) floor plans between the pre-trained Real-ESRGAN (for blind-SR) and the Real-ESRGAN fine-tuned on the custom dataset. The visual assessment of the results corroborates that the fine-tuned Real-ESRGAN (for 30.5k iterations) achieved superior results to the pre-trained Real-ESRGAN regarding upscaling quality and restoration of detail errors. The up-sampling by the custom trained Real-ESRGAN has improved features like walls, windows, and door openings, and these features are better represented with fewer missing details.

Fig. 11 shows the results when continuing the Real-ESRGAN model training from 30.5k to 372.7k iterations. Even though continuing training improves the details and general quality of the generated images by suppressing noise and repairing walls, doors, and windows, the model trained at 372.7k iterations begins to remove notable functional features from the floor plans or to hallucinate windows instead of doors.

3.3.2. Baseline comparison

Fig. 12 illustrates that the Real-ESRGAN has far superior results to the baseline custom-trained SRGAN [67]. The outcome appears much sharper in the Real-ESGAN-based results. Table 11 outlines the quantitative comparison between SRGAN and Real-ESRGAN (both custom-trained) in terms of training duration, energy consumption, and associated carbon dioxide emissions. Although SRGAN has lower energy requirements and environmental cost, its performance was deemed insufficient due to poor generated image quality compared to the more complex Real-ESRGAN architecture. Therefore, it is concluded that the custom-trained Real-ESRGAN architecture is the best option for the image up-sampling module.

3.4. Validation framework

3.4.1. Contextual functionality

Fig. 13 illustrates the 25 randomly selected floor plans for the framework validation. Table 12 lists the (individual and mean) numbers of misaligned doors and windows, as well as the contextual functionality scores obtained from the visual evaluation. It was found that the floor plans have an average of 1.32 misplaced or missing doors. The number of windows with defects was 1.82 per floor plan on average. However, it can be seen that the generated floor plans, in general, have performed well in placing windows and doors in contextually meaningful locations from an architectural perspective. For instance, windows are typically placed in

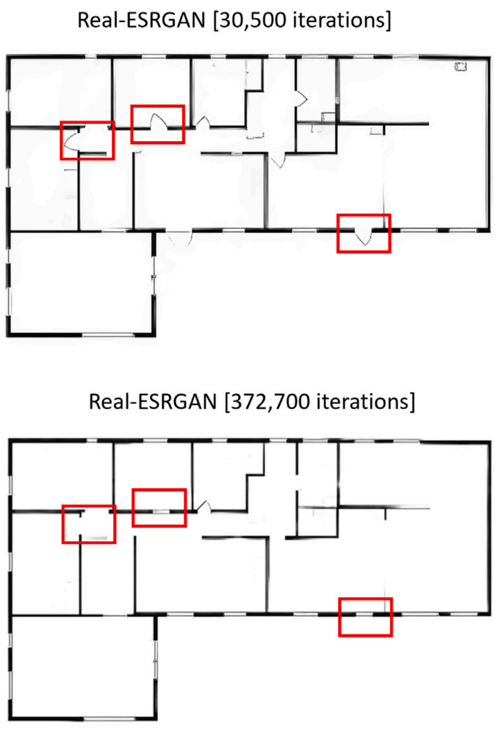
Real-ESRGAN [30,500 iterations]
Real-ESRGAN [372,700 iterations]

Fig. 11. Real-ESRGAN comparison of training iterations. Bottom image shows over-shoot artifacts (in red boxes) caused by the prolonged training.

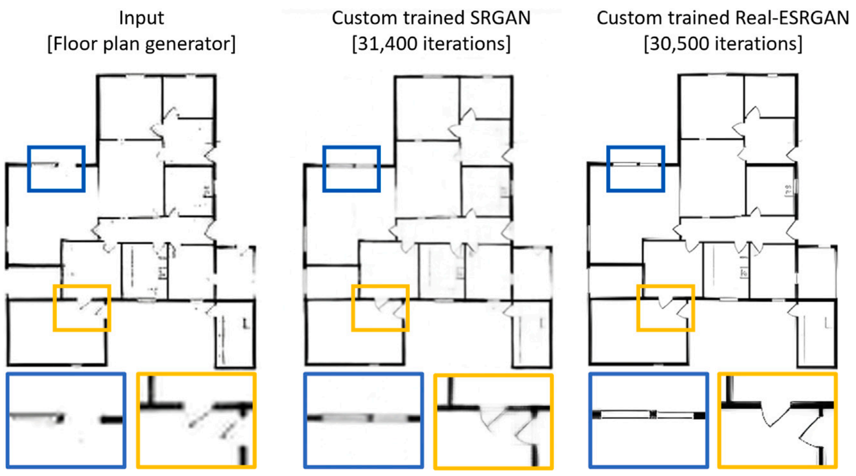

Fig. 12. A comparison of the 4× upscaling process by the pre-trained Real-ESRGAN (2nd column) and the custom fine-tuned Real-ESRGAN (3rd column), based on an input generated image of 256 × 256 pixels (1st column). Top: The complete floor plans. Bottom: Zoomed-in view of the areas inside the blue and yellow boxes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

outer walls and have a well-composed size relative to the room size, whereas doors are often placed so that they open against a wall, which saves space in the layout design.

Structural area span, denoting the distance between bearing points in the layout, was mainly affected by the large, squared rooms where the span between two bearing walls is large. Relative to load and shear walls, the stability of the building relies on a well-composed layout that has both transverse and longitudinal walls, which will help resist wind load. Of course, other techniques do exist that provide wind bracing when spatial open designs are required. In general, the 25 analyzed floor plan designs performed exceptionally well on these two structural design functionalities.

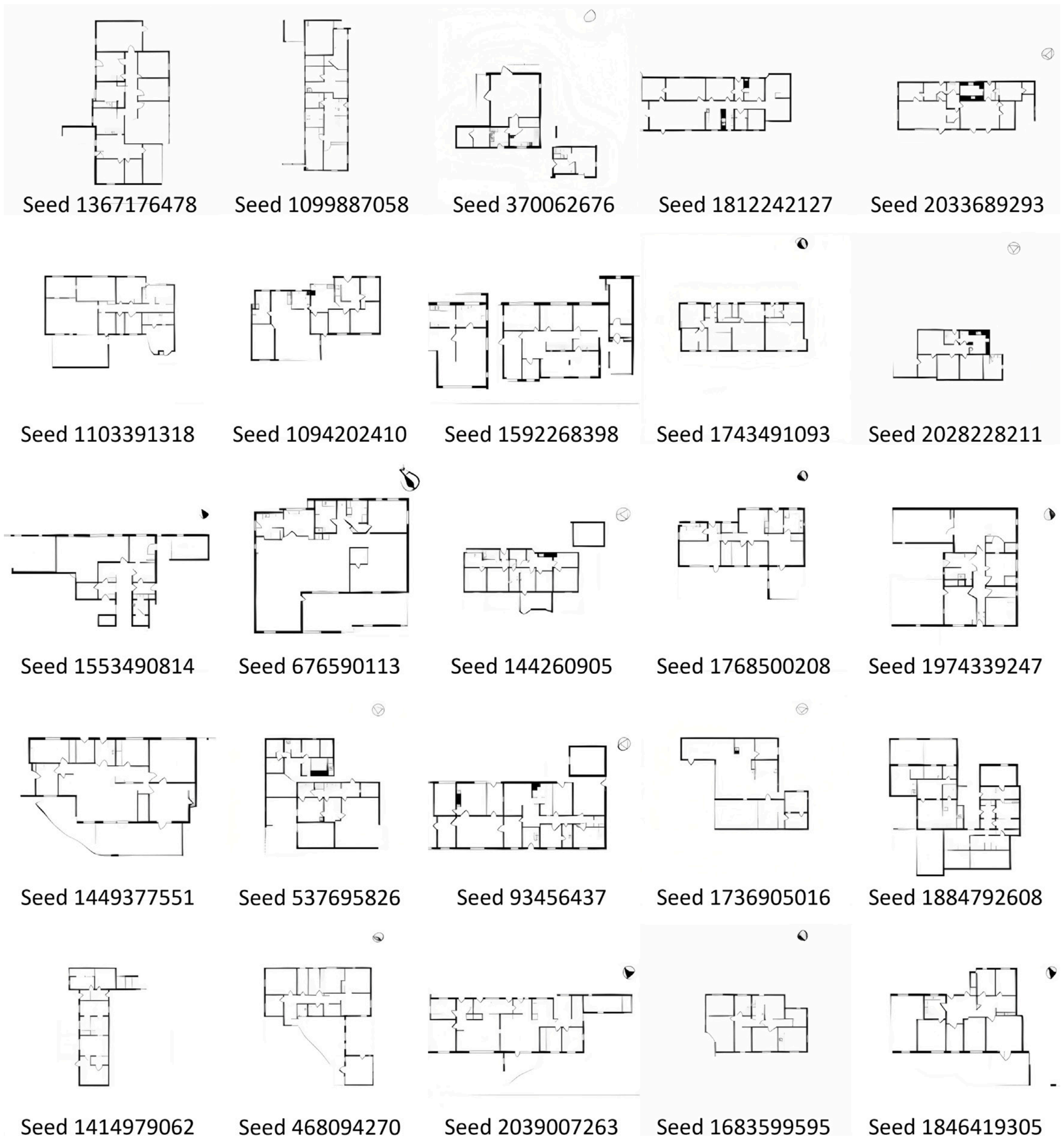

Fig. 13. Image grid of the 25 floor plans used for the validation framework.

Building energy has many regulating parameters, such as outer shape, natural lighting, material selection etc. In fact, the early design phase is to some degree disconnected from the detailed material selection. The main selection criteria are based on the floor plan layout's ability to minimize the linear thermal bridging and optimize the natural light entrance, for which the 25 analyzed designs showed a relatively good performance.

The waste of space in the floor plans gives an indication of the space used in the connecting corridor. The placement of doors and windows also plays a role if they take up unnecessary space by their functioning. The above criterion together with the room connectivity optimization criterion were found in general to underperform to some extent in the 25 analyzed synthetic HR floor plans. In summary, regarding structural and energy considerations, the proposed floor plan generation framework was verified to be consistent with the intended application.

3.4.2. Data privacy with perceptual image hashing

Fig. 14 shows an example of a pair of floor plans with the lowest HD (pHash) and the corresponding 8 × 8 hash images. Some similarities due to the building shape can be observed, but there are no critical signs that indicate overfitting. This result verifies that no direct data copying takes place between the HR real dataset and the HR fake (after SR) dataset.

Table 12
Contextual functionality validation of the 25 analyzed floor plans, out of 10,000 randomly generations. Listed are the (individual and average) numbers of misaligned doors and windows, as well as the scores for each contextual functionality validation criterion. Reported are the mean opinion scores assigned by all co-authors.

Seed	Doors	Windows	Open-area span minimization	Structural stability	Energy efficiency	Access to daylight	Wasted space minimization	Room connectivity
1099887058	3	2	2	2	1	2	1	0
1367176478	0	3	2	2	1	2	0	0
370062676	2	3	1	2	1	2	1	1
1812242127	1	3	2	2	2	2	0	2
2033689293	2	3	2	2	1	1	1	0
1103391318	2	1	2	2	2	0	1	1
1094202410	0	0	2	2	2	1	2	2
1592268398	0	2	2	2	0	1	1	2
1743491093	1	0	2	2	2	2	1	0
2028228211	1	3	2	2	2	2	1	2
1553490814	1	3	2	2	0	1	0	0
676590113	0	1	0	1	1	2	2	2
144260905	1	3	2	2	1	1	1	0
1768500208	0	1	2	2	2	1	1	2
1974339247	3	0	1	2	1	1	0	1
1449377551	0	0	2	2	1	2	0	2
537695826	3	3	2	2	1	0	1	0
93456437	1	3	2	2	1	1	0	0
1736905016	1	3	1	2	1	1	2	2
1884792608	3	3	2	2	0	0	0	0
1414979062	1	3	2	2	0	1	0	0
468094270	2	2	2	2	1	2	1	1
2039007263	1	0	2	2	1	2	0	0
1683599595	3	0	2	2	2	1	1	1
1846419305	1	0	2	2	1	1	1	1
Average	1.32	1.80	1.80	1.96	1.12	1.28	0.76	0.88

Table 13
Framework sampling time based on 10,000 generated floor plan images.

Module	Sampling time [s/image]
Floor plan generator (Projected GAN with FastGAN)	0.02
Up-sampling (Real-ESRGAN)	0.15

3.4.3. *Latent space manipulation with image morphing*

Fig. 15 shows an example of the image morphing transformation technique, demonstrating the floor plan generator's capability of controlling the building layout size and shape styles by selecting seed versions from the learned latent space. It illustrates the transformative power of the floor plan generation module which gradually affects the style from one seed to another (from left to right). The style drifting capability enables a deeper layer of floor plan outputs.

3.4.4. *Downstream BIM implementation*

Fig. 16 shows an example of resketching a generated floor plan using a BIM software environment. Seen is a building layout of 185 m^2 with maximum span of 5862 mm, two sheds with an option for transforming the large one into a garage, two entrances, two bathrooms, a master bedroom with connected bathroom, three rooms for children's rooms/office, a connected kitchen and living room, and a laundry room that connects the house with the shed/garage. All rooms are connected with corridors (or entrances) which can serve as buffer zones for heat differences between the rooms. The building design caters to all common construction principles such as light wall timber frames etc. There is an option for placing shared walls and loading walls in the floor plan design, due to the even distribution of walls. The door openings seem to naturally optimize the space around them, also avoiding collision with other openings. All living spaces (living room with open kitchen) have access to daylight. For the living room, there is also an option to add extra windows. The depth of all rooms relative to the window placement also seems reasonable to ensure sufficient natural daylight. The only challenges of the specific floor plan design are mainly related to the share of outer walls that may reduce the energy performance and the indented areas which may cause limited light in the building.

3.4.5. *Framework inference speed*

Table 13 outlines the average sampling times per image for the Projected GAN with FastGAN floor plan generator (model 9) and the custom-trained Real-ESRGAN up-sampling model. These values verify that the generated floor plan images by the framework are obtained in a swift manner while relying on limited computational resources.

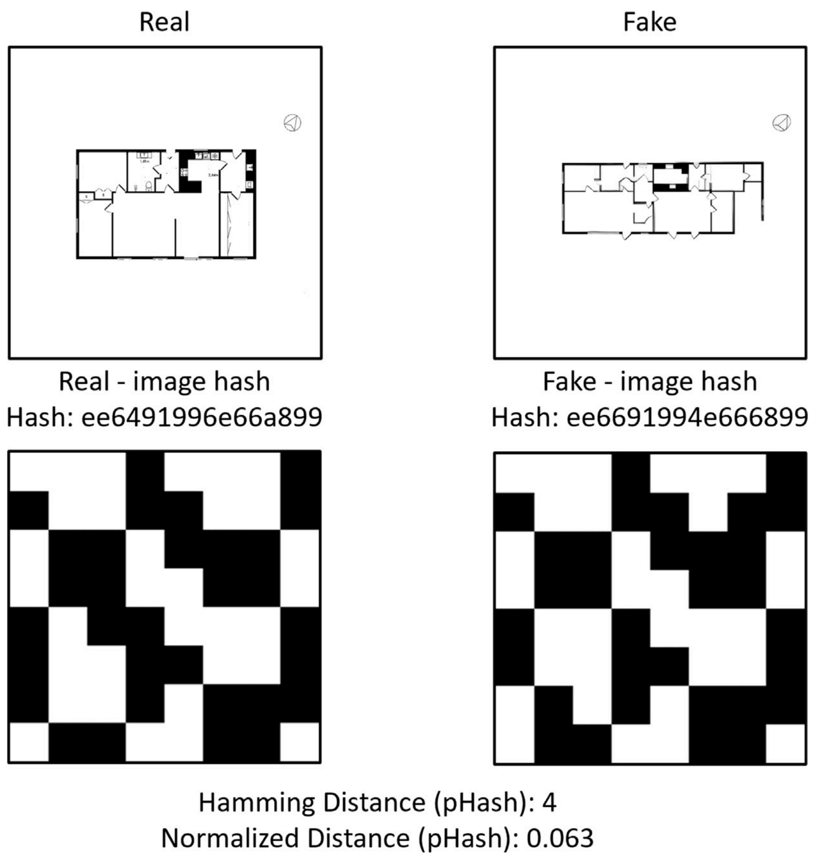

Fig. 14. Overfitting evaluation based on model 9 (seed 2033689293) between a real floor plan image (left), and a synthetically generated floor plan sample (right). Top: The generated floor plans. Bottom: The 8 × 8 hash images. Also given are the Hamming and normalized Hamming distances.

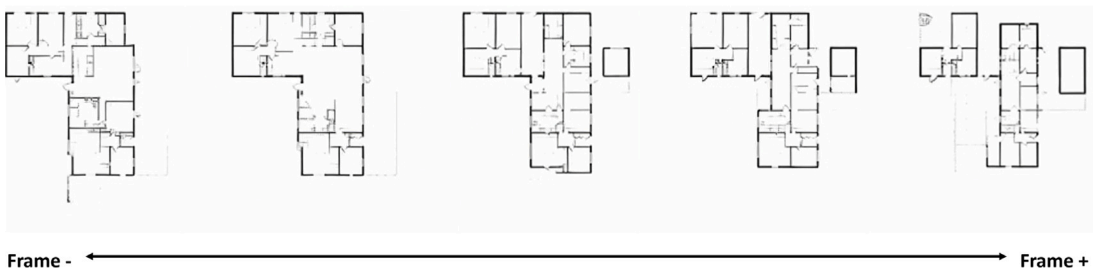

Fig. 15. Floor plan generation with image morphing based on model 9.

4. Discussion and future work

4.1. Main contribution of this study

This study aimed to make progress on the four main challenges that concern the use of GML in construction applications, as they were pointed out in the recent review study by Chai et al. [22]. A novel stable multi-module GAN-stack was proposed for the autonomous generation of high-fidelity and contextually meaningful floor plan layouts by relying on small-scale computational and data resources. The pipeline is made up of a pre-processing (i.e., denoising and 4× down-sampling) module, a floor plan image generation module, and a 4× image up-sampling module. Innovative frameworks were also introduced for: (i) holistic evaluation of the generated floor plans, and (ii) contextual validation of the proposed pipeline.

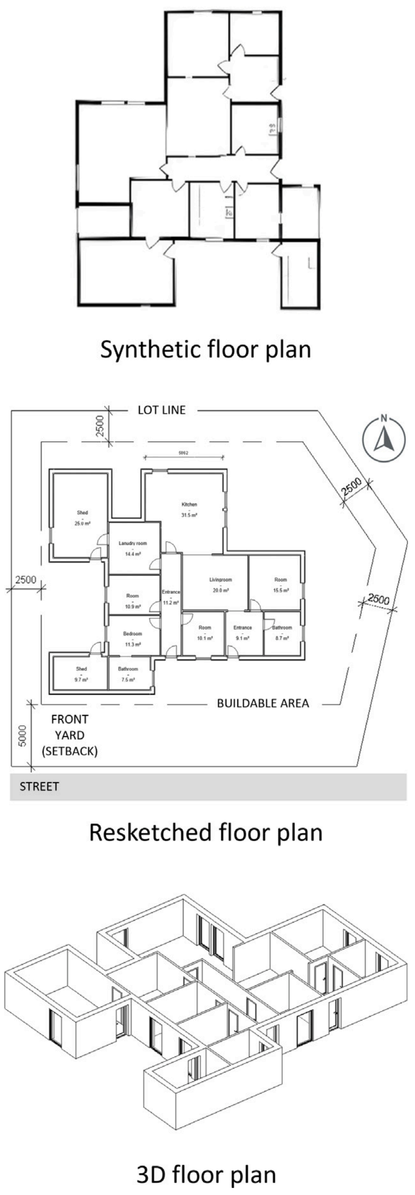

Fig. 16. Architectural resketching (center) using BIM software of a generated output (top) by the framework, together with the corresponding 3D floor plan (bottom), also produced by BIM. The building area is 185 m^2, and the max structural span is 5862 mm.

4.2. Experiments and main findings

All GAN experiments in this study were conducted using minimal training data and computational resources, namely a sub-thousand dataset of high-fidelity architectural floor plan images of single-story residential Danish homes and a single RTX-3090 GPU. To determine the best configuration of the individual modules, this work focused primarily on systematic hyperparameter tuning and model fine-tuning (custom training). In the pre-processing module, a transformer-based denoising approach [47] was deployed. The effect of cleaning datasets from unwanted text content was showcased in a downstream image generation task. Visual assessment was adopted, because using a quantitative metric such as the FID score would not have provided any meaningful insights. The experiments showed that removing text content from the training data significantly improved its practical utility. For down-sampling, a method based on OpenCV's AREA module [46,53] was utilized, which resulted in minimal degradation compared to other approaches. For the floor plan generation module, 10 different models were tested by varying the network architecture, training mode, and set of hyperparameter values. It was found that the best performing model was the one based on the Projected GAN with FastGAN generator [19,21], utilizing half the batch and batch GPU sizes compared to the original implementation. The proposed model gained superior quality, reaching an FID score of 9.4. The respective scores for the traditional StyleGAN3-based models were above 19.1. The comprehensive evaluation demonstrated that the proposed floor plan generation model also achieved the best density (=1.01) and coverage (=0.97) values, and the highest level of truthfulness of the generated images. In addition, it converged in less than 60 h which was almost twice as fast as the other StyleGAN3 end-to-end trained models. Moreover, its energy consumption and carbon emissions were substantially lessened, highlighting its high resource efficiency and low environmental cost. Energy consumption and thereby CF should be relevant to GAN performance evaluation given the growing concerns about the environmental impact of deep learning [13,82,83]. Extending training iterations alone would have been unlikely to yield major improvements due to stagnation in the model development, possibly caused by the limited dataset size. In the up-sampling module, a custom-trained Real-ESRGAN model [48] was selected as it was found to convincingly outperform the baseline method. The novel validation framework was applied to 25 randomly selected floor plans, verifying that the proposed pipeline: (i) preserves data privacy, (ii) can be manipulated with image morphing to suit the designer's needs, (iii) is exploitable in downstream vector-based BIM software implementation tasks, and (iv) can generate contextually meaningful 1024 × 1024 floor plan images in a swift manner by relying on small-scale hardware resources. Lastly, the contextual functionality validation corroborated that the proposed framework is consistent with the intended application regarding door/window placement, structural design, and energy considerations, whereas there is some room for improvement in the space optimization and room connectivity aspects.

4.3. Why it works so well?

The outstanding efficiency and generated image quality performance that was achieved by the proposed multi-module GAN-stack is the result of adopting best practices from a different context (i.e., natural images) for selecting the network architectures and set of techniques that suit the needs and goals of this study. Apart from historical precedent, several crucial network design choices were based on empirical experimentations and intuition. GML architectures, which are able to rapidly sample training datasets and allow substantial control over the generation process, were chosen. The reduced computational cost and training time in the floor plan generation process without compromising image quality is most likely attributed to employing EfficientNet-Lite1 [63] as the pre-trained feature network. The stabilized training of the GAN-stack is in all probability the outcome of utilizing: (i) pre-trained representations in the floor plan image generation (Projected GAN with FastGAN generator) module, and (ii) U-Net design with skip connections together with spectral normalization in the discriminator of the image up-sampling (Real-ESRGAN) module. The lack of overfitting is presumably due to leveraging differentiable augmentation methods [65] in the floor plan generation process.

4.4. Validation framework

A novel validation framework was introduced to confirm that the proposed computerized pipeline captures architectural business rules, attributes, and relationships correctly. Ensuring high-quality validation is a troublesome task considering the strict regulations and the growing sophistication levels of related frameworks. However, validation is a pivotal step towards developing consistent, high-quality products. Domain-specific knowledge is essential for appreciating the validation criteria that matter the most in this type of research. Generating highly realistic-looking floor plan images is not an end in itself — contextual relevance is more critical. In this study, architects with experience in detached single-family home design were interviewed to provide the co-authors with domain-specific insight into the architectural context. Complementary literature was also adjusted [29,84].

4.5. Data privacy

Fair usage, intellectual property issues, and ethical concerns regarding data sourcing have brought GML into question. There is no clear answer yet to these copyright issues [102]. To our knowledge, never before has data privacy been assessed in GAN-based floor plan generation papers. To validate data privacy and overfitting in this study, perceptual image hashing was introduced as a metric for detecting similarities between the real dataset and generated data. Image hashing, while computationally intensive on larger datasets, has the benefit that the generated hash of real images is stationary and can be reused in future evaluations. Rotations and flips were applied to the dataset for a more robust evaluation. Although GANs may avoid overfitting through their built-in classification mechanisms, image hashing provides a transparent and explainable method to ensure data privacy.

4.6. Usefulness of the work

The presented work could inspire new perspectives to the early architectural design phase. As the communication between professional consultants and their clients progresses in this essential phase, the proposed framework has the potential to enable the fast iterative prototyping of conceptual floor plan layouts. A large number of empirical experiments were conducted over the course of this study, which exceed the ones presented in this paper. These tests have provided the co-authors with valuable insights into the training process of various GANs in the context of architectural floor plan design. They have also contributed to a knowledge base for future research aimed at further improving the related training methods. This research advances GML resource efficiency for architectural applications. Efficient frameworks are certain to curb the computational and data requirements for generating realistic floor plan images, which in turn may facilitate democratizing research in this area by enabling more companies to participate. Local training of GML models will also contribute to a higher level of data control and thus data security for business and related data stakeholders in the process. Lastly, this work could lay the foundations for extending GML to other industries that share driving needs and limitations for adopting GML with the architectural design industry [1,2].

4.7. Study limitations

Hyperparameter tuning of GAN models is a rather time-consuming task with each adjustment requiring 10k kimg iterations (i.e. ∼147–160 h). This has limited the ability to more finely search the hyperparameter space. However, GAN training is overly sensitive to hyperparameters, and further experiments with additional hyperparameter value combinations might have provided valuable insight into the optimal setting, especially with regard to the gamma level. The FID score is based on features obtained from the ImageNet pre-trained model. Therefore, caution should be exercised when using this metric for models such as Projected GAN which relies on ImageNet pre-trained representations [64]. In addition, it has been reported that there is some discrepancy between the FID metric and human assessment when evaluating non-ImageNet data [16]. However, this study offered a more reliable generated image quality evaluation across different GAN methods by adopting a combination of three metrics (namely FID, density, CNN detection score) as well as human inspection. Even though downscaling the dataset from 1024 × 1024 to 256 × 256 pixels has sped up the training process, a degradation of the detail level of the floor plan images is inevitable. The up-sampling process can reconstruct some deficiencies, but this degradation remains a limitation worth investigating further. The validation framework of this study did not include criteria related to room modularity and natural ventilation. These were nevertheless deemed too advanced for the early design phase that the proposed pipeline aims to support.

4.8. Future work

While StyleGAN is a promising network architecture for generating image content, it is worth testing and comparing additional GAN models to determine the optimal generator. Licensing aspects might also play a role in the future selection of models. Applying image-to-image style-transfer by using models such as pix2pixHD [103], pix2pix [104], or CycleGAN [105] could enhance the appearance of the generated floor plans and add more details. Converting raster images to vector formats, suitable for BIM environments, is likely to improve the applicability of generated floor plans, as well as the demand to convert existing building layouts into BIM [25,106]. Adding text-based conditions to the proposed framework may offer more control over the generated outputs [107]. Methods for text-to-image generation have been proposed in the literature [15,107,108], and incorporating these methods may further improve output specification. In the Danish architectural context, three main design paradigms influence floor plan layouts from 1950 to today. The style-based floor plan generator of the proposed framework should be able to reflect the above grouping. It may have created a hidden hierarchy in the style space that represents these three design paradigms, offering potential for renovation and reconstruction projects through methods such as style transfer. This task remains to be elucidated. GAN-based pix2pix methods can replace costly computational fluid dynamics (CFD) analysis for layout performance as surrogate models with image-to-image translation to predict contour plots of layout performance to support designers in the early design stage both for 2D and spatial analysis [108]. Furthermore, recent studies have shown promising results using ControlNet-based diffusion model structures, which are capable of generating fast and highly detailed conceptual designs guided by control masks [108]. The modular structure of the framework offers flexibility in updating a specific module as future technological advances come to the fore.

The model's ability to autonomously generate highly realistic floor plans, combined with the control demonstrated by cGAN-based generation, presents a key future challenge in improving the usability of such systems for architectural purposes. A critical aspect of this challenge is enabling the generated floor plans to adapt to specific building plot conditions and meet relevant constraints such as zoning codes and national building regulations.

5. Conclusion

This study aimed at increasing the applicability of generative machine learning (GML) to the automation of early stage building engineering workflows. It made notable strides in addressing the following key challenges on employing generative adversarial networks (GANs) towards this application: Substantial computational and data demands, training instability, and challenges in evaluating results. A stable, multi-module GAN-stack was developed for autonomously generating high-fidelity, contextually meaningful floor plan layouts, while operating on limited computational and data resources. The pipeline consisted of a pre-processing module comprising transformer-based text remover and AREA down-sampling, a floor plan image generation module based on the projected GAN with FastGAN generator network, and an image up-sampling module utilizing the Real-ESRGAN architecture. It was efficiently trained on real-world construction data. Innovative frameworks were also introduced for the holistic evaluation of the generated floor plans and the contextual validation of the proposed pipeline. The comprehensive evaluation results demonstrated that the proposed pipeline can expedite and stabilize the GML-based autonomous generation of 1024 × 1024 floor plan layouts while relying on minimal compute and data resources. The novel validation framework confirmed that the proposed computer-aided pipeline has contextual relevance to architectural design, ensuring adherence to architectural business rules, attributes, and relationships. The presented work could streamline the iterative prototyping of conceptual floor plan layouts in the early design phase, supporting the communication between architects (end users) and clients. The proposed framework has the capability to generate unique high-resolution (HR) contextually meaningful floor plan layouts in a total sampling time of 0.17 s. This research also has potential for broadening access to the GML-based automation in the building engineering research area. It may provide the foundation for extending GML applications to other industries with similar needs and resource constraints.

CRediT authorship contribution statement

Michael Sahl Lystbæk: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Michail J. Beliatis:** Writing – original draft, Supervision, Resources, Conceptualization. **Archontis Giannakidis:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Home A/S for supporting the data collection process. Special thanks to Arkitec A/S, particularly the architect Ole Damgaard for inspiring discussions about perspectives on the professional use of AI-based design generation. We are also grateful to Professor Cand.arch., PhD — Thomas Bo Jensen and Cand.arch., PhD fellow — Mette Boisen Lyhne for their valuable input on the design subject.

Data availability

Data will be made available on request.

References

[1] R.C. Rodrigues, R.B. Duarte, Generating floor plans with deep learning: A cross-validation assessment over different dataset sizes, Int. J. Archit. Comput. 20 (3) (2022) 630–644, http://dx.doi.org/10.1177/14780771221120842.

[2] K. Park, S. Ergan, C. Feng, Quality assessment of residential layout designs generated by relational Generative Adversarial Networks (GANs), Autom. Constr. 158 (2024) http://dx.doi.org/10.1016/j.autcon.2023.105243.

[3] Z. Luo, W. Huang, FloorplanGAN: Vector residential floorplan adversarial generation, Autom. Constr. 142 (2022) 104470, http://dx.doi.org/10.1016/j.autcon.2022.104470.

[4] J. McCormack, P. Hutchings, T. Gifford, M. Yee-King, M.T. Llano, M. D'Inverno, Design considerations for Real-Time Collaboration with creative artificial intelligence, Organ. Sound 25 (1) (2020) 41–52, http://dx.doi.org/10.1017/S1355771819000451.

[5] V. Paananen, J. Oppenlaender, A. Visuri, Using text-to-image generation for architectural design ideation, Int. J. Archit. Comput. (2023) http://dx.doi.org/10.1177/14780771231222783.

[6] Z. Zhang, J.M. Fort, L. Giménez Mateu, Exploring the potential of artificial intelligence as a tool for architectural design: A perception study using Gaudí's Works, Buildings 13 (7) (2023) 1863, http://dx.doi.org/10.3390/buildings13071863.

[7] S. Zheng, StyleGAN-Canvas: Augmenting StyleGAN3 for real-time Human-AI Co-Creation, in: CEUR Workshop Proceedings, Vol. 3359, 2023, pp. 108–120, https://ceur-ws.org/Vol-3359/paper12.pdf. (Accessed 31 July 2025).

[8] M.L. Castro Pena, A. Carballal, N. Rodríguez-Fernández, I. Santos, J. Romero, Artificial intelligence applied to conceptual design. A review of its use in architecture, Autom. Constr. 124 (2021) http://dx.doi.org/10.1016/j.autcon.2021.103550.

[9] P. Pouliou, A.-S. Horvath, G. Palamas, Speculative hybrids: Investigating the generation of conceptual architectural forms through the use of 3D generative adversarial networks, Int. J. Archit. Comput. 21 (2) (2023) 315–336, http://dx.doi.org/10.1177/14780771231168229.

[10] J. Huang, M. Johanes, F.C. Kim, C. Doumpioti, G.C. Holz, On GANs, NLP and architecture: Combining human and machine intelligences for the generation and evaluation of meaningful designs, Technol. Archit. Des. 5 (2) (2021) 207–224, http://dx.doi.org/10.1080/24751448.2021.1967060.

[11] H. Song, J. Ghaboussi, T.-H. Kwon, Architectural design of apartment buildings using the Implicit Redundant Representation Genetic Algorithm, Autom. Constr. 72 (2016) 166–173, http://dx.doi.org/10.1016/j.autcon.2016.09.001.

[12] T. Karras, S. Laine, T. Aila, A Style-Based generator architecture for generative adversarial networks, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 4396–4405, http://dx.doi.org/10.1109/CVPR.2019.00453.

[13] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, T. Aila, Alias-Free generative adversarial networks, in: Advances in Neural Information Processing Systems, Vol. 34, 2021, pp. 852–863, https://proceedings.neurips.cc/paper_files/paper/2021/file/076ccd93ad68be51f23707988e934906-Paper.pdf. (Accessed 31 July 2025).

[14] A. Sauer, K. Schwarz, A. Geiger, Stylegan-xl: Scaling stylegan to large diverse datasets, in: ACM SIGGRAPH 2022 Conference Proceedings, Association for Computing Machinery, 2022, pp. 1–10, http://dx.doi.org/10.1145/3528233.3530738.

[15] A. Sauer, T. Karras, S. Laine, A. Geiger, T. Aila, StyleGAN-T: Unlocking the power of GANs for fast Large-Scale Text-to-Image synthesis, in: Proceedings of the 40th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 202, 2023, pp. 30105–30118, https://proceedings.mlr.press/v202/sauer23a/sauer23a.pdf. (Accessed 31 July 2025).

[16] T. Kynkäänniemi, T. Karras, M. Aittala, T. Aila, J. Lehtinen, The role of ImageNet classes in Fréchet inception distance, 2023, http://dx.doi.org/10.48550/arXiv.2203.06026.

[17] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, T. Aila, Training generative adversarial networks with limited data, in: Advances in Neural Information Processing Systems, Vol. 33, 2020, pp. 12104–12114, https://proceedings.neurips.cc/paper_files/paper/2020/file/8d30aa96e72440759f74bd2306c1fa3d-Paper.pdf. (Accessed 31 July 2025).

[18] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning transferable visual models from natural language supervision, in: Proceedings of the 38th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 139, 2021, pp. 8748–8763, http://proceedings.mlr.press/v139/radford21a/radford21a.pdf. (Accessed 31 July 2025).

[19] B. Liu, Y. Zhu, K. Song, A. Elgammal, Towards faster and stabilized GAN training for High-fidelity Few-shot image synthesis, 2021.

[20] A. Alanov, V. Titov, M. Nakhodnov, D. Vetrov, StyleDomain: Efficient and lightweight parameterizations of StyleGAN for One-shot and Few-shot Domain adaptation, in: 2023 IEEE/CVF International Conference on Computer Vision, ICCV, IEEE, 2023, pp. 2184–2194, http://dx.doi.org/10.1109/ICCV51070.2023.00208.

[21] A. Sauer, K. Chitta, J. Müller, A. Geiger, Projected GANs converge faster, in: Proceedings of the 35th International Conference on Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 17480–17492, https://proceedings.neurips.cc/paper/2021/file/9219adc5c42107c4911e249155320648-Paper.pdf. (Accessed 31 July 2025).

[22] P. Chai, L. Hou, G. Zhang, Q. Tushar, Y. Zou, Generative adversarial networks in construction applications, Autom. Constr. 159 (2024) 105265, http://dx.doi.org/10.1016/j.autcon.2024.105265.

[23] E. Betzalel, C. Penso, A. Navon, E. Fetaya, A study on the evaluation of generative models, 2022, http://dx.doi.org/10.48550/arXiv.2206.10935.

[24] M.F. Naeem, S.J. Oh, Y. Uh, Y. Choi, J. Yoo, Reliable fidelity and diversity metrics for generative models, in: Proceedings of the 37th International Conference on Machine Learning, Vol. 119, 2020, pp. 7176–7185, http://proceedings.mlr.press/v119/naeem20a/naeem20a.pdf. (Accessed 31 July 2025).

[25] J. Parente, E. Rodrigues, B. Rangel, J. Poças Martins, Integration of convolutional and adversarial networks into building design: A review, J. Build. Eng. 76 (2023) 107155, http://dx.doi.org/10.1016/j.jobe.2023.107155.

[26] S. Kookalani, E. Parn, I. Brilakis, S. Dirar, M. Theofanous, A. Faramarzi, M.A. Mahdavipour, Q. Feng, Trajectory of building and structural design automation from generative design towards the integration of deep generative models and optimization: A review, J. Build. Eng. 97 (2024) 110972, http://dx.doi.org/10.1016/j.jobe.2024.110972.

[27] Q. Feng, C. Guo, F. Benitez-Quiroz, A. Martinez, When do GANs replicate? On the choice of dataset size, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV, IEEE, 2021, pp. 6681–6690, http://dx.doi.org/10.1109/ICCV48922.2021.00663.

[28] D. Newton, Deep generative learning for the generation and analysis of architectural plans with small datasets, in: Proceedings of the International Conference on Education and Research in Computer Aided Architectural Design in Europe, Vol. 2, 37th Conference on Education and Research in Computer Aided Architectural Design in Europe, 2019, pp. 21–28, http://dx.doi.org/10.5151/proceedings-ecaadesigradi2019_135.

[29] R.E. Weber, C. Mueller, C. Reinhart, Automated floorplan generation in architectural design: A review of methods and applications, Autom. Constr. 140 (2022) 13, http://dx.doi.org/10.1016/j.autcon.2022.104385.

[30] L. Wang, J. Liu, Y. Zeng, G. Cheng, H. Hu, J. Hu, X. Huang, Automated building layout generation using deep learning and graph algorithms, Autom. Constr. 154 (April) (2023) 105036, http://dx.doi.org/10.1016/j.autcon.2023.105036.

[31] M. Aalaei, M. Saadi, M. Rahbar, A. Ekhlassi, Architectural layout generation using a graph-constrained conditional generative adversarial network (GAN), Autom. Constr. 155 (November) (2023) 105053, http://dx.doi.org/10.1016/j.autcon.2023.105053.

[32] J. Sun, W. Wu, L. Liu, W. Min, G. Zhang, L. Zheng, WallPlan: Synthesizing floorplans by learning to generate wall graphs, ACM Trans. Graph. 41 (4) (2022) 1–14, http://dx.doi.org/10.1145/3528223.3530135.

[33] M. Rahbar, M. Mahdavinejad, M. Bemanian, A. Davaie Markazi, L. Hovestadt, Generating synthetic space allocation probability layouts based on trained Conditional-GANs, Appl. Artif. Intell. 33 (8) (2019) 689–705, http://dx.doi.org/10.1080/08839514.2019.1592919.

[34] C.-W. Zhao, J. Yang, J. Li, Generation of hospital emergency department layouts based on generative adversarial networks, J. Build. Eng. 43 (2021) 102539, http://dx.doi.org/10.1016/j.jobe.2021.102539.

[35] H. Tanasra, T. Rott Shaham, T. Michaeli, G. Austern, S. Barath, Automation in interior space planning: Utilizing conditional generative adversarial network models to create furniture layouts, Buildings 13 (7) (2023) http://dx.doi.org/10.3390/buildings13071793.

[36] I. Karadag, O.Z. Güzelci, S. Alaçam, EDU-AI: a twofold machine learning model to support classroom layout generation, Constr. Innov. 23 (4) (2022) 898–914, http://dx.doi.org/10.1108/CI-02-2022-0034.

[37] M. Rahbar, M. Mahdavinejad, A.H. Markazi, M. Bemanian, Architectural layout design through deep learning and agent-based modeling: A hybrid approach, J. Build. Eng. 47 (2022) http://dx.doi.org/10.1016/j.jobe.2021.103822.

[38] F. Jiang, J. Ma, C.J. Webster, X. Li, V.J. Gan, Building layout generation using site-embedded GAN model, Autom. Constr. 151 (2023) http://dx.doi.org/10.1016/j.autcon.2023.104888.

[39] S. Wang, W. Zeng, X. Chen, Y. Ye, Y. Qiao, C.-W. Fu, ActFloor-GAN: Activity-Guided adversarial networks for Human-Centric floorplan design, IEEE Trans. Vis. Comput. Graphics 29 (3) (2023) 1610–1624, http://dx.doi.org/10.1109/TVCG.2021.3126478.

[40] C. Zhao, J. Yang, W. Xiong, J. Li, Two generative design methods of hospital operating department layouts based on healthcare systematic layout planning and Generative Adversarial network, J. Shanghai Jiaotong Univ. (Sci.) 26 (1) (2021) 103–115, http://dx.doi.org/10.1007/s12204-021-2265-9.

[41] P. Ghannad, Y.-C. Lee, Automated modular housing design using a module configuration algorithm and a coupled generative adversarial network (CoGAN), Autom. Constr. 139 (2022) http://dx.doi.org/10.1016/j.autcon.2022.104234.

[42] J. Shim, J. Moon, H. Kim, E. Hwang, FloorDiffusion: Diffusion model-based conditional floorplan image generation method using parameter-efficient fine-tuning and image inpainting, J. Build. Eng. 95 (2024) 110320, http://dx.doi.org/10.1016/j.jobe.2024.110320.

[43] H. Leng, Y. Gao, Y. Zhou, ArchiDiffusion: A novel diffusion model connecting architectural layout generation from sketches to Shear Wall Design, J. Build. Eng. 98 (2024) 111373, http://dx.doi.org/10.1016/j.jobe.2024.111373.

[44] S. Chaillou, ArchiGAN: Artificial intelligence x architecture, in: P.F. Yuan, M. Xie, N. Leach, J. Yao, X. Wang (Eds.), Architectural Intelligence, Springer Nature Singapore, 2020, pp. 117–127, http://dx.doi.org/10.1007/978-981-15-6568-7.

[45] Z. Du, H. Shen, X. Li, M. Wang, 3D building fabrication with geometry and texture coordination via hybrid GAN, J. Ambient. Intell. Humaniz. Comput. 13 (11) (2022) 5177–5188, http://dx.doi.org/10.1007/s12652-020-02488-9.

[46] G. Rossi, M. Fontani, S. Milani, Neural network for denoising and reading degraded license plates, in: Pattern Recognition. ICPR International Workshops and Challenges, Vol. 12666, Springer International Publishing, 2021, pp. 484–499, http://dx.doi.org/10.1007/978-3-030-68780-9_39.

[47] M. Lystbæk, A. Giannakidis, M.J. Beliatis, M. Olsen, Removing unwanted text from architectural images with multi-scale deformable attention-based machine learning, in: 2023 IEEE International Conference on Imaging Systems and Techniques, IST, IEEE, 2023, pp. 1–5, http://dx.doi.org/10.1109/IST59124.2023.10355658.

[48] X. Wang, L. Xie, C. Dong, Y. Shan, Real-ESRGAN: Training Real-World Blind Super-Resolution with pure synthetic data, in: Proceedings of the IEEE International Conference on Computer Vision, Vol. 2021-Octob, 2021, pp. 1905–1914, http://dx.doi.org/10.1109/ICCVW54120.2021.00217.

[49] R. Schwartz, J. Dodge, N.A. Smith, O. Etzioni, Green AI, Commun. ACM 63 (12) (2020) 54–63, http://dx.doi.org/10.1145/3381831.

[50] K. Tero, A. Miika, L. Samuli, H. Erik, H. Janne, L. Jaakko, A. Timo, StyleGAN3 - Training configurations, 2021, https://github.com/NVlabs/stylegan3/blob/main/docs/configs.md#recommended-configurations. (Accessed 31 July 2025).

[51] E.C. Ifeachor, B.W. Jervis, Digital Signal Processing: A Practical Approach, second ed., Prentice Hall, Harlow, England, 2002, p. 933, https://books.google.com/books?id=slHJIO9xmcEC&pgis=1. (Accessed 31 July 2025).

[52] Home A/S, Home.dk, 2025, https://home.dk. (Accessed 31 July 2025).

[53] Open Source Computer Vision (OpenCV), Geometric image transformations - Image processing, 2024, https://docs.opencv.org/3.4/da/d54/group_imgproc_transform.html. (Accessed 31 July 2025).

[54] X. Zhang, Y. Su, S. Tripathi, Z. Tu, Text spotting transformers, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 9509–9518, http://dx.doi.org/10.1109/CVPR52688.2022.00930.

[55] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[56] I.J. Goodfellow, J. Pouget-abadie, M. Mirza, B. Xu, D. Warde-farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems 27, NIPS, NeurIPS, 2014, pp. 1–9, https://proceedings.neurips.cc/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf. (Accessed 31 July 2025).
[57] Z. Wu, D. Lischinski, E. Shechtman, Stylespace analysis: Disentangled controls for stylegan image generation, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR, 2021, pp. 12858–12867, http://dx.doi.org/10.1109/CVPR46437.2021.01267.
[58] S. Park, How to edit images with gans? Controlling the latent space of GANs, 2021, https://medium.com/codex/afde630e53d1. (Accessed 31 July 2025).
[59] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, D. Lischinski, StyleCLIP: Text-driven manipulation of stylegan imagery, in: Proceedings of the IEEE International Conference on Computer Vision, ICCV, 2021, pp. 2065–2074, http://dx.doi.org/10.1109/ICCV48922.2021.00209.
[60] T. Oanda, Exploring and exploiting the latent style space, 2022, https://blog.paperspace.com/exploring-and-exploiting-the-latent-style-space/. (Accessed 31 July 2025).
[61] M.K. Venturelli, P.H. Gomes, J. Wehrmann, Looks like Magic: Transfer learning in GANs to generate new card illustrations, in: Proceedings of the International Joint Conference on Neural Networks, Vol. 2022-July, IJCNN, 2022, pp. 1–8, http://dx.doi.org/10.1109/IJCNN55064.2022.9892463.
[62] J. Pinkney, StyleGAN3-t LHQ 256 dataset, 2023, https://huggingface.co/justinpinkney/stylegan3-t-lhq-256. (Accessed 31 July 2025).
[63] M. Tan, Q.V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: Proceedings of the 36th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 97, 2019, pp. 6105–6114, http://proceedings.mlr.press/v97/tan19a/tan19a.pdf. (Accessed 31 July 2025).
[64] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Vol. 20, IEEE, 2009, pp. 248–255, http://dx.doi.org/10.1109/CVPR.2009.5206848.
[65] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, S. Han, Differentiable augmentation for data-efficient GAN training, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, Vol. 33, Red Hook, NY, USA, 2020, pp. 7559–7570, https://proceedings.neurips.cc/paper/2020/file/55479c55ebd1efd3ff125f1337100388-Paper.pdf. (Accessed 31 July 2025).
[66] B. Neyshabur, S. Bhojanapalli, A. Chakrabarti, Stabilizing GAN training with multiple random projections, 2017, http://dx.doi.org/10.48550/arXiv.1705.07831, arXiv preprint arXiv:1705.07831.
[67] C. Ledig, L. Theis, F. Husz, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, W. Shi, Photo-Realistic single image super-resolution using a generative adversarial network, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 4681–4690, http://dx.doi.org/10.1109/CVPR.2017.19.
[68] J. Guerreiro, P. Tomás, N. Garcia, H. Aidos, Super-resolution of magnetic resonance images using Generative Adversarial Networks, in: Computerized Medical Imaging and Graphics, Vol. 108, Elsevier Ltd, 2023, 102280, http://dx.doi.org/10.1016/j.compmedimag.2023.102280.
[69] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015, pp. 1–14, http://dx.doi.org/10.48550/arXiv.1409.1556.
[70] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, C.C. Loy, ESRGAN: Enhanced super-resolution generative adversarial networks, in: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, in: LNCS, Vol. 11133, 2019, pp. 63–79, http://dx.doi.org/10.1007/978-3-030-11021-5_5.
[71] A. Jolicoeur-Martineau, The relativistic discriminator: a key element missing from standard GAN, 2018, http://dx.doi.org/10.48550/arXiv.1807.00734.
[72] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, GANs trained by a two Time-Scale update rule converge to a local Nash equilibrium, 2017, https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf. (Accessed 31 July 2025).
[73] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2016-Decem, 2016, pp. 2818–2826, http://dx.doi.org/10.1109/CVPR.2016.308.
[74] M.S.M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, S. Gelly, Assessing generative models via precision and recall, in: Advances in Neural Information Processing Systems, Vol. 31, 2018, pp. 5228–5237, https://proceedings.neurips.cc/paper/2018/file/f7696a9b362ac5a51c3dc8f098b73923-Paper.pdf. (Accessed 31 July 2025).
[75] L. Simon, R. Webster, J. Rabin, Revisiting precision and recall definition for generative model evaluation, in: 36th International Conference on Machine Learning, Vol. June, 2019, pp. 10174–10183, http://proceedings.mlr.press/v97/simon19a/simon19a.pdf. (Accessed 31 July 2025).
[76] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, T. Aila, Improved precision and recall metric for assessing generative models, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vol. 32, 2019, pp. 1–10, https://proceedings.neurips.cc/paper/2019/file/0234c510bc6d908b28c70ff313743079-Paper.pdf. (Accessed 31 July 2025).
[77] L. Nataraj, T.M. Mohammed, B.S. Manjunath, S. Chandrasekaran, A. Flenner, J.H. Bappy, A.K. Roy-Chowdhury, Detecting GAN generated fake images using Co-occurrence matrices, 2019, http://dx.doi.org/10.2352/ISSN.2470-1173.2019.5.MWSF-532.
[78] M. Barni, K. Kallas, E. Nowroozi, B. Tondi, CNN detection of GAN-generated face images based on Cross-Band Co-occurrences analysis, in: 2020 IEEE International Workshop on Information Forensics and Security, WIFS, 2020, pp. 1–6, http://dx.doi.org/10.1109/WIFS49906.2020.9360905.
[79] S. Raj, J. Mathew, A. Mondal, Generalized and robust model for GAN-generated image detection, Pattern Recognit. Lett. 182 (April) (2024) 104–110, http://dx.doi.org/10.1016/j.patrec.2024.04.018.
[80] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9, http://dx.doi.org/10.1109/CVPR.2015.7298594.
[81] Carbontracker: Tracking and predicting the carbon footprint of training deep learning models, in: ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems, 2020, http://dx.doi.org/10.48550/arXiv.2007.03051.
[82] T.A. Bohoran, K.S. Parke, M.P. Graham-Brown, M. Meisuria, A. Singh, J. Wormleighton, D. Adlam, D. Gopalan, M.J. Davies, B. Williams, et al., Resource efficient aortic distensibility calculation by end to end spatiotemporal learning of aortic lumen from multicentre multivendor multidisease CMR images, Sci. Rep. 13 (1) (2023) 21794, http://dx.doi.org/10.1038/s41598-023-48986-6.
[83] A. Siouras, S. Moustakidis, G. Chalatsis, T.A. Bohoran, M. Hantes, M. Vlychou, S. Tasoulis, A. Giannakidis, D. Tsaopoulos, Economical hybrid novelty detection leveraging global aleatoric semantic uncertainty for enhanced MRI-based ACL tear diagnosis, Comput. Med. Imaging Graph. 117 (2024) 102424, http://dx.doi.org/10.1016/j.compmedimag.2024.102424.
[84] E. de Place Hansen, Small Houses - design and function, in: SBi-anvisning, Statens Byggeforskningsinstitut, Aalborg Universitet, 2018, https://vbn.aau.dk/en/publications/small-houses-design-and-function. (Accessed 31 July 2025).
[85] Statistics Denmark, BOL106: Dwellings with Registered Population, Tech. Rep., Statistics Denmark, 2025, https://www.statbank.dk/BOL106. (Accessed 31 July 2025).
[86] C. Meehan, K. Chaudhuri, S. Dasgupta, A Non-Parametric Test to detect Data-Copying in generative models, in: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, Vol. 108, 2020, pp. 3546–3556, https://proceedings.mlr.press/v108/meehan20a/meehan20a.pdf. (Accessed 31 July 2025).
[87] J. Buchner, Github repo: Imagehash, 2021, https://github.com/JohannesBuchner/imagehash. (Accessed 31 July 2025).
[88] Z. Huang, S. Liu, Perceptual image hashing with texture and invariant vector distance for copy detection, IEEE Trans. Multimed. 23 (2021) 1516–1529, http://dx.doi.org/10.1109/TMM.2020.2999188.
[89] S. Mckeown, W.J. Buchanan, Hamming distributions of popular perceptual hashing techniques, Forensic Sci. Int.: Digit. Investig. 44 (2023) http://dx.doi.org/10.1016/j.fsidi.2023.301509.

[90] Z. Huang, Z. Tang, X. Zhang, L. Ruan, X. Zhang, Perceptual image hashing with locality preserving projection for copy detection, IEEE Trans. Dependable Secur. Comput. 20 (1) (2023) 463–477, http://dx.doi.org/10.1109/TDSC.2021.3136163.
[91] M. Tanaka, S. Shiota, H. Kiya, A detection method of operated fake-images using robust hashing, J. Imaging 7 (8) (2021) http://dx.doi.org/10.3390/jimaging7080134.
[92] Y.N. Li, P. Wang, Y.T. Su, Robust image hashing based on selective quaternion invariance, in: IEEE Signal Processing Letters, Vol. 22, IEEE, 2015, pp. 2396–2400, http://dx.doi.org/10.1109/LSP.2015.2487824.
[93] N. Krawetz, Looks like it - Perceptual hash, 2011, https://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html. (Accessed 31 July 2025).
[94] M. Tanaka, H. Kiya, Fake-image detection with robust hashing, in: LifeTech 2021 - 2021 IEEE 3rd Global Conference on Life Sciences and Technologies, IEEE, 2021, pp. 40–43, http://dx.doi.org/10.1109/LifeTech52111.2021.9391842.
[95] R. Abdal, Y. Qin, P. Wonka, Image2StyleGAN: How to embed images into the StyleGAN latent space? in: Proceedings of the IEEE International Conference on Computer Vision, Vol. 2019-Octob, 2019, pp. 4431–4440, http://dx.doi.org/10.1109/ICCV.2019.00453.
[96] A. Bermano, R. Gal, Y. Alaluf, R. Mokady, Y. Nitzan, O. Tov, O. Patashnik, D. Cohen-Or, State-of-the-Art in the architecture, methods and applications of StyleGAN, Comput. Graph. Forum 41 (2) (2022) 591–611, http://dx.doi.org/10.1111/cgf.14503.
[97] Open Source Computer Vision (OpenCV), Color conversions, 2024, https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray/. (Accessed 31 July 2025).
[98] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: Computer Vision – ECCV 2016, Springer International Publishing, 2016, pp. 694–711, http://dx.doi.org/10.1007/978-3-319-46475-6_43.
[99] Mahmood-Hussain, Generative-evaluation-prdc, 2022, https://github.com/Mahmood-Hussain/generative-evaluation-prdc. (Accessed 31 July 2025).
[100] L.F.W. Anthony, B. Kanding, R. Selvan, Github repo: Carbon tracker, 2023, https://github.com/lfwa/carbontracker. (Accessed 31 July 2025).
[101] J. Buchner, Imagehashing, 2022, https://pypi.org/project/ImageHash/. (Accessed 31 July 2025).
[102] R.E. Mitchell, Web scraping with Python: Data Extraction from the Modern Web, third ed., O'Reilly Media, Sebastopol, CA, 2024, https://www.oreilly.com/library/view/web-scraping-with/9781098145347/. (Accessed 31 July 2025).
[103] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, B. Catanzaro, High-resolution image synthesis and semantic manipulation with conditional GANs, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8798–8807, http://dx.doi.org/10.1109/CVPR.2018.00917.
[104] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 5967–5976, http://dx.doi.org/10.1109/CVPR.2017.632.
[105] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired Image-to-Image translation using Cycle-Consistent adversarial networks, in: 2017 IEEE International Conference on Computer Vision, ICCV, Vol. 2017-Octob, IEEE, 2017, pp. 2242–2251, http://dx.doi.org/10.1109/ICCV.2017.244.
[106] M. Urbieta, M. Urbieta, T. Laborde, G. Villarreal, G. Rossi, Generating BIM model from structural and architectural plans using Artificial Intelligence, J. Build. Eng. 78 (2023) 107672, http://dx.doi.org/10.1016/j.jobe.2023.107672.
[107] M.S. Lystbæk, Text-to-image conditional GAN-Based floor plan generator, in: 2025 International Conference on Artificial Intelligence in Information and Communication, ICAIIC, 2025, pp. 0302–0307, http://dx.doi.org/10.1109/ICAIIC64266.2025.10920845.
[108] M.S. Lystbæk, Machine learning-driven processes in architectural building design, Autom. Constr. 178 (2025) 106379, http://dx.doi.org/10.1016/j.autcon.2025.106379.