**Discover** Artificial Intelligence

Research

# Generative approaches for solving tangram puzzles

**Fernanda Miyuki Yamada[1] · Harlen Costa Batagelo[2] · João Paulo Gois[2] · Hiroki Takahashi[1,3]**

Received: 9 November 2023 / Accepted: 23 January 2024
Published online: 08 February 2024

**Abstract**
The Tangram is a dissection puzzle composed of seven polygonal pieces that can form different patterns. Solving the Tangram is an irregular shape packing problem known to be NP-hard. This paper investigates the application of four deep-learning architectures—Convolutional Autoencoder, Variational Autoencoder, U-Net, and Generative Adversarial Network—specifically designed for solving Tangram puzzles. We explore the potential of these architectures in learning the complex spatial relationships inherent in Tangram configurations. Our experiments show that the Generative Adversarial Network competes well with other architectures and converges considerably faster. We further prove that traditional evaluation metrics based on pixel accuracy often fail in assessing the visual quality of the generated Tangram solutions. We introduce a loss function based on a Weighted Mean Absolute Error that prioritizes pixels representing inter-piece sections over those covered by individual pieces. Extending this loss function, we propose a novel evaluation metric as a more fitting measure for assessing Tangram solutions compared to traditional metrics. This investigation advances our understanding of the capabilities of artificial intelligence in complex geometrical problem domains.

**Keywords** Tangram · Puzzle · Deep-learning · Generative adversarial network

## 1 Introduction

The Tangram comprises seven polygonal pieces, forming a geometric puzzle. Achieving the objective involves rearranging these pieces through rigid body transformations to match a specific pattern. A solution is valid only if it contains all the pieces with no overlaps between them. Figure 1 exemplifies a Tangram puzzle, where one can find the pieces, a desired pattern, and a feasible solution. The desired pattern may also be referred to as simply Tangram pattern throughout the text of the present paper.

According to some historians, the Tangram originated from a furniture set during the Song Dynasty and later evolved into a collection of wooden blocks for recreational purposes [1]. Up to this day, even though several geometric puzzles were invented before and after Tangram, it remains the most popular geometric puzzle in the world [2]. Artists and designers have embraced its geometric elements to create artworks, while mathematicians have explored its properties and connections to other mathematical concepts [3, 4]. Tangram puzzles have found extensive use in stimulating

Harlen Costa Batagelo, João Paulo Gois and Hiroki Takahashi are contributed equally to this work.

✉ Fernanda Miyuki Yamada, y2140014@edu.cc.uec.ac.jp; Harlen Costa Batagelo, harlen.batagelo@ufabc.edu.br; João Paulo Gois, joao.gois@ufabc.edu.br; Hiroki Takahashi, rocky@inf.uec.ac.jp | [1]Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofugaoka, Chofu, Tokyo 182-8585, Japan. [2]Center for Mathematics, Computing and Cognition, Federal University of ABC, Av. dos Estados, Santo André, São Paulo 09210-580, Brazil. [3]Artificial Intelligence Exploration Research Center, Meta-Networking Research Center, The University of Electro-Communications, Chofugaoka, Chofu, Tokyo 182-8585, Japan.

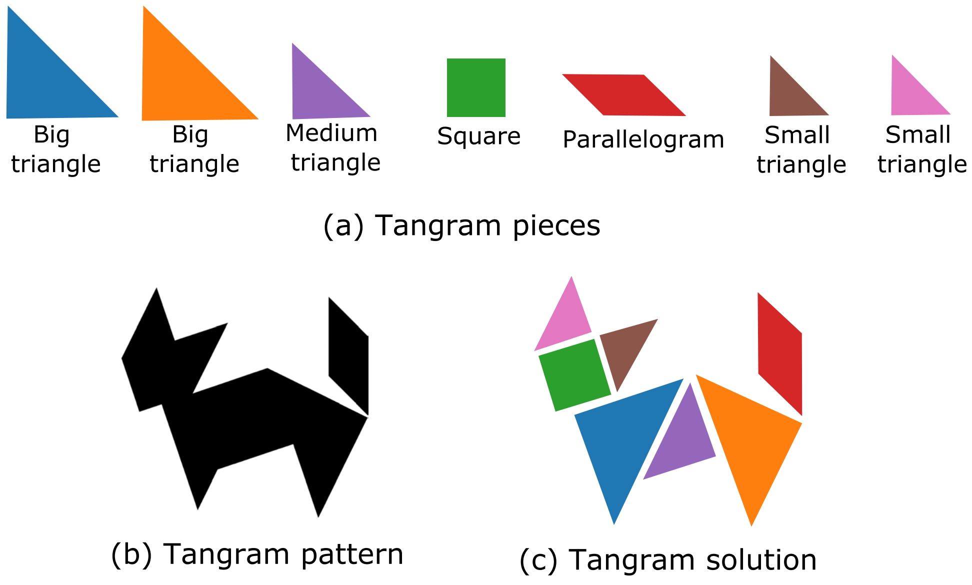


**Fig. 1** Tangram pieces, puzzle, and a feasible solution. Rigid body transformations are applied to the pieces to form the desired pattern, with the condition of no overlaps and no missing pieces

manipulative learning, serving as teaching aids to help young students develop geometric thinking and spatial reasoning skills [5, 6]. The problem-solving strategies employed in Tangram puzzles can also find application in a range of daily visual tasks linked to abstract diagrams, proving beneficial for design-related problems [7, 8]. These diverse instances showcase the historical and practical significance of Tangram as an impacting and versatile puzzle with enduring relevance.

The task of solving Tangram puzzles relates to a more general class of combinatorial problems, such as the nesting problem and the bin packing problem, which are known to be NP-hard problems [9]. Because of that, it is common for two-dimensional optimization problems to restrict transformations to translations and rotations, and employ a single rectangular container. On the other hand, the Tangram assembly process is considerably more complex because it requires irregular containers, and often demands unconstrained rotations for the pieces. Players can also apply the reflection transformation on the pieces, which is a practice that is almost exclusive to the Tangram [10]. Another property that needs to be considered in the assembly process of the Tangram is that the solution is not necessarily unique, which implies that different arrangements can result in the same pattern [11]. This highlights the inherently combinatorial nature of the Tangram, amplifying the challenge of the solving process, given that diverse strategies can result in distinct solutions. Figure 2 shows a Tangram puzzle with different feasible solutions.

Although methods for generating jigsaw puzzle solutions have made significant progress, particularly with generative models, methods dedicated to Tangram are far more primitive [10, 12]. Deep neural network approaches that solve the jigsaw puzzle rely heavily on semantic information contained in the pieces and often consider square pieces with equal sizes. Both factors significantly alleviate the combinatorial challenges inherent in solving jigsaw puzzles, and inhibit their application on textureless problems, such as the Tangram [13].

This paper presents an investigation into the application of four deep-learning architectures in solving Tangram puzzles: Convolutional Autoencoder (CAE) [14], Variational Autoencoder (VAE) [15], U-Net [16], and Generative Adversarial Network (GAN) [17]. We justify our architectural choices based on their distinct attributes and relevance to the problem domain: CAE excels in feature extraction and reconstruction, VAE provides probabilistic representations, U-Net specializes in semantic segmentation and pixel-wise tasks, while GAN offers generative capabilities. The CAE and VAE are inspired

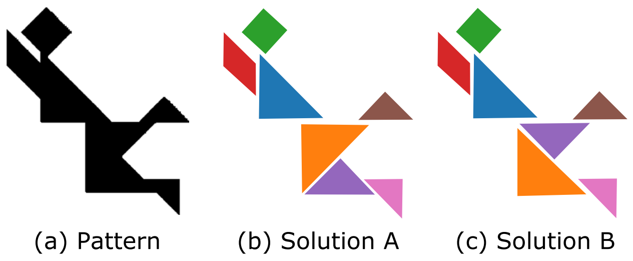


**Fig. 2** Tangram puzzle with different feasible solutions. The same pattern can be obtained by varying the position of the medium triangle and one of the big triangles

by the implementation of Minhas and Zelek [18] for anomaly detection, while U-Net is inspired by the implementation of Zhou et al. [19] for image segmentation. We make our own adjustments to these architectures to fit our task.

Our experiments are divided into two stages. In the first, we evaluate the performance of CAE, VAE and U-Net architectures. Our results show that VAE outperforms the other architectures in the first stage of experiments. For the second, we implement a GAN using the architecture that performs best as the generator, thus forming our VAE-GAN, a GAN encompassing a VAE-based generator. Our experiments indicate that VAE-GAN, paired with our proposed loss function designed for Tangram, presents more refined solutions than the previous architectures. We further analyze whether conventional metrics that are based on pixel accuracy are appropriate to evaluate generated Tangram solutions and propose a novel metric designed to evaluate the visual quality of generated Tangram solutions.

To the best of our knowledge, this study is pioneer in addressing the applicability of different deep-learning architectures in the automatic solution of Tangram puzzles. Therefore, this paper brings the following contributions: (1) comparative analysis of the performance of different deep-learning architectures in solving Tangram puzzles; (2) implementation of a loss function designed for the task of automatizing the solution of Tangram puzzles; (3) assessment of traditional pixel accuracy metrics in evaluating generated Tangram solutions; (4) development of an evaluation metric for evaluating generated Tangram solutions; and (5) new dataset for training and testing neural networks to solve Tangram puzzles.

Section 2 synthesizes the literature on computational Tangram solvers. Section 3 outlines the dataset generation process. Section 4 covers the experimental setup, parameters, metrics and outcomes of our first stage of experiments, while Sect. 5 analogously details our second stage of experiments. Finally, Sect. 6 encompasses our final considerations and future works.

## 2 Related work

The literature to solve different categories of puzzles that do not employ deep-learning techniques is vast. From these methods, we identified two approaches that include extensions to the Tangram [20, 21], and three that focus specifically on the Tangram [10, 22, 23]. Deutsch and Hayes [22] use heuristic programming to split the puzzle into polygons to form the pieces. Bartoněk [20] presents a method based on genetic algorithms for edge-matching puzzles that includes an extension to the Tangram. Kovalsky et al. [21] solve jigsaw puzzles in terms of algebraic concepts, extending its application to Tangram puzzles. Oflazer [23] follows a connectionist approach by representing the placement and orientation of the pieces as a non-restricted Boltzmann machine. Yamada et al. [10] propose a heuristic-based method that uses geometrical techniques commonly applied in the solution of cutting and packing problems. They adopt a raster-based representation that enables the depiction of puzzles with multiple regions and holes. Further, the technique uses mathematical morphology operations to determine if a piece in a certain configuration would fit the puzzle silhouette. A more recent study [24] shows that this heuristic can benefit from different sorting techniques for the pieces.

From deep-learning models that solve other visual tasks, we identified two approaches that include extensions to the Tangram [8, 13]. Li et al. [8] present a GAN architecture that synthesizes layouts by modeling geometric relations of different types of graphical elements. They solve Tangram puzzles as a validation for their approach. Their model generates meaningful solutions like animals and people's poses, although others may be hard to interpret. A drawback of this approach is the limited number of transformations that can be executed on each piece, consisting of only 8 combinations of rotation/reflection. They also limit their tests to only 12 Tangram puzzles. More recently, Lee et al. [13] split different two-dimensional target shapes into multiple fragments of arbitrary polygons by a stochastic partitioning process. They implement a GAN architecture aiming at assembling the target shape given the partitioned fragments while the original poses are hidden. The authors claim that the addressed problem is analogous to the Tangram, although they do not present experiments to sustain this claim. Their tests are limited to regular target shapes, such as squares, pentagons, and hexagons, which do not meet the requirements for modeling Tangram puzzles. We conclude that the application of deep-learning approaches in the solution of Tangram puzzles remains a relatively uncharted field, indicating the need for more comprehensive exploration in this promising area.

One may argue that we could take advantage of deep neural network architectures that solve different categories of puzzles. We have identified a growing interest in solving square jigsaw puzzles using different architectures. When square jigsaw puzzles are treated as an image-to-image translation problem, the model generates each pixel of the output image and has no incentive to transfer a piece unchanged from the input to another location in the output [25]. As a result, the output image often does not accurately represent the rearranged input pieces. In an attempt to tackle this problem, different authors choose to treat the jigsaw puzzle as a piece-to-location mapping problem. The pioneering work in this topic solves the task of predicting the relative position of adjacent fragments given a central fragment, which is managed by learning the high-level semantic information of the data [26]. In recent studies [25, 27–38] the output of the model becomes a placement vector interpreted as the positions of jigsaw pieces or pseudo-labels. The placement vector represents the mapping between each piece and its placement within a grid of possible locations. Different works use jigsaw puzzles as the pretext task for transfer learning and self-supervised learning aiming at solving more complex tasks involving image context [27–29, 33, 39–41]. They push the semantic nature of jigsaw puzzles even further by solving the task where some pieces of a puzzle are replaced with images from other puzzles as a pretext task for self-supervised learning [42]. Further studies add extra complexity to the task by discarding a certain number of pieces, distorting the geometry of some pieces, and changing the color scheme or adding noise to some pieces [28–31, 34, 37, 38, 43, 44]. Regarding limitations in the area, many works are limited to square puzzles, where the tests are often restricted to puzzles with only a few pieces [45]. These approaches are not adequate for the Tangram because they rely heavily on the semantic information contained in each piece to assemble the puzzle. Unlike square jigsaw puzzles, Tangram pieces have distinct geometries, and mapping every possible placement for each piece in the puzzle is impracticable in a reasonable time. Still in contrast to jigsaw puzzles, the Tangram does not suffer from the issue where the output image often does not accurately represent the rearranged input pieces, as the textureless nature presented by the Tangram eliminates the need for forming a coherent picture in the solution.

The literature shows that there are two distinct approaches to modeling and solving the Tangram. Some authors treat the puzzle as a sectioning problem [22, 23] where they attempt to section the pattern into the desired pieces. Other authors treat it as a placement problem [8, 10, 13, 20, 21] where the main idea is to find the exact position of the pieces that form the desired pattern. Both approaches are valid given the rules and goal for assembling Tangram puzzles. In this paper, we opt for using a visual representation of the Tangram, where the aim is for the deep-learning models to learn the right places to section the input pattern, thus outputting an image that depicts a Tangram solution.

## 3 Dataset

Large training datasets and deep complex structures enhance the ability of deep-learning models to learn effective representations for tasks of interest [46]. If the model is fed with poor or insufficient data, it might be unable to generalize accurately [47]. In this scenario, even though it can make accurate predictions for previously seen training data, when tested for new data there is a risk that it will infer inaccurate predictions. Therefore, when assembling a data-driven dataset, it is important to ensure that it contains a diverse and representative set of samples that are accurately consistent with the task of interest. This is especially important for Tangram puzzles since each piece can assume an uncountable number of configurations, resulting in a wide variety of different patterns that can be formed with these pieces.

The challenge in assembling a Tangram dataset lies in determining the adequate number and variability of samples essential for achieving generalization. Consequently, our approach involves not only gathering samples from existing literature but also supplementing the dataset with additional randomly generated samples. Therefore, we collect these data for 182 puzzles from the literature, and 752 randomly generated puzzles, thus accounting for 934 samples. To generate random samples, we use a random Tangram generator implemented in Javascript by Köpp [48]. The inclusion of random samples serves to augment the variability in our data-driven dataset, aiding the model in learning the intricate geometry of Tangram pieces and understanding their interactions to form different arrangements.

Each sample is composed of a $256 \times 256$ grayscale image depicting the puzzle and a $256 \times 256$ grayscale image representing a single feasible solution as its ground truth. We consider that this size is sufficient for clearly depicting the pixels that represent the sections between pieces, thus enabling the deep-learning models to learn how to determine where the puzzle should be cut to generate a solution. Figure 3 depicts the samples of our dataset.

We guarantee that our data puzzles have a similar area by making the pieces the same size. This aspect of our dataset benefits the model convergence since it reduces the necessity of the model to estimate the size of the pieces during the assembly process. We split our dataset into 888 samples for the training set and 46 samples for the testing set, which results in an approximate 95:5 split ratio.

Our dataset is available in our Github repository [49], serving as a useful resource for research and evaluation in dissection puzzles, as well as related optimization problems. To the best of our knowledge, our dataset is the most extensive in the literature regarding the automatic solution of Tangram puzzles.

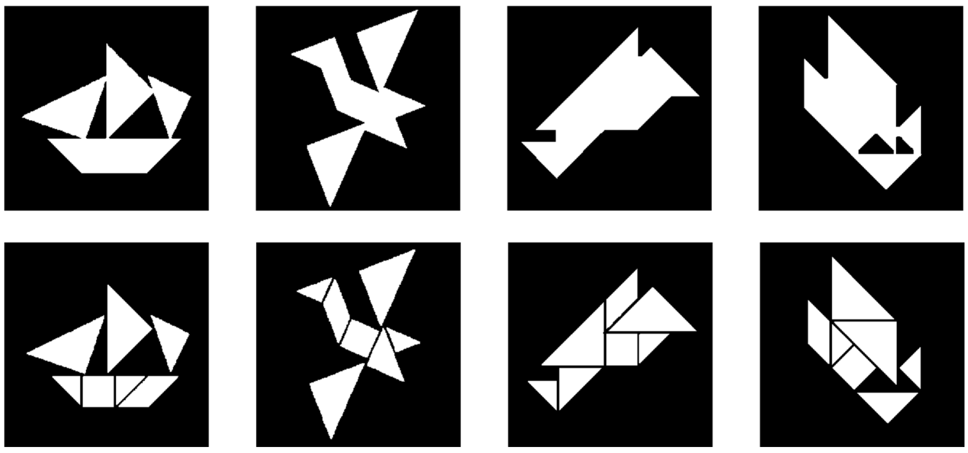


**Fig. 3** Samples included in our dataset. First row presents the Tangram patterns, and second row presents their respective ground truth solutions

## 4 Tangram solvers based on autoencoders

The input of the implemented architectures is 256 × 256 grayscale images. The output also follows the same pattern. Figure 4 presents a flowchart illustrating the workflow of our training approach. The deep-learning architecture block can be substituted by any of the architectures described in Sect. 4.1. It is worth mentioning that this process is done only for the training to increase the variability of our data.

For each iteration, an input batch is selected from our dataset. Then, Gaussian noise is applied on the input batch images as a form of regularization to prevent overfitting and to encourage the model to learn more robust features [50]. We also use online dataset augmentation on the input batch to further improve the accuracy and robustness of the model [51]. By definition, any pattern formed by the seven Tangram pieces is a Tangram puzzle, thus we take advantage of this concept to create variations of the samples we consider in our dataset. We apply a random rotation to the input batch images, ensuring the rotation angle stands at a multiple of 90°. We also randomly reflect the image in the vertical and horizontal axis.

### 4.1 Network architectures

Figure 5 presents the deep-learning architectures we use in the present study. A detailed description over the implementation of these architectures is available in our Github repository [49].

#### 4.1.1 CAE architecture

The concept of autoencoders, including CAE, has a history dating back to the early days of artificial neural networks. Autoencoders, in their basic form, were proposed in the 1980 s as a neural network architecture for dimensionality

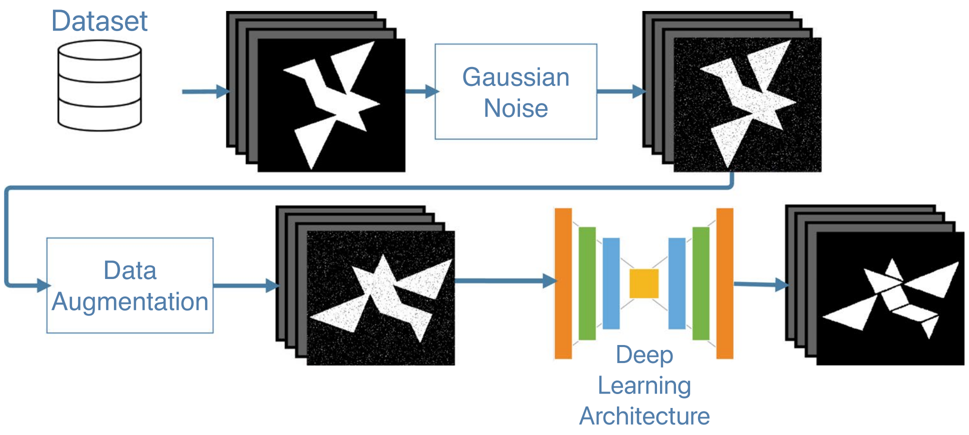


**Fig. 4** Testbed workflow. Gaussian noise and data augmentation are applied on each batch before inputting it into the deep-learning architecture

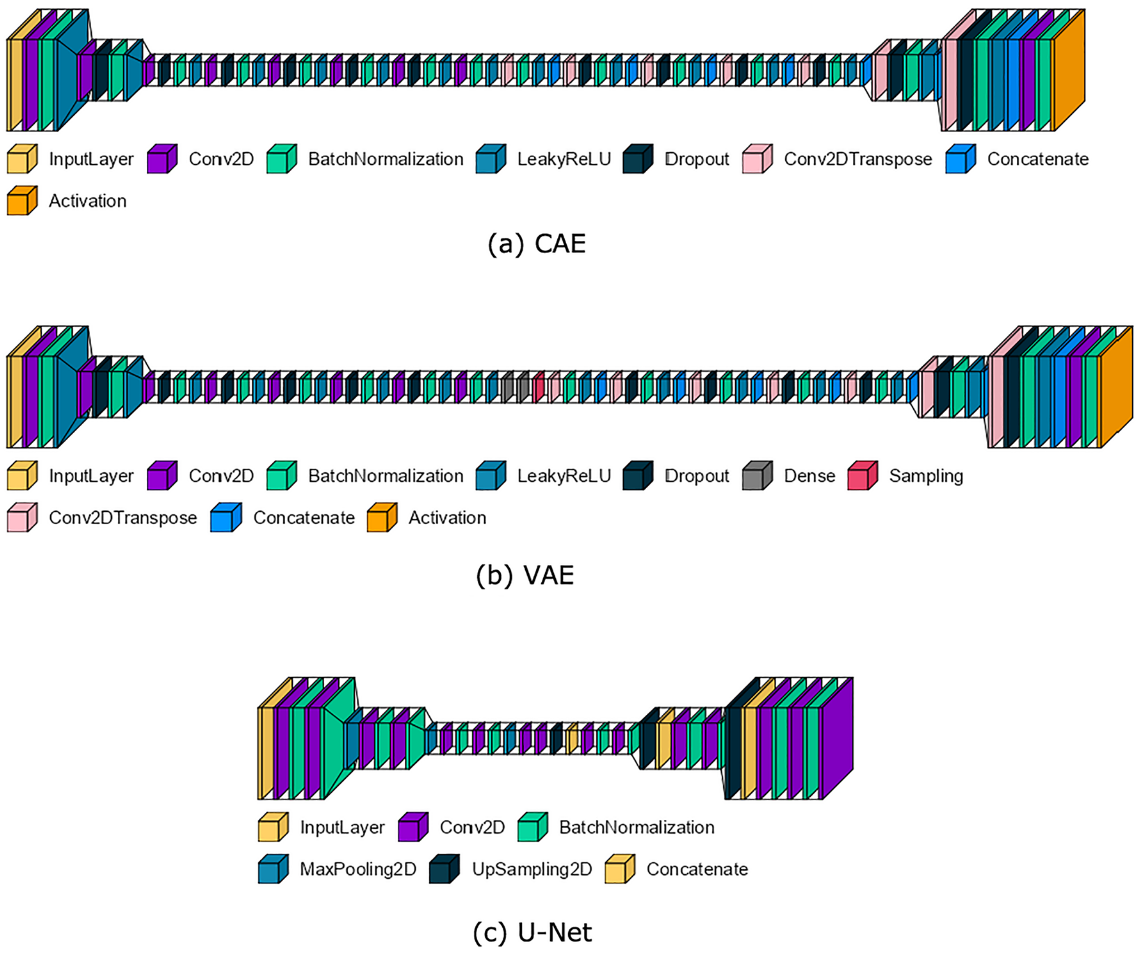


**Fig. 5** Diagrams illustrating the deep-learning architectures we employed in our study. Each layer type is assigned a different color. **a** Diagram of CAE architecture: it consists of an encoder and a decoder with a latent space connecting them. **b** Diagram of VAE architecture: it consists of an encoder and a decoder with a stochastic latent space connecting them. **c** Diagram of U-Net architecture: this architecture presents a U-shaped design, which consists of a contracting path and an expansive path

reduction and feature learning. They were initially used for visual tasks, including data compression and noise reduction [52].

In the CAE architecture presented in Fig. 5a, the primary objective is to section the Tangram pattern following the contour of each piece, therefore revealing the final solution as an image. The CAE model consists of an encoder network that extracts informative features from the Tangram pattern, followed by a decoder network that constructs the Tangram solution. The CAE architecture we implement counts with 8 convolutional layers in the encoder, and 8 deconvolutional layers in the decoder, which permit the network to capture intricate details in both the encoding and decoding stages. In the encoder phase, the network gradually diminishes the feature map dimensions, initiating from (256, 256, 1) and concluding at (4, 4, 48). The encoder component of the CAE is composed of a sequence of convolutional layers, each complemented by batch normalization and LeakyReLU activation functions. The encoder should extract intricate image structures and patterns while simultaneously reducing the dimensionality of the input data.

On the other side of the CAE, the decoder phase mirrors the structure of the encoder, incrementally expanding the encoded feature maps back to (256, 256, 1). Composed of transposed convolutional layers, the decoder constructs the Tangram solution from the latent representation generated by the encoder. Aiming for generalization, dropout, and batch normalization techniques are also integrated into the decoder. These techniques ensure the CAE learns to adapt to various scenarios while maintaining robustness. The CAE encompasses a total of 3,115,301 parameters, of which 3,113,859 are trainable, and 1,442 are non-trainable.

#### 4.1.2 VAE architecture

The VAE architecture, introduced by Kingma and Welling [15], combines the principles of autoencoders and probabilistic graphical models to perform unsupervised learning and generate data that adheres to a specific probability distribution, typically a Gaussian distribution in the latent space. This approach allows for more structured and controllable data generation, making this architecture particularly well-suited for tasks such as image generation, data denoising, and generating novel samples from learned representations.

The VAE architecture presented in Fig. 5b is specifically designed to extract meaningful latent representations from the Tangram pattern images while enabling the generation of Tangram solution images from these learned representations. Similarly to the CAE, the model is composed of an encoder and a decoder. The VAE architecture we employ consists of a total of 22 layers, including 11 convolutional and 11 deconvolutional layers. In the encoder phase, the network progressively reduces the feature map dimensions, starting from (256, 256, 1) and ending at (2, 2, 48). The encoder processes Tangram pattern images through a sequence of convolutional layers, each followed by batch normalization and LeakyReLU activation functions. This architecture allows the encoder to discern intricate patterns and features within the input images while simultaneously reducing the dimensionality. This ultimately results in a condensed latent representation. The VAE incorporates two additional dense layers, at the end of the encoder to facilitate the stochastic nature of VAEs, determining the statistical properties of our latent space [53]. Subsequently, a sampling operation is applied to generate latent vectors that follow a Gaussian distribution, using the mean and log-variance computed by the dense layers. We sample from a single Gaussian distribution for each input in the batch. The decoder phase mirrors the encoder, incrementally expanding the feature maps back to (256, 256, 1).

The decoder employs transposed convolutional layers to upsample the latent vectors back into image dimensions, forming the Tangram solution image from the latent space representations. Batch normalization and LeakyReLU activation functions complement the transposed convolutional layers. Ultimately, the output layer of the decoder produces Tangram solution images that are the same size and channel depth as the input Tangram pattern images. The VAE encompasses a total of 3,060,297 parameters, with 3,058,855 being trainable and 1,442 non-trainable parameters. Similarly to CAE, the VAE architecture also aims to minimize a loss function throughout the training process.

#### 4.1.3 U-Net architecture

The U-Net architecture, introduced by Ronneberger et al. [16], is a U-shaped architecture, consisting of a contracting path for feature extraction and an expansive path for precise pixel-wise segmentation. U-Net is widely used for semantic segmentation tasks, particularly in the field of biomedical image analysis. It is also acclaimed for its ability to capture fine-grained details in images.

The U-Net architecture presented in Fig. 5c is characterized by its symmetric encoder-decoder structure, which enables it to capture intricate image features while preserving spatial information [54]. The U-Net comprises a total of 29 layers, including 14 convolutional layers and 15 deconvolutional layers. It starts with an input shape of (256, 256, 1) and progressively reduces the feature map dimensions through the use of max-pooling layers, resulting in a feature map size of (32, 32, 512). Each block of the contracting path consists of two consecutive convolutional layers, followed by batch normalization and ReLU activation functions. The pooling layers, interspersed between these blocks, progressively reduce the spatial dimensions of the feature maps, facilitating the extraction of hierarchical features.

Subsequently, deconvolutional layers are employed to increase the feature map dimensions back to (256, 256, 1). The architecture strategically incorporates concatenation layers to merge feature maps from both the contracting and expanding paths, ensuring a detailed and accurate representation. In total, the U-Net architecture presents 7,788,929 parameters, with 7,785,345 of them being trainable and 3584 non-trainable.

### 4.2 Loss functions

We use the following loss functions in our experiments: (1) mean square error ($Loss_{MSE}$), and (2) structural similarity ($Loss_{SSIM}$). The $Loss_{MSE}$ encourages the model to minimize the magnitude of errors, making it suitable for tasks where precise numeric predictions are essential. In our case, it is desired to obtain a solution where the area covered by pieces is easily distinguishable from the background and sections between pieces. $Loss_{SSIM}$ is a perceptual loss function designed for image processing applications. It evaluates the structural similarity between the generated and reference images, considering luminance, contrast, and structure. $Loss_{SSIM}$ encourages the model to produce visually similar outputs, making it valuable in image generation and restoration tasks where perceptual quality is critical.

#### 4.2.1 Mean square error loss function

MSE, a commonly used loss function in image reconstruction tasks, measures the pixel-wise discrepancy between predicted and ground truth images. The literature suggests that MSE tends to prioritize pixel-level accuracy, which may produce visually unsatisfactory results, especially when dealing with complex image structures or textures [55]. The following formula presents MSE as a dissimilarity loss function:

$$Loss_{MSE}(I_x, I_y) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(I_x(i,j) - I_y(i,j))^2, \tag{1}$$

where $Loss_{MSE}(\boldsymbol{I}_x, \boldsymbol{I}_y)$ quantifies the dissimilarity between two images $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$. Since $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$ have the same dimensions, $MN$ represents the total number of pixels in these images, with $M$ representing height and $N$ representing width.

#### 4.2.2 Structural similarity loss function

SSIM is designed to capture not only pixel-level differences but also structural and perceptual similarities between images. It is expected that SSIM loss encourages the preservation of structural information and leads to visually more pleasing reconstructions [56]. The SSIM calculation is done as follows:

$$SSIM(I_x, I_y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{2}$$

where $SSIM(\boldsymbol{I}_x, \boldsymbol{I}_y)$ calculates the similarity between images $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$. Variables $\mu_x$ and $\mu_y$ represent the means of the pixel intensities in images $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$, respectively. Additionally, $\sigma_x^2$ and $\sigma_y^2$ correspond to the variances of pixel intensities in images $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$, while $\sigma_{xy}$ denotes the covariance of pixel intensities between these two images. The constants $C_1$ and $C_2$ are small values introduced to prevent division by zero errors and enhance the stability of the loss calculation.

Since we are working with dissimilarity metrics, and SSIM calculates a ratio expressing the similarity of a pair of images, we need to adapt it to transform it into a loss function:

$$Loss_{SSIM}(I_x, I_y) = 1 - SSIM(I_x, I_y), \tag{3}$$

where $Loss_{SSIM}(\boldsymbol{I}_x, \boldsymbol{I}_y)$ calculates the similarity between images $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$ based on the SSIM metric.

### 4.3 Evaluation metrics

For evaluating the performance of our architectures, we use the following evaluation metrics: (1) MSE and (2) SSIM metrics. Our objective is to analyze if conventional metrics based on pixel accuracy are effective in evaluating the performance of deep-learning models that aim to solve Tangram puzzles. For SSIM, we use the adapted calculation that transforms it into a metric of dissimilarity, so it can be compared to the other two evaluation metrics. Therefore, the formula for MSE and SSIM are analogous to the ones described by Eqs. (1) and (3), respectively. Our hypothesis is that conventional metrics fall short in capturing nuanced aspects when comparing the Tangram solution images with ground truth images. Since Tangram pattern images are visually similar to the ground truth images, the generated solution images also end up presenting a close visual resemblance to the ground truth image. This may happen even when the obtained solution is not correct, thus making it difficult to differentiate a correct solution from an incorrect one.

### 4.4 Experimental results

The architectures are implemented in Python 3.9.12 using Tensorflow 2.11.0 library. We run all tests on a Ryzen 3700x 3.6GHz 32GB of RAM with a Nvidia RTX 4090 24GB.

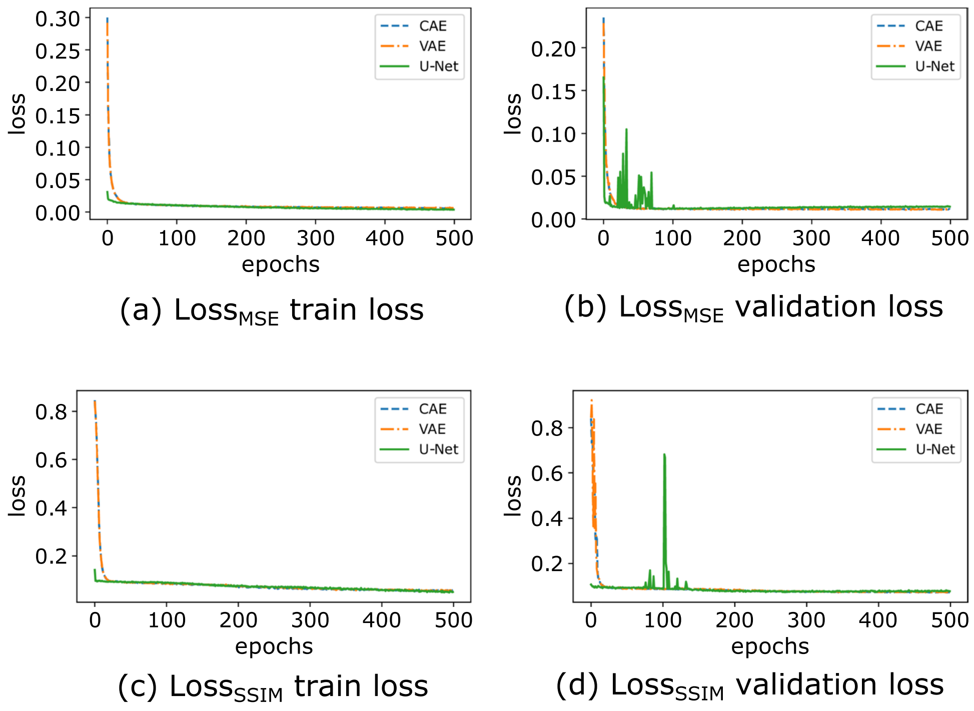


**Fig. 6** Training and validation loss curves for the CAE, VAE, and U-Net when they are submitted to $Loss_{MSE}$ and $Loss_{SSIM}$. Training loss graphs present smooth curves for all architectures. Validation curves for U-Net fluctuate in early training epochs due to parameter adjustment

**Table 1** Experimental results according to evaluation metrics. The best combinations of architecture and loss function for each evaluation metric are highlighted

| Evaluation Metric | Architecture | $Loss_{MSE}$ | $Loss_{SSIM}$ |
|---|---|---|---|
| MSE | CAE | **0.0377** | 0.0559 |
| | VAE | 0.0415 | 0.0578 |
| | U-Net | 0.0579 | 0.0582 |
| SSIM | CAE | 0.1568 | 0.0548 |
| | VAE | 0.1622 | **0.0547** |
| | U-Net | 0.0653 | 0.0569 |

We start by analyzing the loss curves generated throughout the training stage. Figure 6 presents the loss curves for the CAE, VAE, and U-Net when they are submitted to $Loss_{MSE}$ and $Loss_{SSIM}$. The three models converge after a few hundred epochs. The validation curves for U-Net present a significant oscillation in early training stages. This is due to the total number of parameters of this architecture being more than double when compared to CAE and VAE. In early stages of training, U-Net is still adjusting its parameters searching for a fair representation of the data. These random fluctuations lead to oscillations in the validation loss curve as the model searches for optimal parameters.

Additionally, we inspect the output solutions of our trained models when submitted to our testing set and analyze the visual quality of the generated solutions according to conventional metrics based on pixel accuracy. Table 1 shows the results we obtained for our experiments over the aforementioned evaluation metrics and Fig. 7 presents 15 inferences of the trained CAE, VAE, and U-Net on our testing set when combined with $Loss_{MSE}$ and $Loss_{SSIM}$. The first two columns in Table 1 define the evaluation metric and the architecture for each experiment. The next two columns present the average values for $Loss_{MSE}$ and $Loss_{SSIM}$. It is worth noticing that the models apply the same technique of segmenting the Tangram puzzle area following a triangular grid before deciding the sections that represent the contact between pieces as shown in Fig. 7. The employed strategy aligns with a geometrical property of Tangram pieces, which tells that they can all be decomposed as a combination of the small triangular piece [5, 57]. Therefore, the models can learn this property even though it is not directly informed to them, which indicates that they can extract valuable information regarding the geometry of the pieces.

Table 1 shows that for MSE evaluation metric, the architecture that presented the best performance is CAE. Although CAE is the architecture that paired up the best with $Loss_{MSE}$, this combination does not present a clear definition of the sections between pieces and contains several blurred section lines that compromise defining the correct position of the

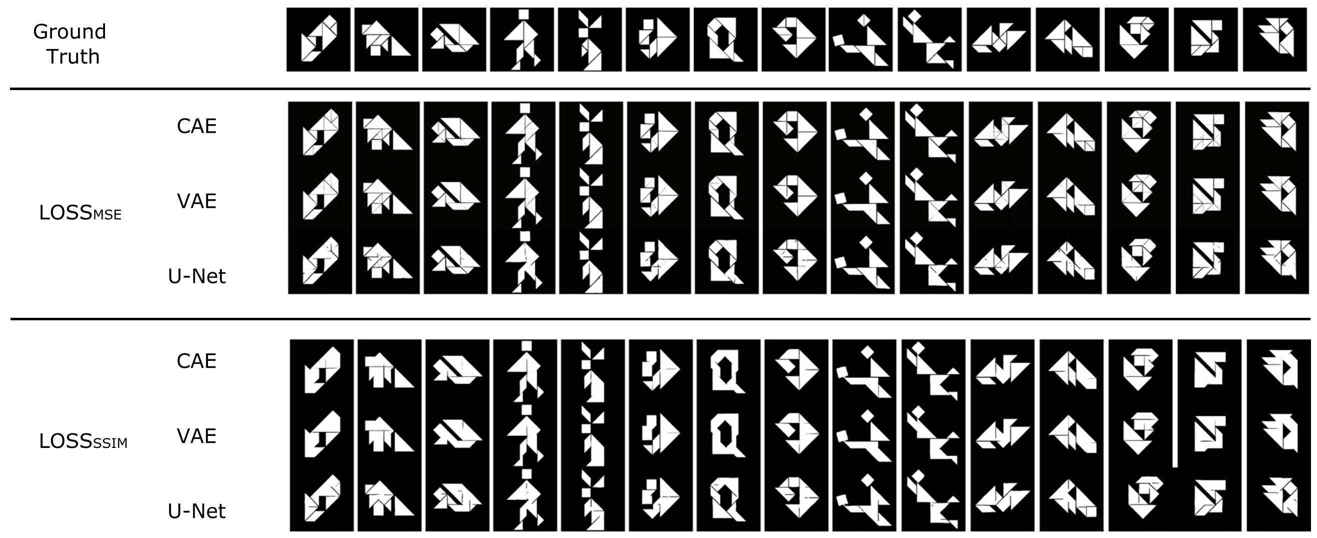


**Fig. 7** Solution images generated by CAE, VAE, and U-Net paired with $Loss_{MSE}$ and $Loss_{SSIM}$

pieces. For $Loss_{MSE}$, it tends to produce overly smooth and blurred results in the context of generative models as shown in Fig. 7. This occurs because $Loss_{MSE}$ is sensitive to both small and large errors, which places equal emphasis on all pixels.

SSIM evaluation metric considers that VAE presented the best performance, although CAE closely matches it. However, from Fig. 7, we notice that both CAE and VAE perform poorly when combined with $Loss_{SSIM}$. Additionally, it is possible to notice that $Loss_{SSIM}$ performs poorly in identifying the correct sections between pieces, thus generating solution images closer to an initial pattern than to a proper solution. In our experiments, we use images with only black and white pixels. It is known that $Loss_{SSIM}$ may not effectively capture the subtle structural differences in such images, since it relies on local patterns and variations in pixel values to assess their visual similarity [56]. In those experiments, the models miss a considerable amount of sections between pieces, generating an output solution that shares more visual similarities with a Tangram pattern than a Tangram solution. To further support our claim, Fig. 8 shows cases where the MSE and SSIM metrics fail to correctly indicate the visual quality of a solution. In the example presented in the first row, it is clear that even though SSIM suggests that the solution image (b) is more similar to the ground truth (a), it is easier to identify the position of the Tangram pieces by looking at the solution image (c). The same can be said about the example presented in the second row, where MSE fails to identify that the image solution (f) presents more consistent sections than (e) when compared to ground truth (d).

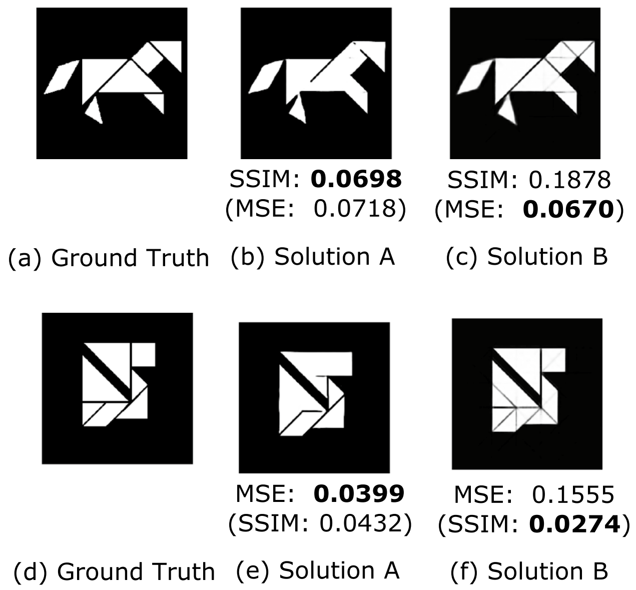


**Fig. 8** Cases where SSIM and MSE fail in evaluating pairs of Tangram solutions. **a** ground truth; **b** Generated using U-Net with $Loss_{SSIM}$; (c) Generated using CAE with $Loss_{MSE}$; (d) ground truth; (e) Generated using U-Net with $Loss_{SSIM}$; (f) Generated using VAE with $Loss_{MSE}$

When analyzing the generated Tangram solutions, an important consideration emerges when deciding between solutions that contain extra sections or those with missing ones. In this context, we prioritize solutions with additional sections over those with omissions. We believe that inferring the correct position of the pieces from an image with extra sections is considerably less challenging than discerning the precise location of missing cuts. This is especially true when the player has some familiarity with the Tangram pieces. We conclude that solution images with extra sections offer a greater degree of interpretability, therefore being closer to the correct solution, at least for the Tangram. Figure 9 presents a visualization of false positives and false negatives in solution images. False positive represents missing sections and false negative shows extra ones. A visual inspection on Fig. 7 shows that both CAE and VAE combined with $Loss_{MSE}$ present extra sections rather than omissions, making it easier to infer the positions of the pieces. However, it is also possible to observe that the solutions generated by VAE are less blurry than the ones presented by CAE, making it easier to comprehend the inferences of the model towards the location of sections.

In summary, we observe that CAE, VAE, and U-Net architectures reveal the limitations of existing evaluation metrics in adequately preserving the perceivable geometrical properties of Tangram pieces. The conventional metrics fail to capture the nuanced intricacies of Tangram shapes, leading to suboptimal results. VAE, however, presents promising solutions by generating well-defined sections between pieces and presenting extra sections rather than omissions. Although $Loss_{MSE}$ is sensitive to both small and large errors because it places equal emphasis on all pixels, $Loss_{MSE}$ can generate an output solution that shares more visual similarities with a Tangram pattern than a Tangram solution comparing with $Loss_{SSIM}$ which misses a considerable amount of sections between pieces. In response to this discussion, in the second part of our experiments, we propose a novel loss function and evaluation metric specifically designed for the perception of geometric characteristics of Tangram solutions. We aim to enhance the fidelity of our models in preserving the crucial geometrical properties of Tangram pieces in generated solutions. Since we cannot rely on conventional evaluation metrics to assess the generated solutions, through a visual inspection, we ultimately decide to employ a novel evaluation metric appropriate for the novel loss function and use VAE architecture as the generator of our GAN in the second stage of experiments. Our hypothesis is that, given the competitive relationship between the generator and discriminator, the discriminator indirectly guides the generator to make decisions towards which generated sections should be retained to represent the boundaries of Tangram pieces. Consequently, it is expected that VAE-GAN will produce solutions that visually closely resemble real Tangram solutions.

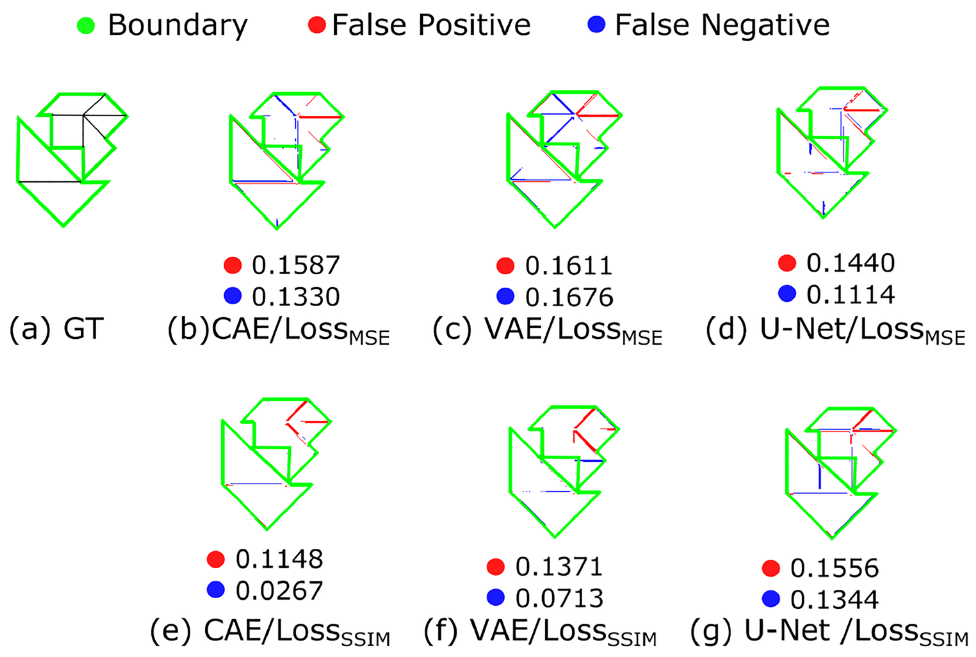


**Fig. 9** Visualization of false positives and false negatives in solution images. Solution images were generated using CAE, VAE, and U-Net with $Loss_{MSE}$ and $Loss_{SSIM}$. Values are normalized by the number of pixels in pieces sections presented in black in the ground truth

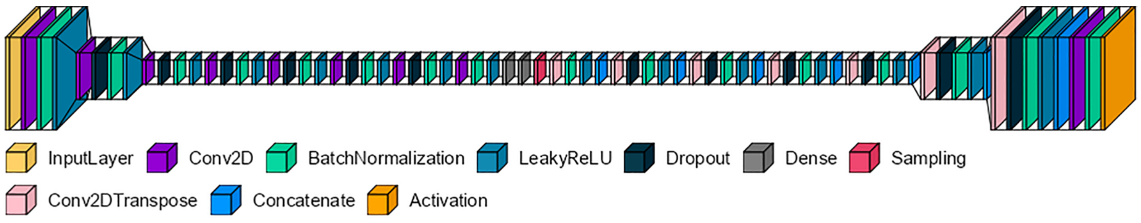
## 5 VAE-GAN model for refinement of tangram geometry perception

The workflow of the second part of experiments is analogous to the flowchart presented in Fig. 4. The input is composed of 256 × 256 grayscale images. The output also follows the same pattern. The deep-learning architecture is substituted by our VAE-GAN architecture. We employ Gaussian noise and online data augmentation to improve the robustness of our model.

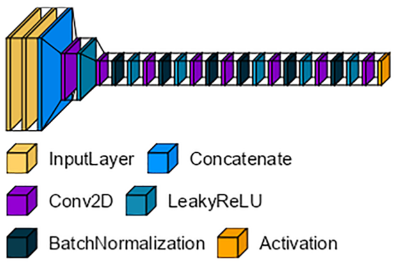
### 5.1 Network architecture

GAN is a class of generative models introduced in 2014 by Goodfellow et al. [17]. It started being used in 2017 with human faces to adopt image enhancement that produces better illustrations at high intensity [58, 59]. A basic GAN architecture is formed by a generator and a discriminator. These are typically implemented using neural networks but could be implemented using any form of differentiable system that maps data from one space to another [60]. As previously mentioned, we implement a VAE-GAN using the aforementioned VAE architecture combined with our discriminator, which is detailed in the following paragraphs. Figure 10 presents the generator and the discriminator we employ. A detailed description of the implementation of these architectures is available in our Github repository [49]. Our discriminator comprises multiple convolutional layers, progressively increasing the number of filters. This depth allows the network to capture intricate and hierarchical features within input images, making it highly effective for discerning ground truth from fake images [60].

The GAN discriminator architecture we implement is presented in Fig. 10b, consisting of a total of 8,276,801 parameters, with 8,273,217 parameters being trainable and 3584 non-trainable parameters. The network begins with two input layers for image pairs with shapes (256, 256, 1), which are concatenated into a single (256, 256, 2) input. The network aims at successively reducing the spatial dimensions of feature maps as it processes images, allowing it to focus on capturing abstract and high-level features for image discrimination. To accomplish that, a series of convolutional layers progressively reduce the feature map dimensions from (256, 256, 2) to (4, 4, 512). Batch normalization and LeakyReLU activation functions are applied at each convolutional layer. Batch normalization technique accelerates convergence, mitigates gradient-related issues, and enhances the overall robustness. LeakyReLU facilitates faster convergence during training, and introduces non-linearity, enabling the network to learn complex decision



(a) VAE-based Generator



(b) Discriminator

**Fig. 10** Diagrams illustrating components of our GAN architecture we employed in our study. Each layer type is assigned a different color. **a** Diagram of VAE-based generator: it consists of an encoder and a decoder with a stochastic latent space connecting them; **b** Diagram of discriminator: as the data progresses through the network, it reduces the spatial dimensions and abstracts the information, and it ends in a sigmoid activation function that determines if the input is fake or not

boundaries and critical aspect of distinguishing real and fake images. The final convolutional layer reduces the feature maps to (4, 4, 1). A sigmoid activation function in the output layer squashes the output into a boolean value, which indicates if the input either is a generated solution or a ground truth.

### 5.2 Weighted mean absolute error loss function

Weighted Mean Absolute Error (WMAE) is a variation of Mean Absolute Error (MAE), with different elements of the error being assigned distinct weights based on their sign and magnitude. It takes inspiration from other versions of weighted versions of MAE [61, 62]. It also mixes these inspirations with Weighted Mean Squared Error [63]. In WMAE, positive errors are scaled by a factor of $c$, while negative errors are treated with a weight of 1. Its concept is inspired by our observation that $Loss_{MSE}$ demonstrates promising results in the task of segmenting Tangram patterns into distinct pieces. However, the sections produced by $Loss_{MSE}$ tend to be blurry and poorly defined. This inspire us to implement $Loss_{\mathrm{WMAE}}$, a loss function based on WMAE, designed to enhance the precision and clarity of sections by assigning variable weights to false positive and false negative errors.

First, both input images $\boldsymbol{I}_x$ and $\boldsymbol{I}_y$ of the same dimensions are normalized to a range between 0 and 1. Their normalized counterparts are defined as $\boldsymbol{I}_x^{norm}$ and $\boldsymbol{I}_y^{norm}$ respectively. Thus, we have the conditions that $0 \leq \boldsymbol{I}_x^{norm}(i,j) \leq 1$ and $0 \leq \boldsymbol{I}_y^{norm}(i,j) \leq 1$ for every coordinate $(i, j)$ present in $\boldsymbol{I}_x^{norm}$ and $\boldsymbol{I}_y^{norm}$. Then, the absolute difference between them, denoted as $\Delta$, is calculated element-wise following the Euclidean norm:

$$\Delta = \| I_y^{norm} - I_x^{norm} \|_2. \tag{4}$$

The next stage is the conditional weighting step. Elements of $\Delta$ are analyzed, and weight term $W$ given by:

$$W = \begin{cases} \boldsymbol{I}_y^{norm} \cdot \mathrm{c} + 1, & \text{if } \Delta > 0 \\ 1, & \text{if } \Delta < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

With the conditional weight in place, the WMAE loss function computes the weighted absolute error, $L$, by element-wise multiplication of the absolute error $\Delta$ and the conditional weights $W$:

$$L = |W \cdot \Delta|. \tag{6}$$

Finally, the following equation treats WMAE as a dissimilarity loss function:

$$Loss_{\mathrm{WMAE}}(I_x, I_y) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} L(i,j). \tag{7}$$

In summary, $Loss_{\mathrm{WMAE}}$ attributes a bigger weight when a pixel that is black is assigned as white, than when the opposite occurs. Doing so, the loss prioritizes the pixels that represent the section between pieces over pixels that represent the area covered by pieces. It is designed to impose a higher penalty on false positives compared to false negatives, which are illustrated in Fig. 9.

### 5.3 Weighted mean absolute error evaluation metric

We extend the utility of $Loss_{\mathrm{WMAE}}$ by re-purposing it as an evaluation metric. By employing the same loss function for assessment, we establish a unified and consistent approach to both training and evaluating our generated solutions. When the metric used for evaluation mirrors the criteria set during training, it ensures that the evaluation process accurately reflects the objectives and criteria that guided the model during training. Thus, our WMAE evaluation metric follows Eq. (7). Our hypothesis is that our WMAE metric is better aligned with the task of evaluating Tangram solutions than traditional metrics, such as MSE and SSIM.

## 5.4 Experimental results

The VAE-GAN architecture is also implemented by the same environment described in Sect. 4.4. We perform experiments using the VAE-GAN combined with $Loss_{\mathrm{WMAE}}$ to show the efficiency of our architecture. Figure 11 depicts the training process of VAE-GAN. It can be noticed an abrupt decrease in training and validation loss curves in early stages of training, while for the generator loss curve, the decrease is more gradual. Comparing Figs. 6 and 11, we observe that VAE-GAN converges faster than the previous architectures and that the VAE-based generator is in fact learning from the judgment of the discriminator. It is also possible to notice that, although the VAE-GAN is using $Loss_{\mathrm{WMAE}}$, we do not observe the fluctuations in the validation curve that are shown in the first part of our experiments. This shows that the discriminator regularizes the training process and leads to smoother loss curves for the VAE-based generator. In other words, it clearly indicates that the incorporation of a discriminator is beneficial because the competitive nature of the GAN stabilizes the training dynamics of the VAE-based generator, making it less susceptible to fluctuations compared to a standalone model.

To support our claim that $Loss_{\mathrm{WMAE}}$ is better suited for solving Tangram puzzles than $Loss_{MSE}$ and $Loss_{SSIM}$, we extended our experiments to architectures CAE, VAE, and U-Net. This extension enables the validation towards the effectiveness of $Loss_{\mathrm{WMAE}}$ across a diverse range of architectures. Thus, Fig. 12 presents 15 inferences of the trained CAE, VAE, and U-Net on our testing set when combined with $Loss_{\mathrm{WMAE}}$. We also compare these solutions with the ones generated by CAE with *Loss MSE* and VAE with $Loss_{SSIM}$. This pair of combinations are the ones that MSE and SSIM metrics judged presenting the best performance in Sect. 4. We notice that the solutions generated using $Loss_{\mathrm{WMAE}}$ present better defined sections when compared to the solutions generated by CAE with *Loss MSE* and VAE with

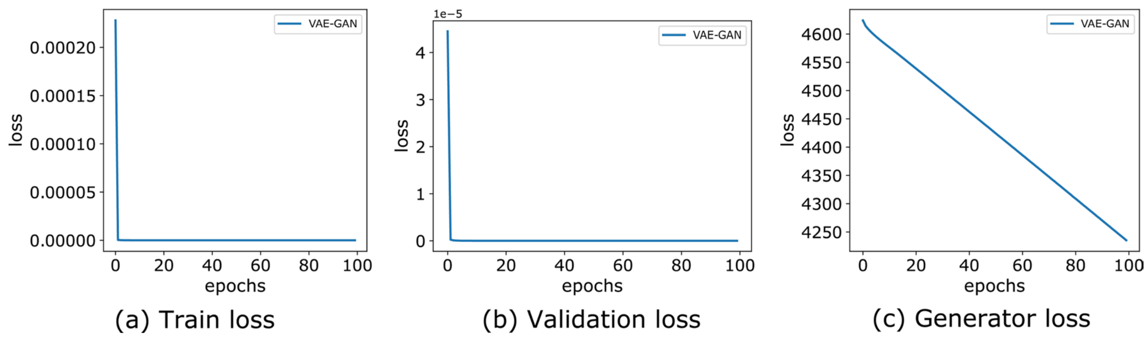


**Fig. 11** Training, validation, and generator loss curves for VAE-GAN. Smooth curves indicate effective learning and stable convergence

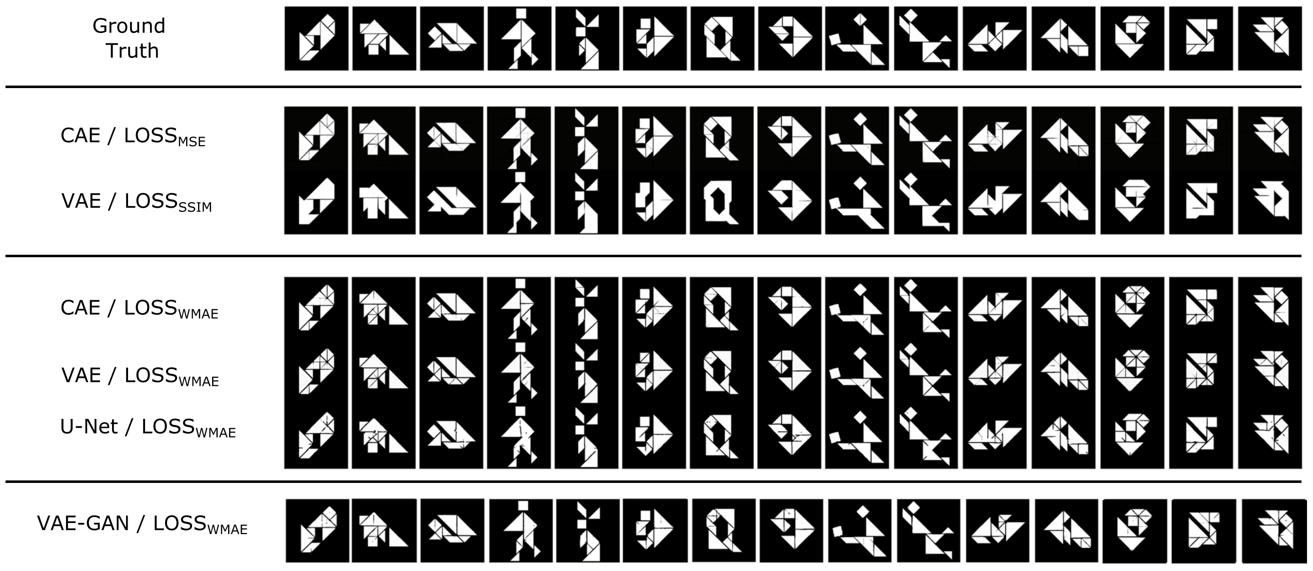


**Fig. 12** Solution images generated by CAE, VAE, U-Net, and VAE-GAN paired with $Loss_{\mathrm{WMAE}}$

$Loss_{SSIM}$. Also, when compared to the standalone VAE, VAE-GAN generates solutions that are closer to the expected ground truth. This shows that the discriminator successfully fulfills its role of pushing the VAE-based generator to produce images that are closer to real Tangram solutions.

We notice that $Loss_{\mathrm{WMAE}}$ presents better qualitative results than the other loss functions. It tends to generate clear extra sections than omissions, making it easier to infer the positions of the pieces. Additionally, VAE-GAN presents solutions similar to the ones by VAE, but with fewer extra sections, thus showing that it discerns better which sections are necessary to form a solution image for Tangram. Even when the sections between pieces are not perfectly depicted in the solution image, a human with familiarity with the Tangram pieces would be capable of inferring the correct position and configuration of the pieces. Furthermore, the tenth column of the figure shows a case where the generated solution by VAE-GAN is feasible, although different from the ground truth, indicating that VAE-GAN can learn geometric features of the Tangram pieces. Figure 13 presents a visualization of false positives and false negatives in solutions generated for the same Tangram pattern. The ratio of false negatives in (e) is noticeably lower than the same ratio in (c). This indicates that the incorporation of the discriminator into VAE, resulting in VAE-GAN, leads to a significant improvement in the model toward discerning which sections are necessary to form a solution image for Tangram. Moreover, a comparison with Fig. 9 reveals that the ratio of false positives reduces expressively with $Loss_{\mathrm{WMAE}}$. For the presented scenarios, the ratio of false positives for VAE with $Loss_{\mathrm{WMAE}}$ is close to 1/10 of the ratios obtained for $Loss_{MSE}$ and $Loss_{SSIM}$. These observations align with our hypothesis regarding the effects of VAE-GAN and $Loss_{\mathrm{WMAE}}$ on the generated solutions.

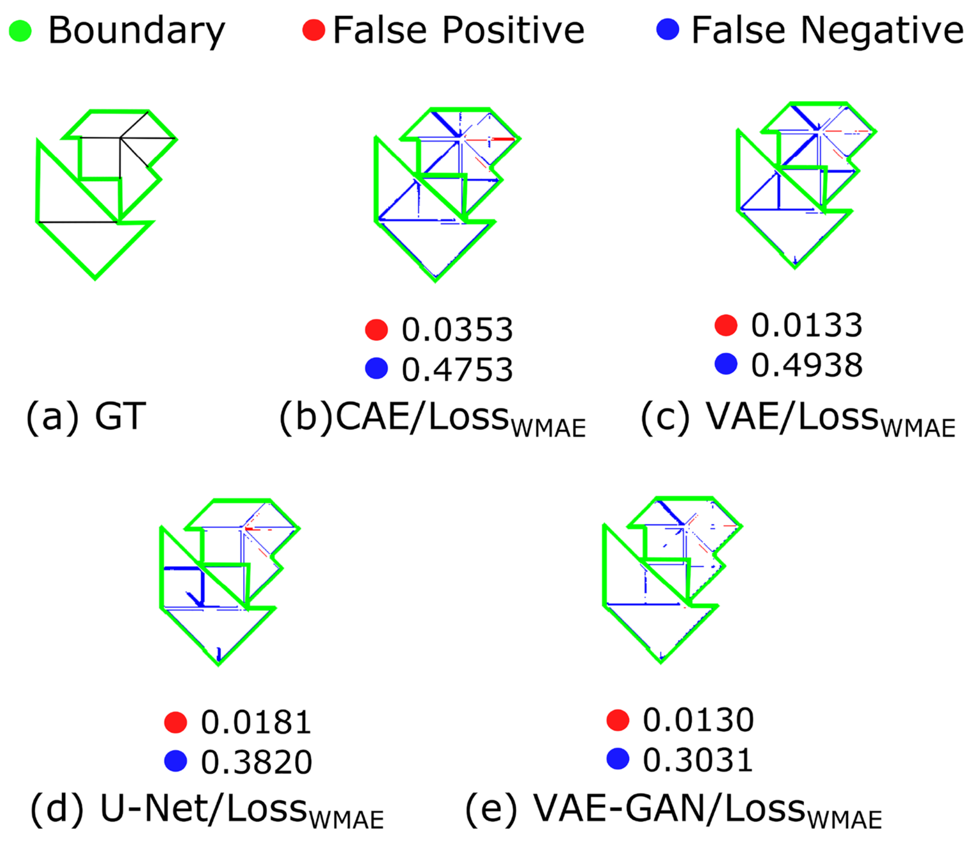

● Boundary
● False Positive
● False Negative
● 0.0353
● 0.4753
● 0.0133
● 0.4938
(a) GT
(b)CAE/Loss$_{\mathrm{WMAE}}$
(c) VAE/Loss$_{\mathrm{WMAE}}$
● 0.0181
● 0.3820
● 0.0130
● 0.3031
(d) U-Net/Loss$_{\mathrm{WMAE}}$
(e) VAE-GAN/Loss$_{\mathrm{WMAE}}$

**Fig. 13** Visualization of false positives and false negatives in solution images. Solution images generated using CAE, VAE, U-Net, and VAE-GAN with $Loss_{\mathrm{WMAE}}$. Values are normalized by the number of pixels in pieces sections presented in black in the ground truth

**Table 2** Experimental results according to WMAE evaluation metric

| Achitecture | WMAE Metric | Inference Time |
|---|---|---|
| CAE | 0.0309 | 0.01377s |
| VAE | 0.0304 | 0.01507s |
| U-Net | 0.0333 | **0.01070s** |
| VAE-GAN | **0.0270** | 0.04193s |

The best architecture according to the WMAE metric and shortest inference time are highlighted

Table 2 shows the results for our experiments over the proposed WMAE evaluation metric. We use $Loss_{\mathrm{WMAE}}$ for all performed experiments. The first column defines the architecture used in each experiment. The second column presents average values for the WMAE evaluation metric. Finally, the last column shows the average inference time each model took to generate the solutions. The results suggest that the WMAE metric is better aligned with the nuances present in Tangram solutions than traditional metrics, such as MSE and SSIM.

As we observe in Fig. 12, solutions generated by VAE-GAN are clear and insightful for determining the correct position and configuration of the Tangram pieces, while U-Net presents incomplete sections, resulting in poor visual performance. These qualitative results closely align with the values shown in the table, since the WMAE metric considers VAE-GAN as the best architecture, and U-Net as the worst one in the visual quality of generated solutions. Moreover, the WMAE tells that the visual quality of the solutions generated by both CAE and VAE is similar, which can also be perceived by their generated solutions. Although the average inference time for VAE-GAN is considerably higher than the other standalone architectures, we consider that this is not a prohibitive limitation, given the notable improvements in solution quality.

It is evident that $Loss_{\mathrm{WMAE}}$ consistently delivers superior qualitative results when compared to the alternative loss functions $Loss_{MSE}$ and $Loss_{SSIM}$. Its preference for generating clear additional sections, rather than omissions, not only enhances the visual quality of the solutions but also significantly facilitates the inference of piece positions. Regarding the employed evaluation metrics, while MSE and SSIM serve as common benchmarks, they often fall short in capturing the nuanced characteristics of a Tangram solution. WMAE evaluation metric demonstrates a superior capacity in assessing the visual quality of Tangram solutions. By assigning priority to the pixel values representing the sections between Tangram pieces, our metric surpasses the limitations associated with conventional methods, thereby providing a more precise evaluation of Tangram solutions.

Finally, we conduct an experiment to address the influence of the coefficient c on the $Loss_{\mathrm{WMAE}}$. We vary the value of this coefficient in a large range of values, doing so we can assess its impact on the VAE-GAN performance. Table 3 shows the results of this experiment over the proposed WMAE metric. The first column presents the value used for c in $Loss_{\mathrm{WMAE}}$. The second column presents the average values for WMAE evaluation metric. We do not consider the average inference time in this experiment because there are no expressive variations regarding this aspect.

The experiment shows that, although $c = 50$ presents a slightly better performance compared to the other considered values, the performance does not change expressively. The same behavior is expected for the WMAE evaluation metric eliminating the necessity of additional tests. Figure 14 presents examples of solutions generated by VAE-GAN for $c = 50$ and $c = 5$ in $Loss_{\mathrm{WMAE}}$ aiming at visual comparison.

It is possible to notice that, as the values in Table 3 suggest there is not considerable visual difference considering this range for $c$ in $Loss_{\mathrm{WMAE}}$.

**Table 3** Experimental results varying coefficient c in $Loss_{\mathrm{WMAE}}$ according to WMAE evaluation metric

| Coefficient c | WMAE Metric |
|---|---|
| 1 | 0.0273 |
| 2 | 0.0278 |
| 5 | 0.0270 |
| 10 | 0.0277 |
| 50 | **0.0268** |
| 100 | 0.0273 |
| 500 | 0.0273 |
| 1000 | 0.0273 |
| 5000 | 0.0276 |
| 10000 | 0.0272 |
| 50000 | 0.0271 |
| 100000 | 0.0275 |

All experiments use VAE-GAN

The best value for c according to the WMAE metric is highlighted

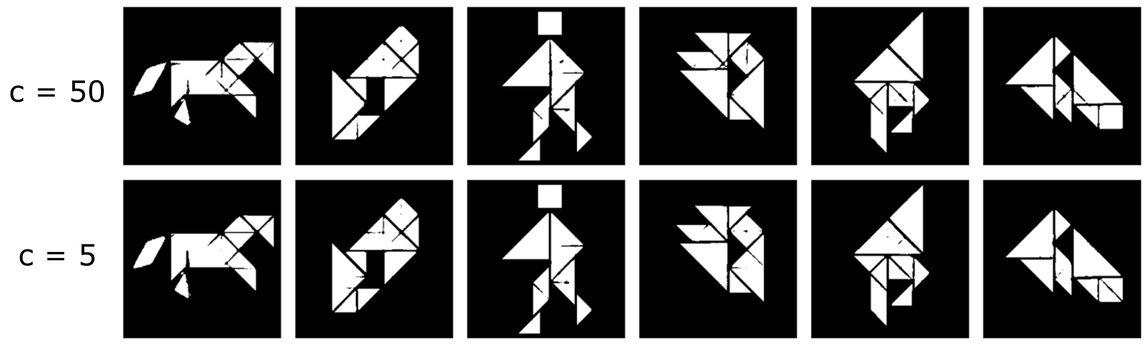


**Fig. 14** Solution images generated by VAE-GAN paired with $Loss_{\mathrm{WMAE}}$ considering coefficient c equal to 50 and 5

## 6 Conclusions

In this paper, we approach the use of deep-learning models for solving Tangram puzzles. At a first moment, we implemented three deep-learning architectures for this task: CAE, VAE, and U-Net. In our experiments, we varied the loss function among $Loss_{MSE}$ and $Loss_{SSIM}$ to determine which architecture performs best in solving Tangram puzzles. To evaluate the generated solution images, we used MSE and SSIM, which are traditional metrics that are based on pixel accuracy. We showed that these metrics fail in properly evaluating the generated Tangram solutions. We observed that VAE outperforms CAE and U-Net regarding the visual quality of the generated Tangram solutions. After this first round of tests, we implemented a VAE-GAN architecture by combining the VAE architecture with a discriminator. Integrated with the VAE-GAN, we implemented a new loss function that was designed for our task. We further propose a novel evaluation metric that has proven more efficient in assessing the visual quality of generated Tangram solutions. We analyzed the results and presented a detailed comparative analysis of the capabilities of deep-learning architectures in automatizing the solution of the Tangram.

Experimental results indicate the models employed a similar approach to segment the Tangram pattern into pieces. This approach consists of segmenting the puzzle area following a triangular grid, and then eliminating segments that do not represent the contact between pieces. This strategy aligns well with a particularity of the Tangram pieces, that tells that they can all be decomposed into small triangles [57]. This shows that the models were able to learn the geometrical properties of Tangram pieces even when these properties were not directly informed to them. Although conventional evaluation metrics were proven not adequate for determining the visual quality of Tangram solutions, a visual examination revealed that VAE exhibited promising results as it produced well-defined sections between pieces, and favored generating additional sections rather than omissions. Our preference for solutions with extra sections aligns with the principle that they are more insightful in the context of interpretation of Tangram puzzles. In line with this perspective, we proposed $Loss_{\mathrm{WMAE}}$, which is a loss function that gives higher importance to pixels representing sections between pieces rather than pixels covered by pieces. We expanded the usage of this loss function to propose the WMAE evaluation metric, aiming at presenting a metric that is better aligned with the task of evaluating Tangram solutions than the aforementioned conventional evaluation metrics. Solutions generated by VAE-GAN combined with $Loss_{\mathrm{WMAE}}$ were clear and insightful for determining the correct position and configuration of the Tangram pieces. The discriminator supports regularization in the training process and leads to smoother loss curves for the generator. Our experiments indicate that VAE-GAN architecture is the most robust choice among the considered architectures, showing stable convergence and strong generalization skills. At the conclusion of our experiments, we applied the WMAE metric to evaluate the performance of our architectures, demonstrating its ability to overcome limitations associated with conventional metrics and, consequently, deliver a more accurate assessment of Tangram solutions. Therefore, we conclude that deep-learning techniques are applicable to the task of solving Tangram puzzles.

This study serves as a basis for future research on application of artificial intelligence in complex geometrical problem domains. As future works, our priority is to implement a more robust model capable of not only accurately segmenting Tangram pieces but also focusing on further extracting their geometric features. This is extremely important for the model to be able to comprehend the different forms the Tangram pieces can interact. We also aim at constructing a larger data-driven dataset. The literature shows that different authors often consider only a limited set of patterns in their experiments, which varies considerably from work to work. We believe that deep-learning models could benefit from having access to a larger variety of Tangram puzzles during the training stage. We also consider that a data-driven dataset would serve as a valuable resource for future research concerning the solution of dissection puzzles, as well as related optimization problems.

**Acknowledgements** This study was financed in part by a scholarship from the Ministry of Education, Culture, Sports, Science, and Technology (MEXT), Government of Japan.

**Author contributions** FMY: Proposed the architectures, carried out the experiments, and wrote the majority of the text. HCB: Assisted with writing and reviewing the manuscript, providing critical feedback, and contributing to the interpretation of the research. JPG: Provided significant input and support in the writing and editing process, and helped review and revise the manuscript. HT: Participated in reviewing and revising the manuscript, ensuring its accuracy and providing valuable insights, and supervised the project. All authors have read and approved the final manuscript for submission.

**Funding** This study was financed in part by a scholarship from the Ministry of Education, Culture, Sports, Science, and Technology (MEXT), Government of Japan.

**Data availability** The data that support the findings of this study are openly available in our Github repository [49].

**Code availability** The computer code associated with the research presented in this paper is available for review and use by interested parties. The code can be accessed on our GitHub repository [49].

## Declarations

**Competing interests** Authors declare no conflict of interest and no competing interests.

## References

1. Liu Z, Liu W. Research on the design of combination furniture based on toy brick style concept. In: Liu Z, editor. 5th international conference on civil engineering and transportation. Amsterdam: Atlantis Press; 2015. p. 1712–7. https://doi.org/10.2991/iccet-15.2015.319.
2. Tchoshanov M. Building students' mathematical proficiency: connecting mathematical ideas using the tangram. Learn Teach Math. 2011;2011(10):16–23.
3. Gao W, Ramani K. Kaleidogami$^{TM}$ : Multi-primitive reconfigurable artistic structures. School of Mechanical Engineering School, Electrical and Computer Engineering, Purdue University: by Courtesy. 2012.
4. Pohl SS, Richter C. The complete characterization of tangram pentagons. Contrib Algebra Geom. 2021;62(1):121–35. https://doi.org/10.48550/arXiv.2006.09698.
5. Kmetová M, Nagyová Lehocká Z. Using tangram as a manipulative tool for transition between 2d and 3d perception in geometry. Mathematics. 2021;9(18):2185. https://doi.org/10.3390/math9182185.
6. Renavitasari IRD, Supianto AA. Educational game for training spatial ability using tangram puzzle. In: Renavitasari IRD, editor. 2018 International conference on sustainable information engineering and technology (SIET). Malang: IEEE; 2018. p. 174–9. https://doi.org/10.1109/SIET.2018.8693164.
7. Zhao Y, Qiu L, Lu P, Shi F, Han T, Zhu S.-C. Learning from the tangram to solve mini visual tasks. Proc AAAI Conf Artif Intell. 2022;36:3490–8. https://doi.org/10.1609/aaai.v36i3.20260.
8. Li J, Yang J, Hertzmann A, Zhang J, Xu T. Layoutgan: generating graphic layouts with wireframe discriminators. arXiv Preprint. 2019. https://doi.org/10.48550/arXiv.1901.06767.
9. Martins T, Tsuzuki MSG. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. Exp Syst Appl. 2010;37(3):1955–72. https://doi.org/10.1016/j.eswa.2009.06.081.
10. Yamada FM, Gois JP, Batagelo HC. Solving tangram puzzles using raster-based mathematical morphology. In: Yamada FM, editor. 2019 32nd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI). Rio de Janeiro: IEEE; 2019. p. 116–23. https://doi.org/10.1109/SIBGRAPI.2019.00024.
11. Bofferding L, Aqazade M. where does the square go?: reinterpreting shapes when solving a tangram puzzle. Educ Stud Math. 2023;112(1):25–47. https://doi.org/10.1007/s10649-022-10166-0.
12. Karras T, Aila T, Laine S, Lehtinen J. Progressive growing of gans for improved quality, stability, and variation. Neural Evol Comp. 2018. https://doi.org/10.48550/arXiv.1710.10196.

13. Lee J, Kim J, Chung H, Park J, Cho M. Learning to assemble geometric shapes. Int Joint Conf Artif Intell. 2022. https://doi.org/10.48550/arXiv.2205.11809.
14. Masci J, Meier U, Cireşan D, Schmidhuber J. Stacked convolutional auto-encoders for hierarchical feature extraction. In: Artificial neural networks and machine learning–ICANN 2011: 21st International conference on artificial neural networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21, 2011; Espoo: Springer. p. 52– 59 . https://doi.org/10.1007/978-3-642-21735-7_7
15. Kingma DP, Welling M. Auto-encoding variational bayes. arXiv Preprint. 2013. https://doi.org/10.48550/arXiv.1312.6114.
16. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF, editors. Medical image computing and computer-assisted intervention-MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18. Munich: Springer; 2015.
17. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. Adv Neural Inform Process Syst. 2014. https://doi.org/10.48550/arXiv.1406.2661.
18. Minhas MS, Zelek J. Semi-supervised anomaly detection using autoencoders. arXiv Preprint. 2020. https://doi.org/10.48550/arXiv.2001.03674.
19. Zhou Y, Huang W, Dong P, Xia Y, Wang S. D-unet: a dimension-fusion u shape network for chronic stroke lesion segmentation. IEEE/ACM Transact Comput Biol Bioinform. 2019;18(3):940–50. https://doi.org/10.1109/TCBB.2019.2939522.
20. Bartoněk D. A genetic algorithm how to solve a puzzle and its using in cartography. Acta Sci Pol Geod Descr Terrarum. 2005;4(2):5–23.
21. Kovalsky SZ, Glasner D, Basri R. A global approach for solving edge-matching puzzles. SIAM J Imaging Sci. 2015;8(2):916–38. https://doi.org/10.1137/140987869.
22. Deutsch ES, Hayes KC Jr. A heuristic solution to the tangram puzzle. Mach Intell. 1972;7:205–40.
23. Oflazer K. Solving tangram puzzles: a connectionist approach. Int J Intell Syst. 1993;8(5):603–16. https://doi.org/10.1002/int.4550080502.
24. Yamada FM, Takahashi H, Batagelo HC, Gois JP. An extended approach for the automatic solution of tangram puzzles using permutation heuristics. In: Yamada FM, editor. 2020 Nicograph International (NicoInt). Tokyo: IEEE; 2020. p. 47–50. https://doi.org/10.1109/NicoInt50878.2020.00016.
25. Rafique A, Iftikhar T, Khan N. Adversarial placement vector learning. In: Rafique A, editor. 2019 2nd International conference on advancements in computational sciences (ICACS). Lahore: IEEE; 2019. p. 1–7. https://doi.org/10.23919/ICACS.2019.8689004.
26. Doersch C, Gupta A, Efros AA. Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE International conference on computer vision. 2015. p. 1422– 1430. https://doi.org/10.48550/arXiv.1505.05192.
27. Noroozi M, Favaro P. Unsupervised learning of visual representations by solving jigsaw puzzles. In: Leibe B, Matas J, Sebe N, Welling M, editors. Computer vision-ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VI. Cham: Springer; 2016. p. 69–84. https://doi.org/10.48550/arXiv.1603.09246.
28. Taleb A, Lippert C, Klein T, Nabi M. Self-supervised learning for medical images by solving multimodal jigsaw puzzles. IEEE Transact Med Imaging. 2017;12729:661–73.
29. Kim D, Cho D, Yoo D, Kweon IS. Learning image representations by completing damaged jigsaw puzzles. In: Kim D, editor. 2018 IEEE winter conference on applications of computer vision (WACV). Lake Tahoe: IEEE; 2018. p. 793–802. https://doi.org/10.1109/WACV.2018.00092.
30. Paumard MM, Picard D, Tabia H. Jigsaw puzzle solving using local feature co-occurrences in deep neural networks. In: Paumard MM, editor. 2018 25th IEEE international conference on image processing (ICIP). Athens: IEEE; 2018. p. 1018–22. https://doi.org/10.1109/ICIP.2018.8451094.
31. Paumard M-M, Picard D, Tabia H. Image reassembly combining deep learning and shortest path problem. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018. pp. 153– 167 . https://doi.org/10.48550/arXiv.1809.00898.
32. Zhang Y, Hare J, Prügel-Bennett A. Learning representations of sets through optimized permutations. Int Conf Learn Represent. 2018. https://doi.org/10.48550/arXiv.1812.03928.
33. Pang K, Yang Y, Hospedales TM, Xiang T, Song YZ. Solving mixed-modal jigsaw puzzle for fine-grained sketch-based image retrieval. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. pp. 10347– 10355.
34. Paumard M-M, Picard D, Tabia H. Deepzzle: solving visual jigsaw puzzles with deep learning and shortest path optimization. IEEE Transact Image Process. 2020;29:3569–81. https://doi.org/10.1109/TIP.2019.2963378.
35. Le C, Li X. Jigsawnet: shredded image reassembly using convolutional neural network and loop-based composition. IEEE Transact Image Process. 2019;28(8):4000–15. https://doi.org/10.1109/TIP.2019.2903298.
36. Li R, Liu S, Wang G, Liu G, Zeng B. Jigsawgan: auxiliary learning for solving jigsaw puzzles with generative adversarial networks. IEEE Transact Image Process. 2021;31:513–24. https://doi.org/10.1109/TIP.2021.3120052.
37. Talon D, Del Bue A, James S. Ganzzle: reframing jigsaw puzzle solving as a retrieval task using a generative mental image. In: Talon D, editor. 2022 IEEE international conference on image processing (ICIP). Bordeaux: IEEE; 2022. p. 4083–7. https://doi.org/10.1109/ICIP46576.2022.9897553.
38. Khoroshiltseva M, Traviglia A, Pelillo M, Vascon S. Relaxation labeling meets gans: solving jigsaw puzzles with missing borders. In: Sclaroff S, Distante C, Leo M, Farinella GM, Tombari F, editors. Image analysis and processing-ICIAP 2022: 21st international conference, Lecce, Italy, May 23–27, 2022, proceedings, part III. Cham: Springer; 2022. p. 27–38. https://doi.org/10.48550/arXiv.2203.14428.
39. Santa Cruz R, Fernando B, Cherian A, Gould S. Visual permutation learning. IEEE Transact Pattern Anal Mach Intell. 2018;41(12):3100–14. https://doi.org/10.1109/TPAMI.2018.2873701.
40. Salehi M, Eftekhar A, Sadjadi N, Rohban MH, Rabiee HR. Puzzle-ae: novelty detection in images through solving puzzles. arXiv preprint. 2020. https://doi.org/10.48550/arXiv.2008.12959.
41. Baykal G, Ozcelik F, Unal G. Exploring deshufflegans in self-supervised generative adversarial networks. Pattern Recogn. 2022;122: 108244. https://doi.org/10.1016/j.patcog.2021.108244.
42. Noroozi M, Vinjimoor A, Favaro P, Pirsiavash H. Boosting self-supervised learning via knowledge transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. p. 9359– 9367. https://doi.org/10.48550/arXiv.1805.00385.
43. Bridger D, Danon D, Tal A. Solving jigsaw puzzles with eroded boundaries. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. pp. 3526– 3535 . https://doi.org/10.48550/arXiv.1912.00755.

44. Hosseini S, Shabani MA, Irandoust S, Furukawa Y. Jigsawplan: room layout jigsaw puzzle extreme structure from motion using diffusion models. arXiv preprint. 2022. https://doi.org/10.48550/arXiv.2211.13785.
45. Markaki S, Panagiotakis C. Jigsaw puzzle solving techniques and applications: a survey. Vis Comput. 2022. https://doi.org/10.1007/s00371-022-02598-9.
46. Wang W, Zhang M, Chen G, Jagadish H, Ooi BC, Tan K-L. Database meets deep learning: challenges and opportunities. ACM Sigmod Rec. 2016;45(2):17–22. https://doi.org/10.1145/3003665.3003669.
47. Bansal MA, Sharma DR, Kathuria DM. A systematic review on data scarcity problem in deep learning: solution and applications. ACM Comput Surv. 2022;54(10s):1–29. https://doi.org/10.1145/3502287.
48. Köpp W. Random generation of tangrams. Interdisciplinary project in mathematics, Technische Universitat München. 2013.
49. Yamada FM. TangramDeep. GitHub. 2023. https://github.com/fernandamiyukiyamada/TangramDeep.
50. Tian C, Fei L, Zheng W, Xu Y, Zuo W, Lin C-W. Deep learning on image denoising: an overview. Neural Netw. 2020;131:251–75. https://doi.org/10.1016/j.neunet.2020.07.025.
51. Fawzi A, Samulowitz H, Turaga D, Frossard P. Adaptive data augmentation for image classification. In: Fawzi A, editor. 2016 IEEE international conference on image processing. Phoenix: IEEE; 2016. p. 3688–92. https://doi.org/10.1109/ICIP.2016.7533048.
52. Zhai J, Zhang S, Chen J, He Q. Autoencoder and its various variants. In: Zhai J, editor. 2018 IEEE international conference on systems, man, and cybernetics (SMC). Miyazaki: IEEE; 2018. p. 415–9. https://doi.org/10.1109/SMC.2018.00080.
53. Vahdat A, Kautz J. Nvae: a deep hierarchical variational autoencoder. Adv Neural Inform Process Syst. 2020;33:19667–79.
54. Yin X-X, Sun L, Fu Y, Lu R, Zhang Y, et al. U-net-based medical image segmentation. J Healthc Eng. 2022. https://doi.org/10.1155/2022/4189781.
55. Dong C, Loy CC, He K, Tang X. Image super-resolution using deep convolutional networks. IEEE transact Pattern Anal Mach Intell. 2015;38(2):295–307. https://doi.org/10.1109/TPAMI.2015.2439281.
56. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE transact Image Process. 2004;13(4):600–12. https://doi.org/10.1109/TIP.2003.819861.
57. Wang H. Using DFS search and enumerate method to find all solutions in 13 convex figures in tangram game. In: Wang H, editor. 2021 International conference on computer information science and artificial intelligence (CISAI). Kunming: IEEE; 2021. p. 505–9. https://doi.org/10.1109/CISAI54367.2021.00103.
58. Aggarwal A, Mittal M, Battineni G. Generative adversarial network: an overview of theory and applications. Int J Inform Manag Data Insights. 2021;1(1): 100004. https://doi.org/10.1016/j.jjimei.2020.100004.
59. Trevisan de Souza VL, Marques BAD, Batagelo HC, Paulo Gois J. A review on generative adversarial networks for image generation. Comput Graph. 2023;114:13–25. https://doi.org/10.1016/j.cag.2023.05.010.
60. Gui J, Sun Z, Wen Y, Tao D, Ye J. A review on generative adversarial networks: algorithms, theory, and applications. IEEE transact Knowl Data Eng. 2021;35(4):3313–32. https://doi.org/10.1109/TKDE.2021.3130191.
61. Ameer S, Basir O. Objective image quality measure based on weber-weighted mean absolute error. In: Ameer S, editor. 2008 9th International conference on signal processing. Beijing: IEEE; 2008. p. 728–32. https://doi.org/10.1109/ICOSP.2008.4697233.
62. Hao S, Li S. A weighted mean absolute error metric for image quality assessment. In: Hao S, editor. 2020 IEEE international conference on visual communications and image processing (VCIP). Macau: IEEE; 2020. p. 330–3. https://doi.org/10.1109/VCIP49819.2020.9301889.
63. Hu S, Jin L, Wang H, Zhang Y, Kwong S, Kuo C-CJ. Objective video quality assessment based on perceptually weighted mean squared error. IEEE Transact Circuits Syst Video Technol. 2016;27(9):1844–55. https://doi.org/10.1109/TCSVT.2016.2556499.