Solving the Tangram Puzzle: Mathematical Morphology and Deep Learning Approaches

by

Fernanda Miyuki Yamada

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Engineering – Dr.Eng.

Department of Informatics
Graduate School of Informatics & Engineering
The University of Electro-Communications, Tokyo, Japan

JUNE 2024

Advisor: Hiroki Takahashi, Dr.Eng.

Solving the Tangram Puzzle: Mathematical Morphology and Deep Learning Approaches

Approved by:

Chair of Committee	Professor Hiroki Takahashi (The University of Electro-Communications)
Member of Committee	Professor Hayaru Shouno (The University of Electro-Communications)
Member of Committee	Professor Hiroyuki Sato (The University of Electro-Communications)
Member of Committee	Professor Keiji Yanai (The University of Electro-Communications)
Member of Committee	Professor Tomonori Hashiyama (The University of Electro-Communications)

Acknowledgements

My sincere gratitude to Professor Hiroki Takahashi, for his support and encouragement throughout my Ph.D. studies. I am grateful for the opportunity to be part of his laboratory and for his guidance on the topics related to my research. Special thanks to my collaborators, Professor Harlen Costa Batagelo and Professor João Paulo Gois, for their generosity in sharing their knowledge and providing insightful feedback.

I would also like to acknowledge the committee members of this dissertation, whose useful hints, and contributions have played a significant role in refining and improving the content of this dissertation. The Graduate School of Informatics & Engineering at The University of Electro-Communications deserves recognition for providing a conducive academic environment and essential resources that contributed to the completion of my Ph.D. studies.

My heartfelt appreciation goes out to my family and friends for their constant encouragement. Without their support, this dissertation would not have been possible. I would like to give a special mention to my boyfriend, Gabriel Peixoto de Carvalho, for the pieces of advice and constant support. His proficiency in deep learning and valuable insights have not only enhanced the quality of my work but have also been a constant source of inspiration. My warmest thanks to him for making each step forward more meaningful.

Lastly, I would like to express my sincere gratitude to the Ministry of Education, Culture, Sports, Science, and Technology, Government of Japan, for awarding me a scholarship. This financial support has been instrumental in facilitating my research and academic pursuits. I am deeply thankful for the opportunity to further my studies and contribute to the academic community with the assistance of this prestigious scholarship.

論文概要

タングラムは，七つの多角形のピースからなる幾何学的なパズルである．これらのピースを剛体変換を行って配置して特定のパターンに一致させることでパズルを解く必要がある．有効な解はピースの重なりがなく，全てのピースが含まれる場合のみである．タングラムパズルの解法は，Nesting問題やBin Packing問題などの一般的な組み合わせ最適化問題に関連しており，これらはNP完全な問題として知られている．2次元の最適化問題では，通常，平行移動と有限な回転に変換を制限し，一つの長方形のコンテナを使用する．一方，タングラムでは多角形状のコンテナに対し，ピースの任意回転を必要とする複雑なピースの組み合わせ処理が求められることがよくある．本論文では，任意のタングラムパターンを対象とし，(1)パターンの構造とピースの無制限な幾何学変換に基づき，タングラムパズルを単純，複雑なパターンに分類したベースラインデータセット，および(2)複雑なパターンに対しても適切な解を得るヒューリスティック，ディープラーニングアプローチを提案した．最後に(3)両アプローチの得失を解析した．

第1章では，本研究の位置付けを述べ，計算機でタングラムパズル解を得るために考慮すべき幾何学的な特性と一般的なシナリオについて述べる．本章では，タングラムパズルで構成される形状を解析し，その形状を単純な形状と複雑な形状に分類，複雑な形状に適した表現手段と変換の必要性について述べている．また，本論文の構成について概観する．

本論文では，従来研究では扱われていなかった複雑なタングラムパズル解を得るためにヒューリスティックアプローチとディープラーニングアプローチを提案している．第2章と第3章では，提案アプローチの理解に必要な理論的概念を紹介する．第2章では，提案手法の着想を得た，無駄な領域を最小限にして複数の形状をコンテナに配置するCutting & Packing問題を紹介し，ヒューリスティックアプローチで用いるNo-fit PolygonやCollision-free Areaなどの概念について述べる．第3章では，提案ディープラーニングアプローチの理解に不可欠ないくつかの概念とアーキテクチャを定義とともにData Augmentationの概念についても述べる．

ジグソーパズルの解法，特に生成モデルを用いた手法にはめざましい進歩がみられるが，タングラムに特化した手法はいまだに未熟なままである．ジグソーパズルに対するディープニューラルネットワークアプローチは，ピースの意味情報に大きく依存し，等サイズの正方形ピースに基づく知見が利用される場合もある．これらは，ジグソーパズルに内在する組み合わせ的な課題の軽減に大きく寄与する．第4章では，従来のタングラムソルバーを分析し，解析的手法とディープラーニングに基づく手法に大別し，それぞれの得失について述べる．

これまで，各著者が独自に選定した一部の限定されたパターンに対する実験を行っていることが多いため，比較可能なベースラインとしてのタングラムパズルデータセットが強く望まれ

る．このデータセットには，単純なパターンと複雑なパターンの両方のタングラムパズルが含まれる必要がある．第5章では，従来文献で用いられたデータを含むベースラインデータセットを提案し，その統計的な分析も行った．

第6章では数学的モルフォロジーに基づくヒューリスティックアプローチについて述べるとともに，タングラムパズルの複雑な形状を表現するためのラスター表現についても詳しく説明している．提案手法は，タングラムパターンをラスター表現し，面積の大きいタングラムピースから順に，そのピースの配置可能領域から最適な配置位置を探索する．また，提案ヒューリスティックアプローチが実際に複雑なパズルを解決できることを示すために，従来手法から選んだ全ての複雑なパターンを含む30種類のタングラムパターンを対象とした予備実験結果を示した．予備実験の結果，46.67%のパターン解法を得ることができ，平均51.042秒の実行時間であることを示した．

第7章では，Generative Adversarial Network (GAN)を用いたディープラーニングアプローチの予備調査を行った．まず，Convolutional AutoEncoder (CAE), Variational AutoEncoder (VAE)とU-netを用いたアーキテクチャ，また，Mean Square Error (MSE), Structural SIMilarity (SSIM), Weighted Mean Absolute Error (WMAE) Lossの組み合わせで，タングラムパズル解法に対する得失を検討した．888種の学習パターンと46種のテストパターンに対する予備実験を行い，VAEとWMAE Lossの組み合わせが視覚的判断に基づく定性的観点から最も適切な解を得た．その後，VAE を用いたGANを構成し，VAEのみを用いた場合よりも適切な解を得ることができることを示した．しかし，タングラムパズル解法の詳細が充分適切に生成できないケースがあること，また，MSEとSSIMを用いて解法の定量的な評価も行ったが，それらの評価値と視覚的な解法の把握に矛盾が生じるケースがあることが明らかにした．

第8章では，第7章で得られた知見をもとに，Huモーメントを用いた幾何学的不変特徴量に基づくLossを利用することで，タングラムパズル解の各ピース配置にも着目にするとともに，あるタングラムパズルに対する複数の解にも対応可能な手法を提案した．また，Huモーメントに基づく損失を拡張し，ピクセル情報そのものに着目する代わりにに生成された解の幾何学的な情報を考慮する評価メトリックを提案した．提案メトリックは一般的な幾何学的最適化の問題に適しているとともに，第7章で明らかにしたMSEとSSIMの問題を克服していることを示した．

第9章では，第6章および第8章で提案したヒューリスティックアプローチとディープラーニングアプローチの両方に対して，第5章で作成したパターンを用いた評価実験を行った．テストデータ100種のパターンに対して，数学的モルフォロジーを用いた手法では6分以内の制限を設定した場合に24.0%が平均184.148秒で解け，ディープラーニングアプローチでは平均0.003秒で70.0-87.0%のパズルが主観的に適切な解法が得られることを示した．また，各手法の詳細な得失の解析を行った．

最後に，第10章では，ヒューリスティックとディープラーニングアプローチの得失について本論文の結論としてまとめるとともに今後の展望について述べる．

Abstract

The Tangram is a dissection puzzle composed of seven polygonal pieces. To solve this puzzle, the pieces must be arranged through rigid transformations to match a specific pattern. A solution is valid only if all pieces are included without any overlap. The Tangram puzzle-solving process relates to combinatorial optimization problems such as Nesting Problem and Bin Packing, known to be NP-complete. In two-dimensional optimization problems, transformations are typically limited to translations and finite rotations, using a single rectangular container. However, Tangram puzzles often require complex piece arrangements involving unconstrained rotations within a polygonal container. This dissertation targets general Tangram puzzles and proposes (1) a baseline dataset classifying Tangram puzzles into simple and complex based on their structure and unrestricted geometric transformations; and (2) heuristic and deep learning approaches that effectively solve even complex patterns. Finally, (3) an analysis of the strengths and weaknesses of both approaches is presented.

Chapter 1 outlines the background of the dissertation, discussing the geometric properties and general scenarios to consider when solving Tangram puzzles computationally. It analyzes the patterns formed in Tangram puzzles, categorizes them into simple and complex puzzles, and discusses the need for appropriate representation methods and transformations for complex puzzles. This chapter also provides an overview of the structure of the dissertation.

This dissertation proposes heuristic and deep learning approaches to solve complex Tangram puzzles, which have not been adequately addressed in previous research. Chapters 2 and 3 introduce theoretical concepts essential for understanding these approaches. Chapter 2 discusses the Cutting & Packing problem, which inspired the heuristic method by addressing the task of minimizing wasted space when arranging multiple shapes in a container. It introduces concepts such as no-fit polygons and collision-free areas used in the heuristic approach. Chapter 3 defines several concepts and architectures crucial for understanding the proposed deep learning approach and discusses data augmentation.

While there has been significant progress in jigsaw puzzle-solving methods, particularly those using generative models, Tangram-specific methods remain underdeveloped. Deep neural network approaches for solving jigsaw puzzles rely heavily on the semantic information of the pieces and often use knowledge based on equal-sized square pieces. These factors significantly mitigate the combinatorial challenges inherent in solving jigsaw puzzles. Chapter 4 analyzes traditional Tangram solvers, categorizing them into analytical and deep learning-based methods, and discusses their respective strengths and weaknesses.

Given that previous experiments often involved a limited selection of Tangram patterns chosen independently by each author, there is a strong need for a Tangram puzzle dataset that serves as a baseline for

comparison. This dataset should include both simple and complex Tangram puzzles. Chapter 5 proposes a baseline dataset, including data used in previous literature, and provides statistical analyses of this dataset.

Chapter 6 discusses the heuristic approach based on mathematical morphology, providing a detailed explanation of the raster representation used for complex Tangram puzzles. The proposed method represents the Tangram pattern as a raster representation, and searches for the optimal placement position of each Tangram piece from its possible placement area, starting with the Tangram piece with the largest area. Preliminary experiments using a toy dataset demonstrated that the heuristic approach could solve complex puzzles, achieving a solution rate of 46.67% with an average execution time of 51.042 seconds.

Chapter 7 investigates a deep learning approach using a Generative Adversarial Network (GAN). Preliminary experiments compared architectures such as Convolutional AutoEncoder (CAE), Variational AutoEncoder (VAE), and U-Net, combined with loss functions like Mean Square Error (MSE), Structural Similarity (SSIM), and Weighted Mean Absolute Error (WMAE). The combination of VAE and WMAE Loss produced the most appropriate solutions based on qualitative visual judgment. Further, using VAE in a GAN setup provided better solutions than using VAE alone. However, some cases failed to generate sufficiently detailed Tangram solutions, and inconsistencies between MSE/SSIM evaluation values and visual judgments were observed.

Chapter 8 builds on the insights from Chapter 7, proposing a loss function based on Hu Moments, focusing on geometric invariant features for Tangram piece placements. This method can handle multiple solutions for a given Tangram puzzle. The proposed metric, which considers geometric information rather than only pixel data, is suitable for general geometric optimization problems and addresses the issues identified with MSE and SSIM.

Chapter 9 evaluates the heuristic and deep learning approaches proposed in Chapters 6 and 8 using the samples presented in Chapter 5. For 100 test patterns, the heuristic approach solved 24.0% within a six-minute limit, with an average time of 184.148 seconds. The deep learning approach achieved subjectively appropriate solutions for 70.0-87.0% of puzzles in an average of 0.003 seconds. Detailed analyses of the strengths and weaknesses of each method are provided.

Finally, Chapter 10 summarizes the conclusions of the dissertation on the strengths and weaknesses of heuristic and deep learning approaches and discusses future directions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Puzzles have fascinated human minds for centuries by conveying a unique blend of challenge, creativity, and intellectual stimulation. Pictorial puzzles are among the most popular forms of puzzles and include jigsaw and edge-matching puzzles [1]. Beyond the recreational nature, the automatic solution of puzzles has been widely applied in several fields, such as the restoration of archeological artifacts [2], reconstruction of fragmented wall painting [3], repair of shredded documents [4], image stitching in computer vision [5], assembly of dissected maps in cartography [6], molecular docking problem [7], and even bone fracture medical treatment [8]. Researchers also suggest that regular puzzle-solving can enhance memory, improve concentration, boost logical reasoning, and promote overall mental agility [9].

In contrast to pictorial puzzles, apictorial puzzles are especially challenging to solve. Since there is no visual information associated with each piece, the puzzle assembly process is based only on the geometry of the pieces [3]. In terms of popularity, dissection puzzles are amongst the most well-known apictorial puzzles [3]. Dissection puzzles require assembling a common set of pieces into multiple distinct forms [10]. The first known dissection puzzle dates to the third century B.C., was of Greek origin and was called the Loculus Archimedes or the Stomachion [11, p. 147]. The Stomachion puzzle had 14 pieces in the form of different polygons. Since then, a variety of dissection puzzles have been invented and used in different countries. In Japan, for example, a dissection puzzle of 7 pieces known as Sei Shonagon's Wisdom Plates was used starting in the 10^{th} century A.D. [12]. Later, in 1796, the 14 Ingenious Pieces puzzle was also invented in Japan, whose main distinction was the inclusion of a circular piece along with other polygon pieces [13].

This dissertation addresses the Tangram, a Chinese dissection puzzle [14]. According to some historians, the Tangram originated from the furniture set during the Song Dynasty and later became a set of wooden blocks for playing [15]. In the year 1802, an American crew gave the Tangram pieces to his son, making the Tangram known to the Western world [16]. Up to this day, even though several dissection puzzles were invented before and after Tangram, it remains the most popular dissection puzzle in the world [12]. Artists and designers have embraced its geometric elements to create artworks, while mathematicians have explored its properties and connections to other mathematical concepts [17, 18]. Tangram puzzles have also

been extensively used to stimulate manipulative learning and teaching aids that help young students acquire geometry thinking and reasoning processes [19].

The Tangram is formed by seven polygonal pieces: two big triangles, one square, one parallelogram, one medium triangle, and two small triangles. The goal is to rearrange the seven pieces using rigid body transformations to fit them into a given pattern. An arrangement of pieces is accepted as a solution only if it contains all the pieces and presents no overlaps between them. To this day there are more than 3000 known Tangram patterns [20]. Different studies are dedicated to cataloging possible patterns that can be formed by Tangram pieces [18, 21]. Figure 1.1 illustrates how Tangram puzzles are presented and interacted with in the real world, in contrast to modern digital representations. Figure 1.1 (a) shows a Tangram puzzle made of wood surrounded by patterns composed by different arrangements of pieces. Figure 1.1 (b) shows an Italian Tangram book containing numerous target patterns and their solutions, entitled "New and Delightful Chinese puzzle" by Lorenzo Bardi in Florence [22].

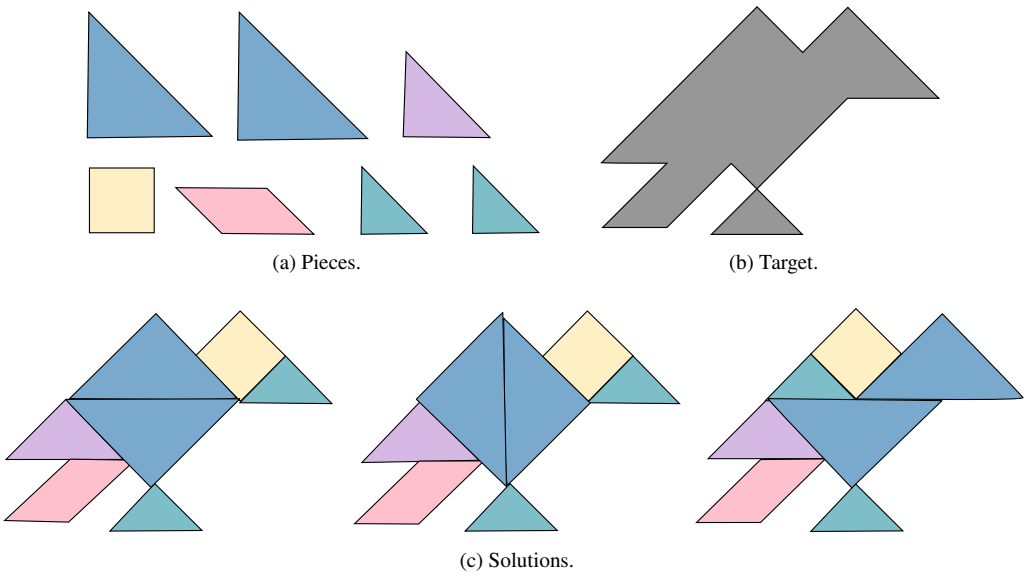
(a) Wooden Tangram puzzle.

(b) Book with Tangram challenges.

Figure 1.1: Physical versions of the Tangram.

In computational theory, 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 NP-complete problems [23]. 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 [24]. Many consider the Tangram as a particular case of the irregular cutting and packing problem in which the number of pieces is fixed and, at the end of the pieces placement process, there is no space left in a container of limited size [25]. The Tangram stands out from other puzzles due to some interesting geometric properties and particularities, which emphasize its uniqueness and challenge the player to use distinctive strategies to assemble the puzzle. An interesting geometric 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 [26]. Therefore, different strategies can lead to distinct arrangements of pieces for the same desired pattern. Figure 1.2 shows the Tangram pieces, a desired pattern, and three feasible solutions for that pattern.



(a) Pieces. (b) Target.

(c) Solutions.

Figure 1.2: Tangram puzzle with different feasible solutions.

Another geometric property found in the Tangram pieces is that they can all be decomposed as a combination of the small triangle piece [27, 19]. Due to this property, Tangram pieces are considered a set of Precious Polygons, in which a set of different polygons can be used to form a similar but larger version of the original set. [28, 29]. This highlights the similarities between the Tangram and the tiling problem, whose goal involves the coverage of a designated space with pre-defined geometric shapes, avoiding overlaps or gaps [30]. Figure 1.3 shows the decomposition of the pieces into small triangles. The pieces are identified with keywords, and dotted black lines indicate where they should be sectioned to form a set of small triangles.

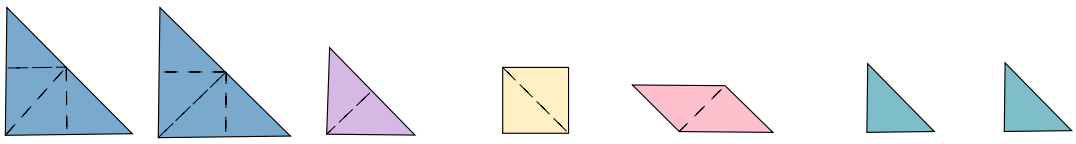


Figure 1.3: Tangram pieces decomposed into combinations of small triangles.

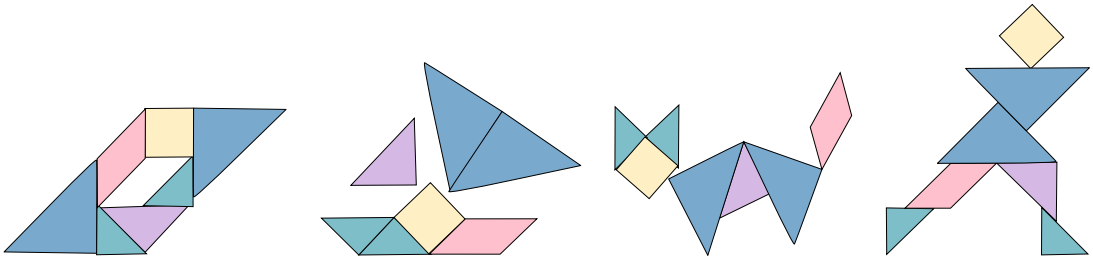
Considering the aforementioned characteristics that distinguish Tangram from other puzzles and general C&P problems, Tangram puzzles are classified into simple Tangram puzzles, and complex Tangram puzzles [31]. The first combines a set of translations and rotations constrained to multiples of 45^{o} to form a pattern composed of a single region, while the second presents at least one of the following characteristics:

1. Contains multiple connected components;
2. Contains holes within the puzzle area;

3. Demand for unconstrained rotations for the piece;

4. Demand for the reflection transformation of the parallelogram.

Complex puzzles are defined as the ones that present at least one of the particular characteristics that are not often observed in other categories of puzzles. While simple puzzles can be described by a simply connected contour, and have a fairly reduced number of configurations that each piece can assume, complex puzzles demand more complex forms of representation and demand more computational effort to be solved. One may wonder why only the parallelogram can be reflected. This is because the parallelogram piece is the only one that does not have any lines of symmetry, which results in it having a pair of enantiomers [32]. It is also possible to notice that reflecting the other pieces corresponds to rotating them to a certain angle. Figure 1.4 exemplifies complex Tangram puzzles with different complex aspects. The arrangement presented in Figure 1.4 (a) represents a polygon that contains a hole. The arrangement presented in Figure 1.4 (b) shows a boat where the sails are formed by multiple regions. The arrangement presented in Figure 1.4 (c) represents a cat with a body and tail formed by unconstrained rotations. Finally, Figure 1.4 (d) represents a person where the parallelogram is reflected when compared to the other patterns.



(a) Holes. (b) Multiple. (c) Unconstrained. (d) Flip.

Figure 1.4: Complex Tangram puzzles and complex aspects

Two of the aspects that characterize complex Tangram puzzles regard the representation of the pattern and pieces. As a consequence, a method that claims to be capable of solving complex puzzles should adequately choose a puzzle and pieces representation that can depict puzzles with holes and multiple regions. The other two aspects concern the transformations that can be applied to the pieces during the assembly process. Therefore, a method that claims to be capable of solving complex puzzles should also be able to recognize pieces in unconstrained angles and differentiate the enantiomers of the parallelogram.

1.2 Problem Statement

Humans show an excellent ability to deal with general assembly problems only by analyzing a pattern and its pieces. On the other hand, machines still fall short of the level of intelligence and often suffer from the combinatorial nature of assembly problems [33]. For this reason, there is a growing interest in developing computational methods for the automatic solution of different puzzles. Whereas methods for generating jigsaw puzzle solutions have made significant recent progress, particularly with deep learning approaches,

methods dedicated to Tangram are far more primitive [34]. Deep neural network approaches that solve the jigsaw puzzle cannot be adapted to solve Tangram puzzles because they rely heavily on semantic information contained in the pieces and often consider square pieces with equal sizes [3]. Both factors greatly mitigate the combinatorial challenges of the problem [33].

Additionally, the number of works that treat complex Tangram puzzles is scarce [24]. Many authors focus on other visual tasks and use the Tangram to assess the versatility of their approach. These works often diminish the particularities that characterize complex Tangram puzzles and limit their tests to only simple Tangram puzzles. Another problem present in the literature on Tangram solvers is the lack of baselines for comparison. Different works often consider only a limited set of patterns in their experiments, which varies considerably from work to work. This highlights the necessity of a dataset for Tangram puzzles to be used as a baseline of comparison. To properly evaluate the ability of each method to solve different types of puzzles, a baseline dataset must include both simple and complex Tangram puzzles. The applications of a baseline dataset for Tangram would not have its application limited to dissection puzzle methods, it would be a valuable resource for works focused on optimization problems in general.

Considering all these factors it is possible to observe that the current literature on computational methods for solving puzzles, particularly Tangram puzzles, is limited and lacks advanced approaches that address the unique combinatorial challenges posed by this puzzle. Existing deep learning methods, predominantly focus on jigsaw puzzles, and rely heavily on semantic information and uniform piece sizes, which makes them unsuitable for Tangram puzzles. Additionally, there is a notable absence of standardized datasets for benchmarking the efficacy of Tangram solvers, with most studies only considering a limited and variable set of patterns. This highlights the need for dedicated computational methods for Tangram puzzles and a comprehensive dataset to facilitate meaningful comparisons and advancements in this field.

1.3 Objectives

In this dissertation, two novel approaches dedicated to the automatic solution of Tangram puzzles are proposed. First, a heuristic method inspired by mathematical morphology techniques that are used in the solution of C&P problems is presented. Then a more modern deep learning approach that aims at extracting geometric information from the pieces and understanding the different forms they can interact throughout the training process is proposed. To the extent of the literature review, this is the very first deep learning approach that is dedicated to solving Tangram puzzles, not treating it as an extension. Both approaches proposed in this dissertation take advantage of a raster representation to allow for fast placements of the pieces and support for complex puzzles. Apart from comparing their clear conceptual difference, the idea behind the proposal of the heuristic method and the deep learning approach is to contrast the way they attack the presented problem. While the heuristic method exhaustively attempts to place the pieces in the correct arrangement one by one, the deep learning approach aims at extracting patterns that inform the geometry of the pieces and the interaction between them by outputting an image that should depict a solution. The following subsections define the main objective of this dissertation and a plan to achieve it.

1.3.1 General Objective

The primary goal of this dissertation is to significantly advance the state-of-the-art in solving the Tangram puzzle by addressing and overcoming the limitations identified in the existing literature, focusing on representation and combinatorial challenges related to the solution of non-contiguous patterns and transformation constraints for the pieces that characterize complex Tangram puzzles.

1.3.2 Specific Objectives

Specific objectives are listed below:

1. Conduct a literature review on existing Tangram solvers, aiming to discern inherent limitations of the area and identify potential research prospects.
2. Collect a data-driven dataset designed for automating Tangram puzzle solutions, ensuring diversity and representativeness.
3. Propose a heuristic method designed for the automatic solution of Tangram puzzles, addressing the limitations identified in the literature and taking inspiration from C&P problem-solving techniques.
4. Explore different deep learning architectures in the task of solving Tangram puzzles, tracing potential methodologies for the final deep learning approach.
5. Establish a set of metrics for comparative analysis, enabling a robust evaluation of the efficacy of deep learning architectures in the assembly of Tangram puzzles.
6. Execute initial experiments on the considered deep learning architectures, evaluating their performance to identify their strengths and weaknesses to be considered in the final deep learning approach.
7. Present a final deep learning architecture that competes with the heuristic method, based on insights from comparative experiments.
8. Design an experimental framework for the comparative analysis between the final deep learning architecture and the heuristic method.
9. Execute the planned experiments and analyze the results to obtain meaningful insights into the performance of the proposed methodologies, highlighting their strengths and weaknesses.
10. Document concluding remarks and define future directions for the research field, synthesizing the main findings of the present dissertation.

1.4 Contributions

The present dissertation is significant as it not only addresses the practical challenges in solving the NP-complete Tangram puzzle but also contributes novel methodologies, metrics, and datasets that have the potential to advance the broader field of artificial intelligence in geometric problem-solving. The outcomes

of this research are expected to apply to a wide range of dissection puzzles and optimization problems. The main contributions of this dissertation encompass the following:

1. **Heuristic Approach:** Introduction of a traditional heuristic approach based on mathematical morphology techniques addresses identified limitations in representing complex Tangram patterns.

2. **Deep Learning Approach:** Proposal of a contemporary deep learning approach based on generative models that showcase competitive accuracy with traditional methods and possibly faster inferences.

3. **Generalizable Loss Function and Novel Evaluation Metric:** Development of a generalizable loss function based on Hu Moments and the introduction of a novel evaluation metric enhance the robustness and effectiveness of proposed methodologies.

4. **Novel Dataset:** Generation of a novel dataset, surpassing any reported in the existing literature in number of samples, which serves as a valuable resource for research and evaluation in dissection puzzles and related optimization problems.

5. **Significance and Advancements in Artificial Intelligence:** Significantly advancements regarding the current understanding of artificial intelligence in geometric problem domains, specifically within the context of dissection puzzles and related real-world applications.

1.5 Dissertation Structure

Chapter 2 introduces the fundamental theoretical foundations and techniques essential to understanding cutting and packing problems, focusing on their application to dissection puzzles like the Tangram. Cutting and packing problems involve optimizing the arrangement of shapes within a given space, addressing challenges ranging from irregular shape packing to efficient material usage in manufacturing processes. Emphasis is placed on the adaptability of some geometric techniques to dissection puzzles, showcasing their relevance in irregular shape packing problems.

Chapter 3 explores the application of deep learning concepts and architectures in the context of solving the Tangram. The chapter explores the underlying mechanisms of different architectures and discusses their potential applications to Tangram. The chapter concludes by examining data augmentation strategies within the Tangram context.

Chapter 4 synthesizes the literature on computational Tangram solvers. This review encompasses works dedicated exclusively to Tangram, as well as those addressing broader tasks that extend their applicability to the Tangram context. The works are separated according to the application or not of deep learning techniques. The primary objective is to concisely outline each approach, evaluating their strengths and limitations in addressing both simple and complex Tangram puzzles.

Chapter 5 outlines the dataset generation process. It aims to underscore the thoughtful considerations behind creating a data-driven dataset that can be used for training and testing in neural networks. The proposed dataset is also useful for approaches that do not use machine learning by using only the testing set to evaluate the performance of these approaches. A study of the statistics of the proposed dataset is also conducted, focusing on statistics regarding simple and complex puzzles included as samples.

Chapter 6 presents the proposed heuristic approach. It covers the proposed raster representations that support the complex representation of the Tangram puzzle. It also details each step of the pieces placement procedure and the validation process. It also presents a proof of concept where a toy dataset is used to prove that the heuristic can solve complex puzzles.

Chapter 7 assesses the application of different deep learning architectures in solving Tangram puzzles. The chapter explores the potential of these architectures in learning the complex spatial relationships inherent in Tangram puzzles. It further assesses traditional evaluation metrics based on pixel accuracy in assessing the visual quality of the generated Tangram solutions. The investigation presented in this chapter serves as a foundation for the development of the final deep learning approach proposed in this dissertation.

Chapter 8 presents the deep learning approach. It details the components of the proposed GAN-based architecture. It also introduces a loss function based on Hu Moments, which is another contribution of the dissertation. A major problem emerges when choosing a loss function that enables the model to properly learn geometric features from the objects depicted in an image due to traditional loss functions being based on pixel accuracy. Instead of using metrics based on pixel accuracy, a better practice would be to use a loss function that can extract geometric features from each piece presented in the ground truth image and attempt to find the same geometric features in the output image from the generator. For this reason, the dissertation proposes a novel loss function that is based on Hu moments and is capable of telling the discrepancy between two images by taking into consideration geometric features.

Chapter 9 presents the final experiments executed using both the proposed heuristic approach and the deep learning approach. The experimental setup is presented with a detailed description of all the set parameters. For comparison, aspects such as the visual quality of generated solutions and the average running time to assemble a single Tangram puzzle are considered. It also presents an analysis of the overall performance of the heuristic approach and the deep learning approach by outlining the advantages and disadvantages of each strategy.

Finally, Chapter 10 summarizes the main outcomes of the dissertation and discusses the future directions. It not only synthesizes the key contributions but also offers reflections on the broader implications of the research presented in this dissertation. Additionally, it outlines potential avenues for future exploration, guiding subsequent researchers in extending the scope of knowledge in the field.

Chapter 2

Cutting and Packing Problem

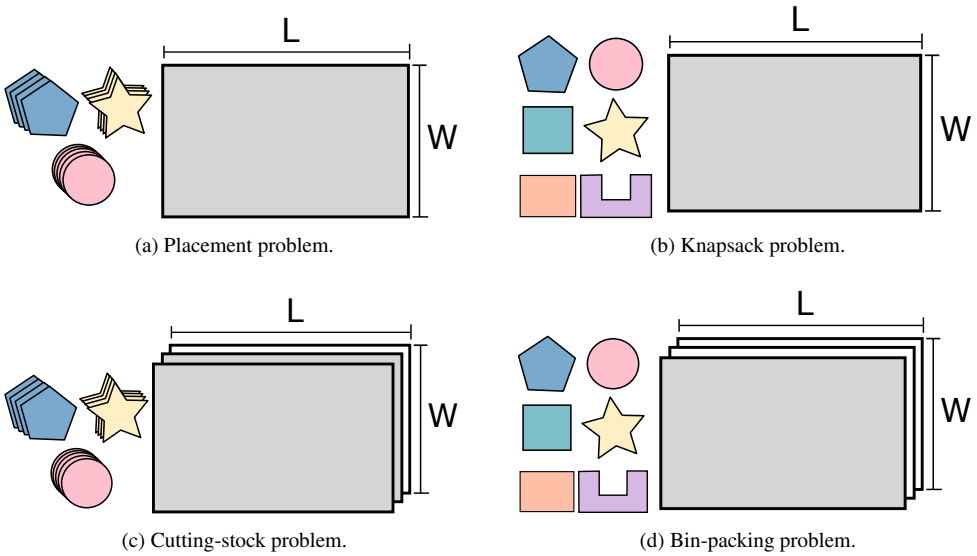
Effective representations and algorithms for solving puzzle games have applications beyond theoretical interest [35]. An industrial application that is frequently mentioned is the C&P problem. The C&P problem involves the placement of a certain number of shapes onto a container to minimize the waste of area, where shapes must not overlap and they must stay within the limits of the container [36]. In the industrial environment, these problems are recurrent and to find a feasible solution experienced workers attempt to build layouts with computer-aided design systems. [37]. Examples include the metal, glass, and wood industries, where a set of items required by customer orders must be cut from larger sheets or boards [38]. According to the typology by Mundim et al. [36], two-dimensional versions of C&P problems, with irregular shapes and limited-size containers, can be classified into maximization problems and minimization problems [36].

Maximization problems aim at maximizing the use of a single container and include the placement problems and the knapsack problems [36]. The difference between the placement problems and the knapsack problems lies in the diversity of the types of shapes. In placement problems, the set of shapes is weakly heterogeneous, whereas in knapsack problems the set of shapes is strongly heterogeneous [39].

Minimization problems consist of assigning a finite set of shapes to the least possible number of containers [36]. The minimization problems include the cutting-stock problems, and the bin packing problems [39]. Similarly to maximization problems, the difference between the cutting-stock problems and the bin-packing problems lies in the diversity of the respective shape sets. In cutting-stock problems, the set of shapes is weakly heterogeneous, whereas in bin-packing problems the set of shapes is strongly heterogeneous [36].

Figure 2.1 presents examples of different categories of C&P problems according to the typology by Mundim et al. [36]. In the presented examples, shapes with the same type are assigned with the same color, and the containers have fixed dimensions $L \times W$. Also, occupied containers are highlighted in gray. Notice that only problems with limited-size containers are addressed, not including open-dimension problems, such as the variant presented by Cherri et al. [40].

To avoid local optima, many methods focus on finding a feasible solution, but not necessarily an optimal one, in reasonable computational times by the application of evolutionary algorithms, hybrid algorithms, and other metaheuristic methods [36]. Over the years, some approaches have been proposed to solve the problem of detecting and preventing overlaps among shapes. The traditional strategies include the front-line method, the scanning-line method, and the no-fit polygon method [41]. Among all these techniques, the most widely used tool for checking whether two polygons overlap is the no-fit polygon [42].



(a) Placement problem.

(b) Knapsack problem.

(c) Cutting-stock problem.

(d) Bin-packing problem.

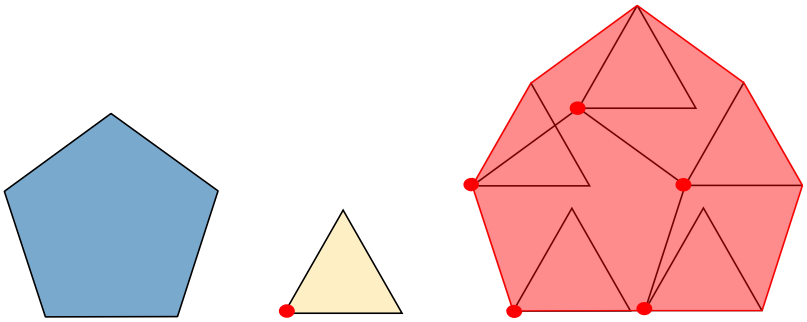
Figure 2.1: Different categories of C&P problems.

Researchers consider dissection puzzles as particular cases of a knapsack maximization C&P problem in which the number of pieces is fixed and, at the end of the pieces placement process, there is no space left in a container of limited size [25]. However, the Tangram presents some particularities that are rarely observed in general C&P. For instance, it is common for C&P problems to restrict transformations to translations and rotations constrained to multiples of 180º, 90º or 45º, and make use of a single rectangular container [24]. In contrast, the Tangram puzzles process usually requires irregularly shaped containers, unconstrained rotations of the pieces, and the reflection transformation for the parallelogram [24]. Despite the differences in combinatorial complexity, the computational Tangram solvers may take advantage of geometric techniques applied in C&P problems to avoid overlaps and reduce the distance between shapes. Therefore, this chapter is dedicated to covering such geometric techniques and concepts that may be used in the automatic solution of Tangram puzzles.

2.1 No-fit Polygon

The no-fit polygon is a basic graphic tool used for calculating the relative positions of the two polygons in which they either touch or overlap [43]. It can be easily obtained by using the Minkowski sum algorithm, which can be calculated very efficiently for convex polygons [44]. Non-convex polygons can be decomposed into convex polygons since the isometric transformations applied do not affect such decomposition [45].

The Minkowski sum is obtained by adding each point in A to each point in B [46]. An equivalent kinematic interpretation describes the Minkowski sum as the result of the translatory motion of B such that its reference point runs on the surface A [47]. Figure 2.2 presents the Minkowski sum of two polygons A and B, taking the highlighted vertex in polygon B as the reference point. The resulting Minkowski sum is colored in red.



(a) Polygon A. (b) Polygon B. (c) Minkowski sum $A \oplus B$.

Figure 2.2: Minkowski sum between polygons A and B.

The Minkowski sum is used in different tasks related to computational geometry, particularly in the context of robotics, collision detection, and motion planning [48]. For instance, in robotics, this operation can be visualized as sweeping a robot around the perimeter of an obstacle, and the Minkowski sum represents the envelope of all possible positions of the combined sets [49]. The Minkowski sum is crucial for designing algorithms that involve the interaction of shapes in various fields, especially in scenarios where it is necessary to analyze or plan movements in the presence of obstacles. The formal definition of the Minkowski sum of two polygons A and B is denoted as $A \oplus B$, and can be calculated as [45]:

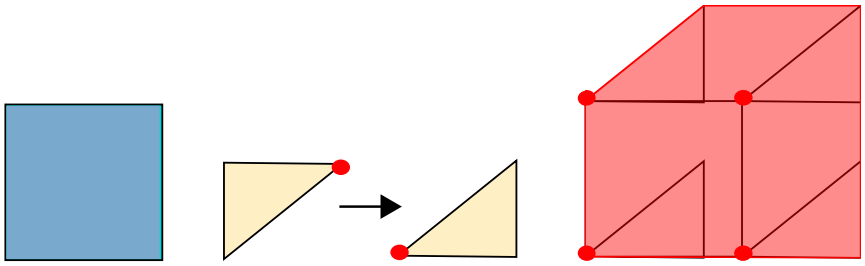
$$A \oplus B = \{a + b : a \in A, b \in B\}. \tag{2.1}$$

The no-fit polygon is calculated by the execution of the Minkowski sum A $\oplus$ (-B) [50], which is the locus of points traced by the reference point associated with -B, when this piece slides along the external contour of A [51]. The opposed polygon is obtained by inverting the signal of all coordinates of the original polygon [25]. Simple vector algebra can be used to show the necessity of negating B for calculating the no-fit polygon [50, 52]. Therefore, the formal definition of the no-fit polygon induced by polygons A and B, noted as $NFP(A, B)$, is defined as [45]:

$$NFP(A, B) = A \oplus -B = \{a + b : a \in A, b \in -B\}, \tag{2.2}$$

where the Minkowski sum operation $A \oplus -B$ is defined as the set of all possible vector sums of elements from A and elements from the negation of B, denoted as $-B$. It can be said that this process is analogous to a dilation operation [53]. Figure 2.3 illustrates the process for obtaining $NFP(A, B)$ for a pair of polygons A and B, where the resulting no-fit polygon is presented in red.

In summary, the concept of no-fit polygons is useful because it enables more efficient and optimized solutions in a wide range of applications, such as packing problems, resource optimization problems, and collision avoidance in robotics [54]. It helps to guide algorithms and decision-making processes by identifying spatial constraints and regions where certain configurations are not feasible, ultimately leading to improved resource utilization and cost savings. The implications of the no-fit polygon, as outlined by Carravilla et al. [55], offer a systematic approach to interpreting the spatial relationship between the two polygons A and B based on the reference point chosen for B and the resulting $NFP(A, B)$:

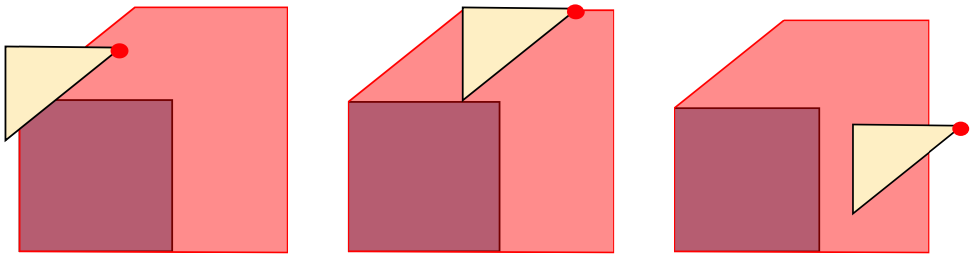


(a) Polygon A. (b) Inversion of polygon B. (c) No-fit polygon $NFP(A, B)$.

Figure 2.3: No-fit polygon.

1. If the reference point of B is placed in the interior of $NFP(A, B)$, then B overlaps A.
2. If the reference point of B is placed on the boundary of $NFP(A, B)$, then B touches A.
3. Otherwise, it implies that B does not overlap or touch A.

The enumerated cases are presented in Figure 2.4, where the depicted polygons are the same as the ones used in Figure 2.3.



(a) Case 1. (b) Case 2. (c) Case 3.

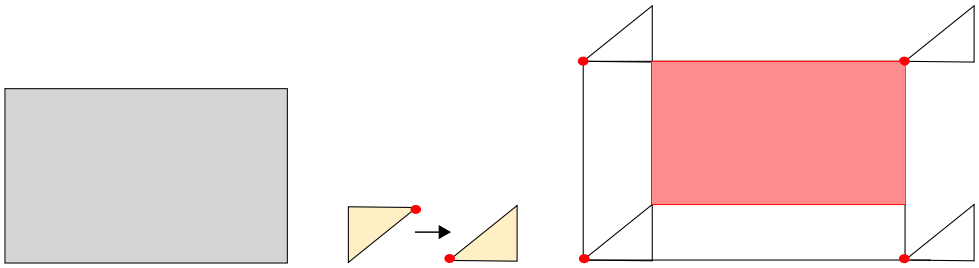
Figure 2.4: Spatial relationship between polygons according to the no-fit polygon.

2.2 Inner-fit Polygon

The concept of the inner-fit polygon is derived from the no-fit polygon and represents the feasible placement positions of polygon B inside a container C [56]. Analogous to the no-fit polygon, the inner-fit polygon can be computed by sliding a polygon along the internal contour of the container [55]. The formal definition of the inner-fit polygon, considering polygon B and container C, is presented as [45]:

$$IFP(B, C) = C \ominus B = \{c - b : c \in C,\ b \in -B\}, \tag{2.3}$$

where $C \ominus B$ provides the collection of vectors obtained by subtracting each point in $-B$ from each point in C, analogous to an erosion operation [45]. Figure 2.5 illustrates the obtaining process of $IFP(B, C)$, considering polygon B and container C, where the resulting inner-fit polygon is presented in red.



(a) Container c. (b) Inversion of polygon B. (c) Inner-fit polygon $IFP(B, C)$.

Figure 2.5: Inner-fit polygon.

Analogously to the no-fit polygon, it is possible to interpret the spatial relationship between the polygon B and container A based on the reference point chosen for B and the resulting $IFP(B, C)$:

1. If the reference point of shape B is placed in the interior of $IFP(B, C)$, then B is within C.
2. If the reference point of shape B is placed on the boundary of $IFP(B, C)$, then B touches C.
3. Otherwise it implies that B leaves the limits of C.

2.3 Collision-free Area

When sequential placement of items is adopted, the placement heuristic must take into account previously placed items, as well as the container to obtain a feasible layout [45]. The collision-free areas (regions) represent all possible translations for an item to be placed without overlaps and staying within the limits of the container [45]. It is obtained by the boolean subtraction from the inner–fit polygon derived from the container and the boolean union of the no–fit polygons induced by all items already placed [57]. The operation must consider uniquely the interior of the no-fit polygon. As a result, collision-free areas can result in a set of multiple disconnected polygons with holes, disconnected edges, or vertices [45].

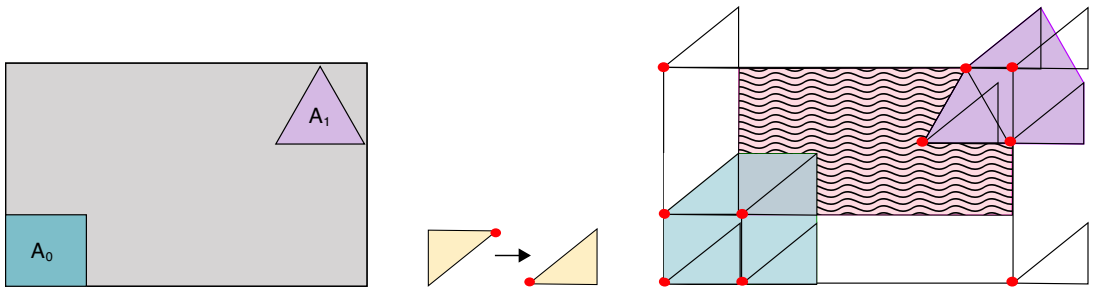
Aiming at producing a tight layout when placing shapes into a container, each shape should have its reference point on the boundaries of one of the collision-free areas [51]. This ensures that the shape will always be connected to at least one already placed polygon or to the boundaries of the container, thus reducing the waste of space [57]. Formally, the collision-free area is defined as [45]:

$$CFR(A, B, C) = IFP(B, C) \bigcap_i \overline{NFP}(A_i, B), \tag{2.4}$$

where B is the shape to be placed inside container C, and A represents a collection of polygons that are already placed in container C. The index i is responsible for identifying each shape A_i placed inside he container C. The term $\overline{NFP}(A_i, B)$ denotes the complement of $NFP(A_i, B)$, and indicates that the shape B should be placed outside of the no-fit polygon formed with A_i.

Figure 2.6 presents an example of the obtaining process of the collision-free area. In the presented example, two shapes are already placed inside the container, thus $A = \{A_0, A_1\}$. The no-fit polygons are

colored according to the colors assigned to their respective shapes. The inner-fit polygon obtained for the addressed container and the considered shape is presented in pink, while the resulting collision-free area is presented in a wavy pattern.



(a) Container C.

(b) Inversion of polygon B.

(c) Collision-free area $CFR(A, B, C)$.

Figure 2.6: Collision-free area.

In summary, the present chapter explores the crucial concepts integral to addressing C&P problems, namely the no-fit polygon, inner-fit polygon, and collision-free area. By exploring these concepts, it is possible to gain insights into effective strategies for avoiding overlaps and fostering a compact layout. Through the explored concepts and techniques, it is possible to perceive the significance of meticulous planning for optimizing spatial arrangements and enhancing efficiency in the packing process.

Chapter 3

Deep Learning

Deep learning, a subfield of machine learning inspired by the intricate workings of the human brain, evolved from the rudimentary perceptrons to the sophisticated neural networks of today [58, 59]. Neural networks consist of interconnected nodes, or neurons, organized into layers [60]. These layers process information hierarchically, transforming input data into meaningful representations. Thus, the term "deep" signifies the incorporation of multiple layers within these networks, allowing for the extraction of hierarchical representations from data. This depth enables neural networks to automatically extract hierarchical features and representations from raw data, allowing them to tackle complex tasks with unprecedented accuracy [61].

The inception of neural networks dates back to the late 1950s when Frank Rosenblatt introduced perceptrons, though limited to learning linearly separable functions [58]. However, the 1970s and 1980s witnessed a neural network winter, as researchers grappled with the challenges of training deeper networks and the absence of effective learning algorithms. A turning point came in 1986 with the development of the backpropagation algorithm by Rumelhart et al. [62], which enabled the efficient training of multi-layer neural networks by updating weights based on the reverse direction of the error gradient. Despite this advancement, the computational power required for deep learning remained a significant constraint. The late 1990s and early 2000s marked a neural network renaissance, as researchers explored more sophisticated architectures and training techniques [63]. However, it was the 2010s that ushered in the deep learning boom, fueled by the convergence of large labeled datasets, powerful GPUs, and algorithmic refinements [64]. In 2012, the deep learning model AlexNet triumphed in the ImageNet Large Scale Visual Recognition Challenge, showcasing the superiority of deep neural networks in image classification [64]. This success paved the way for subsequent architectures, including GoogleNet [65], VGGNet [66], and ResNet [67], each contributing to enhanced performance in specific tasks. Nowadays, deep learning stands as a cornerstone of modern artificial intelligence, with its continuous evolution driven by the combination of algorithmic innovations, increasing amounts of data, and powerful hardware [68]. Currently, the application of deep learning methods extends to complex challenges across various domains, such as computer vision, natural language processing, and speech recognition. In the following sections, some deep learning architectures and concepts that are useful for understanding the proposal of this dissertation are discussed.

3.1 Convolutional Autoencoder

Autoencoder is a type of artificial neural network that is capable of learning the representation of the given data through an encoding and decoding process in an unsupervised manner [69]. However, using basic fully connected layers fails to capture the patterns in pixel data since they do not hold the neighboring information. For this reason, researchers proposed the Convolutional Autoencoders (CAE), in which convolutional layers are used in autoencoders aiming at a good capture of the image data in latent variables [70]. As an auto-encoder, it is based on the encoder-decoder paradigm [71]. The idea behind this architecture is that the encoder performs feature extraction and dimensionality reduction by using the convolution filters and pooling layers of the convolutional layers, while the decoder performs the reverse operation [72]. Convolution layers are used for encoding and deconvolution layers for decoding instead of the fully connected layers [73]. It is trained in an unsupervised fashion allowing it to extract generally useful features from unlabeled data, to detect and remove input redundancies, and to present essential aspects of analyzing data in robust and discriminative representations [71]. With the popularity of various deep learning models, especially generative models, autoencoder has been brought to the forefront of generative modeling [70]. Figure 3.1 illustrates the basic structure of a CAE architecture.

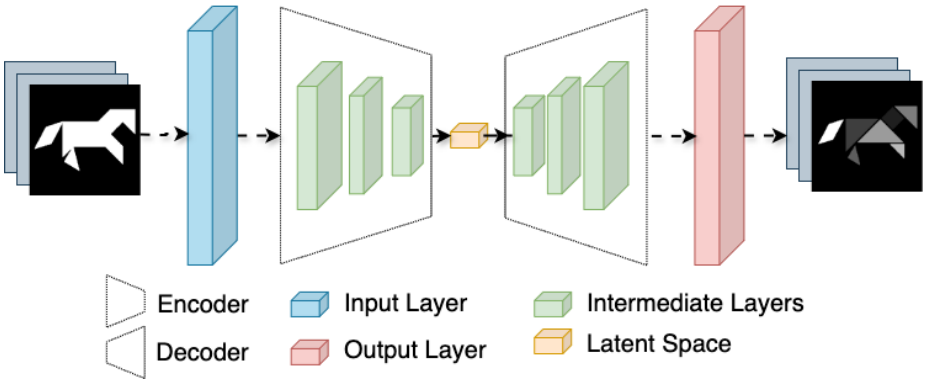


Figure 3.1: Simple CAE architecture.

3.2 Variational Autoencoder

A Variational Autoencoder (VAE) is a type of artificial neural network used in unsupervised machine learning and generative modeling. It is designed to capture and represent complex data in a lower-dimensional latent space while simultaneously generating new data samples that resemble the original input data [74]. VAEs consist of two main components: an encoder and a decoder [75]. The encoder maps input data into a probability distribution in the latent space, typically a Gaussian distribution, with both a mean and a variance. The decoder then takes samples from this distribution and reconstructs the original data. This probabilistic approach allows VAEs to model the inherent uncertainty in data, making them powerful for tasks like data compression, denoising, and generating new data samples with controllable features. VAEs have found applications in various domains, including image and text generation, anomaly detection, and feature learning.

Figure 3.2 illustrates the basic structure of a VAE architecture.

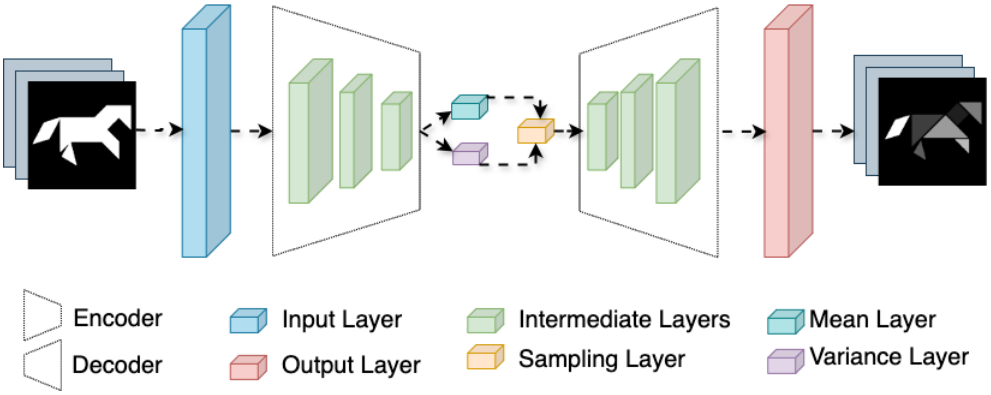


Figure 3.2: Simple VAE architecture.

While both VAEs and CAEs aim to capture meaningful representations of input data, they differ in their fundamental approach. A CAE primarily uses convolutional layers, which are well-suited for image-related tasks, to encode and decode the data. CAEs focus on learning spatial hierarchies and local features within images. However, CAEs typically lack the probabilistic nature of VAEs, meaning they do not explicitly model uncertainty in the latent space. VAEs, on the other hand, not only learn data representations but also model the distribution of these representations, making them more versatile for probabilistic generative tasks and applications where uncertainty quantification is crucial [76].

3.3 U-Net

U-Net architecture is a convolutional neural network architecture designed for semantic image segmentation tasks, where the goal is to classify each pixel in an input image into one of several predefined classes [77]. U-Net is characterized by its U-shaped architecture, with a contracting path on one side and an expansive path on the other [78]. The contracting path consists of a series of convolutional and pooling layers that progressively reduce the spatial dimensions of the input image while learning abstract features. The expansive path then uses transposed convolutions to upsample the feature maps and recover the original spatial dimensions. Skip connections between corresponding layers in the contracting and expansive paths enable the network to capture both high-level semantic information and fine-grained details, making it particularly effective for tasks like medical image segmentation, where precise boundaries and structures need to be delineated. U-Net has become a widely adopted architecture in the field of computer vision due to its exceptional performance in a variety of segmentation tasks. Figure 3.3 presents an example of a U-Net architecture.

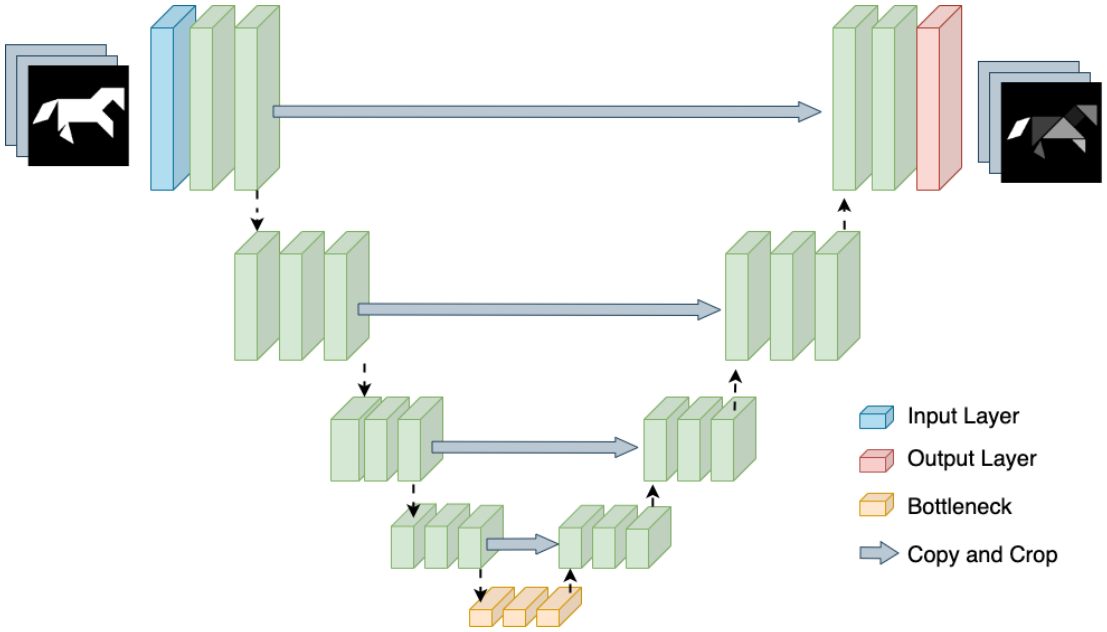


Figure 3.3: Simple U-Net architecture.

3.4 Generative Adversarial Network

Generative Adversarial Network (GAN) is a class of generative models introduced in 2014 by Goodfellow et al. [79]. It started being used in 2017 with human faces to adopt image enhancement that produces better illustrations at high intensity [80]. A basic GAN architecture is formed by a generator and a discriminator [81]. Figure 3.4 presents a simple GAN architecture.

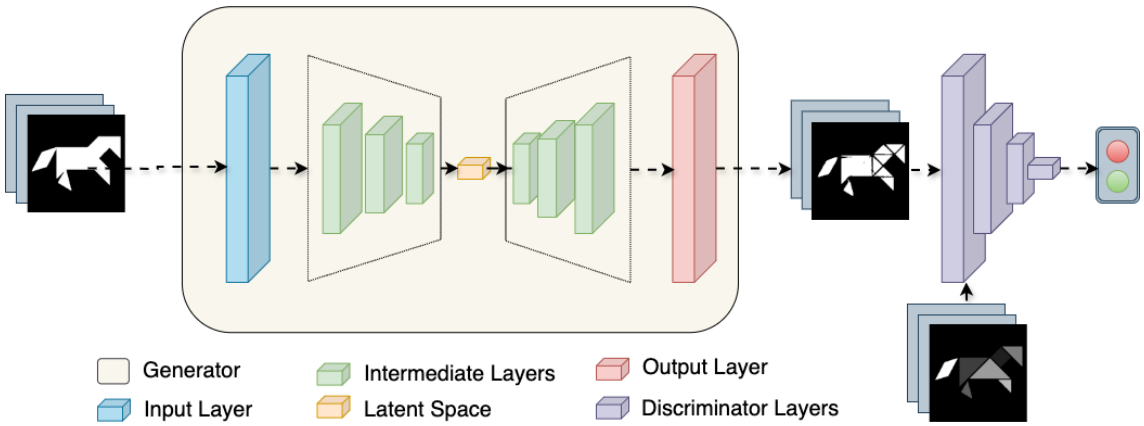


Figure 3.4: Simple GAN architecture.

The idea behind GANs is that they are based on a game, in the sense of game theory, between two machine learning models that are typically implemented using neural networks [82]. The generator tries to capture the distribution of true examples and generate new data examples, while the discriminator is usually

a binary classifier used to discriminate generated examples from true examples as accurately as possible [81]. This process of optimal learning is done as a min-max game problem [83]. In this dynamic, the generator has no direct access to ground truth samples and the only way it learns is through its interaction with the discriminator[84]. On the other hand, the discriminator has access to both the samples generated by the generator, as well as a stack of ground-truth samples [84].

The error signal to the discriminator is provided through the simple ground truth of knowing whether the image came from the real stack or the generator [84]. The same error signal is transferred back to the generator to produce data that are more similar to the ground-truth data [85]. The adversarial relationship between the generator and the discriminator continues until the generated samples cannot be distinguished by the discriminator [86]. At this point, the model has trained so that the liability rate of the network can be increased and the discriminator network can be fooled by producing such candidates that are not synthesized [80].

3.5 Data Augmentation

A common issue that researchers face when training deep learning models is overfit. This issue is common when dealing with small datasets, in which the discriminator overfits the training examples and its feedback to the generator becomes meaningless and training starts to diverge [87]. As a consequence, the model loses the ability to properly generalize the data [88]. In almost all areas of deep learning, dataset augmentation is the standard solution against overfitting [87]. Data augmentation is the process of generating samples by transforming training data, with the target of improving the accuracy and robustness of classifiers [89]. Usual data augmentation operations on images are rotating, cropping, zooming, noise injection, and changes in color scheme [90]. The choice of which operations should be implemented in the data augmentation depends on the task the model is attacking. Inappropriate choices of data augmentation schemes are likely to result in augmented samples that are not informative enough, which leads to no effect or detrimental effect on the accuracy and robustness of classifiers [89]. Figure 3.5 shows some operations of data augmentation that can be performed on an image to be fed to a deep learning model. The image shows augmentations done considering rotation, translation, flip, and scale. Notice that some of these transformations have to be limited when handling Tangram puzzles because they often cause cropping in the Tangram pattern.

(a) Original. (b) Augmented A. (c) Augmented B. (d) Augmented C. (e) Augmented D.

Figure 3.5: Data augmentation performed on a Tangram pattern.

Chapter 4

Related Work

As a convention, it is assumed that to be considered a Tangram puzzle solver a method must be able to assemble at least the simple Tangram puzzles which can be fully characterized by a set of translations and discrete rotations to form a pattern composed of a single connected polygon. It is desired that this method is also able to solve complex Tangram puzzles. These may be composed of multiple regions, possibly with holes. They also may demand the implementation of unconstrained angles of rotation and the reflection of the parallelogram. The following sections present an overview of computational methods that can be used in the automatic solution of Tangram puzzles. This literature review also includes works that are dedicated to other tasks but mention the Tangram as an extension. The works are separated according to whether they use deep learning methods. The focus is to summarize each approach and determine its advantages and limitations in solving both simple puzzles and complex puzzles.

4.1 Non-Deep Learning Tangram Solvers

The solution to Tangram puzzles dates back to 1972 with the heuristic programming method by Deutsch & Hayes [91]. Before the concept of deep learning became well-established, many authors proposed to solve the Tangram using more traditional approaches. They often relied on information regarding the geometry of the pieces and faced the assembly process as an optimization problem. The following paragraphs present the works identified in the literature that are not based on deep learning approaches.

Deutsch & Hayes [91] solve Tangram puzzles using heuristic programming. Their algorithm performs attempts and tests on partitioning a desired polygonal pattern into smaller parts called sub-puzzles. It follows the contour of the pattern and, in convex corners, generates extension lines that determine a possible section of the pattern. The method applies ten rules that consider the arrangement of the edges of the pattern and the extension lines with the pieces and the composites. Composite is a term introduced in their paper and refers to convex regions formed by a set of pieces. The algorithm rearranges the pieces correctly for 9 out of the 10 puzzles used for testing, but the authors present cases that their approach could not solve. No performance metric is mentioned except for the number of solved puzzles. In addition, the approach is limited to patterns without holes and only allows rotations of pieces by angles at multiples of 45^o.

Oflazer [92] follows a connectionist approach by representing the placement and orientation of the pieces as a non-restricted Boltzmann machine. The pieces are initially laid out on a regular grid. Possible positions

and rotations are represented by neural units that receive excitatory connections from input units that define the puzzle, and lateral inhibitory connections of conflicting units. Also, the grid points are labeled with coordinates (i, j), in which i indicates the row and j indicates the column in the grid in standard matrix notation. The author states that the eight units: Left (L), Right (R), Down (D), Up (U), Right-Up (RU), Right-Down (RD), Left-Up (LU), and Left-Down (LD) associated with each grid point indicate in which orientations the puzzle area or boundary extends around that grid point. To test the proposed approach, the author considers 10 different puzzles and runs the method 100 times for each puzzle. The method can solve all the considered puzzles, with around 70% of the runs converging in less than 500 epochs. The method is flexible in solving patterns that require the reflection transformation for the parallelogram, and patterns with holes. However, it limits the rotations of pieces by angles at multiples of 45^o due to the regular grid.

Bartoněk [93] presents an evolutionist approach to solving polygonal jigsaw puzzles. This method presents an extension for the solution of Tangram puzzles in which pieces are represented by string codes. In a string code, the edges and angles are represented by integer numbers invariant to rigid body transformations. Based on the theory of cluster analysis, each piece is assigned to a certain group according to the calculated similarity between the piece and the other pieces belonging to the same group based on its string codes. A fitness function determines how many groups the pieces will be divided into. The advantage of the developed algorithm consists in the selection of fragments that have a suitable evaluation value, reducing computing costs. The author affirms that the performance of their algorithm depends not only on the number of segments but also on their variability, i.e., the number of string codes. The author mentions the possibility of extending the evolutionist approach to Tangram but does not perform experiments on Tangram puzzles. This approach cannot be used to solve complex Tangram puzzles of any kind.

Kovalsky et al. [35] present a method for solving jigsaw puzzles in terms of algebraic concepts. The puzzle is modeled as a system of polynomial equations so that any solution of the system is a solution of the puzzle as a complete representation. The authors first propose to solve edge-matching puzzles. However, they show how to apply their approach to Tangram puzzles by considering it as an edge-matching puzzle in which all pieces have the same color. It is possible to notice that the orientation of each piece is fixed. The authors argue that the method can be modified to assimilate the solution of puzzles with rotations, although limited to a discrete set of rotations, although they do not present any example considering this modification. They further extend their method to variants of edge-matching puzzles including higher-dimensional puzzles, and puzzles with irregular pieces. They test their method on only 2 Tangram puzzles, both of which are successfully solved. Their tests for Tangram exhibit an undue level of simplicity, with no performance metric being considered except for the number of solved puzzles. This approach cannot be used to solve complex Tangram puzzles of any kind.

Domokos & Kato [94] propose to solve the shape realignment problem, where given a template image of an object and its broken fragments, the goal is to find an aligning transformation that reassembles the complete template object. Their method consists of constructing a polynomial system of equations that describe the problem whose solution provides the parameters for alignment. To validate their approach, the authors conduct experiments on a limited set of Tangram puzzles. They showcase successful solutions to seven Tangram puzzles, achieving an average completion time of 50. Although their approach is not explicitly tested on puzzles with unconstrained rotations, it is designed to handle this type of transformation. An important

aspect to be noticed from their experiments is that they start with a feasible solution to the puzzle, then they shuffle the pieces to obtain their initial position. This characteristic diminishes the combinatorial nature of the problem, as the initial piece arrangement is already considerably close to the desired solution. This aspect is related to another drawback of this method because the parallelogram has to be pre-reflected in the correct enantiomer before applying their method, making it not prepared to handle this aspect in complex puzzles.

4.2 Deep Learning Tangram Solvers

There is a growing interest in developing deep learning architectures for solving jigsaw puzzles. One may think that these architectures could serve as major inspirations for the implementation of my architecture for the automatic solution of Tangram puzzles. However, deep learning approaches dedicated to jigsaw puzzles present significant drawbacks [3]. Many authors limit their approaches to square jigsaw puzzles and address the position of the pieces as indexes or pseudo-labels [95]. Unlike square jigsaw puzzles, Tangram pieces have distinct geometries, and mapping every possible placement for each piece in the puzzle is impracticable in reasonable running time. They also rely heavily on the semantic information contained in each piece to assemble the puzzle, which is an unrealistic approach for the Tangram because its pieces are apictorial. Therefore, to solve Tangram puzzles using deep learning models, it is more adequate to implement architectures that are dedicated to this task. These architectures should consider the geometric properties of the Tangram pieces, including the ones that characterize complex puzzles.

Li et al. [96] present a GAN architecture that synthesizes layouts by modeling geometric relations of different types of graphical elements. The generator takes as input a set of randomly placed graphic elements and uses self-attention modules to refine their labels and geometric parameters jointly to produce a realistic layout. The authors propose a novel differentiable wireframe rendering layer that maps the generated layout to a wireframe image, upon which a CNN-based discriminator is used to optimize the layouts in image space. Although it is not their main focus, they solve some Tangram puzzles as a validation for their approach. They collect 149 Tangram graphic designs including animals, people, and objects. In the performed experiments, the authors consider eight configurations by varying rotation/reflection poses for each piece. They randomize the Tangram pieces and the model has to move them to the desired pattern. Their model can generate meaningful solutions like fox and person, although others may be hard to interpret. Through visual analysis, it is possible to infer that their method can assemble 5 out of 12 puzzles considered in the experiments Due to a lack of metrics that can determine how good is a generated solution compared to the ground truth, the evaluation for Tangram puzzles ends up depending only on visual resemblance.

Lee et al. [33] present a problem formulation and then propose an efficient and effective learning-based approach to solving this problem. They split different two-dimensional target shapes into multiple fragments of arbitrary polygons by a stochastic partitioning process. They then proceed to implement an agent to assemble the target shape given the partitioned fragments while the original poses are hidden. Given a target object and a set of candidate fragments, the proposed model learns to select one of the fragments and place it into the right place. Unlike an approach based on the backtracking method, which considers how the remaining fragments will fit into the unfilled area of the target shape, they attempt to solve such

a problem with a learning-based method. The authors claim that the proposed method effectively learns to tackle different assembly scenarios. They further explore the level of generalization of their method by submitting their model to different scenarios, such as cases with missing fragments, distorted fragments, and different levels of rotation discretization. They claim that the addressed problem is analogous to the Tangram, but do not present experiments on Tangram puzzles. Although their representation can depict puzzles with holes or multiple regions, they do not test their approach to these patterns. They also do not implement the reflection transformation.

4.3 Summary of Literature

Several research problems can be inferred from this literature review when it comes to the solution of both simple and complex Tangram puzzles. Table 4.1 presents a comparison of the methods in this present literature review, where it analyzes the employed approach, the presence of tests considering the Tangram, and their capability of solving different Tangram puzzles. For aspects that describe complex puzzles, refer to the terms used in Figure 1.4.

Table 4.1: Comparison of solvers included in the literature review.

Work	Approach	Exp. Tangram	Complex Puzzles			
			Holes	Multiple	Unconstrained	Flip
Deutsch & Hayes [91]	Heuristic Method	✓				✓
Oflazer [92]	Neural Network	✓	✓			✓
Bartoněk [93]	Genetic Algorithm					
Kovalsky et al. [35]	Algebraic Concepts	✓				
Domokos & Kato [94]	Algebraic Concepts	✓			✓	
Li et al. [96]	Generative Model	✓				
Lee et al. [33]	Transformer Model		✓	✓	✓	

Many works focus on other visual tasks and use the Tangram to assess the versatility of their approach. As a result, they end up ignoring some aspects that are unique to the Tangram and characterize complex Tangram puzzles. One critical aspect that defines whether an approach will be able to solve patterns with holes or multiple regions is how the puzzle and pieces are represented. Some works assume that the puzzle area can be described as a single connected contour, which is valid for many cases but disregards the existence of more complex arrangements of pieces. The literature shows that raster-based or image-based representations are versatile in the sense of being able to represent any kind of puzzle.

Many works also do not consider the implementation of unconstrained rotations and the execution of the reflection transformation, which serves as a way to reduce the number of configurations a piece can assume. This strategy is useful in some cases, especially considering that many Tangram puzzles are formed by pieces rotated in multiples of 45^o and do not require the reflection transformation. However, it mitigates the combinatorial nature of the Tangram and ignores puzzles that demand more complex piece configurations.

When they test their approaches to Tangram puzzles, different authors often consider only a limited set of patterns in their experiments, which varies considerably from work to work. This highlights the necessity of a dataset for Tangram puzzles to be used as a baseline of comparison. Another notable observation in the literature is the absence of a uniform baseline for comparison, with some authors using running time and others using the number of iterations to assess the performance of their respective methods.

Chapter 5

Dataset

This chapter is dedicated to presenting the process for generating the proposed data-driven dataset for the automatic solution of the Tangram puzzles task. The goal is to present a data-driven dataset to be used for training and testing deep learning models. This dataset is also useful for approaches that do not use machine learning by using only the testing set to evaluate the performance of these approaches. Therefore, this dataset can be used as a basis for comparison of methods that aim at solving the Tangram, independently whether they demand a training stage or not. In the following paragraphs, some key aspects that are considered during the assembly of the proposed dataset are discussed.

Large training datasets and deep complex structures enhance the ability of deep learning models for learning effective representations for tasks of interest [97]. If the model is fed with poor or insufficient data, it might be unable to generalize accurately [98]. 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 dataset for deep learning, 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 dataset needs to contain a sufficient amount of data aiming for the model generalization. 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. It is desired that the data is sufficiently diverse for the model to learn the geometry of the pieces, and understand how these pieces can interact to form patterns. Additionally, it is also necessary to guarantee that this dataset contains a sufficient amount of samples of both simple and complex puzzles. The dataset should contain a sufficient amount of each type of complex puzzle for the model to be able to learn the aspects that characterize them. A straightforward approach is to calculate statistics for this dataset regarding a taxonomical analysis. These statistics are useful to determine the number of puzzles that contain the aspects that characterize complex puzzles and determine if they are sufficient for the model to learn such features.

It is also important to guarantee that this dataset is diverse regarding the geometry and morphology of the included Tangram puzzles. To accomplish that, a possible approach is to compare every pair of patterns included in this dataset using a metric that tells the visual similarity between them. By doing that, it is possible to analyze if this dataset presents a sufficient variation in its samples or not. Another possible approach would be using a technique that can extract morphological features from each pattern included in this dataset. These techniques are more focused on the geometry being depicted in the image and some

of them are invariant to translation, rotation, and scale. By using these two approaches, it is possible to determine whether the samples present variation not only regarding visual features but also concerning geometric and morphological features.

The following sections present the process for forming the proposed dataset. First, an outline to detail how the data is organized into sets to be fed to the model is presented. Additionally, a detailed statistical analysis of the dataset regarding taxonomy statistics and data correlation statistics is also performed. The literature review suggests that this dataset is the most extensive in the literature. It is also the very first data-driven dataset for Tangram puzzles, being also a groundbreaker in contemplating all types of complex puzzles. It is expected that this new dataset can serve as a valuable resource for future research concerning the solution of dissection puzzles, as well as related optimization problems [1].

5.1 Dataset Collection

The proposed data-driven dataset contains 6,000 samples. To form this dataset 163 samples were collected from the literature and generated extra 5,837 random samples. The samples from the literature were collected from the Tangram Channel [99]. The following paragraphs are dedicated to explaining this data collection, and some criteria used to assemble this dataset. To generate random samples, a random Tangram generator implemented in Javascript by Köpp [100] is used. The first thing the generator does to generate a pattern is to randomly pick one of the enantiomers of the parallelogram to be used in the pattern composition. After that, the generator shuffles the order of the pieces and places the first piece in the center of the drawing space forming an intermediate pattern. It picks the next piece and randomly selects one of the corners of this piece to be attached to the intermediate pattern. It randomly selects one of the corners of the intermediate pattern to be attached to the next piece and then calculates a probabilistic distribution to find which orientation of the next piece increases the length of shared edges and contact with the intermediate pattern. It places then the next piece in this orientation forming a new intermediate pattern. This process continues until all the pieces are used.

As stated before, the generator can form puzzles using the reflection transformation for the parallelogram. It also enables the generation of patterns with holes. However, the rotations of the pieces are always limited to multiples of 45°, and the resulting pattern is always formed by a single connected region. Therefore, the generator is not prepared to generate puzzles with unconstrained rotations, or with multiple regions. For this reason, the original code is modified to generate such patterns, thus making it capable of generating all kinds of complex Tangram puzzles.

In the original implementation, the rotation is done using a rotation matrix presented in Equation 5.1. This matrix rotates a point $v = (x, y)$ counterclockwise through an angle θ about the origin of a two-dimensional Cartesian coordinate system [101].

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{5.1}$$

[1] Dataset available on: github.com/fernandamyamada1/TANGAN/.

where θ is the angle of rotation. The process for obtaining the new coordinate of a point $v = (x, y)$ after the rotation consists of a matrix multiplication. This process is illustrated in Equation 5.2.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{5.2}$$

where x' and y' represent the coordinates of point $v = (x, y)$ after the rotation in the horizontal and vertical axes respectively. The original implementation assumes that the values for θ are only multiples of 45°, which prevents the generator from constructing patterns with unconstrained rotations. This implementation is modified by varying the value for θ during the generation process. Determining how many and which pieces will be rotated using unconstrained rotations is done randomly. This way it is possible to guarantee that this dataset contains patterns with unconstrained rotations.

The procedure employed in the original implementation always links one corner of the next piece to one corner of the intermediate pattern. This implies that the pieces are always connected by at least one point to one another. To enable the generator to form patterns with multiple regions, whenever the generator picks the next piece it randomly determines whether the next piece will be attached to the intermediate pattern or not. If it is determined that it will not be attached to the intermediate pattern, it is randomly placed in the surroundings of the intermediate pattern, thus forming a new connected region. This new connected region becomes the new intermediate pattern, to which the next pieces might be attached to. By using this strategy, it is possible to generate patterns that are composed of multiple regions.

Image processing techniques are used to retrieve a pair of images that represent a pattern and a solution for each generated sample. This standardizes the representation of the samples that were collected from the literature and samples that were randomly generated. Therefore, at this point, for both samples collected from the literature and randomly generated samples, a single sample is composed of a 512×512 grayscale image depicting the puzzle and a 512×512 grayscale image depicting a feasible solution. Figure 5.1 presents some examples of Tangram samples as binary images. The first row, represents the desired pattern, while the second row presents a feasible solution for each presented pattern. The first two puzzles on the left side are samples collected from the literature, while the other two are randomly generated ones.

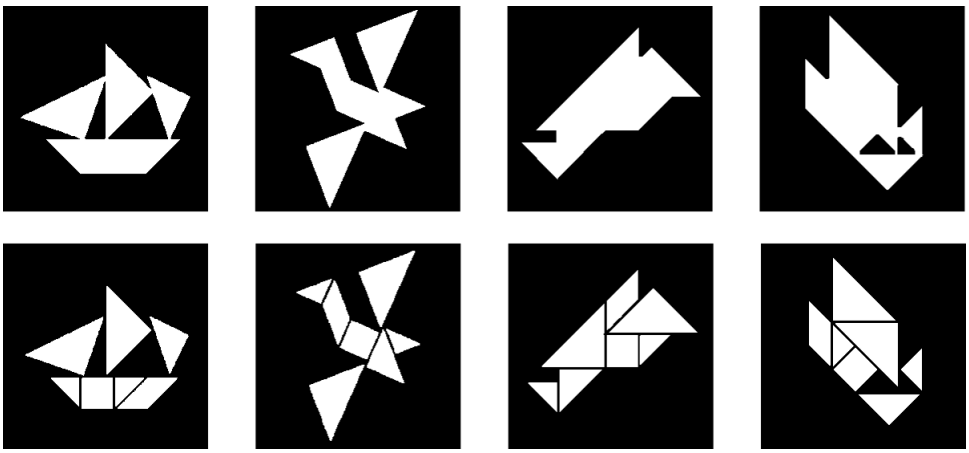


Figure 5.1: Binary images representing literature and generated samples.

Another procedure is applied to the dataset to make it more sophisticated and enhance the visual distinction of the pieces in the Tangram solution. The procedure consists of assigning different grayscale tones to each piece. The hypothesis is that the different tones help the model to extract geometric features from the pieces and understand how they interact with the Tangram patterns. The solutions are susceptible to some level of imprecision originating from image processing applied when converting the images to grayscale. It is also important to make sure that the puzzles have approximately the same area by standardizing the sizes of the pieces. This aspect of this dataset benefits the model conversion since it reduces the necessity of the model to estimate the size of the pieces during the assembly process. Specifically for this dataset, the area condition is that the puzzle needs to be close to 14395 pixels within a $\pm 5\%$ tolerance. It is assumed that this number of pixels is sufficient for representing a puzzle inside a 512×512 area while enabling the distinction of pieces in the correspondent solution. Figure 5.2 presents some examples of samples that are included in this dataset.

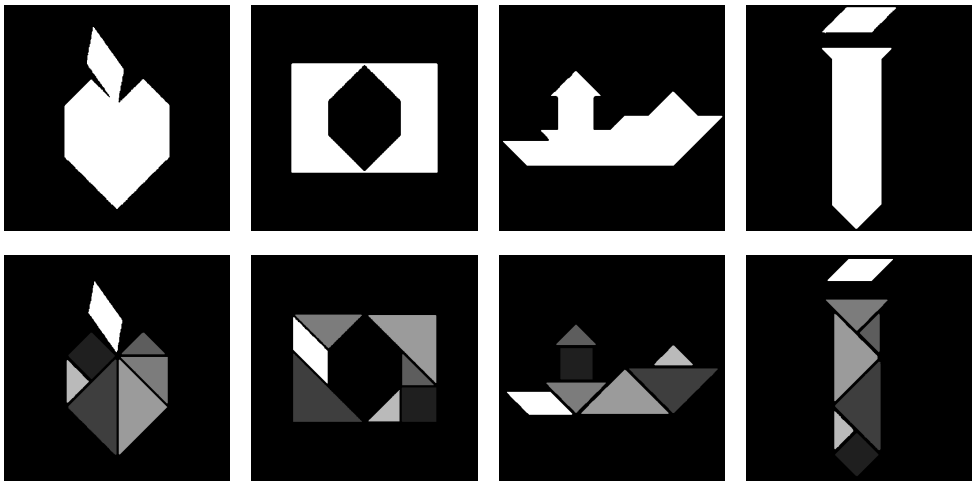


Figure 5.2: Final representation of samples included in the dataset.

5.2 Dataset Outline

The dataset is split into 5,900 samples for the training set and 100 samples for the testing set. At the beginning of the training procedure, if validation is needed, the model may randomly select 100 samples from the training set to form the validation set. Therefore, there are 5,800 samples for training, 100 samples for testing, and 100 samples for validation.

An important aspect to be noticed from the testing set is that it contains samples showing Tangram puzzles with varied geometric features. It is also desired to have samples that represent the different aspects that characterize complex puzzles. This way, it is possible to accurately assess the ability of the model to handle various geometric patterns and their applicability in different scenarios, thus guaranteeing its robustness and overall confidence and reliability. Another crucial consideration is the ability to compare the generated solutions with experiments conducted by other researchers in the literature. To address this, 15 samples within the testing set were intentionally chosen due to being used in experiments detailed in related

works or to exhibit noteworthy similarities to puzzles investigated in these studies. Figure 5.3 presents these samples. Deutsch & Hayes [91] experiments on sample (i), while samples (f), (l), and (m) share some similarities with other puzzles considered in their work. Oflazer [92] considers (o) for testing, while sample (j) shares some similarity with another sample used by the author. The only pair of Tangram puzzles used for testing by Kovalsky et al. [35] are samples (n) and (o). Domokos & Kato [94] use samples (e), (i), and (k) in their experiments, while some other considered puzzles resemble samples (c) and (h). Li et al. [96] tests on samples (b), (d) and (g), while samples (a) and (n) closely resemble other considered puzzles.

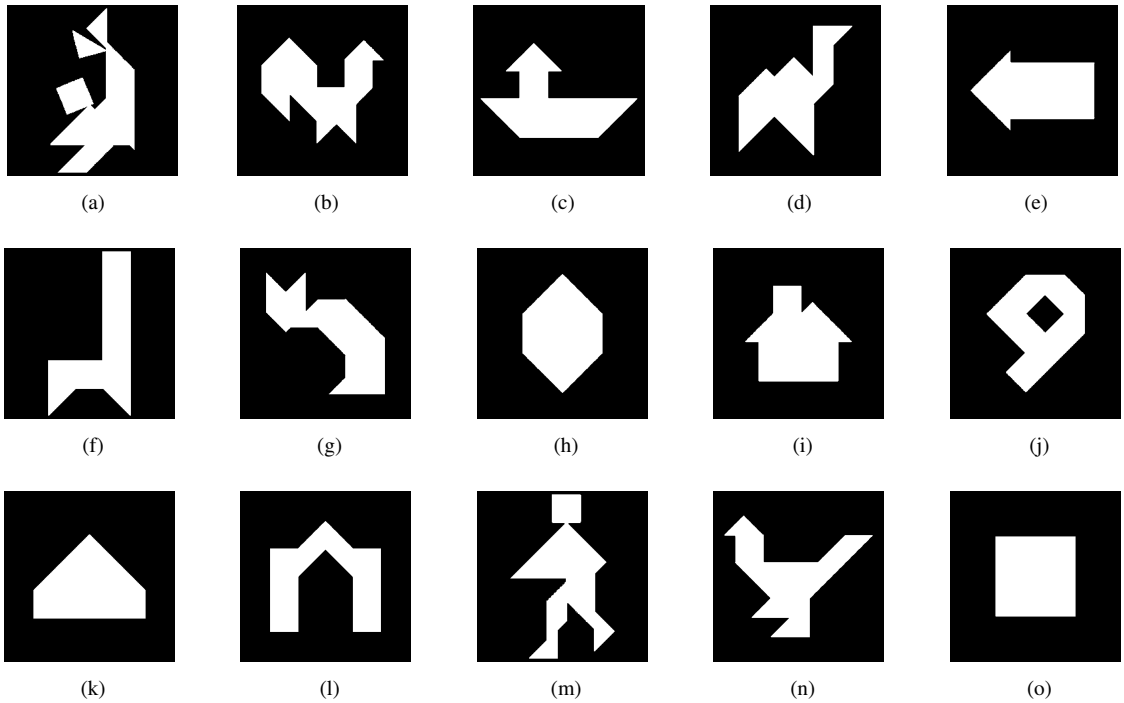


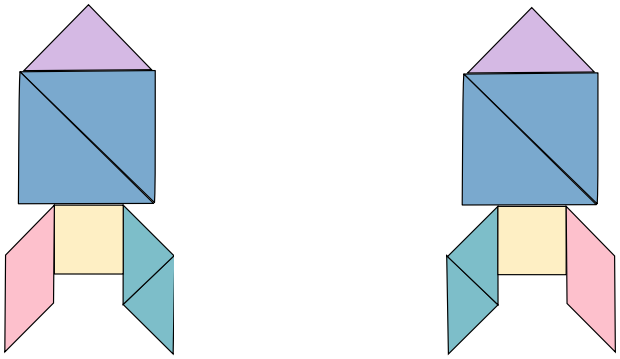
Figure 5.3: Samples included in the dataset similar to the literature.

5.3 Dataset Statistics

Since a data-driven dataset is proposed, it is important to guarantee that this dataset is not unbalanced and presents a sufficient amount of samples with different features. In this case, the features that this dissertation focuses on concern the taxonomy of the computational complexity of Tangram puzzles. It is also desired to include in this dataset samples that are diverse in terms of their geometry and morphology. This enhances the learning process by assuring that the model will have access to a diverse range of examples. Therefore, taxonomy and data correlation analyses are performed on this dataset. The former examines how diverse is this dataset in terms of taxonomical features, while the latter examines how diverse is this dataset in terms of visual, geometric, and morphological features.

5.3.1 Taxonomical Statistics

Considering the main objective of this dissertation, the most logical approach is to analyze this dataset regards the taxonomy of its samples. However, there is a problem when classifying puzzles regarding the reflection of the parallelogram. When looking at a Tangram pattern, determining if it has multiple regions or holes is straightforward. The same can be said of the demand for unconstrained rotations to a certain extent. However, determining if a Tangram puzzle demands the reflection transformation is not usually easy. Generating a Tangram pattern using a flipped parallelogram does not guarantee that this particular pattern cannot be assembled using the unflipped parallelogram. This is due to the fact that the solution of a Tangram puzzle is not necessarily unique [102]. This difficulty persists even in patterns where the parallelogram is isolated from the other pieces. This happens because it is possible to combine two small triangles to form a parallelogram that is congruent to the parallelogram piece. In Figure 5.4, there are two solutions for the same pattern containing distinct enantiomers for the parallelogram.



(a) Solution with enantiomer A.

(b) Solution with enantiomer B.

Figure 5.4: Solutions for the same pattern with different enantiomers for the parallelogram.

Table 5.1 shows a taxonomical analysis of the proposed dataset samples regarding the aspects that characterize complex Tangram puzzles. The flip ratio refers to the distribution of both enantiomers of the parallelogram found in the considered ground truth set, being the first enantiomer present in Figure 5.4 (a) and the second enantiomer present in Figure 5.4 (b).

Set	Holes	Multiple	Unconstrained	Flip Ratio
Training	514	1984	1521	3255 / 2645
Testing	34	40	14	70 / 30
Total	548	2024	1535	3325 / 2675

Table 5.1: Dataset statistics regarding complex Tangram puzzles.

The taxonomical analysis presented in the current section is sufficient to demonstrate that this dataset

contains an adequate amount of samples for simple, as well as every type of complex Tangram puzzle. Both puzzles with unconstrained rotations and puzzles with multiple regions cover an expressive number of samples in this dataset. Puzzles with holes are the ones that contemplate the least number of samples but still represent more than 10% of the entire dataset. The aspects are also well balanced in the training and testing sets, presenting a fair distribution of the samples that will be used in the learning process and its assessment.

Moreover, regarding the parallelogram reflection, it is possible to notice that the samples are well distributed among both enantiomers in the training set. The same can be said about the distribution of samples in the training and testing sets. There are no unbalanced conditions in any of these sets, which is a favorable factor for the learning process of a model. Additionally, for deep learning models, a viable alternative solution to mitigate the parallelogram reflection problem is to perform data augmentation by randomly flipping each input included in the batch on the horizontal and vertical axis. This practice ensures that the model will be fed with samples containing both enantiomers of the parallelogram, even if the dataset does not include such samples. The strategy for data augmentation is detailed in Chapter 8.

This taxonomical analysis shows that in a preliminary examination, this dataset seems to meet the aforementioned requirements for the construction of a data-driven dataset. This taxonomical analysis is important to support an adequate learning process for deep learning models. This taxonomical investigation has never been performed before in the works that address the automatic solution of Tangram puzzles. This fact establishes this dataset as the very first data-driven dataset with a strong analytical basis that is dedicated to the solution of the Tangram.

5.3.2 Morphological Statistics

Another aspect that is important to be analyzed is the level of diversification concerning the Tangram patterns that are included in this dataset. For the Tangram, addressing this issue involves analyzing the geometric features presented in each puzzle. To perform the data geometric analysis on this dataset, Hu moments are considered. These mathematical moments are used in image processing and computer vision to characterize the shape or contour of objects in binary or grayscale images [103]. These moments are invariant to translation, rotation, and scale, making them valuable for shape analysis, pattern recognition, and object classification tasks. To compute Hu moments, it is necessary to first calculate the normalized central moments η_{pq} using the following formula:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}}, \tag{5.3}$$

where μ_{pq} is the central moment of order $p+q$ and μ_{00} is the zeroth-order central moment.

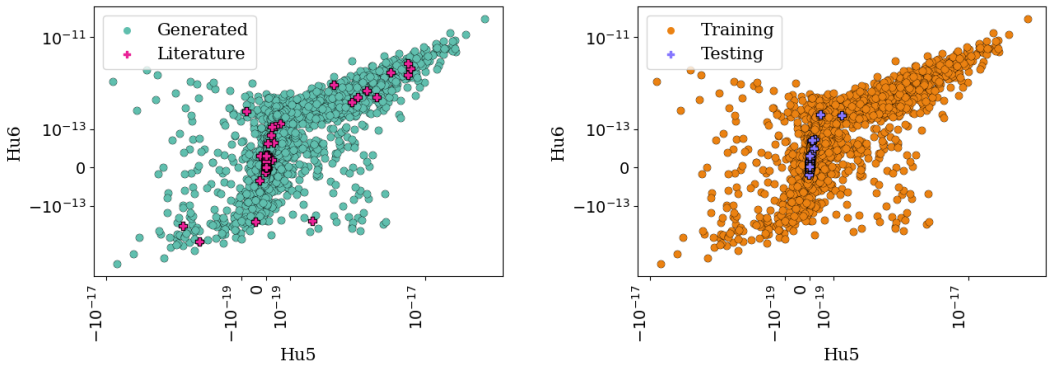
Central and normalized moments are invariant to scaling and translation already [104]. The challenge is to find rotational invariants. These normalized central moments are then used to compute the Hu moments as linear combinations of the central moments, each representing distinct geometric properties and relationships of the object.

$$
\begin{aligned}
\mathrm{Hu}_1 &= \frac{\eta_{20} + \eta_{02}}{\eta_{00}^2}, \\
\mathrm{Hu}_2 &= \frac{(\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2}{\eta_{00}^4}, \\
\mathrm{Hu}_3 &= \frac{(\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2}{\eta_{00}^5}, \\
\mathrm{Hu}_4 &= \frac{(\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2}{\eta_{00}^5}, \\
\mathrm{Hu}_5 &= \frac{(\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]}{\eta_{00}^{10}}, \\
\mathrm{Hu}_6 &= \frac{(20\eta_{20} - 20\eta_{02})^2 + 6(30\eta_{11})^2}{\eta_{00}^{10}}, \\
\mathrm{Hu}_7 &= \frac{\eta_{21} \cdot (\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{03} \cdot (\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]}{\eta_{00}^{10}},
\end{aligned}
\tag{5.4}
$$

where Hu_i represents the i-th Hu moment, η_{pq} is the normalized central moment of order $p + q$, and η_{00} is the zeroth-order normalized central moment.

Two scatter plot illustrating the Hu moment values is presented in Figure 5.5. The horizontal axis corresponds to the Hu_5 value, while the vertical axis corresponds to the Hu_6 value. The practice of choosing two representative moments to be used to compare the morphological structure of images has been done in the literature [105]. Hu_5 quantifies how the shape is oriented concerning its longest dimension [103]. As a consequence, shapes that are not perfectly symmetrical are expected to have higher Hu_5 values, as they are more sensitive to orientation changes. Hu_6 captures the deviation of the shape from being a perfect circle [103]. It is related to the angularity of the shape, which implies that shapes with sharp corners, edges, or irregular shapes will have higher Hu_6 values. For instance, an elongated ellipse is expected to have a high Hu_5 value, but a low Hu_6 value. These two moments are the ones that better describe the morphology of a Tangram pattern because they are sensitive to its arrangement of pieces and its geometric features. Each dot in the plot represents a pattern included in this dataset, and a color code indicates which dots are part of the training set and testing set. In both plots, the mean value for Hu_5 is 2.697×10^{-19} and the mean value for Hu_6 is $.249 \times 10^{-13}$. For better visualization, the graph uses a symmetrical logarithmic scale where the linear threshold parameter for the x-axis is the mean value for Hu_5, and for the y-axis is the mean value for Hu_6.

In Figure 5.5 (a), each dot in the plot represents a pattern included in this dataset, and a color code indicates the samples are from the literature and the generated ones. It shows that the generated samples closely resemble the geometry of the literature samples. Since the literature samples represent real-world Tangram puzzles with a meaning behind their construction, the graph indicates a favorable characteristic that opens the possibility of utilizing the generated samples to train a deep learning approach for solving real-world problems. When the generated samples exhibit a close alignment with real-world scenarios, it signifies that they capture essential features and patterns present in the actual data. This theory underscores



(a) Generated and literature sets.

(b) Training and testing sets.

Figure 5.5: Morphological analysis of samples included in the dataset.

the importance of generating high-quality synthetic data that faithfully mirrors the characteristics of the desired real-world dataset. It highlights the potential for leveraging such generated samples in the training process, ultimately leading to more robust and effective machine-learning models.

In Figure 5.5 (b), each dot in the plot represents a pattern included in this dataset, and a color code indicates which samples are part of the training set and testing set. The testing set distribution seems to be well aligned with this training set distribution. This is important because it indicates that during the training stages, the model is fed with examples that are close to the ones that are used to address its performance. Therefore, it shows that the model should be able to make good predictions considering that during training, it has access to plenty of training samples that are fairly similar to the testing ones. If the model is not able to make good predictions, the issue is probably related to the model itself or the designed learning schedule.

Chapter 6

Heuristic Approach

This chapter is dedicated to presenting the proposed heuristic approach for the automatic solution of Tangram puzzles. Figure 6.1 presents the heuristic method diagram.

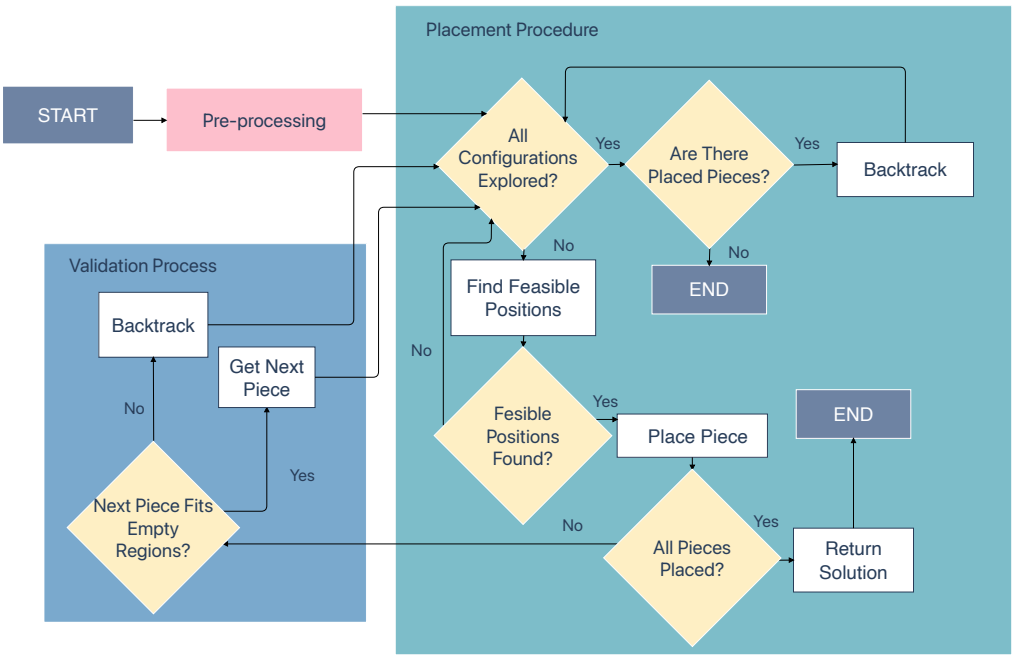


Figure 6.1: Flowchart presenting the main stages of the proposed heuristic method.

The heuristic is separated into three main blocks. The pre-processing block, detailed in Section 6.1, is responsible for receiving the desired Tangram pattern and calculating all the possible configurations each piece can assume throughout the execution of the heuristic. The heuristic generates a list of all the possible configurations that each piece can assume considering orientations and reflection transformation. The placement procedure block, described in Section 6.2, is responsible for executing the assembly process considering the input Tangram pattern. The assembly process fits the pieces one by one and follows the largest

first heuristic, which determines that the heuristic must consider the pieces according to their area from largest to smallest [106]. Last, Section 6.3 presents the validation block, which is responsible for verifying whether an obtained intermediate solution of the puzzle allows the positioning of the remaining pieces.

After presenting each stage of the heuristic approach, the present chapter presents in Section 6.4, a proof of concept that shows that the proposed heuristic approach can efficiently deal with the aspects that characterize complex Tangram puzzles. For this preliminary investigation, only 30 puzzles are considered, among which are included both simple and complex puzzles.

6.1 Pre-processing

This section details the representation chosen for describing the geometry of the input Tangram pattern and the Tangram pieces. An important aspect of the heuristic method is that it is raster-based, thus it receives a binary mask referred to as a pattern mask that represents the desired Tangram pattern. In this binary mask, the puzzle region is represented as white pixels, while black pixels represent areas outside the puzzle region. The foremost advantage of this representation is that it easily permits the depiction of complex Tangram puzzles with holes and multiple regions. Areas covered by holes and gaps between puzzle regions are assigned as black pixels, indicating that they cannot be filled by pieces. As stated before, many previous works treat the Tangram pattern as a single connected contour, thus ignoring the existence of some complex puzzles. To foster understanding, areas composed of black pixels may be referred to as occupied areas, while areas composed of white pixels are often referred to as unoccupied areas. Some examples of pattern masks are presented in Figure 6.2. Throughout the assembly process, whenever a piece is placed on an occupied area, the pixels corresponding to the raster conversion of that piece turn black in the pattern mask. By the end of the assembly process, it is expected that all of the pixels of the pattern mask (or at least most of them) will be assigned black.

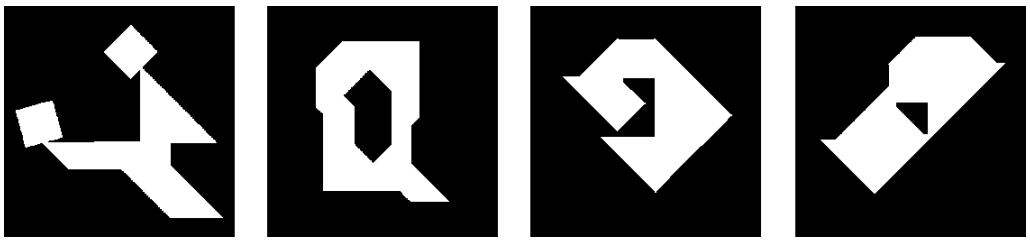


Figure 6.2: Pattern masks in raster representation.

The Tangram pieces are originally represented in vector format inside a local coordinate system as a list of vertices that have their origin located in the center of the correspondent piece. Thus, the Tangram pieces can be rotated and reflected without losing precision in their representation. Also, the size of the Tangram pieces is determined according to the empty regions of the Tangram pattern, since the total number of pixels of the Tangram pieces has to be the same as the total number of pixels of the Tangram pattern empty regions. Each Tangram piece to be placed inside the puzzle area contains the following properties:

1. t: an array containing two values belonging to [0, 1] that represents the translation of the piece in the x and y axis relative to the pattern mask.

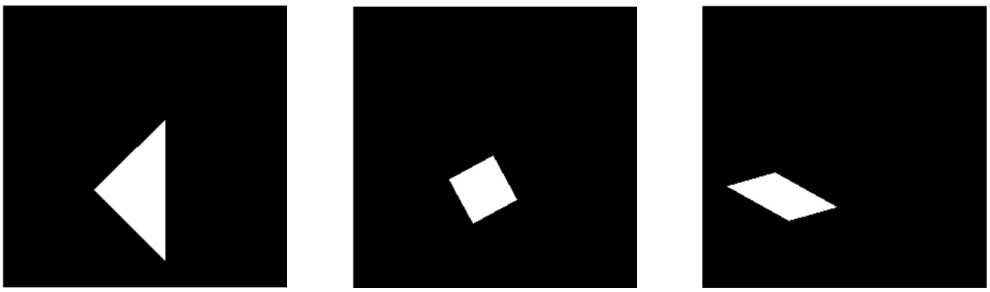
2. θ: integer between 0^o and 359^o representing the angle of rotation of the current piece.

3. f: binary value expressing the execution or not of the reflection transformation in the current piece.

As stated before, each piece has its list of translations, angle of rotation, and a flag indicating whether the piece is reflected. As a pre-processing step, the method generates a list of all the possible configurations that each piece can assume considering orientations and reflection transformation. Since many Tangram patterns do not demand unconstrained rotations and the reflection transformation, configurations that combine angles with multiples of 45º and no reflection transformation have priority and are considered first in the puzzle assembly process.

Initially, the properties for the pieces are set in their corresponding original state, i.e. their placement in the initial square. Thus, during the puzzle assembly process, the values assigned to each piece property consider the piece original state centroid as a starting point to execute the isometric transformations. In addition, the values assigned to these Tangram pieces properties are constantly changed in the process of finding a feasible placement for each piece. Furthermore, to determine the current positioning of a certain piece inside the puzzle area, the information concerning this particular piece is converted to a binary mask with the same dimensions as the pattern mask, which will be referred to as piece mask throughout the dissertation. At the end of the assembly process of the Tangram puzzle, it is expected that the values stored in the data structure pieces produce a feasible solution for this Tangram puzzle.

Figure 6.3 presents some examples of piece masks generated from the pieces information. Figure 6.3 (a) presents the piece mask generated from one of the largest triangles and corresponds to this set of data: t = [0.48 0.65], $\theta = 180^o$, f = 0. In addition, Figure 6.3 (b) presents the piece mask generated from the square and corresponds to this set of data: t = [0.45 0.64], $\theta = 16^o$, f = 0. Finally, Figure 6.3 (c) presents the piece mask generated from the parallelogram and corresponds to this set of data: t = [0.27 0.67], $\theta = 28^o$, f = 0.



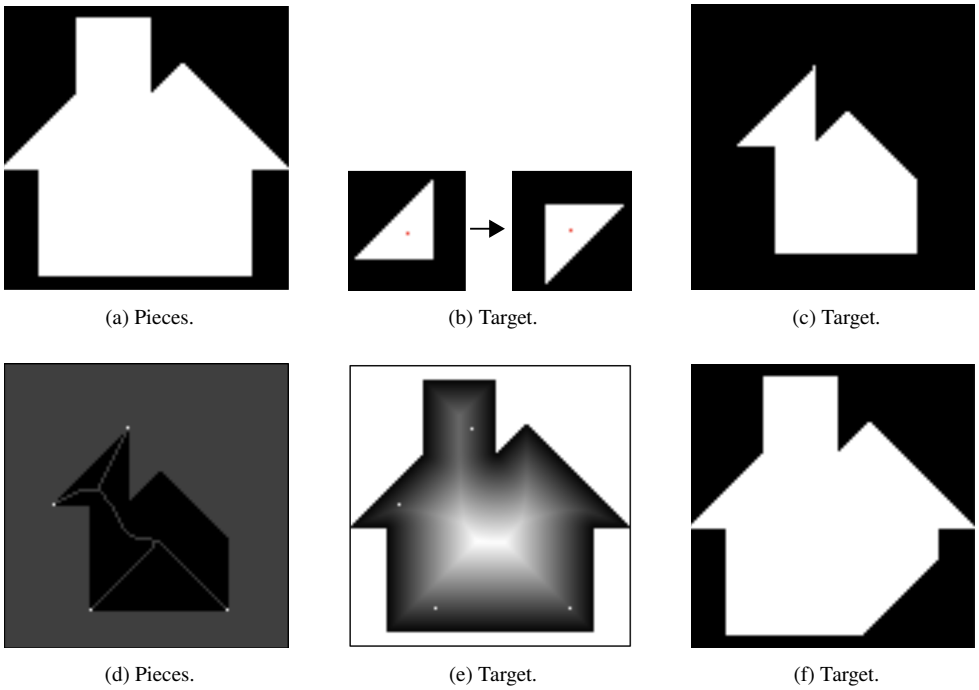
(a) (b) (c)

Figure 6.3: Examples of piece masks generated from the pieces information.

6.2 Placement Procedure

In each iteration, the heuristic selects a piece following the largest-first heuristic. It determines that the pieces should be considered from largest to smallest in area [106]. The heuristic then selects one of the configurations of the list and proceeds to search for feasible positions for that configuration. To do that the heuristic calculates the raster collision-free area by dilating the pattern mask using the reflected configuration mask of the current piece as the structuring element of the dilation. The collision-free area represents all possible translations for an item to be placed [107]. The idea of using the collision-free area to avoid overlaps while reducing the distance between polygons is inspired by works that address the cutting and packing problem[41, 42].

To reduce the number of positions to be considered, the method computes the endpoints of the morphological skeleton of the collision-free area. The skeleton of a binary image is composed of a set of points whose distance from the nearest boundary of the shape is locally maximum [108]. The endpoints of the morphological skeleton correspond to concave corners of the collision-free area, which provide a better interaction between the piece corners and the corners of the occupied areas. These endpoints represent the candidate positions $\vec{x_0}, \vec{x_1}, ..., \vec{x_k}$ for the current configuration. Figure 6.4 presents an example of this process. After obtaining the candidate positions, the idea is to calculate the cost associated with each one of them, and then pick the one that produces the lowest cost.



(a) Pieces. (b) Target. (c) Target.

(d) Pieces. (e) Target. (f) Target.

Figure 6.4: Process for obtaining the collision-free area and endpoints.

The distance transform approach helps the heuristic to decide the best candidate position by acting as a cost matrix. It maps each pixel into its shortest distance to the regions of interest [109]. The most natural

metric for computing distance in most applications is the Euclidean distance [110]. The idea is to approximate the current piece to Tangram pattern borders, as well as other Tangram pieces already placed inside the puzzle area. The heuristic calculates the distance transform mask that determines the distance between each element in the unoccupied area and the elements of the occupied area. Then, an element-wise multiplication is executed between the current piece mask and the distance transform mask to obtain the placement cost matrix $M(\overrightarrow{x_p})$ that represents the placement cost associated with an arbitrary candidate position $\overrightarrow{x_p}$, where $p \in \{0, 1, ..., k\}$. The cost of each candidate position is calculated according to Equation 6.1:

$$c(\overrightarrow{x_p}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \left(M(\overrightarrow{x_p}) \right)_{ij}, \tag{6.1}$$

where c is the cost function, $\overrightarrow{x_p}$ is an array representing a candidate position for the current piece, $M(\overrightarrow{x_p})$ is the placement cost matrix that represents the placement cost associated with $\overrightarrow{x}$, n and m are the number of rows and columns of $M(\overrightarrow{x_p})$. The heuristic calculates the costs associated with all candidate positions $\overrightarrow{x_0}, \overrightarrow{x_1}, ..., \overrightarrow{x_k}$ and picks the candidate position that generates the lowest cost.

After placing the piece in the position that generates the lowest cost, it checks if there are remaining pieces to be placed. If there are no remaining pieces, it finishes its execution and returns the final solution with all pieces correctly placed inside the puzzle area. Otherwise, it proceeds to the validation process.

If the heuristic gets to the point in which none of the possible configurations produced a feasible placement for the current piece, it verifies if there is at least one piece placed inside the puzzle area. In this case, the method backtracks to the previously placed piece. It resets the list of possible configurations of the current piece, removes the previous piece considering the largest-first heuristic, and attempts to find a new placement for that piece considering the next possible configuration. Otherwise, if there is not any piece left in the puzzle area, the method finishes its execution with none of the pieces placed inside the puzzle area, indicating that it was not possible to find a solution for the given Tangram pattern.

6.3 Validation Process

After positioning the current configuration and updating the pattern mask, the heuristic performs a validation procedure. This validation is used in the proposed method to determine whether an intermediate solution is feasible, taking into consideration the following iterations of the proposed method following the largest-first heuristic. It determines that the following conditions have to be satisfied: (1) the largest empty region must be greater than the largest piece regarding the number of pixels and (2) the maximum value of the distance transform considering the largest piece must be shorter than the maximum value of the distance transform considering the empty regions. Through the validation procedure, it is possible to determine early whether an intermediate solution can lead the heuristic to a final solution for the desired Tangram pattern. Figure 6.5 shows an example that would not pass the validation procedure, where it is possible to notice that the medium triangle cannot be placed in the empty regions.

If the placement validation process considers that the current piece placement does not prevent the placement of the subsequent pieces, then the method proceeds to the placement procedure of the following piece determined by the largest-first heuristic. Otherwise, the method removes the current piece from the puzzle

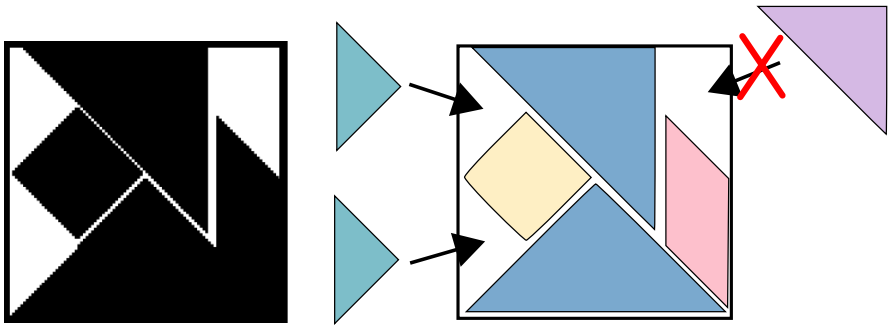


Figure 6.5: Scenario that fails the validation procedure.

area and returns to the step in which it attempts to get the next possible configuration of the current piece.

6.4 Proof of Concept with Limited Data

This section presents a toy experiment that verifies if the proposed heuristic method can solve Tangram puzzles. The model takes a 256×256 and tries to assemble the pieces to form the desired pattern. In the input image, the puzzle area is presented in white and the background is presented in black. In this toy experiment, the limited dataset comprises 30 samples collected from the literature. This investigation did not use the random Tangram generator by Köpp [100], thus it serves to check if the heuristic method can solve at least the puzzles present in the literature. A time limit of 360s is defined for the heuristic to assemble each puzzle, otherwise, it is considered a timeout. The scale of the pieces is reduced by 5% to the scale of the input target pattern to avoid overlaps. Also, rotations are implemented with 1-degree increments. Given that Tangram is a visual puzzle, discerning a 1-degree difference in rotation for a specific piece is typically unnoticeable to a human player. Figure 6.6 presents some solutions obtained in this study, where the pieces are assigned the same color for better visualization of the sections between them.

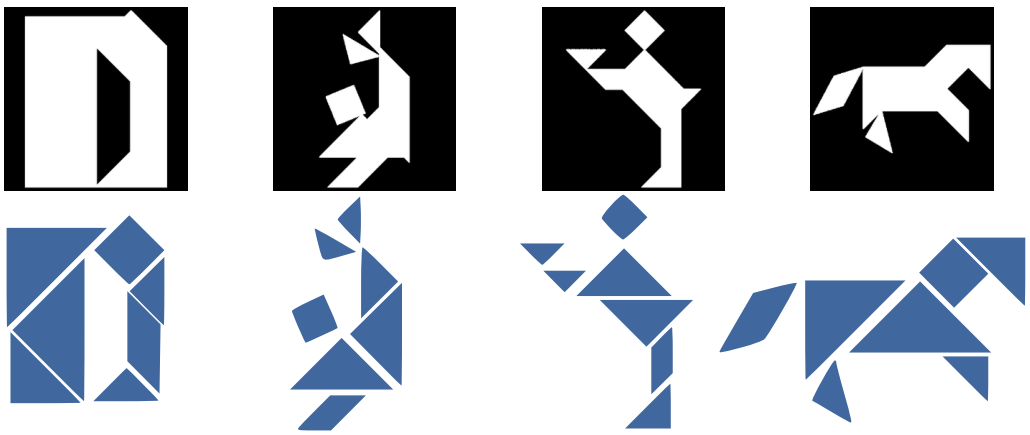


Figure 6.6: Solutions considering limited data.

The heuristic was able to assemble 46.67% samples in an average time of 51.042, generating solutions that are visually accurate and align well with the input pattern. Even though the number of solved Tangram puzzles is considerably limited, the results show that the heuristic method has proven efficient in assembling at least one sample for each type of complex Tangram puzzles. It has also proven efficient in assembling puzzles with different scales, being able to infer the size of the pieces from the input pattern. This study is important to show that analytical approaches can be used to assemble complex Tangram puzzles, although they may be sensitive to the increase in level of complexity, and take a long time if they consider each piece configuration one by one. It is possible to verify that the heuristic method depends heavily on the precise representation of the input Tangram pattern. If its representation is inaccurate and presents small imperfections along the edges of the Tangram pattern, the process for obtaining the collision-free area and endpoints is compromised. Therefore, it is possible that the image processing techniques employed to obtain the randomly generated Tangram puzzles may compromise the performance of the heuristic method.

Chapter 7

Assessment of Deep Learning Architectures

This chapter presents an investigation into the application of four deep learning architectures in solving Tangram puzzles: CAE, VAE, U-Net, and GAN. The architectural choices are 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, and GAN offers generative capabilities. The CAE and VAE are inspired by the implementation of Minhas & Zelek [111] for anomaly detection, while U-Net is inspired by the implementation of Zhou et al. [112] for image segmentation. Some adjustments are introduced to these architectures to fit the Tangram assembly task. The hypothesis is that deep learning techniques can be applied to the automatic solution of Tangram puzzles.

The experiments are divided into two stages. In the first, the performance of CAE, VAE, and U-Net architectures are evaluated. Results show that VAE outperforms the other architectures in the first stage of experiments. For the second, a GAN is implemented by using the architecture that performs best as the generator, thus forming the VAE-GAN, a GAN encompassing a VAE-based generator. Experimental results indicate that VAE-GAN, paired with the proposed loss function designed for Tangram, presents more refined solutions than the previous architectures. Also, it is necessary to 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. The findings of this investigation help to decide which architecture is the best to be used as the final model for the automatic solution of Tangram puzzles. It also supports the directions chosen in the final dataset presented in Chapter 5, and the implementation of the proposed loss function presented in Chapter 8.

7.1 Limited Dataset

The toy dataset is formed by collecting 934 samples, where 182 samples are from the literature and 752 samples are randomly generated. To generate random samples, the random Tangram generator implemented in Javascript by Köpp [100] is used without any modification. Each sample is composed of a 256×256 image depicting the puzzle and a 256×256 image representing a single feasible solution. It is important to ensure that these puzzles all have approximately the same area by standardizing the sizes of the pieces. The dataset is split into 888 samples for the training set and 46 samples for the testing set, which results in

an approximate 95:5 split ratio. Figure 5.1 serves as a representation of this toy dataset, where the first row contains the inputs and the second row contains the ground truths.

7.2 Tangram Solvers Based on Autoencoders

The input of the implemented architectures is 256×256 grayscale images. The output also follows the same pattern. Figure 7.1 presents a flowchart illustrating the workflow of the training approach. The deep learning architecture block can be substituted by any of the architectures described in Section 7.2.1. It is worth mentioning that this process is done only for the training to increase the variability of the data.

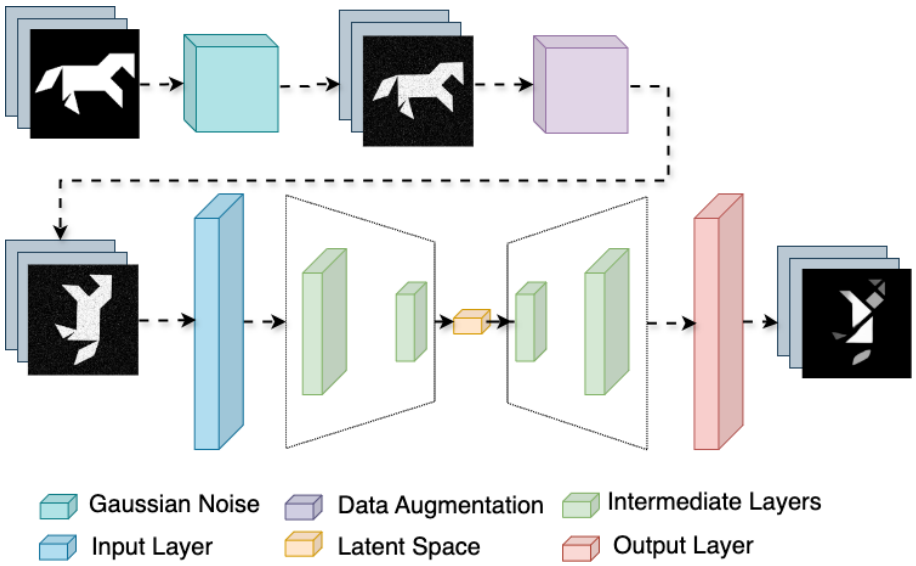


Figure 7.1: Testbed workflow for preliminary assessment.

For each iteration, an input batch is selected from the 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 [113]. Online dataset augmentation is applied on the input batch to further improve the accuracy and robustness of the model [89]. By definition, any pattern formed by the seven Tangram pieces is a Tangram puzzle, thus this idea can be used to create variations of the samples included in the dataset. The data augmentation procedure applies a random rotation to the input batch images, ensuring the rotation angle stands a multiple of 90°. It also randomly reflects the image on the vertical and horizontal axis.

7.2.1 Network Architectures

In the following, the idea behind the conception of each employed network architecture is presented followed by a brief description of its layers and parameters. The network architectures are detailed in the Appendix 10.2.

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 1980s as a neural network architecture for dimensionality reduction and feature learning. They were initially used for visual tasks, including data compression and noise reduction [70].

In the employed CAE architecture, 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 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.

VAE Architecture

The VAE architecture, introduced by Kingma & Welling [74], 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 employed VAE architecture 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 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 process 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 the latent space [76]. Subsequently, a sampling operation is applied to generate a

latent vectors that follow a Gaussian distribution, using the mean and log-variance computed by the dense layers. It 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.

U-Net Architecture

The U-Net architecture, introduced by Ronneberger et al. [77], 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 is characterized by its symmetric encoder-decoder structure, which enables it to capture intricate image features while preserving spatial information [114]. The employed 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 3,584 non-trainable.

7.2.2 Loss Functions

The following loss functions are used in the 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. For the Tangram, 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.

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 [115]. The following formula presents MSE as a dissimilarity loss function:

$$Loss_{MSE}(I_x, I_y) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (I_x(i,j) - I_y(i,j))^2, \tag{7.1}$$

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

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 [116]. The SSIM calculation is done as follows:

$$SSIM(I_x, I_y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{7.2}$$

where $SSIM(I_x, I_y)$ calculates the similarity between images I_x and I_y. Variables μ_x and μ_y represent the means of the pixel intensities in images I_x and I_y, respectively. Additionally, σ_x^2 and σ_y^2 correspond to the variances of pixel intensities in images I_x and I_y, while σ_{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 dissimilarity metrics are considered, and SSIM calculates a ratio expressing the similarity of a pair of images, it is necessary to adapt it to transform it into a loss function:

$$Loss_{SSIM}(I_x, I_y) = 1 - SSIM(I_x, I_y), \tag{7.3}$$

where $Loss_{SSIM}(I_x, I_y)$ calculates the similarity between images I_x and I_y based on the SSIM metric.

7.2.3 Evaluation Metrics

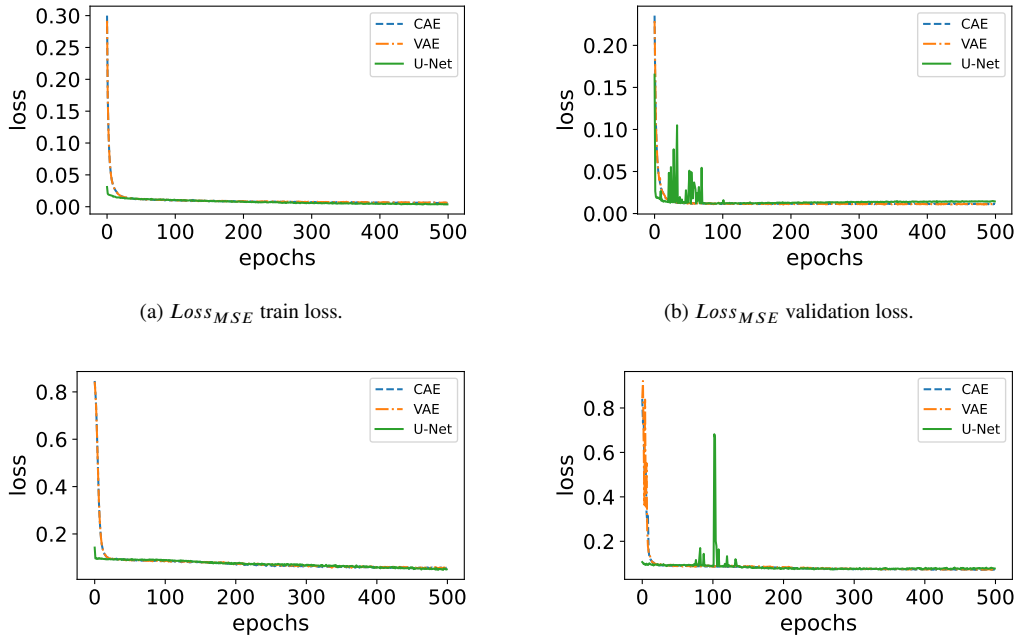
For evaluating the performance of the architectures, the following evaluation metrics are employed: (1) MSE and (2) SSIM metrics. The goal is to analyze if conventional metrics based on pixel accuracy are effective in evaluating the performance of deep learning models that aim to solve Tangram puzzles. For SSIM, the adapted calculation that transforms it into a metric of dissimilarity is used, 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. 7.1 and 7.3, respectively. The 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.

7.2.4 Experimental Results

The architectures are implemented in Python 3.9.12 using Tensorflow 2.11.0 library. Tests are executed on a Ryzen 3700x 3.6GHz 32GB of RAM with an Nvidia RTX 4090 24GB. Batch size is set to 2.

The analysis of experimental results starts by evaluating the loss curves generated throughout the training stage. Figure 7.2 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 the 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 the 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.



(a) $Loss_{MSE}$ train loss.

(b) $Loss_{MSE}$ validation loss.

(c) $Loss_{SSIM}$ train loss.

(d) $Loss_{SSIM}$ validation loss.

Figure 7.2: Loss curves for the CAE, VAE, and U-Net.

Additionally, it is necessary to inspect the output solutions of the trained models when submitted to the testing set and analyze the visual quality of the generated solutions according to conventional metrics based on pixel accuracy. Table 7.1 shows the obtained results for the executed experiments over the aforementioned evaluation metrics and Figure 7.3 presents 15 inferences of the trained CAE, VAE, and U-Net on the testing set when combined with $Loss_{MSE}$ and $Loss_{SSIM}$. The first two columns in Table 7.1 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 Figure 7.3. The employed strategy aligns with a geometric property of Tangram pieces, which tells that they can all be decomposed as a combination of the small triangular piece [27, 19]. 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 7.1: Experimental results according to evaluation metrics.

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

SSIM evaluation metric considers that VAE presented the best performance, although CAE closely matches it. However, from Figure 7.3, it is possible to 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 the experiments, images with only black and white pixels are considered. 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 [47]. 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 this claim, Figure 7.4 shows cases where the MSE and SSIM metrics fail to correctly indicating 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).

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, solutions with additional sections over those with omissions are prioritized. Inferring the correct position of the pieces from an image with extra sections is considerably less challenging than discerning the precise location of

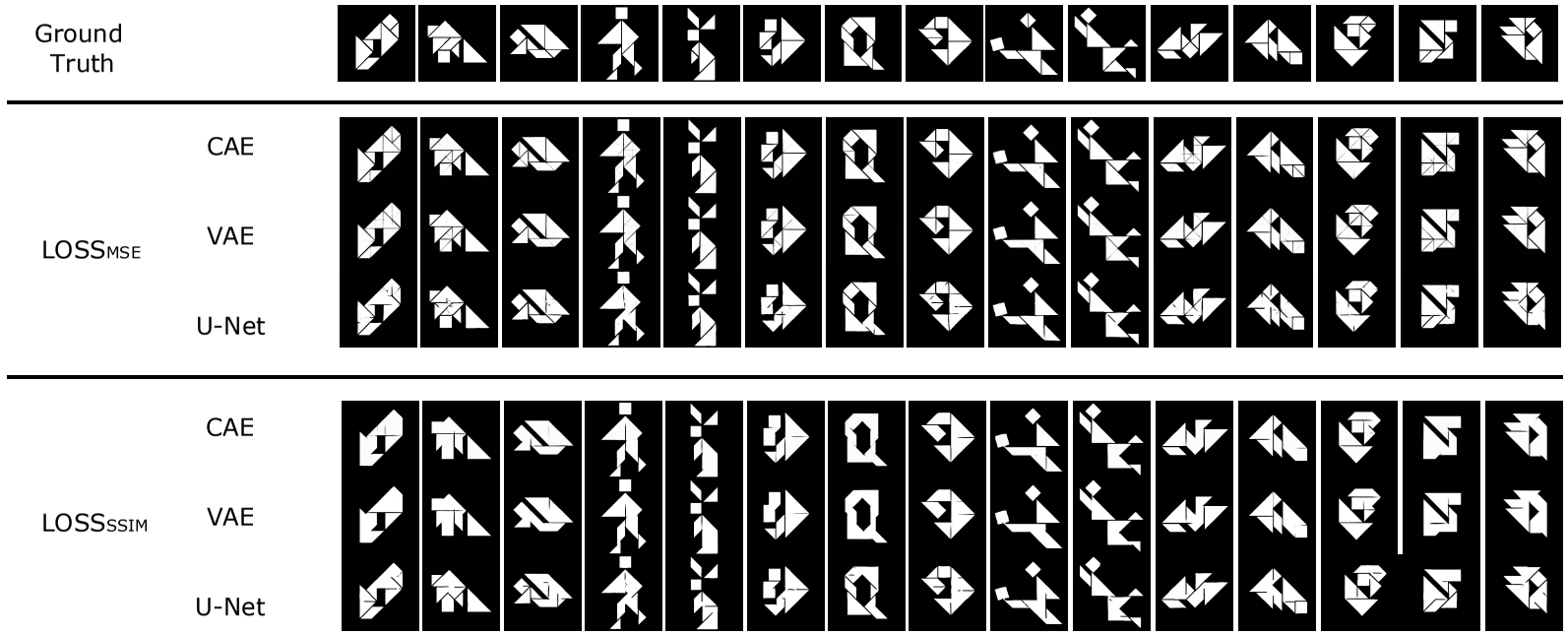
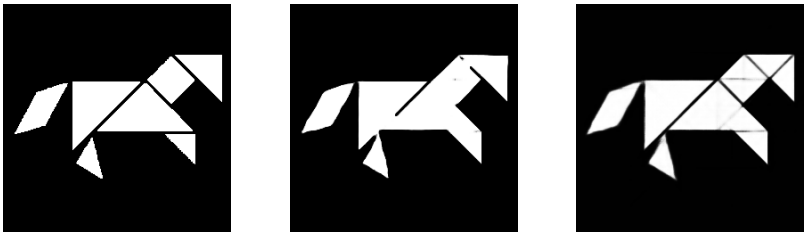


Figure 7.3: Solution images generated by CAE, VAE, and U-Net.



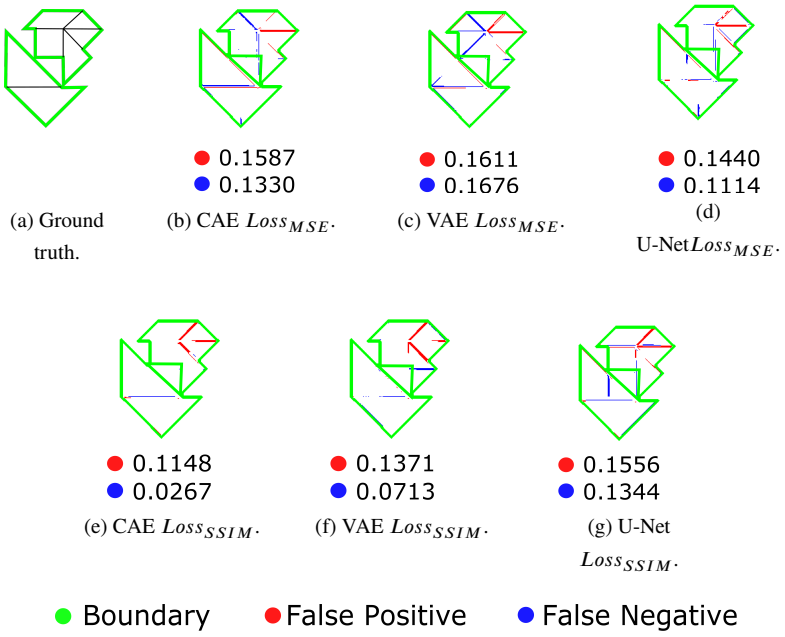
(a) Ground truth. (b) MSE = 0.0718 **SSIM = 0.0698**. (c) MSE = 0.0670 **SSIM = 0.1878**.

(d) Ground truth. (e) **MSE = 0.0399** SSIM = 0.0432. (f) **MSE = 0.1555** SSIM = 0.0274.

Figure 7.4: Cases where SSIM and MSE fail in evaluating pairs of Tangram solutions.

missing cuts. This is especially true when the player has some familiarity with the Tangram pieces. It is possible to 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 7.5 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 Figure 7.3 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, it is possible to observe that CAE, VAE, and U-Net architectures reveal the limitations of existing evaluation metrics in adequately preserving the perceivable geometric 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 the experiments, a novel loss function and evaluation metric specifically designed for the perception of geometric characteristics of Tangram solutions is proposed. The objective is to enhance the fidelity of the models in preserving the crucial geometric properties of Tangram pieces in generated solutions. Since it is not possible to rely on conventional evaluation metrics to assess the generated solutions, through a visual



(a) Ground truth. (b) CAE $Loss_{MSE}$. (c) VAE $Loss_{MSE}$. (d) U-Net $Loss_{MSE}$. (e) CAE $Loss_{SSIM}$. (f) VAE $Loss_{SSIM}$. (g) U-Net $Loss_{SSIM}$.

Figure 7.5: Visualization of false positives and false negatives in solution images.

inspection, it is necessary to implement a novel evaluation metric appropriate for the novel loss function and use VAE architecture as the generator of the GAN in the second stage of experiments. It is expected 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.

7.3 Generative Model for Refinement of Tangram Geometry

The workflow of the second part of the experiments is analogous to the flowchart presented in Figure 7.1. The input is composed of 256×256 grayscale images. The output also follows the same pattern. The deep learning architecture is substituted by the VAE-GAN architecture. Gaussian noise and online data augmentation are used to improve the robustness of the model.

7.3.1 Network Architecture

GAN is a class of generative models introduced in 2014 by Goodfellow et al. [79]. It started being used in 2017 with human faces to adopt image enhancement that produces better illustrations at high intensity [80, 117]. 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 [81]. As previously mentioned, a VAE-GAN using the aforementioned

VAE architecture combined with the discriminator is implemented, which is detailed in the following paragraphs. A detailed description of the implementation of these architectures is available on Appendix 10.2. The 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 [81].

The GAN discriminator architecture consists of a total of 8,276,801 parameters, with 8,273,217 parameters being trainable and 3,584 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. The 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 boundaries and critical aspects 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 is a ground truth.

7.3.2 Weighted Mean Absolute Error Loss Function

Weighted Mean Absolute Error (WMAE) is a variation of 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 Mean Absolute Error [118, 119]. It also mixes these inspirations with Weighted Mean Squared Error [120]. 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 the 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 serves as an inspiration to implement $Loss_{WMAE}$, 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 I_x and I_y of the same dimensions are normalized to a range between 0 and 1. Their normalized counterparts are defined as I_x^{norm} and I_y^{norm} respectively. Thus, There are conditions that $0 \leq I_x^{norm}(i,j) \leq 1$ and $0 \leq I_y^{norm}(i,j) \leq 1$ for every coordinate (i,j) present in I_x^{norm} and I_y^{norm}. Then, the absolute difference between them, denoted as Δ, is calculated element-wise:

$$\Delta = I_y^{norm} - I_x^{norm}. \tag{7.4}$$

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

$$W = \begin{cases} \boldsymbol{I}_y^{norm} \cdot \mathrm{c} + 1, & \text{if } \Delta > 0 \\ 1, & \text{if } \Delta < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{7.5}$$

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

$$L = |W \cdot \Delta|. \tag{7.6}$$

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

$$Loss_{WMAE}(I_x, I_y) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} L(i,j). \tag{7.7}$$

In summary, $Loss_{WMAE}$ 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 Figure 7.5.

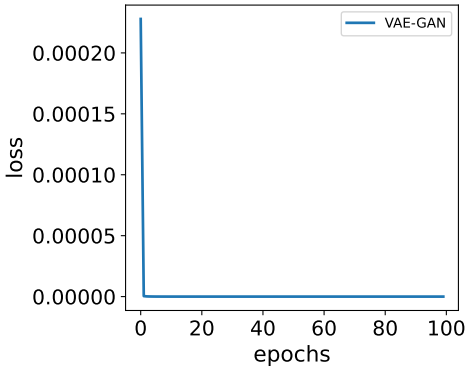
7.3.3 Weighted Mean Absolute Error Evaluation Metric

The $Loss_{WMAE}$ is extended by re-purposing it as an evaluation metric. By employing the same loss function for assessment, it is possible to establish a unified and consistent approach to both training and evaluating the 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, the WMAE evaluation metric follows Equation 7.7. The hypothesis is that the WMAE metric is better aligned with the task of evaluating Tangram solutions than traditional metrics, such as MSE and SSIM.

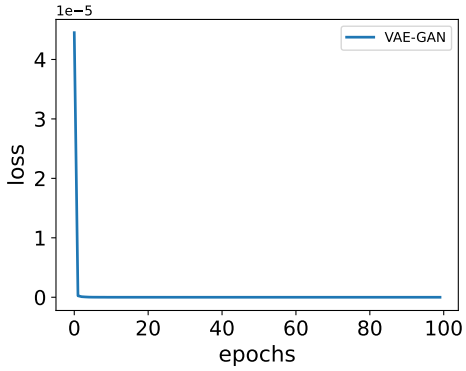
7.3.4 Experimental Results

The VAE-GAN architecture is also implemented by the same environment described in Section 7.2.4. Experiments are performed by using the VAE-GAN combined with $Loss_{WMAE}$ to show the efficiency of the architecture. Figure 7.6 depicts the training process of VAE-GAN. It can be an abrupt decrease in training and validation loss curves in the early stages of training, while for the generator loss curve, the decrease is more gradual. Comparing Figs. 7.2 and 7.6, it is possible to notice that VAE-GAN converges faster than the previous architectures and that the VAE-based generator is learning from the judgment of the discriminator. It is also possible to notice that, although the VAE-GAN is using $Loss_{WMAE}$, the fluctuations in the validation curve that are shown in the first part of the experiments are not perceived. This shows that the discriminator regularizes the training process and leads to smoother loss curves for the VAE-based generator. In other words, it indicates that the incorporation of a discriminator is beneficial because the competitive

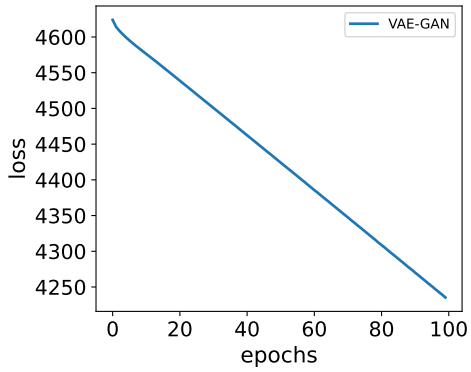
nature of the GAN stabilizes the training dynamics of the VAE-based generator, making it less susceptible to fluctuations compared to a standalone model.



(a) Train loss.



(b) Validation loss.



(c) Generator loss.

Figure 7.6: Loss curves for VAE-GAN.

To support the claim that $Loss_{WMAE}$ is better suited for solving Tangram puzzles than $Loss_{MSE}$ and $Loss_{SSIM}$, the experiments are extended to architectures CAE, VAE, and U-Net. This extension enables the validation of the effectiveness of $Loss_{WMAE}$ across a diverse range of architectures. Thus, Figure 7.7 presents 15 inferences of the trained CAE, VAE, and U-Net on the testing set when combined with $Loss_{WMAE}$. It is also necessary to compare these solutions with the ones generated by CAE with $LossMSE$ and VAE with $Loss_{SSIM}$. This pair of combinations are the ones that MSE and SSIM metrics judged presenting the best performance in Section 7.2. It is possible to observe that the solutions generated using $Loss_{WMAE}$ present better-defined sections when compared to the solutions generated by CAE with $LossMSE$ and VAE with $Loss_{SSIM}$. 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.

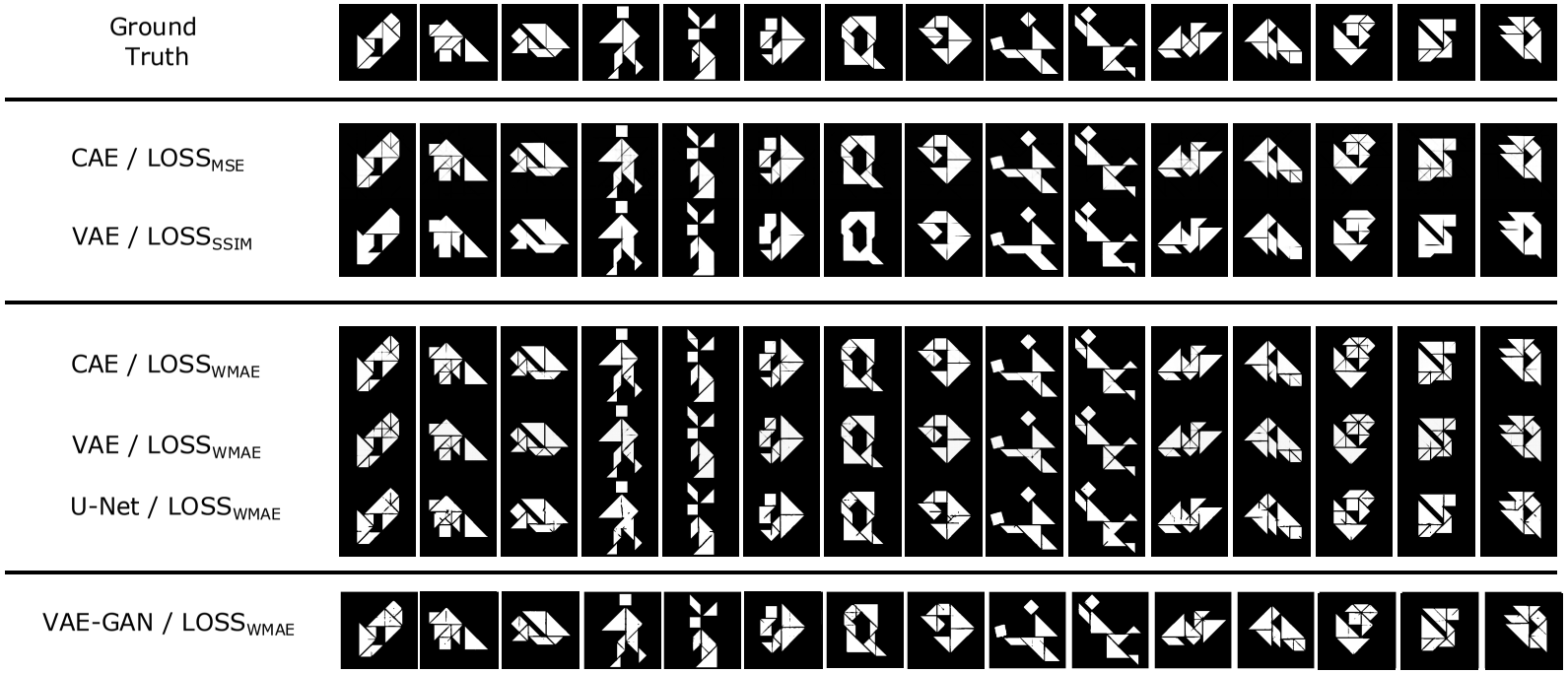
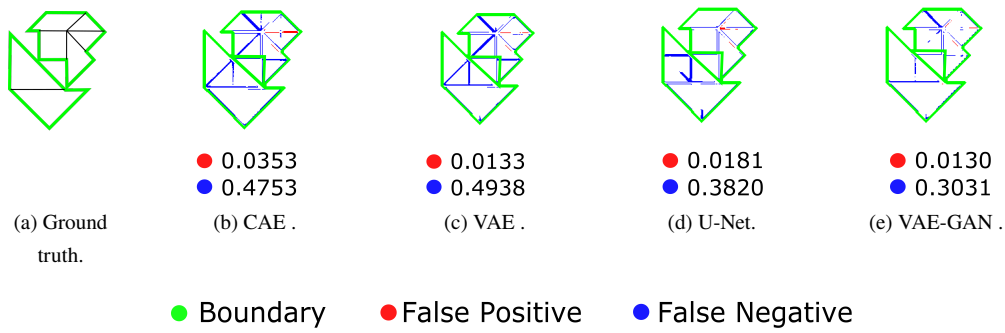


Figure 7.7: Solution images generated by CAE, VAE, U-Net, and VAE-GAN with $Loss_{WMAE}$.

It is noticeable that $Loss_{WMAE}$ presents better qualitative results than the other loss functions. It tends to generate clearer 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 7.8 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 Figure 7.5 reveals that the ratio of false positives reduces expressively with $Loss_{WMAE}$. For the presented scenarios, the ratio of false positives for VAE with $Loss_{WMAE}$ is close to 1/10 of the ratios obtained for $Loss_{MSE}$ and $Loss_{SSIM}$. These observations align with the hypothesis regarding the effects of VAE-GAN and $Loss_{WMAE}$ on the generated solutions.



(a) Ground truth. (b) CAE . (c) VAE . (d) U-Net. (e) VAE-GAN .

Figure 7.8: Visualization of false positives and false negatives in solution images using $Loss_{WMAE}$.

Table 7.2 shows the results for performed experiments over the proposed WMAE evaluation metric. The $Loss_{WMAE}$ is employed for all the 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 it is possible to observe in Figure 7.7, 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 solu-

Table 7.2: 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

tions. Although the average inference time for VAE-GAN is considerably higher than the other standalone architectures, this is not a prohibitive limitation, given the notable improvements in solution quality.

It is evident that $Loss_{WMAE}$ 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 of 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, this metric surpasses the limitations associated with conventional methods, thereby providing a more precise evaluation of Tangram solutions.

Finally, an experiment to address the influence of the coefficient c on the $Loss_{WMAE}$ is conducted. The idea is to vary the value of this coefficient in a large range of values, in doing so it is possible to evaluate its impact on the VAE-GAN performance. Table 7.3 shows the results of this experiment over the proposed WMAE metric. The first column presents the value used for c in $Loss_{WMAE}$. The second column presents the average values for WMAE evaluation metric. This experiment does not consider the average inference time because there are no expressive variations regarding this aspect.

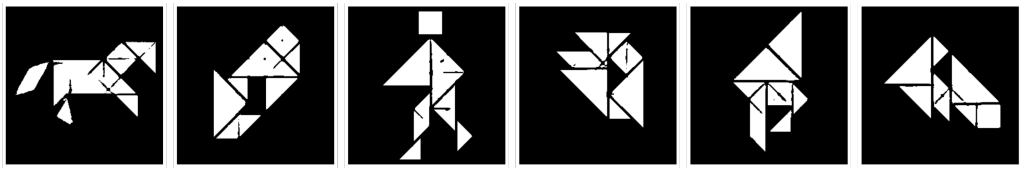
The experiment shows that, although $c = 50$ presents a slightly better performance compared to the other considered values, the performance does not change expressively. The same behavior is expected for the WMAE evaluation metric eliminating the necessity of additional tests. Figure 7.9 presents examples of solutions generated by VAE-GAN for $c = 50$ and $c = 5$ in $Loss_{WMAE}$ aiming at visual comparison.

It is possible to notice that, as the values in Table 7.3 suggest there is not considerable visual difference considering this range for c in $Loss_{WMAE}$.

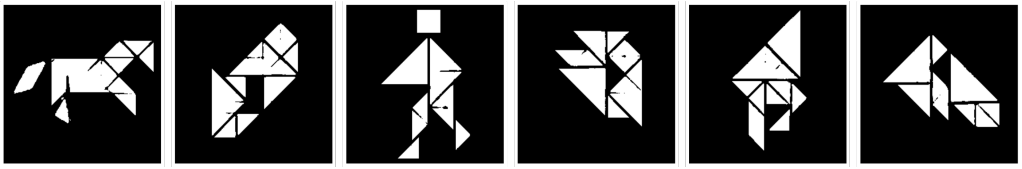
Figure 7.10 presents the training process of VAE-GAN considering the variation of coefficient c in $Loss_{WMAE}$ using the range presented in Table 7.3. The training, validation, and generator loss curves are presented in distinct colors and makers for each value of c. Aiming for a better visualization, each graph is on a different scale. The training loss curves are plotted on a symmetrical logarithmic scale, a combination of linear and logarithmic scales, with a threshold value set to 1×10^{-5}. This means that for data points close to zero, the scale behaves linearly, providing a clear representation of small variations. As the values increase beyond the threshold, the scale transitions to a logarithmic scale, allowing the visualization of larger

Table 7.3: Experimental results varying coefficient c in $Loss_{WMAE}$.

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



(a) Solutions for c = 50.

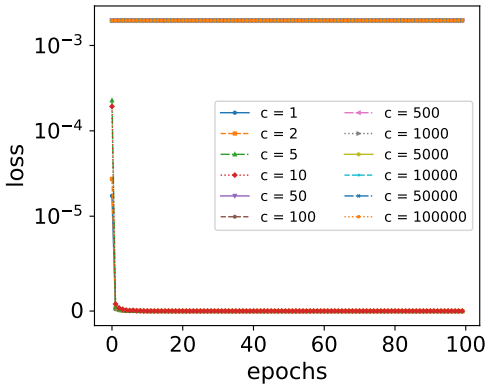


(b) Solutions for c= 5.

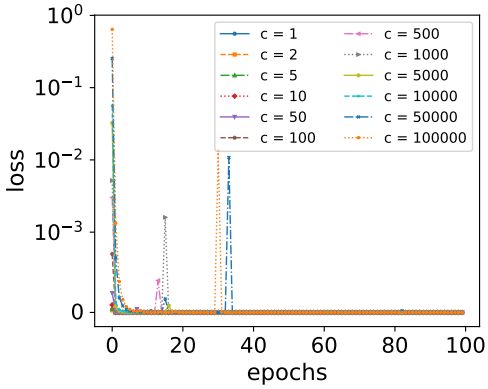
Figure 7.9: Solution images generated by VAE-GAN paired with $Loss_{WMAE}$ with varied coefficients.

magnitudes with improved clarity. Validation loss curves also use a symmetrical logarithmic scale but with a threshold value set to 1×10^{-3}. Finally, the generator loss curves are plotted on a regular logarithmic scale.

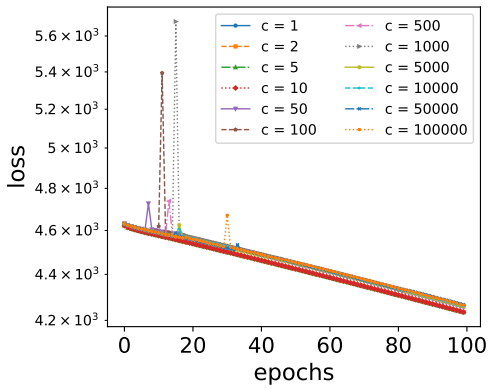
From the presented loss curves, it is possible to observe that certain values of c promote more stability in the training process. When higher values are assigned to c, sudden peaks are observed in the generator and validation loss curves that may be associated with instability in the training process, possibly due to issues with gradients during backpropagation. In the training loss curves, it is possible to notice that the training schedules that considered higher values for c faced convergence issues, thus indicating that the value assigned to c has a considerable impact on the learning dynamics of the model. Therefore, it is possible to notice better stability in training is achieved when $c \leq 10$ considering the presented dataset and the task of automatically solving the Tangram puzzle. Moreover, the generator loss curves suggest that the values



(a) Train loss.



(b) Validation loss.



(c) Generator loss.

Figure 7.10: Loss curves for VAE-GAN under different values for parameter c.

for $c = 5$ and $c = 10$ achieve a lower loss value by the end of training, suggesting that these values are more effective for the given task and dataset by facilitating a faster or more effective learning process. It is also possible to conclude that the values for c with the better-performing curve lead to a more robust model that is less prone to overfitting on the training data. In case $Loss_{WMAE}$ is applied in other contexts, a hyperparameter sensitivity analysis may indicate the range of values for c that lead to better convergence for a specific task.

7.4 Outcomes

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 [27]. This shows that the models were able to learn the geometric properties of Tangram pieces even when these properties were not

directly informed to them. Although conventional evaluation metrics were proven not inadequate 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. The preference for solutions with extra sections aligns with the principle that they are more insightful in the context of the interpretation of Tangram puzzles. In line with this perspective, $Loss_{WMAE}$ is proposed, which is a loss function that gives higher importance to pixels representing sections between pieces rather than pixels covered by pieces. The usage of this loss function is extended to 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 $Loss_{WMAE}$ 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. Experimental results indicate that VAE-GAN architecture is the most robust choice among the considered architectures, showing stable convergence and strong generalization skills. After the experiments, the WMAE metric is employed to evaluate the performance of the architectures, demonstrating its ability to overcome limitations associated with conventional metrics and, consequently, deliver a more accurate assessment of Tangram solutions. Therefore, it is possible to conclude that deep learning techniques are applicable to the task of solving Tangram puzzles.

Chapter 8

Deep Learning Approach

As demonstrated in Chapter 7 and sustained by works in the literature [96, 33], generative models have proven efficient in the task of automatically solving Tangram puzzles. In the following, the proposed GAN architecture and its components are presented in detail. The present chapter also outlines the proposed loss function based on Hu Moments, and a derived evaluation metric for assessing the accuracy of the generated solutions.

8.1 Architecture Components

As stated before, a GAN consists of two intricately linked components: a generator and a discriminator. The input of the model is an image that represents a Tangram pattern in black and white pixels. White pixels represent the puzzle region, while black pixels represent areas outside of the puzzle region. The generator is tasked with the unique ability to synthesize grayscale images that closely resemble the intricate patterns and complexities found in authentic Tangram solutions, while the discriminator attempts to discern the authenticity of these generated images. The proposed GAN model, named TANGAN, is presented in Figure 8.1.

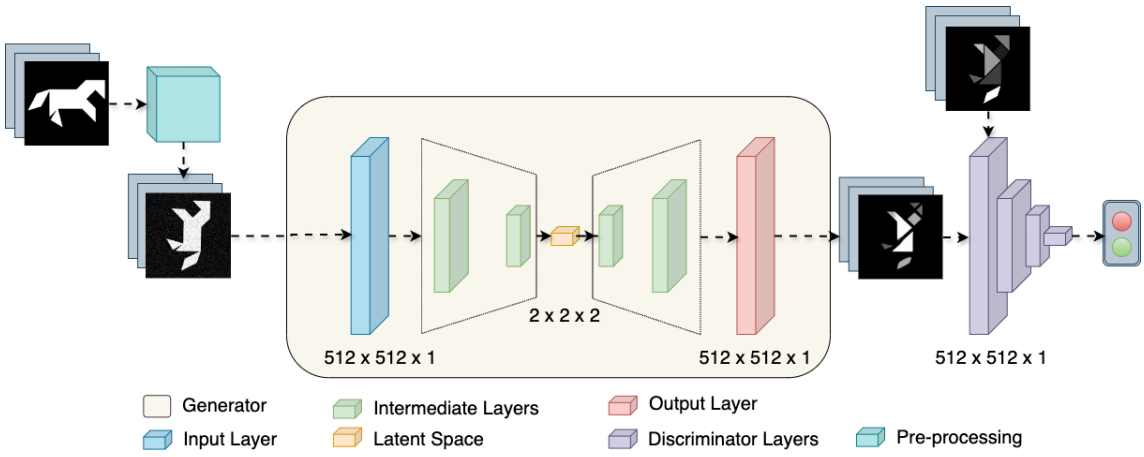


Figure 8.1: Workflow for TANGAN architecture.

The pre-processing block shows a process similar to the one described in Figure 7.1, where Gaussian noise is applied on the input batch images as a form of regularization, and online dataset augmentation is applied on the input batch to further improve generalization of the model. A recurrent problem when using GANs is the amount of data. Studies show that with small datasets, the discriminator tends to overfit the training examples [34]. As a consequence, the feedback to the generator becomes meaningless and training starts to diverge. For this reason, data augmentation is applied on the batches that are input into TANGAN, similarly to the process described in Chapter 7. As mentioned before, by definition, any pattern formed by the Tangram pieces can be understood as a Tangram puzzle. The proposed data augmentation process takes advantage of this concept to create variations of the samples in the batch. After selecting a batch of samples as input, the model performs a random rotation to the image, where this rotation is a multiple of 90º. It also randomly flips the image in the vertical and horizontal axis. Finally, it employs random image scaling, reducing the scale within a limit of 10% of the original size to avoid any cropping.

The following steps are similar to the dynamic illustrated in Figure 3.4. The generator and discriminator are trained competitively. The generator aims to produce data that is realistic enough to fool the discriminator, while the discriminator strives to become better at telling real from fake. Finally, backpropagation is used as part of the training process to update the parameters of both the generator and the discriminator networks. The architecture components are presented in detail in Appendix 10.2 where it is possible to see their summary and number of parameters described in detail.

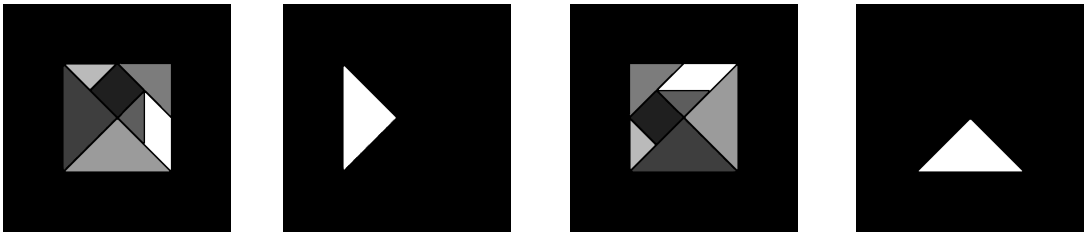
The VAE-based generator is a deep convolutional neural network that starts with a 512x512 grayscale input and progressively applies multiple Conv2D layers with 48 filters each, interspersed with batch normalization, LeakyReLU activations, and dropout layers to downsample the input, culminating in a dense layer and a sampling layer to create a latent space representation. The model then up-samples using Conv2DTranspose layers, concatenating intermediate layers for skip connections, each followed by batch normalization, LeakyReLU activations, and dropout layers, eventually producing a 512x512 output with a single channel. The model consists of 3,804,969 parameters in total, of which 3,803,335 are trainable.

The employed discriminator is a convolutional neural network designed for image processing tasks, consisting of two 512x512 grayscale input layers concatenated into a single 512x512x2 input, followed by a series of convolutional layers with increasing filter sizes and LeakyReLU activations, interspersed with batch normalization and dropout layers to enhance training stability and prevent overfitting. The network concludes with a global average pooling layer and a dense layer producing a single output. The model has a total of 11,153,857 parameters, of which 11,150,017 are trainable.

8.2 Hu Moments Loss Function

Besides the architecture components, another important aspect that dictates the learning process of the model is the loss function. A crucial problem that emerges when assessing generated Tangram solutions related to its one-to-many correspondence, since a single pattern may have multiple solutions. According to what is observed in the real world, a single Tangram pattern should be associated with a set of ground truth solutions. However, it is uncommon to pair a single input with multiple ground truth references in generative models, as the standard practice is to have a one-to-one correspondence, ensuring a straightforward and unambiguous

learning process [121]. This occurs because the association of multiple correct answers for a given input can compromise the adjustment of machine learning methods, and convergence is thus not guaranteed [122]. This aspect makes the application of generative models to dissection puzzles considerably challenging. For example, Figure 8.2 presents a scenario where two different arrangements of the Tangram pieces can form the same square pattern. Imagine that solution image S_y is the ground truth being considered by the model, while solution image S_x is the generated one. Designing a loss function that can effectively recognize the equivalence of solution images S_x and S_y for the square pattern is difficult. This is because performing a pixel value mask operation and comparing the position of each piece is not sufficient in this case, as it is possible to see when one considers the positioning of one of the big triangle pieces.



(a) Solution S_x (b) Big triangle in S_x (c) Solution S_y (d) Big triangle in S_y

Figure 8.2: Difficulty in treating multiple solutions for Tangram.

Considering the presented issue, a better practice would be to design a loss function that can extract geometric features from each piece presented in the ground truth image and attempt to find the same geometric features in the generated image. For this reason, this dissertation proposes a novel loss function that is based on Hu Moments and is capable of telling the discrepancy between two images by taking into consideration geometric features [103].

First, it is necessary to verify the correspondence between the ground truth pattern and the generated one. Notice that the Hu Moments are not applicable in this task because it aims at comparing the patterns precisely, and the invariant nature of Hu Moments can potentially affect that comparison. Thus, analogously to the $Loss_{WMAE}$ presented in Chapter 7, a variation of the WMAE metric is included as a component of the proposed loss function. This component is nominated as $Loss_{CWMAE}$ and is crucial for driving the model to infer the correct tones for the pieces, as well as for maintaining the pieces within the areas of the puzzle area.

The solution images I_x and I_y are given to the loss function as the image representation of solutions S_x and S_y respectively. Their normalized counterparts are defined as I_x^{norm} and I_y^{norm} respectively. Thus, There are conditions that $0 \leq I_x^{norm}(i,j) \leq 1$ and $0 \leq I_y^{norm}(i,j) \leq 1$ for every coordinate (i,j) present in I_x^{norm} and I_y^{norm}. Then, the absolute difference between them, denoted as Δ, is calculated element-wise following the Euclidean norm:

$$\Delta(i,j) = \| I_y^{norm}(i,j) - I_x^{norm}(i,j) \|_2. \tag{8.1}$$

Then, it is possible to calculate the conditional weights. The only big difference from this to the previously proposed $Loss_{WMAE}$ is that this weighting function assumes the image has grayscale tones that vary

in the [0, 1] interval. Therefore, elements of Δ are analyzed, and the penalty term P given by:

$$P(i,j) = \begin{cases} c \cdot \Delta, & \text{if } I_y^{norm}(i,j) = 0 \\ \Delta, & \text{otherwise,} \end{cases} \tag{8.2}$$

where c represents a penalty factor, and the condition $I_y^{norm} = 0$ covers cases where a particular pixel in I_y^{norm} is assigned black. The function effectively penalizes instances where the generator incorrectly predicts a pixel that should be black. Its primary objective is to prioritize pixels delineating sections between pieces and discerning the background from areas covered by pieces. Thus, the final value of the loss function is defined as a sum of all the penalties:

$$Loss_{CWMAE}(S_x, S_y) = \frac{1}{m \cdot n} \sum_{i=1}^{n} \sum_{j=1}^{m} P(i,j) \tag{8.3}$$

After this point, it is possible to proceed to the next step of the loss function calculation, which concerns the assessment of the geometry of the pieces. The concept of Hu Moments was previously mentioned in the present dissertation in Chapter 5. However, it is pertinent to revisit and delve deeper into this concept, since the loss function stands as a main contribution in the present dissertation. To compute Hu moments, it is necessary to first calculate the normalized central moments η_{pq} using the following formula:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}}, \tag{8.4}$$

where μ_{pq} is the central moment of order $p+q$ and μ_{00} is the zeroth-order central moment.

Notice that, in the context of normalized moments, normalization refers to scaling the moments by a measure of dispersion, typically the standard deviation. The purpose is to make the moments comparable across datasets or distributions that may have different scales [123]. Central and normalized moments are invariant to scaling and translation already. The challenge is to find rotational invariants. These normalized central moments are then used to compute the Hu moments as linear combinations of the central moments, each representing distinct geometric properties and relationships of the object.

$$
\begin{aligned}
\mathrm{Hu}_1 &= \frac{\eta_{20} + \eta_{02}}{\eta_{00}^2}, \\
\mathrm{Hu}_2 &= \frac{(\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2}{\eta_{00}^4}, \\
\mathrm{Hu}_3 &= \frac{(\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2}{\eta_{00}^5}, \\
\mathrm{Hu}_4 &= \frac{(\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2}{\eta_{00}^5}, \\
\mathrm{Hu}_5 &= \frac{(\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]}{\eta_{00}^{10}}, \\
\mathrm{Hu}_6 &= \frac{(20\eta_{20} - 20\eta_{02})^2 + 6(30\eta_{11})^2}{\eta_{00}^{10}}, \\
\mathrm{Hu}_7 &= \frac{\eta_{21} \cdot (\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{03} \cdot (\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]}{\eta_{00}^{10}},
\end{aligned}
\tag{8.5}
$$

where Hu_i represents the i-th Hu moment, η_{pq} is the normalized central moment of order $p + q$, and η_{00} is the zeroth-order normalized central moment.

By using the list of tones that were assigned to the pieces, it performs a pixel value mask operation on the ground truth and the solution considering each tone. Therefore, for each Tangram piece, it picks the grayscale tone assigned to it and performs a pixel value mask operation on both solution and ground truth images. The function obtains a pair of binary masks, where True values represent the pixels that have that specific tone, while False values represent pixels dissimilar to the specific tone. It proceeds to calculate the ratio concerning the pixels present in the obtained piece masks. This is necessary to guarantee that the pieces represented in the generated solution image S_x and in the ground truth image S_y have the same scale. It is important to mention that the Hu Moments are invariant to scale, as a consequence objects with different scale and similar geometric features may have similar moments. Therefore the ratio is calculated as:

$$
r(P_{xi}, P_{yi}) = \frac{|T(P_{xi}) - T(P_{yi})|}{|T(P_{xi}) - T(P_{yi})| + 1}, \tag{8.6}
$$

where $T(I)$ counts the number of True values in binary mask I, and piece masks P_{xi} and P_{yi} are related to piece i in S_x and S_y, respectively. The denominator serves as a smooth approximation that makes the ratio value stay within the [0, 1] interval.

It then proceeds to calculate the Hu Moments of the pair of binary masks and calculate the average difference between the moments. This process is illustrated in Figure 8.2 for solutions A and B considering one of the big triangular pieces, where the obtained masks should have approximately the same Hu Moments. It is important to mention that the proposed loss function uses only the Hu Moments Hu_1, Hu_2, Hu_3, Hu_4, Hu_5, and Hu_6. The moment Hu_7 stands out as particularly sensitive to the reflection transformation [124]. When an image undergoes a reflection transformation, such as being flipped, Hu_7 undergoes a noticeable change, thereby providing a distinctive characteristic that aids in distinguishing between the object and its

mirrored counterparts [125]. This aspect contributes to the utility of Hu_7 in scenarios where recognizing mirrored or flipped versions of objects is crucial for accurate image analysis and pattern recognition. Still, it may introduce unwanted variability for some other scenarios, including recognizing the geometry of Tangram pieces. Therefore the function that measures the difference in the geometry of pieces is expressed as:

$$H(S_x, S_y) = \sum_{i=1}^{7} \sum_{j=1}^{6} \frac{|Hu_j(P_{xi}) - Hu_j(P_{yi})|}{|Hu_j(P_{xi}) - Hu_j(P_{yi})| + 1} \tag{8.7}$$

where $Hu_j(M)$ corresponds to the j-th Hu Moment in binary mask M. Piece masks P_{xi} and P_{yi} are related to piece i in S_x and S_y, respectively. The order of the pieces does not affect the value of $Loss_{Pieces}(S_x, S_y)$. The divider of $Loss_{Pieces}(S_x, S_y)$ is the number of pieces times the number of considered Hu Moments, which is important to guarantee that $Loss_{Pieces}(S_x, S_y)$ has a balanced impact concerning $Loss_{CWMAE}(S_x, S_y)$ when these are combined in the final proposed loss function. Notice that, analogously to Equation 8.6, the denominator serves as a smooth approximation that makes the ratio value stay within the [0, 1] interval

Next, it is necessary to divide the calculated geometric difference between the number of considered moments and the number of pieces:

$$Loss_{Hu}(S_x, S_y) = \frac{H(S_x, S_y)}{6 \cdot 7} \tag{8.8}$$

At this point, both components of the loss function, $Loss_{CWMAE}$ and $Loss_{Hu}$, are already calculated. Thus, the previously calculated components are combined and the final proposed loss function is given by:

$$Loss_{WMAE-Hu}(S_x, S_y) = \frac{Loss_{CWMAE}(S_x, S_y) + Loss_{Hu}(S_x, S_y)}{2}, \tag{8.9}$$

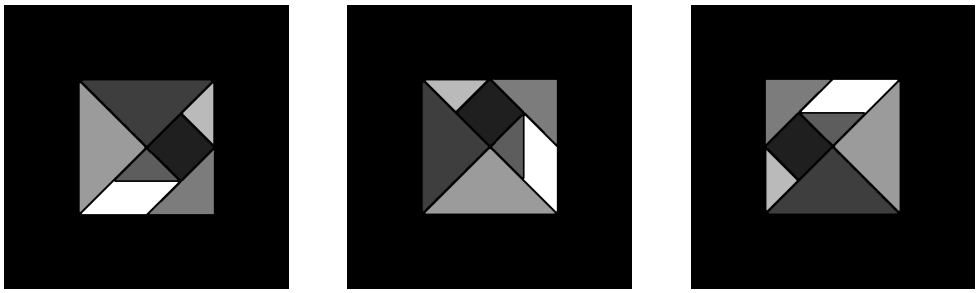
where $Loss_{WMAE-Hu}(S_x, S_y)$ evaluates the dissimilarity between generated solution image S_x in relation to the ground truth image S_y. Also, $Loss_{CWMAE}(S_x, S_y)$ aims at assessing the similarity between the patterns depicted in S_x and S_y. Finally, $Loss_{Pieces}(S_x, S_y)$ calculates the level of geometric dissimilarity between the pieces depicted in S_x and S_y. It is worth noticing that both loss components $Loss_{CWMAE}$ and $Loss_{Pieces}$ both fall in the [0, 1] interval, thus summing them and dividing by 2 guarantees that the value of $Loss_{MAE-Hu}$ also stays in the same interval.

The usage of Hu Moments in the loss function makes it possible to check if the geometry of the pieces is correct, instead of checking their specific position or configuration. The idea is to bring the loss function as close as possible to the perception of a human player, where they would be able to recognize the equivalency of solutions even when the generated solution was different from the ground truth. This is beneficial to not only Tangram but also other dissection puzzles because it is not sensitive to multiple solutions. The criteria for a solution to be considered close to the ground truth are that it has to represent the same pattern, and is composed of the same geometric pieces. The proposed loss function offers a novel perspective on understanding image content, not only relying on pixel values but also on the spatial relationships and geometric structures within the data. It has the potential to enhance the performance of various computer vision tasks, such as object recognition and segmentation, by accounting for object shape and structure, making neural networks more adaptable and versatile in real-world applications.

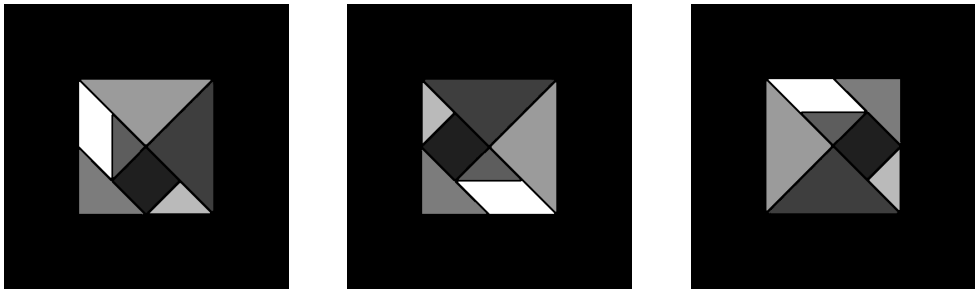
8.3 Hu Moments Evaluation Metric

Expanding upon the loss function established in Section 8.2, this dissertation proposes the application of the $Loss_{WMAE-Hu}$ for the task of evaluating Tangram solutions. In addition to refining the loss function to account for geometric features of Tangram pieces within the solution image, the present dissertation proposes the introduction of a novel evaluation metric denominated the Weighted Mean Absolute Error incorporating Hu Moments (WMAE-Hu). By integrating geometric properties alongside image data, this metric offers a more comprehensive evaluation of solution fidelity and geometric accuracy. While traditional metrics such as MSE[115] and SSIM[116] rely solely on pixel-level assessments, the proposed WMAE-Hu metric refines the evaluation process by capturing geometric nuances critical to Tangram solution accuracy. This extension reflects a deeper understanding of the geometric intricacies inherent in Tangram puzzles, thereby showing a logical solution process and proper understanding of the geometric features present in the Tangram.

To demonstrate the efficiency of WMAE-Hu, a mock scenario is presented where the desired pattern is the initial configuration of Tangram pieces in which they are boxed into a square. It starts with a feasible solution s_0 as the expected ground truth. However, a list of possible solutions also includes several other arrangements of pieces, such as s_1, s_2, s_3, s_4, s_5. In theory, an efficient evaluation metric should be able to identify that $s_0, s_1, s_2, s_3, s_4, s_5$ represent equally feasible solutions for the desired pattern. Therefore, this study serves to understand whether WMAE-Hu is prone to recognizing equally feasible solutions that form the same target pattern. Figure 8.3 shows the solutions being considered in this mock scenario.



(a) s_0. (b) s_1. (c) s_2.



(d) s_3. (e) s_4. (f) s_5.

Figure 8.3: Multiple arrangements of pieces that form the same Tangram pattern.

Traditional MSE and SSIM evaluation metrics are considered for comparison. Additionally, the metric

CWMAE that is a direct extension of $Loss_{CWMAE}$ is also examined, referencing the experiments performed in Section 7. The hypothesis is that WMAE-Hu will outperform those metrics when dealing with multiple feasible solutions, due to its ability to consider geometric information regarding the pieces in contrast to metrics that are more sensitive to pixel-level differences. Figure 8.4 shows the results obtained when evaluating the dissimilarity between s_1, s_2, s_3, s_4, s_5 when compared to the ground truth s_0.

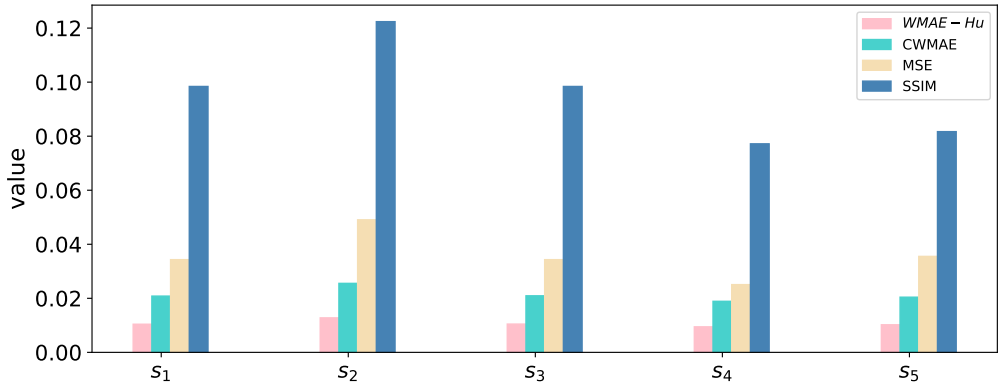


Figure 8.4: Metric values when comparing ground truth with other feasible solutions.

The average loss value for WMAE-Hu is 0.0107, for CWMAE it is 0.0213, for MSE it is 0.0356, and for SSIM it is 0.0956. It is important to mention that in the executed experiments, a slight amount of dissimilarity might be present among the feasible solutions. This can be attributed to inherent challenges associated with pixel representation and the inaccuracies from used image processing techniques. These factors can introduce subtle variations in the images, and it is not uncommon to observe minor discrepancies, even when the underlying content is similar. Overall, the results suggest that for all the executed experiments, WMAE-Hu is the one that produces the lowest average level of dissimilarity between s_0 and its variants, thus showing that it is consistently efficient in assessing Tangram solutions even when they are different from the expected ground truth. In particular, SSIM presented a poor performance by attributing substantially high levels of dissimilarity for the considered feasible solutions. This is because SSIM performs local comparisons between corresponding patches in the images. When an image is flipped or rotated, the local structures and orientations of features can change, thus significantly impacting the calculation of SSIM. Regarding MSE, it could assess the similarity between the feasible solutions fairly well, but it has proven less sensitive than WMAE-Hu and CWMAE in all the performed evaluations. It is also possible to notice that the loss values for WMAE-Hu are always close to half of the correspondent value for WMAE, which indicates that the values $Loss_{Hu}$ is close to zero based on the Equation 8.9. The non-expressibility of $Loss_{Hu}$ proves that the pieces in multiple solutions of the same Tangram pattern have close values for Hu Moments, thus making these invariant moments notably efficient in perceiving the geometry of the pieces.

To address the dispersion of the calculated levels of dissimilarity according to each metric, the coefficient of variation (CV) is considered [126]. It consists of a statistical measure used to quantify the relative variability or dispersion of a data distribution. It is calculated as the ratio of the standard deviation to the mean of the data. The coefficient provides a standardized measure of variability that allows for the com-

parison of variability between datasets with different units or scales. A higher CV indicates greater relative variability compared to the mean, while a lower CV indicates less relative variability. For this particular mock scenario, the average CV value for WMAE-Hu and CWMAE is 0.1046, for MSE it is 0.2155, and for SSIM it is 0.1665. Therefore, it indicates that WMAE-Hu and CWMAE present a lower variation when evaluating equivalent solutions for the same pattern, indicating that these metrics are notably consistent in identifying Tangram solutions that are equivalent in forming the same pattern. The other metrics are less consistent in this sense, attributing more variability to the calculated levels of dissimilarity although the considered solutions are equally feasible.

Therefore, this mock scenario serves to demonstrate the efficiency of the proposed evaluation metric WMAE-Hu in perceiving the similarity between multiple solutions for the same Tangram puzzle. It is important to highlight that this evaluation metric can also be applied to other tasks that permit multiple solutions, such as tilling problems [28], polyominoes [127, 128], furniture layout planning [129], and tessellation [130].

Chapter 9

Experiments

This chapter presents the main results of the proposed heuristic and deep learning approaches. Section 9.1 presents the experimental setup for the conducted experiments, while Section 9.2 presents the obtained results. Finally, Section 9.3 compares the employed approaches and references the findings of Chapter 7.

9.1 Experimental Setup

Implementation was done using Python 3.9.12 and Tensorflow 2.11.0 library. All tests were run on a Ryzen 3700x 3.6GHz 32GB of RAM with an Nvidia RTX 4090 24GB.

For the heuristic method, the dataset used for tests is the same as the one presented in Chapter 5 but the training set is not used due to the absence of a training stage. Thus the heuristic approach is submitted to 100 samples. A time limit of 360s is defined to assemble each Tangram puzzle, otherwise, it is considered that the method reached a timeout. It is assumed that 360s is the maximum time a user is willing to wait for a feasible solution for a given Tangram puzzle. It is also worth mentioning that the scale of the pieces is reduced by 5% to the scale of the input target pattern to avoid overlaps. Rotations are implemented in 1-degree increments. Since Tangram is a visual puzzle, this difference is typically unnoticeable to a human player, making this approach adequate for puzzles requiring unconstrained rotations.

For the deep learning approach, the dataset used for tests is the same as the one presented in Chapter 5. During training, the TANGAN uses a batch size of 20 to balance speed and memory efficiency. It uses the Adam Optimizer [131], where the learning rate starts at 0.0002 and the first-moment decay rate at 0.5. It also employs a binary cross-entropy loss for the discriminator and $Loss_{WMAE-Hu}$ for the generator, weighted by a factor of 100 relative to the discriminator loss. Regarding the parameter setting, empirical tests show that the best results are achieved using $c = 5$ for the penalty factor in Equation 8.2. The training is divided into two stages. In the first stage, the approach selects 100 samples from the training set to serve as a validation set, as indicated in Chapter 5. The loss function curves are constantly monitored to determine the number of epochs the model converges. The idea is that if the model is properly learning, both the training and validation loss curves will present a significant decrease. However, there is a point in training where the validation loss starts increasing, which indicates that the model is overfitting. This point in training indicates the number of epochs for the second stage of training. In the second stage, the validation set is reincorporated

into the training set and the testing set is finally used for testing. The model is retrained considering these sets are subjected to the number of epochs obtained in the previous stage of training.

9.2 Experimental Results

9.2.1 Results for Heuristic Approach

The heuristic method did not present a good performance, being able to assemble only 24% of the input Tangram puzzles in an average running time of 184.148s. Figure 9.1 presents some examples of solutions obtained using the heuristic method. The first row presents the input patterns, the second row presents a corresponding feasible solution, and the third row presents the obtained solution.

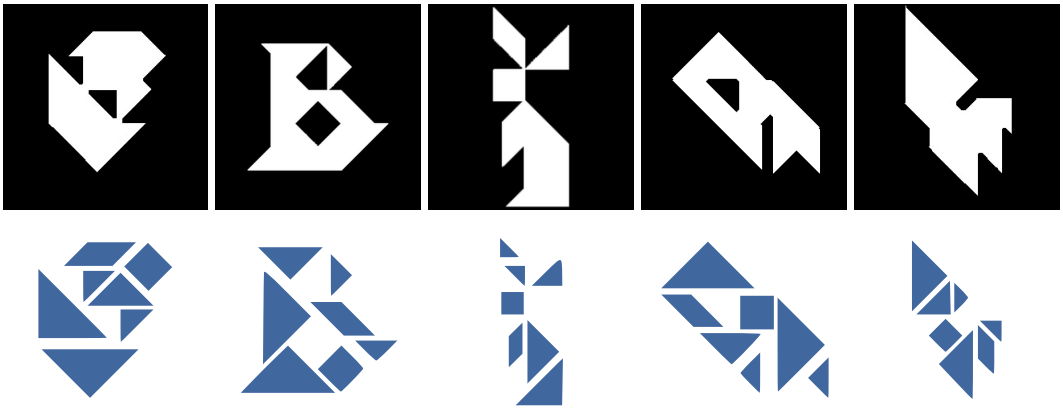


Figure 9.1: Solutions obtained by the heuristic approach.

Table 9.1 presents the number of solved puzzles categorized according to the characteristics that describe complex Tangram puzzles. The first row presents the number of solved puzzles considering each category of complex puzzles. The second row alludes to Table 5.1 and presents the total number of puzzles included in the proposed dataset that possess each aforementioned characteristic. The last row of the table summarizes the information of the previous rows by presenting the ratio of solved puzzles according to each characteristic. The flip ratio refers to the distribution of both enantiomers of the parallelogram found in the considered ground truth set, being the first enantiomer present in Figure 5.4 (a) and the second enantiomer present in Figure 5.4 (b).

	Holes	**Multiple**	**Unconstrained**	**Flip Ratio**
Solved	6	5	1	15 / 9
Total	34	40	14	70 / 30
Ratio	17.65%	12.50%	7.14%	21.14% / 30.00%

Table 9.1: Results for heuristic approach.

Regarding comparison with the methods present in the literature, 15 puzzles of the testing set were purposefully selected for sharing similarities with some puzzles investigated by other authors. The heuristic could solve 3 out of the 15 selected puzzles, thereby accounting for 20% of accuracy. This percentage is close to the overall 24% of puzzles correctly solved from the testing set.

An extra experiment is performed where the established time limit is doubled and then tripled to examine the impacts on the performance heuristic approach. The main idea is to examine the reason for the reduced number of solved complex puzzles by the heuristic approach. Table 9.2 summarizes the performance changes regarding average time and number of solved puzzles according to different time limits.

Time Limit	Solved Puzzles	Average Time
360s	24	184.148s
720s	30	274.917s
1080s	34	340.353s

Table 9.2: Effects of time limit on the heuristic approach.

Although the number of solved puzzles increases slightly when the time limit is extended, there is a substantial rise in the average time. Additionally, there are some interesting observations regarding the aspects of the puzzles solved when the time limit is increased. For instance, 3 out of the 4 extra puzzles that are solved when the time limit goes from 720s to 1080s demand unconstrained rotations, which indicates that the heuristic encounters difficulties in solving puzzles with unconstrained rotations in reasonable time due to the complexity associated with these puzzles.

9.2.2 Results for Deep Learning Approach

The average time for TANGAN to infer on a Tangram pattern image is 0.003s. A major issue emerges when analyzing the Tangram solutions generated by TANGAN since they are image-based and inherently ambiguous. Thus, Figure 9.2 illustrates a visual classification that is performed for assessing the generated Tangram solutions. Pieces are colored in blue for better visualization of sections between them. The images are obtained by thresholding black pixels to white and non-black pixels to blue. High-quality solutions are the ones that clearly represent the sections between pieces, even if the tones are off. Medium-quality images have at most, a missing or extra section, which implies that although they present some pieces well-defined, they demand some familiarity with Tangram pieces to infer their correct placements. Images with poor quality have a great portion of the pieces not well defined. Most of the poor-quality images are closer to the initial input pattern than an actual solution.

According to the visual quality classification of solutions, 70 samples were high quality, 17 were medium quality, and 13 were poor quality. This suggests that depending on their familiarity with the Tangram pieces they could solve between 70.0% and 87.0% of the samples by looking at the images generated with the TANGAN. Although not presenting the pieces clearly in the image, the solutions classified as poor may also be insightful, because even though the section between pieces is not properly defined, it often gets the

(a) High-quality solutions.

(b) Medium-quality solutions.

(c) Poor-quality solutions.

Figure 9.2: Solutions generated by TANGAN according to visual classification.

assigned grayscale tones correctly, at least to the point of being possible to determine which pixels are part of each piece.

Regarding the taxonomy of the generated solutions, Table 9.3 shows the number of solved patterns considering the visual quality classification and the aspects that characterize complex puzzles. The last row shows the percentage of solutions that fall between medium-quality and high-quality categories, thus indicating how many puzzles a player could solve only by looking at the images generated by TANGAN and subject to their familiarity with the Tangram pieces. For aspects that describe complex puzzles, refer to the terms used in Figure 1.4. For aspects that describe complex puzzles, refer to the terms used in Figure 1.4. The flip ratio refers to the distribution of both enantiomers of the parallelogram found in the considered ground truth set, being the first enantiomer present in Figure 5.4 (a) and the second enantiomer present in Figure 5.4 (b).

As previously stated, 15 puzzles of the testing set were purposefully selected due to similarities with some puzzles investigated by other authors. TANGAN presents certain difficulties in assembling those puzzles having 5 of them classified as medium-quality and 10 of them classified as poor-quality in the visual quality classification.

Classification	Holes	Multiple	Unconstrained	Flip Ratio
High	29	32	4	44 / 26
Medium	3	8	7	15 / 2
Poor	2	0	3	11 / 2
Total	34	40	14	70 / 30
Ratio	85% ∼ 94%	80% ∼ 100%	29% ∼ 79%	63% ∼ 84% / 87% ∼ 94%

Table 9.3: Taxonomical analysis on the generated solutions.

9.3 Experimental Analysis

In the following, the results obtained from the heuristic and the deep learning approaches are discussed. For the deep learning approach, a combination of TANGAN architecture and $Loss_{WMAE-Hu}$ proposed in Chapter 8 is employed. Additionally, a combination of VAE-GAN architecture proposed in Chapter 7 and CWMAE presented in Chapter 8 is used for comparison. This comparison is important for analyzing the effects of the introduction of $Loss_{WMAE}$ and modifications made to the architecture on the generated Tangram solutions. Table 9.4 compares the results obtained by the heuristic approach, VAE-GAN, and TANGAN regarding running time and complex aspects. Notice that the solutions output by the heuristic approach are not image-based, and do not need to undergo visual classification. For aspects that describe complex puzzles, refer to the terms used in Figure 1.4. The flip ratio refers to the distribution of both enantiomers of the parallelogram found in the considered ground truth set, being the first enantiomer present in Figure 5.4 (a) and the second enantiomer present in Figure 5.4 (b).

Method	Time	Classification	Holes	Multiple	Unconstrained	Flip Ratio
Heuristic	184.148s		17.65%	12.50%	7.14%	21.14% / 30.00%
VAE-GAN	0.002s	High	35.29%	55.00%	14.29%	56.67% / 31.43%
		Medium	38.24%	35.00%	42.86%	16.67% / 37.14%
		Poor	26.47%	10.00%	42.86%	26.67% / 31.43%
TANGAN	0.003s	High	85.29%	80.00%	28.57%	62.86% / 86.67%
		Medium	8.82%	20.00%	50.00%	21.43% / 6.67%
		Poor	5.88%	0.00%	21.43%	15.71% / 6.67%

Table 9.4: Comparative results of TANGAN, the VAE-GAN, and the heuristic approach.

The heuristic approach is commended for its straightforwardness and ease of application directly to testing sets, eliminating the need for a training stage. It proves advantageous when precise solutions are required, as it provides detailed information about piece positioning, rotation, and reflection by the end of

its execution. However, it did not present a good performance, especially for generated Tangram patterns. This poor performance can be attributed to how it calculates the feasible positions when fitting a piece. The heuristic approach depends on well-defined shapes with no imperfections along the edges of the Tangram pattern. The problem comes from how the Tangram patterns are generated in the dataset. As mentioned before, a random generator is used [100], then image processing techniques are applied to obtain the Tangram patterns and solutions. These image processing procedures may generate small imperfections along the border of Tangram patterns. This combined with the mathematical morphology techniques used for the piece placement may result in the generation of an imprecise morphological skeleton and an unnecessary number of endpoints. Another factor that compromises the performance of the heuristic approach is the fact that its complexity grows with the number of possible configurations for the pieces, which affects mainly complex puzzles. As previously stated, at the beginning of its execution, the heuristic generates a list of possible configurations for each piece to be tested one by one during its placement procedure. It prioritizes simple configurations, where the rotation is a multiple of 45^o and the parallelogram is not flipped by placing them in the initial positions of the list. This seems a fair strategy considering that most of the Tangram patterns in the literature represent simple puzzles, but since this dissertation focuses on complex puzzles, the number of complex features in the testing set is high, as shown in Table 5.1. This can influence the performance of the heuristic approach because it would take much time for the method to get to the correct configurations when considering each element in the list when dealing with unconstrained rotations and the reflection transformation. This observation is reinforced by the taxonomical analysis of the solutions obtained by the heuristic approach, which shows that the heuristic struggles to solve puzzles with unconstrained rotations. This is due to the unconstrained rotations having a notable influence on the increase in the number of possible configurations, and thus on the complexity of the solution process.

Since the solutions generated by TANGAN and VAE-GAN are image-based, they may be assessed by the evaluation metrics mentioned in Chapters 7 and 8. Therefore, Table 9.5 reports average values of MSE, SSIM, CWMAE, and WMAE-Hu for assessing the level of dissimilarity of the generated solutions and their respective ground truths. For all metrics, the best results are highlighted.

Method	Time	Classification	MSE	SSIM	CWMAE	WMAE-Hu
VAE-GAN	0.002s	High	0.0086	0.0523	0.0235	0.0118
		Medium	0.0121	0.0625	0.0261	0.0131
		Poor	0.0172	0.0897	0.0308	0.0154
		Average	**0.0123**	0.0667	0.0265	0.0133
TANGAN	0.003s	High	0.0162	0.0472	0.0233	0.0116
		Medium	0.0195	0.0643	0.0264	0.0132
		Poor	0.0378	0.0896	0.0356	0.0178
		Average	0.0196	**0.0556**	**0.0254**	**0.0127**

Table 9.5: Comparative results of VAE-GAN and TANGAN regarding evaluation metrics.

The results indicate that TANGAN outperforms VAE-GAN since it presents a higher ratio of high-quality solutions considering all the complex aspects in Table 9.4. Regarding the evaluation metrics in Table 9.5, the results reinforce that MSE is not adequate for assessing generated Tangram solutions, going against the results presented in Table 9.4 and suggesting that VAE-GAN generates solutions that are more accurate than TANGAN. The remaining metrics present scales that align better with the visual classification, establishing a lower average dissimilarity value for high-quality solutions, and a higher average dissimilarity value for poor-quality solutions. Considering the presented results and the findings of the mock scenario presented in Chapter 8, it is possible to conclude that WMAE-Hu is the most adequate metric for assessing Tangram solutions among the considered evaluation metrics. By focusing on geometric features rather than only on pixel-wise accuracy, this loss function addresses the challenge of one-to-many correspondence and enhances the ability of deep learning models to generalize across multiple feasible solutions. This underscores the importance of considering geometric properties in the design of loss functions for geometric tasks.

In comparison with the other two approaches, TANGAN offers considerable speed advantages over the heuristic approach, while outperforming VAE-GAN regarding all complex aspects. Its efficiency and speed make it a compelling choice for solving puzzles with unconstrained rotations that might be time-consuming for the heuristic approach. It is worth noticing that, in contrast to the heuristic method, TANGAN did not demand extra effort when dealing with complex puzzles when compared with simple puzzles. From the taxonomical analysis, it is possible to observe that TANGAN did not have problems in dealing with complex aspects related to the representation of the pattern, namely puzzles with multiple regions and holes within the puzzle area. This indicates that the chosen raster-based representation is adequate for depicting these aspects in a pattern in a way that TANGAN can properly differentiate between the puzzle area and the background area. Another finding evident from the taxonomical analysis is that the aspect that represented the biggest challenge for TANGAN was the unconstrained rotations, presenting a fairly balanced distribution between the visual quality grades. Moreover, unlike the heuristic approach, TANGAN demonstrates resilience to imperfections along the borders of puzzle areas. Figure 9.3 shows the progress in the assembly of a Tangram pattern throughout the training process.

Similarly to the work by Deutsch & Hayes [91], TANGAN tackles the Tangram as a sectioning problem. It uses convex corners along the borders of the container to find the lines that represent the segmentation of the pieces. It then takes advantage of the geometric property presented in Figure 1.3 by segmenting the pattern into small triangles and then determining the position of each piece based on the configuration of small triangles. This is an interesting observation because it shows that the model could learn a previously documented solution strategy without being directly presented to it, which reinforces the idea that it can learn complex geometric relations between the pieces and the target pattern. This strategy has proven more efficient for slim patterns than wider patterns that do not present many convex corners along the pattern borders that might assist in finding the sections between pieces. This is the reason for the poor performance of the heuristic when assembling the 15 puzzles addressed by other authors since many of them have wider shapes. Some examples can be found in Figure 5.3 (e), Figure 5.3 (h), Figure 5.3 (k), and Figure 5.3 (o). It is possible to notice that the random generator by Köpp [100] tends to generate samples where the pieces are more distributed, and often contain convex corners that can be used to segment the puzzle area. Therefore, a reasonable idea would be to include more samples of Tangram puzzles containing composites [91] in the

(a) EPOCH = 110. (b) EPOCH = 120. (c) EPOCH = 210.

(d) EPOCH = 270. (e) EPOCH = 280. (f) EPOCH = 600.

Figure 9.3: Progress of solutions over the epochs considering a sample from the testing set.

training set, thus encouraging deep learning models to learn how the Tangram pieces can interact to form convex polygons. By incorporating such samples, the employed model can better generalize to a wider variety of puzzle configurations.

Chapter 10

Conclusions

10.1 Conclusive Remarks

In this dissertation, two novel approaches for the automatic solution of Tangram puzzles were presented: a heuristic approach and a deep learning approach. The heuristic approach uses a raster representation for the puzzle and pieces and takes advantage of mathematical morphology techniques that are traditionally applied to C&P problems to avoid overlaps between pieces. The deep learning approach is based on a GAN architecture. The proposed approaches aim at solving both simple and complex Tangram puzzles [24]. While simple puzzles can be described by a simply connected contour, and have a fairly reduced number of configurations that each piece can assume, complex puzzles demand more complex forms of representation and demand more computational effort to be solved. Complex puzzles may be composed of multiple connected regions, possibly with holes. In addition, they may require non-discrete rotations, as well as the reflection operation for the parallelogram piece. Many authors in the literature ignore the aspects that characterize complex puzzles, focusing only on the solution of simple puzzles. The literature review indicates that there is no work in the literature that solves every type of complex Tangram puzzle.

The heuristic approach presents some strong points over the deep learning approach. It does not need a training process like general deep learning approaches. This eliminates the necessity of gathering a huge amount of data to be used for generalization. Moreover, it is adaptable to different puzzle scales, being able to infer the size of the pieces from the puzzle area. In this sense, the heuristic is more versatile for being adaptable to different sizes of puzzles without the necessity of pre-processing. The heuristic approach is also more indicated when the user wants to know the exact translation and rotation of the pieces by the end of the assembly process. On the other hand, the heuristic method is notably slow and may take a few minutes to solve a Tangram puzzle. The morphological operations are especially time-consuming and may slow down the process of fitting Tangram pieces. It is also susceptible to imperfections along the edges of the Tangram pattern.

The strong points of the deep learning approach lie in the fact that it is considerably faster than the heuristic approach in presenting a Tangram solution. It is also more adaptable than the heuristic approach in the sense that it can solve the Tangram puzzles even with imperfections along the borders of the Tangram pattern. Additionally, the deep learning approach has shown the ability to solve simple and complex

puzzles without any significant difference in the required effort, while the heuristic approach demands more resources and running time to solve a complex puzzle. On the other hand, the deep learning approach is not indicated when the user does not have the time to train the model. Another case where the usage of the deep learning approach is not recommended is when the user wants to know the exact configurations of the pieces to obtain the desired Tangram pattern because the deep learning approach outputs an image that is a representation of the solution and is open for interpretation. It might also present some difficulty in solving patterns that are not scaled according to the samples used for training. The learning process of the deep learning approach has proven to be sensitive to the geometric features the model can extract from the pieces. Therefore, if the scale of the pieces changes, the model might get lost in inferring a feasible solution.

It is possible to conclude that the choice of using the heuristic approach or the deep learning approach in the Tangram solution is relative to the desired outcomes. Nonetheless, both approaches have proven capable of solving both simple and complex Tanrgam puzzles, overcoming critical limitations observed in the literature. The proposed dataset also poses as a valuable resource for research and evaluation in dissection puzzles, as well as related optimization problems. The literature shows that the proposed dataset is the most extensive in the literature regarding the automatic solution of Tangram puzzles. Additionally, along with the deep learning approach, a loss function designed for assessing Tangram solutions was also proposed and has proven efficient in assessing Tangram solutions. The usage of Hu Moments in the loss function allows for the assessment of the geometry of the pieces, rather than their specific position or configuration. By focusing on geometric features, the loss function overcomes limitations inherent in traditional metrics by better accommodating multiple feasible solutions, making it suitable not only for Tangram puzzles but also for other dissection puzzles. Therefore, this dissertation advances the current understanding of artificial intelligence in complex geometrical problem domains and provides a robust foundation for future research.

10.2 Future Directions

Regarding expanding the outcomes of this dissertation to other domains, one possibility is to combine both approaches into a hybrid method. For many puzzles, the deep learning approach can discern the correct placement of pieces. However, there were cases where it did not clearly define the section between a pair of pieces. Those solutions were considered of medium quality in the visual assessment. Although the solution image depicts the correct placement of most of the pieces, under a more rigorous perspective and strictly following the rules of the Tangram, these solutions should be perceived as incorrect because it does not contain all the pieces. To avoid those cases, a viable strategy would be to combine the deep learning approach with the heuristic approach. This methodology would start with the application of the trained deep learning model on the desired pattern binary image, thus generating a solution image that partially depicts the pieces in their correct placements. Then, a method based on image processing can be used to identify which pieces are correctly placed and the missing pieces. After that, it can form another binary mask, where the white areas represent the regions not covered by correctly placed pieces, while the areas covered by correctly placed pieces and background can be presented in black. From this point, the heuristic method becomes responsible for placing the missing pieces in the regions depicted in the binary mask. This approach is feasible because the heuristic approach is adaptable to any number of pieces. Therefore,

this hybrid method combines the fast inference of the deep learning approach with the adaptability of the heuristic method.

Another possible extension of the present dissertation would be to use cross-domain transfer learning to extend the knowledge acquired by the deep learning model. Similarly to the methodology presented by Zhao et al. [129], it is possible to extend the principles of morphology and spatial geometry learned by the trained model for solving other tasks. The application includes not only other types of geometric puzzles but also robotics, digital arts, and industrial applications. For instance, Spencer [29] shows that the Tangram pieces can be used to produce patchwork designs, which can be used to produce mosaics, quilts, and animation. Another feasible direction for the proposed approaches is to expand them to C&P problems, whose application extends to textile, sheet metal, leather, glass, and paper industries [43]. Over the years, research on C&P problems has received more attention and exploration compared to dissection puzzles. As a result, methods for solving C&P problems benefit from a more robust baseline for comparison, with established metrics and datasets. For tests in C&P problems, researchers may use the instances from ESICUP [132] for benchmarking and comparison with the literature.

Last, it is also possible to explore the didactic nature of the Tangram and expand the outcomes of the present dissertation to research related to education for children. As an educative tool, Tangram offers an intriguing blend of visual perception, spatial reasoning, and logical deduction [133]. Given the silhouette of a desired pattern, it is possible to incorporate devices, such as webcams and smartphones, to capture solutions formed by a physical Tangram. Then, it is possible to use the proposed evaluation metric to assess how good the presented solution is. One of the key advantages of using technology in this way is the ability to provide immediate feedback to children. As they arrange the Tangram pieces, the webcam captures their solutions in real-time, and the smartphone instantly assesses the correctness of their arrangements. Moreover, the usage of interactive devices adds an element of excitement to the activity, making it more engaging and appealing to learners. Instead of passively receiving instructions, they become active participants by physically interacting with the Tangram pieces and using technology to assess their solutions. As they experiment with different arrangements of the Tangram pieces, children are encouraged to think critically, use spatial reasoning, and apply logical thinking to find solutions. A similar approach is presented in the work of Lin et al. [134] and Singh et al. [135].

Bibliography

[1] M. Brand, "No easy puzzles: Hardness results for jigsaw puzzles," Theoretical Computer Science, vol.586, pp.2–11, 2015.

[2] N. Alajlan, "Solving square jigsaw puzzles using dynamic programming and the hungarian procedure," American Journal of Applied Sciences, vol.6, no.11, p.1941, 2009.

[3] S. Markaki and C. Panagiotakis, "Jigsaw puzzle solving techniques and applications: a survey," The Visual Computer, pp.1–17, 2022.

[4] F. Richter, C.X. Ries, N. Cebron, and R. Lienhart, "Learning to reassemble shredded documents," IEEE Transactions on multimedia, vol.15, no.3, pp.582–593, 2012.

[5] Z. Hammoudeh and C. Pollett, "Clustering-based, fully automated mixed-bag jigsaw puzzle solving," Computer Analysis of Images and Patterns: 17th International Conference, CAIP 2017, Ystad, Sweden, August 22-24, 2017, Proceedings, Part II 17, pp.205–217, Springer, 2017.

[6] M. Norgate, "Cutting borders: Dissected maps and the origins of the jigsaw puzzle," The Cartographic Journal, vol.44, no.4, pp.342–350, 2007.

[7] D.A. Gschwend, A.C. Good, and I.D. Kuntz, "Molecular docking towards drug discovery," Journal of Molecular Recognition: An Interdisciplinary Journal, vol.9, no.2, pp.175–186, 1996.

[8] Y.C. Kim, K.H. Min, J.W. Choi, K.S. Koh, T.S. Oh, and W.S. Jeong, "Patient-specific puzzle implant preformed with 3d-printed rapid prototype model for combined orbital floor and medial wall fracture," Journal of Plastic, Reconstructive & Aesthetic Surgery, vol.71, no.4, pp.496–503, 2018.

[9] C.C. Hsu and T.I. Wang, "Applying game mechanics and student-generated questions to an online puzzle-based game learning system to promote algorithmic thinking skills," Computers & Education, vol.121, pp.73–88, 2018.

[10] K. Tang, P. Song, X. Wang, B. Deng, C.W. Fu, and L. Liu, "Computational design of steady 3d dissection puzzles," 2019.

[11] D. Ausonius and R. Green, Decimi Magni Ausonii Opera, Oxford Classical Texts, Typographeo Clarendoniano, 1999.

[12] M. Tchoshanov, "Building students' mathematical proficiency: connecting mathematical ideas using the tangram," Learning and Teaching Mathematics, vol.2011, no.10, pp.16–23, 2011.

[13] E. Fox-Epstein, K. Katsumata, and R. Uehara, "The convex configurations of "sei shonagon chie no ita," tangram, and other silhouette puzzles with seven pieces," IEICE transactions on fundamentals of electronics, communications and computer sciences, vol.99, no.6, pp.1084–1089, 2016.

[14] J. Slocum, "Tangram: The world's first puzzle craze," Published by Sterling, 2003.

[15] Z. Liu and W. Liu, "Research on the design of combination furniture based on toy brick style concept," 5th International Conference on Civil Engineering and Transportation, pp.1712–1717, Atlantis Press, 2015.

[16] E. Pascual, Tangram Proficiency Leading to Numeracy Skills Enhancement, LAP LAMBERT Academic Publishing, 2020.

[17] W. Gao and K. Ramani, "Kaleidogami™: Multi-primitive reconfigurable artistic structures," School of Mechanical Engineering School, Electrical and Computer Engineering, Purdue University: by Courtesy, 2012.

[18] S.S. Pohl and C. Richter, "The complete characterization of tangram pentagons," Beiträge zur Algebra und Geometrie/Contributions to Algebra and Geometry, vol.62, no.1, pp.121–135, 2021.

[19] M. Kmetová and Z. Nagyová Lehocká, "Using tangram as a manipulative tool for transition between 2d and 3d perception in geometry," Mathematics, vol.9, no.18, p.2185, 2021.

[20] K. Khairiree, "Creative thinking in mathematics with tangrams and the geometer's sketchpad," Proceedings of the 20th Asian Technology Conference in Mathematics, pp.153–161, 2015.

[21] S. Li, "Splicing all possible convex and non-convex pentagons with tangram using dfs," 2022 International Conference on Cloud Computing, Big Data Applications and Software Engineering (CBASE), pp.60–64, IEEE, 2022.

[22] M. Tombuloğlu, "Grafik tasarım bağlamında tangram; ankaranın kültür denizi tangram oyun seti," Master's thesis, Güzel Sanatlar Enstitüsü, 2019.

[23] E.D. Demaine, M. Korman, J.S. Ku, J.S. Mitchell, Y. Otachi, A. van Renssen, M. Roeloffzen, R. Uehara, and Y. Uno, "Symmetric assembly puzzles are hard, beyond a few pieces," Computational Geometry, vol.90, p.101648, 2020.

[24] F.M. Yamada, J.P. Gois, and H.C. Batagelo, "Solving tangram puzzles using raster-based mathematical morphology," 2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp.116–123, IEEE, 2019.

[25] T. Martins and M.S.G. Tsuzuki, "Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions," Expert Systems with Applications, vol.37, no.3, pp.1955–1972, 2010.

[26] L. Bofferding and M. Aqazade, ""where does the square go?": reinterpreting shapes when solving a tangram puzzle," Educational Studies in Mathematics, vol.112, no.1, pp.25–47, 2023.

[27] H. Wang, "Using dfs search and enumerate method to find all solutions in 13 convex figures in tangram game," 2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI), pp.505–509, IEEE, 2021.

[28] S. Spencer, "An introduction to the tiling properties of the tangram and their application in two and three dimensions," Bridges: Mathematical Connections in Art, Music, and Science, pp.71–78, 2004.

[29] S. Spencer, "Introducing the precious tangram family," Bridges London: Mathematics, Music, Art, Architecture, Culture, pp.73–78, 2006.

[30] N. Vereshchagin, "A family of non-periodic tilings of the plane by right golden triangles," Discrete & Computational Geometry, vol.68, no.1, pp.188–217, 2022.

[31] F.M. Yamada and H.C. Batagelo, "A comparative study on computational methods to solve tangram puzzles," Workshop of Works in Progress (WIP) in the 30th Conference on Graphics, Patterns and Images (SIBGRAPI'17), October 2017.

[32] A.P. Gantapara, W. Qi, and M. Dijkstra, "A novel chiral phase of achiral hard triangles and an entropy-driven demixing of enantiomers," Soft Matter, vol.11, no.44, pp.8684–8691, 2015.

[33] J. Lee, J. Kim, H. Chung, J. Park, and M. Cho, "Learning to assemble geometric shapes," International Joint Conference on Artificial Intelligence, 2022.

[34] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," International Conference on Learning Representations, 2018.

[35] S.Z. Kovalsky, D. Glasner, and R. Basri, "A global approach for solving edge-matching puzzles," SIAM Journal on Imaging Sciences, vol.8, no.2, pp.916–938, 2015.

[36] L.R. Mundim, M. Andretta, M.A. Carravilla, and J.F. Oliveira, "A general heuristic for two-dimensional nesting problems with limited-size containers," International Journal of Production Research, pp.1–24, 2017.

[37] A.M. Gomes and J.F. Oliveira, "Solving irregular strip packing problems by hybridising simulated annealing and linear programming," European Journal of Operational Research, vol.171, no.3, pp.811–829, 2006.

[38] D.J. Chalmers and B. Rabern, "Two-dimensional semantics and the nesting problem," Analysis, vol.74, no.2, pp.210–224, 2014.

[39] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," European journal of operational research, vol.183, no.3, pp.1109–1130, 2007.

[40] L.H. Cherri, M.A. Carravilla, C. Ribeiro, and F.M.B. Toledo, "Optimality in nesting problems: New constraint programming models and a new global constraint for non-overlap," Operations Research Perspectives, vol.6, p.100125, 2019.

[41] J.j. Xu, X.s. Wu, H.m. Liu, and M. Zhang, "An optimization algorithm based on no-fit polygon method and hybrid heuristic strategy for irregular nesting problem," Control Conference (CCC), 2017 36th Chinese, pp.2858–2863, IEEE, 2017.

[42] Y. Qin, F.T. Chan, S. Chung, T. Qu, and B. Niu, "Aircraft parking stand allocation problem with safety consideration for independent hangar maintenance service providers," Computers & Operations Research, vol.91, pp.225–236, 2018.

[43] J.A. Bennell and X. Song, "A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums," Computers & Operations Research, vol.35, no.1, pp.267–281, 2008.

[44] Y. Hu, S. Fukatsu, H. Hashimoto, S. Imahori, and M. Yagiura, "Efficient overlap detection and construction algorithms for the bitmap shape packing problem," Journal of the Operations Research Society of Japan, vol.61, no.1, pp.132–150, 2018.

[45] A.K. Sato, T.C. Martins, and M.S.G. Tsuzuki, "Collision free region determination by modified polygonal boolean operations," Computer-Aided Design, vol.45, no.7, pp.1029–1041, 2013.

[46] B.A. Junior, P.R. Pinheiro, and R.D. Saraiva, "Tackling the irregular strip packing problem by hybridizing genetic algorithm and bottom-left heuristic," Evolutionary Computation (CEC), 2013 IEEE Congress on, pp.3012–3018, IEEE, 2013.

[47] H. Mühlthaler and H. Pottmann, "Computing the minkowski sum of ruled surfaces," Graphical Models, vol.65, no.6, pp.369–384, 2003.

[48] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," 2010 IEEE intelligent vehicles symposium, pp.518–522, IEEE, 2010.

[49] G. Varadhan and D. Manocha, "Accurate minkowski sum approximation of polyhedral models," 12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings., pp.392–401, IEEE, 2004.

[50] J.A. Bennell, K.A. Dowsland, and W.B. Dowsland, "The irregular cutting-stock problem—a new procedure for deriving the no-fit polygon," Computers & Operations Research, vol.28, no.3, pp.271–287, 2001.

[51] A.M. Gomes and J.F. Oliveira, "A grasp approach to the nesting problem," 4th Metaheuristics International Conference MIC'2001, 2001.

[52] H.T. Dean, Y. Tu, and J.F. Raffensperger, "An improved method for calculating the no-fit polygon," Computers & operations research, vol.33, no.6, pp.1521–1539, 2006.

[53] A.K. Sato, G.E.S. Bauab, T. de Castro Martins, M.d.S.G. Tsuzuki, and A.M. Gomes, "A study in pairwise clustering for bi-dimensional irregular strip packing using the dotted board model," IFAC-PapersOnLine, vol.51, no.11, pp.284–289, 2018.

[54] Y. Rao and Q. Luo, "Intelligent algorithms for packing and cutting problem," 2022.

[55] M.A. Carravilla, C. Ribeiro, and J.F. Oliveira, "Solving nesting problems with non-convex polygons by constraint logic programming," International Transactions in Operational Research, vol.10, no.6, pp.651–663, 2003.

[56] A.K. Sato, T. de Castro Martins, and M.d.S.G. Tsuzuki, "Rotational placement using simulated annealing and collision free region," IFAC Proceedings Volumes, vol.43, no.4, pp.234–239, 2010.

[57] T.d.C. Martins and M.S. Tsuzuki, "Rotational placement of irregular polygons over containers with fixed dimensions using simulated annealing and no-fit polygons," Journal of the Brazilian Society of Mechanical Sciences and Engineering, vol.30, no.3, pp.205–212, 2008.

[58] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," Psychological review, vol.65, no.6, p.386, 1958.

[59] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618," Genetic programming and evolvable machines, vol.19, no.1-2, pp.305–307, 2018.

[60] P. Antsaklis, "Neural networks for control systems," IEEE Transactions on Neural Networks, vol.1, no.2, pp.242–244, 1990.

[61] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," IEEE transactions on pattern analysis and machine intelligence, vol.35, no.8, pp.1915–1929, 2012.

[62] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning representations by back-propagating errors," nature, vol.323, no.6088, pp.533–536, 1986.

[63] G. Montavon, G. Orr, and K.R. Müller, "Neural networks: tricks of the trade," 2012.

[64] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol.60, no.6, pp.84–90, 2017.

[65] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.1–9, 2015.

[66] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.770–778, 2016.

[68] W.G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," IEEE Access, vol.6, pp.24411–24432, 2018.

[69] R. Chen and E.M.K. Lai, "Convolutional autoencoder for single image dehazing.," ICIP, pp.4464–4468, 2019.

[70] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," 2018 IEEE international conference on systems, man, and cybernetics (SMC), pp.415–419, IEEE, 2018.

[71] V. Turchenko and A. Luczak, "Creation of a deep convolutional auto-encoder in caffe," 2017.

[72] Z. Salekshahrezaee, J.L. Leevy, and T.M. Khoshgoftaar, "A class-imbalanced study with feature extraction via pca and convolutional autoencoder," 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), pp.63–68, IEEE, 2022.

[73] D. Jana, J. Patil, S. Herkal, S. Nagarajaiah, and L. Duenas-Osorio, "Cnn and convolutional autoencoder (cae) based real-time sensor fault detection, localization, and correction," Mechanical Systems and Signal Processing, vol.169, p.108723, 2022.

[74] D.P. Kingma and M. Welling, "Auto-encoding variational bayes," arXiv preprint arXiv:1312.6114, 2013.

[75] D.P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder," 2014.

[76] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," Advances in neural information processing systems, vol.33, pp.19667–19679, 2020.

[77] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp.234–241, Springer, 2015.

[78] X.X. Yin, L. Sun, Y. Fu, R. Lu, and Y. Zhang, "[retracted] u-net-based medical image segmentation," Journal of healthcare engineering, vol.2022, no.1, p.4189781, 2022.

[79] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," Advances in neural information processing systems, vol.27, 2014.

[80] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," International Journal of Information Management Data Insights, vol.1, no.1, p.100004, 2021.

[81] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," IEEE transactions on knowledge and data engineering, vol.35, no.4, pp.3313–3332, 2021.

[82] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," Communications of the ACM, vol.63, no.11, pp.139–144, 2020.

[83] L. Gonog and Y. Zhou, "A review: generative adversarial networks," 2019 14th IEEE conference on industrial electronics and applications (ICIEA), pp.505–510, IEEE, 2019.

[84] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A.A. Bharath, "Generative adversarial networks: An overview," IEEE signal processing magazine, vol.35, no.1, pp.53–65, 2018.

[85] X. Chang, F. Chao, C. Shang, and Q. Shen, "Sundial-gan: A cascade generative adversarial networks framework for deciphering oracle bone inscriptions," Proceedings of the 30th ACM International Conference on Multimedia, pp.1195–1203, 2022.

[86] J. Ma, W. Yu, C. Chen, P. Liang, X. Guo, and J. Jiang, "Pan-gan: An unsupervised pan-sharpening method for remote sensing image fusion," Information Fusion, vol.62, pp.110–120, 2020.

[87] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," Advances in neural information processing systems, vol.33, pp.12104–12114, 2020.

[88] C. Shorten and T.M. Khoshgoftaar, "A survey on image data augmentation for deep learning," Journal of big data, vol.6, no.1, pp.1–48, 2019.

[89] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard, "Adaptive data augmentation for image classification," 2016 IEEE international conference on image processing (ICIP), pp.3688–3692, Ieee, 2016.

[90] P. Chlap, H. Min, N. Vandenberg, J. Dowling, L. Holloway, and A. Haworth, "A review of medical image data augmentation techniques for deep learning applications," Journal of Medical Imaging and Radiation Oncology, vol.65, no.5, pp.545–563, 2021.

[91] E.S. Deutsch and K.C. Hayes Jr., "A heuristic solution to the tangram puzzle," Machine Intelligence, vol.7, pp.205–240, 1972.

[92] K. Oflazer, "Solving tangram puzzles: A connectionist approach," International journal of intelligent systems, vol.8, no.5, pp.603–616, 1993.

[93] D. Bartoněk, "A genetic algorithm how to solve a puzzle and its using in cartography," Acta Scientiarum Polonorum. Geodesia et Descriptio Terrarum, vol.4, no.2, pp.15–23, 2005.

[94] C. Domokos and Z. Kato, "Realigning 2d and 3d object fragments without correspondences," IEEE transactions on pattern analysis and machine intelligence, vol.38, no.1, pp.195–202, 2015.

[95] A. Rafique, T. Iftikhar, and N. Khan, "Adversarial placement vector learning," 2019 2nd International Conference on Advancements in Computational Sciences (ICACS), pp.1–7, IEEE, 2019.

[96] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu, "Layoutgan: Generating graphic layouts with wireframe discriminators," arXiv preprint arXiv:1901.06767, 2019.

[97] W. Wang, M. Zhang, G. Chen, H. Jagadish, B.C. Ooi, and K.L. Tan, "Database meets deep learning: Challenges and opportunities," ACM Sigmod Record, vol.45, no.2, pp.17–22, 2016.

[98] M.A. Bansal, D.R. Sharma, and D.M. Kathuria, "A systematic review on data scarcity problem in deep learning: solution and applications," ACM Computing Surveys (CSUR), vol.54, no.10s, pp.1–29, 2022.

[99] Tangram-Channel, "Tangram channel website." https://www.tangram-channel.com/. Accessed on June 11th, 2024.

[100] W. Köpp, "Random generation of tangrams," Interdisciplinary Project in Mathematics, Technische Universitat München, 2013.

[101] V.G. Zakharov, "Rotation properties of 2d isotropic dilation matrices," International Journal of Wavelets, Multiresolution and Information Processing, vol.16, no.01, p.1850001, 2018.

[102] B. Baran, B. Dogusoy, and K. Cagiltay, "How do adults solve digital tangram problems? analyzing cognitive strategies through eye tracking approach," in Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments, pp.555–563, Springer, 2007.

[103] M.K. Hu, "Visual pattern recognition by moment invariants," IRE transactions on information theory, vol.8, no.2, pp.179–187, 1962.

[104] M. Lukic, E. Tuba, and M. Tuba, "Leaf recognition algorithm using support vector machine with hu moments and local binary patterns," 2017 IEEE 15th international symposium on applied machine intelligence and informatics (SAMI), pp.000485–000490, IEEE, 2017.

[105] I. Kramer, N. Schmidt, R. Memmesheimer, and D. Paulus, "Evaluation of physical therapy through analysis of depth images," 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pp.1–6, IEEE, 2019.

[106] K.A. Dowsland, S. Vaid, and W.B. Dowsland, "An algorithm for polygon placement using a bottom-left strategy," European Journal of Operational Research, vol.141, no.2, pp.371–381, 2002.

[107] L.R. Mundim, M. Andretta, and T.A. de Queiroz, "A biased random key genetic algorithm for open dimension nesting problems using no-fit raster," Expert Systems with Applications, vol.81, pp.358–371, 2017.

[108] P.E. Trahanias, "Binary shape recognition using the morphological skeleton transform," Pattern recognition, vol.25, no.11, pp.1277–1288, 1992.

[109] R. Fabbri, L.D.F. Costa, J.C. Torelli, and O.M. Bruno, "2d euclidean distance transform algorithms: A comparative survey," ACM Computing Surveys (CSUR), vol.40, no.1, p.2, 2008.

[110] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time euclidean distance transform algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.17, no.5, pp.529–533, 1995.

[111] M.S. Minhas and J. Zelek, "Semi-supervised anomaly detection using autoencoders," arXiv preprint arXiv:2001.03674, 2020.

[112] Y. Zhou, W. Huang, P. Dong, Y. Xia, and S. Wang, "D-unet: a dimension-fusion u shape network for chronic stroke lesion segmentation," IEEE/ACM transactions on computational biology and bioinformatics, vol.18, no.3, pp.940–950, 2019.

[113] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.W. Lin, "Deep learning on image denoising: An overview," Neural Networks, vol.131, pp.251–275, 2020.

[114] X.X. Yin, L. Sun, Y. Fu, R. Lu, Y. Zhang, et al., "U-net-based medical image segmentation," Journal of Healthcare Engineering, vol.2022, 2022.

[115] C. Dong, C.C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," IEEE transactions on pattern analysis and machine intelligence, vol.38, no.2, pp.295–307, 2015.

[116] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE transactions on image processing, vol.13, no.4, pp.600–612, 2004.

[117] V.L. Trevisan de Souza, B.A.D. Marques, H.C. Batagelo, and J. ao Paulo Gois, "A review on generative adversarial networks for image generation," Computers & Graphics, vol.114, pp.13–25, 2023.

[118] S. Ameer and O. Basir, "Objective image quality measure based on weber-weighted mean absolute error," 2008 9th International Conference on Signal Processing, pp.728–732, IEEE, 2008.

[119] S. Hao and S. Li, "A weighted mean absolute error metric for image quality assessment," 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), pp.330–333, IEEE, 2020.

[120] S. Hu, L. Jin, H. Wang, Y. Zhang, S. Kwong, and C.C.J. Kuo, "Objective video quality assessment based on perceptually weighted mean squared error," IEEE Transactions on Circuits and Systems for Video Technology, vol.27, no.9, pp.1844–1855, 2016.

[121] X. Tan, T. Qin, J. Bian, T.Y. Liu, and Y. Bengio, "Regeneration learning: A learning paradigm for data generation," arXiv preprint arXiv:2301.08846, 2023.

[122] R. Unni, K. Yao, and Y. Zheng, "Deep convolutional mixture density network for inverse design of layered photonic structures," ACS Photonics, vol.7, no.10, pp.2703–2712, 2020.

[123] F. Solís, D. Martínez, and O. Espinoza, "Automatic mexican sign language recognition using normalized moments and artificial neural networks," Engineering, vol.8, no.10, pp.733–740, 2016.

[124] L. Basavaraj and R. Sudhaker Samuel, "Offline handwritten character detection using image components," 2007.

[125] R. Swetha, P. Bende, K. Singh, S. Gorthi, A. Biswas, B. Li, D.C. Weindorf, and S. Chakraborty, "Predicting soil texture from smartphone-captured digital images and an application," Geoderma, vol.376, p.114562, 2020.

[126] H. Abdi, "Coefficient of variation," Encyclopedia of research design, vol.1, no.5, 2010.

[127] N. Kita, "Dissection puzzles composed of multicolor polyominoes," 2023.

[128] C. Yang, "Tiling the plane with a set of ten polyominoes," International Journal of Computational Geometry & Applications, pp.1–10, 2023.

[129] Y. Zhao, L. Qiu, P. Lu, F. Shi, T. Han, and S.C. Zhu, "Learning from the tangram to solve mini visual tasks," 2022.

[130] Q. Du, M. Gunzburger, and L. Ju, "Advances in studies and applications of centroidal voronoi tessellations," Numerical Mathematics: Theory, Methods and Applications, vol.3, no.2, pp.119–142, 2010.

[131] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[132] ESICUP, "Special interest group in cutting and packing," 2007. Accessed on December 1st, 2023.

[133] I.R.D. Renavitasari and A.A. Supianto, "Educational game for training spatial ability using tangram puzzle," 2018 International Conference on Sustainable Information Engineering and Technology (SIET), pp.174–179, IEEE, 2018.

[134] C.Y. Lin, H.C. Chai, J.y. Wang, C.J. Chen, Y.H. Liu, C.W. Chen, C.W. Lin, and Y.M. Huang, "Augmented reality in educational activities for children with disabilities," Displays, vol.42, pp.51–54, 2016.

[135] K. Singh, A. Shrivastava, K. Achary, A. Dey, and O. Sharma, "Augmented reality-based procedural task training application for less privileged children and autistic individuals," Proceedings of the 17th International Conference on Virtual-Reality Continuum and its Applications in Industry, pp.1–10, 2019.

Publication Lists

Related Publications

Journal

1. F. M. Yamada, H. C. Batagelo, J. P. Gois, and H. Takahashi, “Generative Approaches for Solving Tangram Puzzles”, Discover Artificial Intelligence, Vol. 4, Article no. 12, pp. 1-20, Feb. 2024. (Chapter 7).

International Conference

1. F. M. Yamada, H. Takahashi, H. C. Batagelo, and J. P. Gois, "Raster-based mathematical morphology for cutting and packing problems", Proceedings of the International Workshop on Advanced Imaging Technology, vol. 11766, pp. 204-209, Mar. 2021. (Chapter 6).
2. F. M. Yamada, H. Takahashi, H. C. Batagelo, and J. P. Gois, "An Extended Approach for the Automatic Solution of Tangram Puzzles Using Permutation Heuristics", 2020 Nicograph International, pp. 47-50, Jun. 2020. (Chapter 6).
3. F. M. Yamada, J. P. Gois, and H. C. Batagelo, "Solving tangram puzzles using raster- based mathematical morphology", SIBGRAPI Conference on Graphics, Patterns and Images, pp. 116-123, Oct. 2019. (Chapter 6).
4. F. M. Yamada and H. C. Batagelo, "A comparative study on computational methods to solve tangram puzzles", Workshop of Works in Progress in SIBGRAPI Conference on Graphics, Patterns and Images, Oct. 2017. (Chapter 4).

Unrelated Publications

Journal

1. F. M. Yamada, T. Ribeiro, and N. P. Ghilardi-Lopes, "Assessment of the prototype of an educational game on climate change and its effects on marine and coastal ecosystems", Brazilian Journal of Computers in Education, vol. 27, no. 03, pp. 01-31, 2019.

Awards

1. E. Pinhata, F.M. Yamada, F. A. S. Berchez, J. C. Braga, L. Silva, N. P. Ghilardi-Lopes, R. L. F. Silva, S. R. Freitas, and T. Ribeiro (2014). Apicum Game. Patent No. BR512016000178-0. Instituto Nacional da Propriedade Industrial.

Other Contributions

1. Development of an educational software addressing global environmental change and their effects on coastal and marine ecosystems, Oct. 2015.

Appendices

Appendix A

Figures 1 and 2 present the solutions generated by VAE-GAN, while the solutions generated by TANGAN are compiled in Figures 3 and 4.

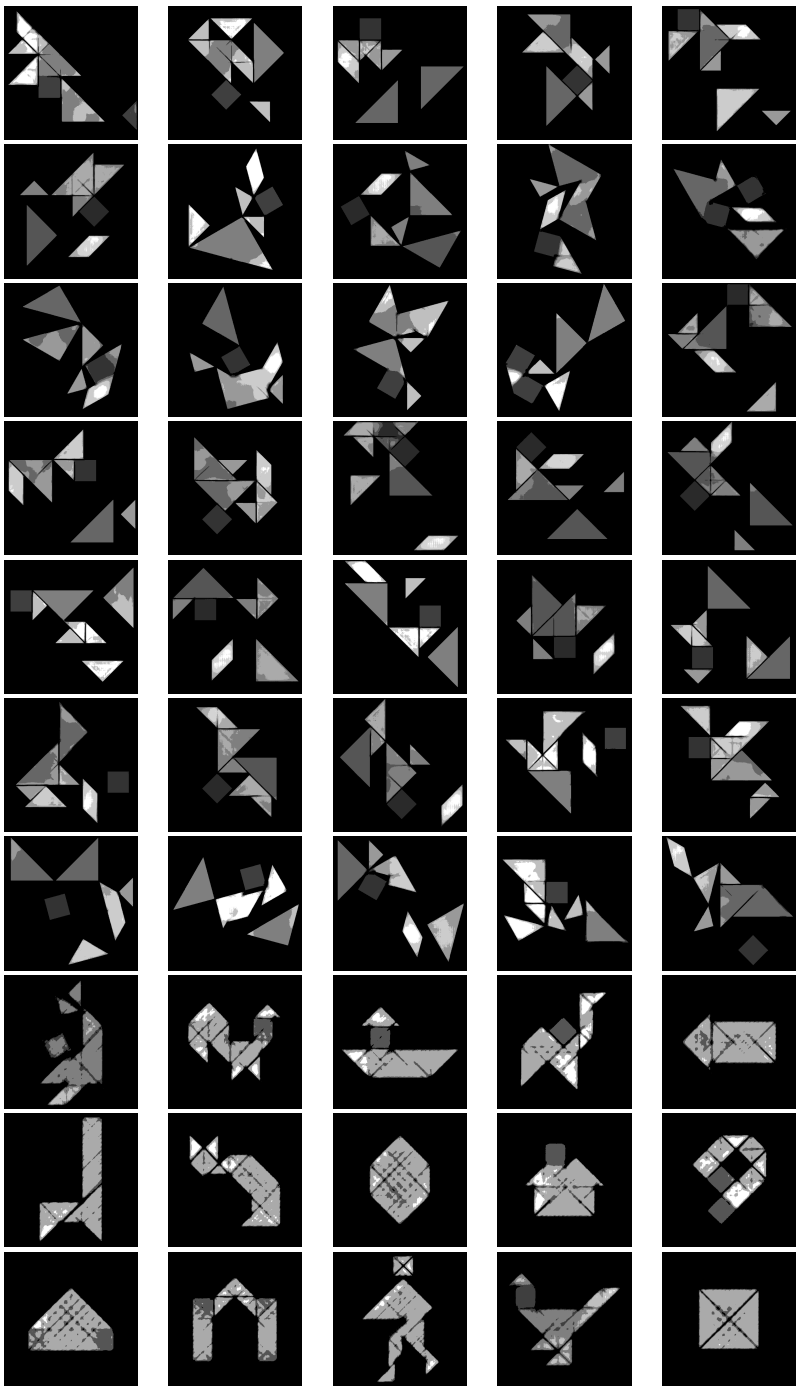


Figure 1: Solutions generated by VAEGAN (part 1 of 2).

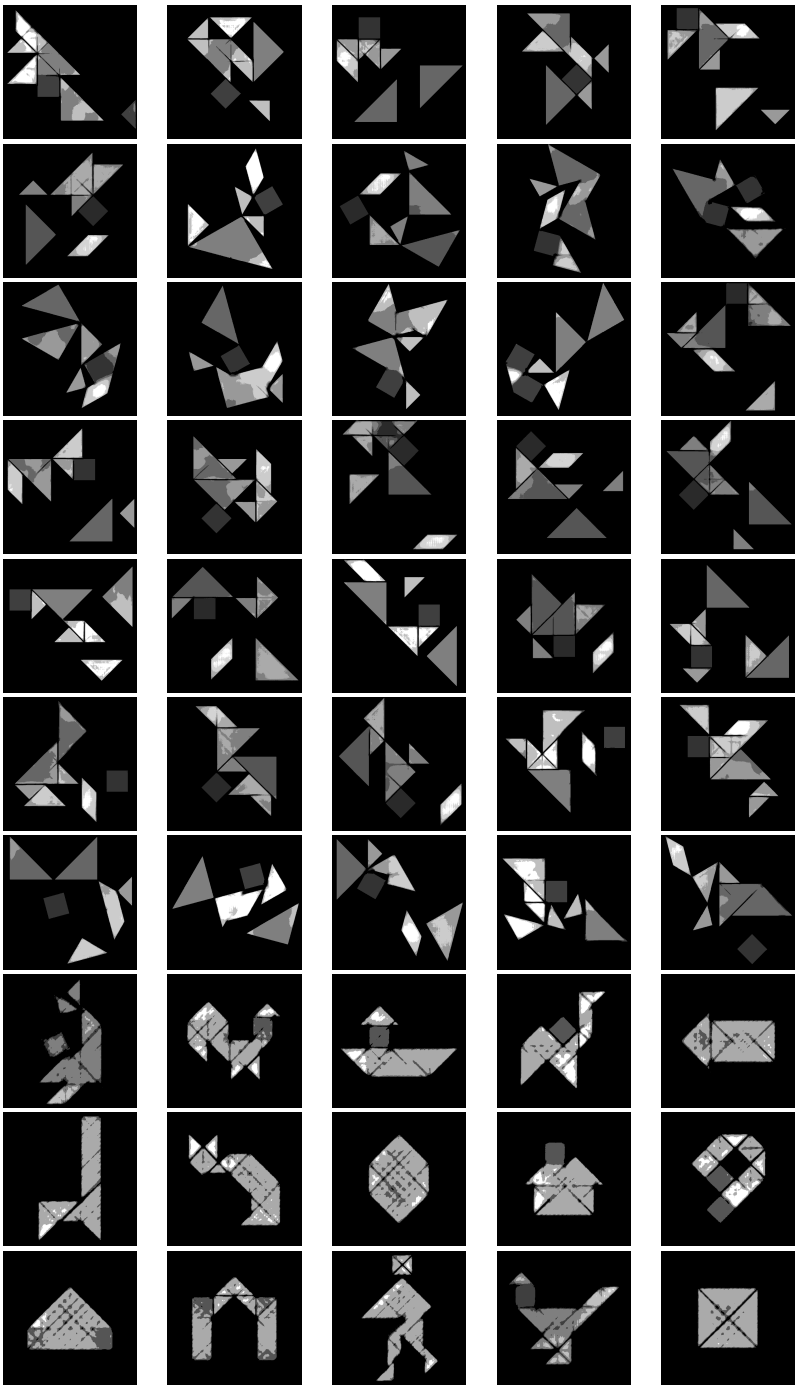


Figure 2: Solutions generated by VAEGAN (part 2 of 2).

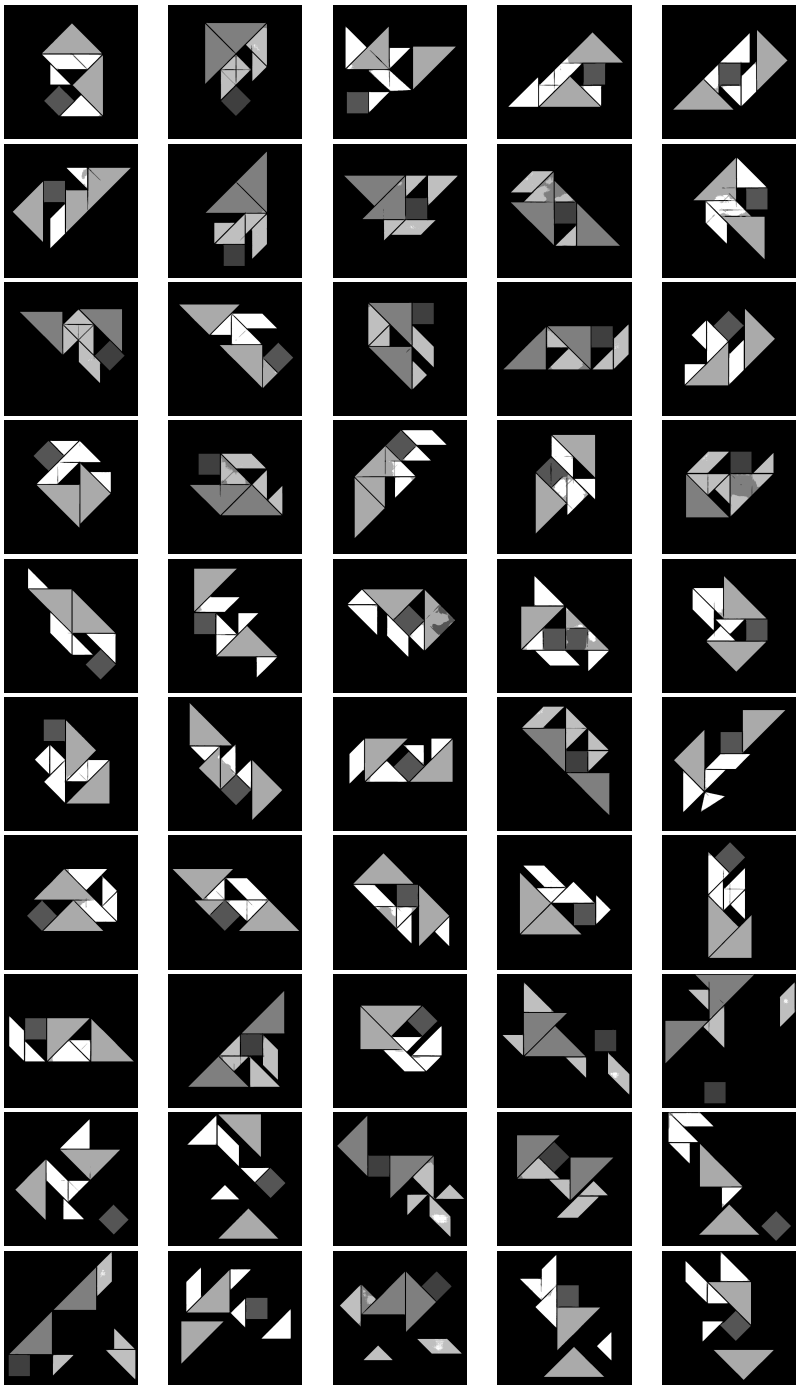


Figure 3: Solutions generated by TANGAN (part 1 of 2).

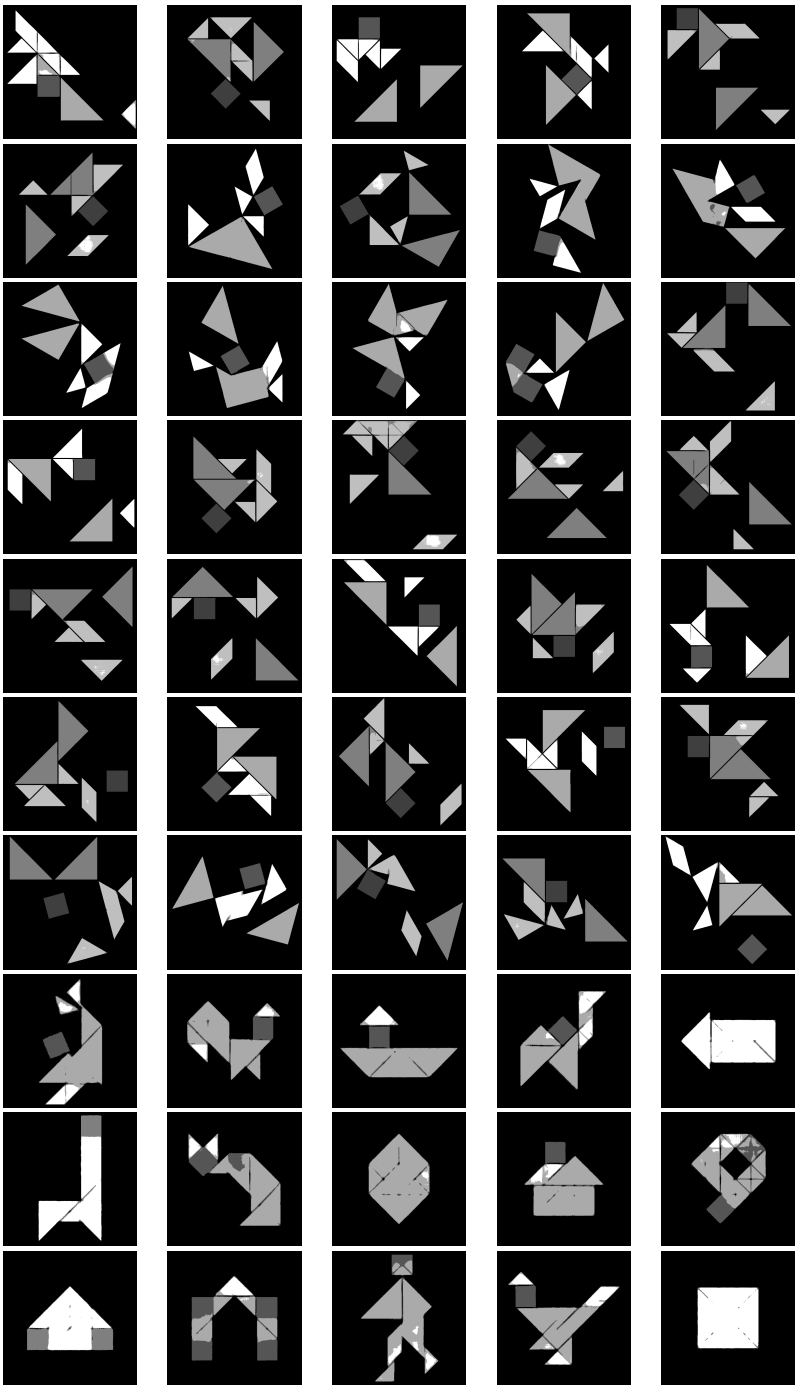


Figure 4: Solutions generated by TANGAN (part 2 of 2).

Appendix B

This appendix presents detailed summaries of the deep learning networks used throughout the present dissertation. Tables 1 and 2 present the CAE architecture from Chapter 7. Tables 3 and 4 present the VAE architecture from Chapter 7. Table 5 presents the U-Net architecture from Chapter 7. Table 6 presents the discriminator of VAE- GAN architecture from Chapter 7. Finally, Tables 7, 8 and 9 describe the generator, while Table 10 summarizes the discriminator of TANGAN from Chapter 8.

Type	Shape	Params #
InputLayer	(256, 256, 1)	0
Conv2D	(256, 256, 48)	5856
BatchNormalization	(256, 256, 48)	192
LeakyReLU	(256, 256, 48)	0
Conv2D	(128, 128, 48)	186672
Dropout	(128, 128, 48)	0
BatchNormalization	(128, 128, 48)	192
LeakyReLU	(128, 128, 48)	0
Conv2D	(64, 64, 48)	186672
Dropout	(64, 64, 48)	0
BatchNormalization	(64, 64, 48)	192
LeakyReLU	(64, 64, 48)	0
Conv2D	(32, 32, 48)	112944
Dropout	(32, 32, 48)	0
BatchNormalization	(32, 32, 48)	192
LeakyReLU	(32, 32, 48)	0
Conv2D	(16, 16, 48)	112944
Dropout	(16, 16, 48)	0
BatchNormalization	(16, 16, 48)	192
LeakyReLU	(16, 16, 48)	0
Conv2D	(8, 8, 48)	57648
Dropout	(8, 8, 48)	0
BatchNormalization	(8, 8, 48)	192
LeakyReLU	(8, 8, 48)	0
Conv2D	(4, 4, 48)	57648
Dropout	(4, 4, 48)	0
BatchNormalization	(4, 4, 48)	192
LeakyReLU	(4, 4, 48)	0

Table 1: Summary of CAE architecture (part 1 of 2).

Type	Shape	Params #
Conv2D	(2, 2, 48)	20784
BatchNormalization	(2, 2, 48)	192
LeakyReLU	(2, 2, 48)	0
Conv2DTranspose	(4, 4, 48)	57648
BatchNormalization	(4, 4, 48)	192
LeakyReLU	(4, 4, 48)	0
Concatenate	(4, 4, 96)	0
Conv2DTranspose	(8, 8, 48)	225840
Dropout	(8, 8, 48)	0
BatchNormalization	(8, 8, 48)	192
LeakyReLU	(8, 8, 48)	0
Concatenate	(8, 8, 96)	0
Conv2DTranspose	(16, 16, 48)	225840
Dropout	(16, 16, 48)	0
BatchNormalization	(16, 16, 48)	192
LeakyReLU	(16, 16, 48)	0
Concatenate	(16, 16, 96)	0
Conv2DTranspose	(32, 32, 48)	373296
Dropout	(32, 32, 48)	0
BatchNormalization	(32, 32, 48)	192
LeakyReLU	(32, 32, 48)	0
Concatenate	(32, 32, 96)	0
Conv2DTranspose	(64, 64, 48)	373296
Dropout	(64, 64, 48)	0
BatchNormalization	(64, 64, 48)	192
LeakyReLU	(64, 64, 48)	0
Concatenate	(64, 64, 96)	0
Conv2DTranspose	(128, 128, 48)	557616
Dropout	(128, 128, 48)	0
BatchNormalization	(128, 128, 48)	192
LeakyReLU	(128, 128, 48)	0
Concatenate	(128, 128, 96)	0
Conv2DTranspose	(256, 256, 48)	557616
Dropout	(256, 256, 48)	0
BatchNormalization	(256, 256, 48)	192
LeakyReLU	(256, 256, 48)	0
Concatenate	(256, 256, 96)	0
Conv2D	(256, 256, 1)	97
BatchNormalization	(256, 256, 1)	4
Activation	(256, 256, 1)	0
Total params: 3,115,301 (11.88 MB)		
Trainable params: 3,113,859 (11.88 MB)		
Non-trainable params: 1442 (5.63 KB)		

Table 2: Summary of CAE architecture (part 2 of 2).

Type	Shape	Params #
InputLayer	(256, 256, 1)	0
Conv2D	(256, 256, 48)	5856
BatchNormalization	(256, 256, 48)	192
LeakyReLU	(256, 256, 48)	0
Conv2D	(128, 128, 48)	186672
Dropout	(128, 128, 48)	0
BatchNormalization	(128, 128, 48)	192
LeakyReLU	(128, 128, 48)	0
Conv2D	(64, 64, 48)	186672
Dropout	(64, 64, 48)	0
BatchNormalization	(64, 64, 48)	192
LeakyReLU	(64, 64, 48)	0
Conv2D	(32, 32, 48)	112944
Dropout	(32, 32, 48)	0
BatchNormalization	(32, 32, 48)	192
LeakyReLU	(32, 32, 48)	0
Conv2D	(16, 16, 48)	112944
Dropout	(16, 16, 48)	0
BatchNormalization	(16, 16, 48)	192
LeakyReLU	(16, 16, 48)	0
Conv2D	(8, 8, 48)	57648
Dropout	(8, 8, 48)	0
BatchNormalization	(8, 8, 48)	192
LeakyReLU	(8, 8, 48)	0
Conv2D	(4, 4, 48)	57648
Dropout	(4, 4, 48)	0
BatchNormalization	(4, 4, 48)	192
LeakyReLU	(4, 4, 48)	0
Conv2D	(2, 2, 48)	20784
BatchNormalization	(2, 2, 48)	192
LeakyReLU	(2, 2, 48)	0
Dense	(2, 2, 2)	98
Dense	(2, 2, 2)	98
Sampling	(2, 2, 2)	0

Table 3: Summary of VAE architecture (part 1 of 2).

Type	Shape	Params #
Conv2DTranspose	(4, 4, 48)	2448
BatchNormalization	(4, 4, 48)	192
LeakyReLU	(4, 4, 48)	0
Concatenate	(4, 4, 96)	0
Conv2DTranspose	(8, 8, 48)	225840
Dropout	(8, 8, 48)	0
BatchNormalization	(8, 8, 48)	192
LeakyReLU	(8, 8, 48)	0
Concatenate	(8, 8, 96)	0
Conv2DTranspose	(16, 16, 48)	225840
Dropout	(16, 16, 48)	0
BatchNormalization	(16, 16, 48)	192
LeakyReLU	(16, 16, 48)	0
Concatenate	(16, 16, 96)	0
Conv2DTranspose	(32, 32, 48)	373296
Dropout	(32, 32, 48)	0
BatchNormalization	(32, 32, 48)	192
LeakyReLU	(32, 32, 48)	0
Concatenate	(32, 32, 96)	0
Conv2DTranspose	(64, 64, 48)	373296
Dropout	(64, 64, 48)	0
BatchNormalization	(64, 64, 48)	192
LeakyReLU	(64, 64, 48)	0
Concatenate	(64, 64, 96)	0
Conv2DTranspose	(128, 128, 48)	557616
Dropout	(128, 128, 48)	0
BatchNormalization	(128, 128, 48)	192
LeakyReLU	(128, 128, 48)	0
Concatenate	(128, 128, 96)	0
Conv2DTranspose	(256, 256, 48)	557616
Dropout	(256, 256, 48)	0
BatchNormalization	(256, 256, 48)	192
LeakyReLU	(256, 256, 48)	0
Concatenate	(256, 256, 96)	0
Conv2D	(256, 256, 1)	97
BatchNormalization	(256, 256, 1)	4
Activation	(256, 256, 1)	0
Total params: 3,060,297 (11.67 MB)		
Trainable params: 3,058,855 (11.67 MB)		
Non-trainable params: 1442 (5.63 KB)		

Table 4: Summary of VAE architecture (part 2 of 2).

Type	Shape	Params #
InputLayer	(256, 256, 1)	0
Conv2D	(256, 256, 64)	640
BatchNormalization	(256, 256, 64)	256
Conv2D	(256, 256, 64)	36928
BatchNormalization	(256, 256, 64)	256
MaxPooling2D	(128, 128, 64)	0
Conv2D	(128, 128, 128)	73856
BatchNormalization	(128, 128, 128)	512
Conv2D	(128, 128, 128)	147584
BatchNormalization	(128, 128, 128)	512
MaxPooling2D	(64, 64, 128)	0
Conv2D	(64, 64, 256)	295168
BatchNormalization	(64, 64, 256)	1024
Conv2D	(64, 64, 256)	590080
BatchNormalization	(64, 64, 256)	1024
MaxPooling2D	(32, 32, 256)	0
Conv2D	(32, 32, 512)	1180160
Conv2D	(32, 32, 512)	2359808
UpSampling2D	(64, 64, 512)	0
Concatenate	(64, 64, 768)	0
Conv2D	(64, 64, 256)	1769728
BatchNormalization	(64, 64, 256)	1024
Conv2D	(64, 64, 256)	590080
BatchNormalization	(64, 64, 256)	1024
UpSampling2D	(128, 128, 256)	0
Concatenate	(128, 128, 384)	0
Conv2D	(128, 128, 128)	442496
BatchNormalization	(128, 128, 128)	512
Conv2D	(128, 128, 128)	147584
BatchNormalization	(128, 128, 128)	512
UpSampling2D	(256, 256, 128)	0
Concatenate	(256, 256, 192)	0
Conv2D	(256, 256, 64)	110656
BatchNormalization	(256, 256, 64)	256
Conv2D	(256, 256, 64)	36928
BatchNormalization	(256, 256, 64)	256
Conv2D	(256, 256, 1)	65
Total params: 7,788,929 (29.71 MB)		
Trainable params: 7,785,345 (29.70 MB)		
Non-trainable params: 3584 (14.00 KB)		

Table 5: Summary of U-Net architecture.

Type	Shape	Params #
InputLayer	(256, 256, 1)	0
InputLayer	(256, 256, 1)	0
Concatenate	(256, 256, 2)	0
Conv2D	(128, 128, 64)	2112
LeakyReLU	(128, 128, 64)	0
Conv2D	(64, 64, 128)	131200
BatchNormalization	(64, 64, 128)	512
LeakyReLU	(64, 64, 128)	0
Conv2D	(32, 32, 128)	262272
BatchNormalization	(32, 32, 128)	512
LeakyReLU	(32, 32, 128)	0
Conv2D	(16, 16, 256)	524544
BatchNormalization	(16, 16, 256)	1024
Conv2D	(8, 8, 256)	1048832
BatchNormalization	(8, 8, 256)	1024
LeakyReLU	(8, 8, 256)	0
Conv2D	(4, 4, 512)	2097664
BatchNormalization	(4, 4, 512)	2048
LeakyReLU	(4, 4, 512)	0
Conv2D	(4, 4, 512)	4194816
BatchNormalization	(4, 4, 512)	2048
LeakyReLU	(4, 4, 512)	0
Conv2D	(4, 4, 1)	8193
Activation	(4, 4, 1)	0
Total params: 8,276,801		
Trainable params: 8,273,217		
Non-trainable params: 3,584		

Table 6: Summary of the discriminator of VAE-GAN.

Type	Shape	Params #
InputLayer	(512, 512, 1)	0
Conv2D	(512, 512, 48)	5856
BatchNormalization	(512, 512, 48)	192
LeakyReLU	(512, 512, 48)	0
Conv2D	(256, 256, 48)	186672
Dropout	(256, 256, 48)	0
BatchNormalization	(256, 256, 48)	192
LeakyReLU	(256, 256, 48)	0
Conv2D	(128, 128, 48)	186672
Dropout	(128, 128, 48)	0
BatchNormalization	(128, 128, 48)	192
LeakyReLU	(128, 128, 48)	0
Conv2D	(64, 64, 48)	186672
Dropout	(64, 64, 48)	0
BatchNormalization	(64, 64, 48)	192
LeakyReLU	(64, 64, 48)	0
Conv2D	(32, 32, 48)	112944
Dropout	(32, 32, 48)	0
BatchNormalization	(32, 32, 48)	192
LeakyReLU	(32, 32, 48)	0
Conv2D	(16, 16, 48)	112944
Dropout	(16, 16, 48)	0

Table 7: Summary of the generator of TANGAN (part 1 of 3).

Type	Shape	Params #
BatchNormalization	(16, 16, 48)	192
LeakyReLU	(16, 16, 48)	0
Conv2D	(8, 8, 48)	57648
Dropout	(8, 8, 48)	0
BatchNormalization	(8, 8, 48)	192
LeakyReLU	(8, 8, 48)	0
Conv2D	(4, 4, 48)	57648
Dropout	(4, 4, 48)	0
BatchNormalization	(4, 4, 48)	192
LeakyReLU	(4, 4, 48)	0
Conv2D	(2, 2, 48)	20784
BatchNormalization	(2, 2, 48)	192
LeakyReLU	(2, 2, 48)	0
Dense	(2, 2, 2)	98
Dense	(2, 2, 2)	98
Sampling	(2, 2, 2)	0
Conv2DTranspose	(4, 4, 48)	2448
BatchNormalization	(4, 4, 48)	192
LeakyReLU	(4, 4, 48)	0
Concatenate	(4, 4, 96)	0
Conv2DTranspose	(8, 8, 48)	225840
Dropout	(8, 8, 48)	0
BatchNormalization	(8, 8, 48)	192
LeakyReLU	(8, 8, 48)	0
Concatenate	(8, 8, 96)	0
Conv2DTranspose	(16, 16, 48)	225840
Dropout	(16, 16, 48)	0
BatchNormalization	(16, 16, 48)	192
LeakyReLU	(16, 16, 48)	0
Concatenate	(16, 16, 96)	0

Table 8: Summary of the generator of TANGAN (part 2 of 3).

Type	Shape	Params #
Conv2DTranspose	(32, 32, 48)	373296
Dropout	(32, 32, 48)	0
BatchNormalization	(32, 32, 48)	192
LeakyReLU	(32, 32, 48)	0
Concatenate	(32, 32, 96)	0
Conv2DTranspose	(64, 64, 48)	373296
Dropout	(64, 64, 48)	0
BatchNormalization	(64, 64, 48)	192
LeakyReLU	(64, 64, 48)	0
Concatenate	(64, 64, 96)	0
Conv2DTranspose	(128, 128, 48)	557616
Dropout	(128, 128, 48)	0
BatchNormalization	(128, 128, 48)	192
LeakyReLU	(128, 128, 48)	0
Concatenate	(128, 128, 96)	0
Conv2DTranspose	(256, 256, 48)	557616
Dropout	(256, 256, 48)	0
BatchNormalization	(256, 256, 48)	192
LeakyReLU	(256, 256, 48)	0
Concatenate	(256, 256, 96)	0
Conv2DTranspose	(512, 512, 48)	557616
Dropout	(512, 512, 48)	0
BatchNormalization	(512, 512, 48)	192
LeakyReLU	(512, 512, 48)	0
Concatenate	(512, 512, 96)	0
Conv2D	(512, 512, 1)	97
BatchNormalization	(512, 512, 1)	4
Activation	(512, 512, 1)	0
Total params: 3,804,969		
Trainable params: 3,803,335		
Non-trainable params: 1,634		

Table 9: Summary of the generator of TANGAN (part 3 of 3).

Type	Shape	Params #
InputLayer	(512, 512, 1)	0
InputLayer	(512, 512, 1)	0
Concatenate	(512, 512, 2)	0
Conv2D	(256, 256, 64)	2112
LeakyReLU	(256, 256, 64)	0
Conv2D	(128, 128, 128)	131200
BatchNormalization	(128, 128, 128)	512
LeakyReLU	(128, 128, 128)	0
Conv2D	(64, 64, 256)	524544
BatchNormalization	(64, 64, 256)	1024
LeakyReLU	(64, 64, 256)	0
Dropout	(64, 64, 256)	0
Conv2D	(32, 32, 512)	2097664
BatchNormalization	(32, 32, 512)	2048
LeakyReLU	(32, 32, 512)	0
Dropout	(32, 32, 512)	0
Conv2D	(32, 32, 1024)	8389632
BatchNormalization	(32, 32, 1024)	4096
LeakyReLU	(32, 32, 1024)	0
Dropout	(32, 32, 1024)	0
GlobalAveragePooling2D	(1024)	0
Dense	(1)	1025
Total params: 11,153,857		
Trainable params: 11,150,017		
Non-trainable params: 3,840		

Table 10: Summary of the discriminator of TANGAN.