Università
Ca'Foscari
Venezia

Master's Degree in Computer Science

Thesis

A Self-Supervised Deep Metric Learning Approach for Jigsaw Puzzle Reconstruction

Supervisor

Ch. Prof. Marcello Pelillo

Co-supervisor

Dr. Alessandro Torcinovich

Graduand

Giosuè Zannini 873810

Academic Year

2022/2023

Abstract

In this study, we focus on solving jigsaw puzzles and introduce a novel approach using self-supervised deep metric learning to analyze the adjacency relationships between puzzle tiles and arrange them in the correct order. Our methodology involves constructing a Siamese Neural Network (SNN) and exploring various configurations to capture the compatibility between image tiles. Initially, we treat the task as a supervised learning problem to identify the optimal configuration for our model. Subsequently, we leverage self-supervised learning, a subtype of unsupervised learning, to enhance the model's capability without the need for labeled data. Our objective is to train the network exclusively on the particular puzzle we aim to solve. This approach allows the network to grasp the intrinsic information from the specific problem. Finally, we contrast two methods: Relaxation Labeling (ReLab) and Puzzle Solving by Quadratic Programming (PSQP), and assess the performance of our model by testing it against some of the most effective hand-crafted compatibility metrics designed for puzzle solving. These evaluations are conducted on publicly available datasets, demonstrating the practicality and effectiveness of our proposed methodology.

Acknowledgements

Before delving into further discussion, I would like to take a moment to express my gratitude to the individuals who have contributed to my paper and have been by my side throughout my personal and professional journey.

First and foremost, I extend my sincere appreciation to my thesis advisor Marcello Pelillo for his unwavering support, professionalism, and invaluable guidance that has enriched my educational experience.

I am also grateful to my co-rapporteur Alessandro Torcinovich for his assistance and insightful suggestions that have greatly enhanced the project.

I owe a profound debt of gratitude to my family for their unwavering support and encouragement throughout the years, propelling me to achieve this milestone.

To all my classmates and close friends, I am immensely thankful for your contributions in making this journey meaningful and rewarding.

Lastly, I extend a special thanks to Aurora for her unwavering assistance and support throughout this journey.

Contents

Chapter 1

Introduction

In 1760, John Spilsbury, a British cartographer and engraver, is believed to have created the first jigsaw puzzle. He intended to use these puzzles as a tool for teaching European geography to the children in 18th-century London. To achieve this, he affixed a world map to a piece of wood and intricately carved each country into separate pieces. Originally termed "dissections" due to their nature of breaking down a map, the puzzles later adopted the name "jigsaw puzzle" around 1880. Recognizing the value of his invention, Spilsbury turned it into a business, producing puzzles based on various countries and regions. Initially, these puzzles were exclusive to the wealthy, including their use in teaching geography to royal children. As the popularity of puzzles grew, they transcended geography education and began featuring diverse subjects, the first version for adults came on the market around 1900.

1.1 Puzzle Solving

A jigsaw puzzle is a recreational activity that involves assembling a certain number of pieces into a combined and well-fitting unit without creating gaps between adjacent pieces. Formally speaking, this can be defined as the systematic exploration for finding the right permutation of those individual pieces.
Two-dimensional puzzles are divided into two main categories (as shown in Fig. 1.1):

- The "apictorial" in which the only information available is the shape of the pieces.
- The "pictorial" which takes into account not only the shape of the pieces but also their content.

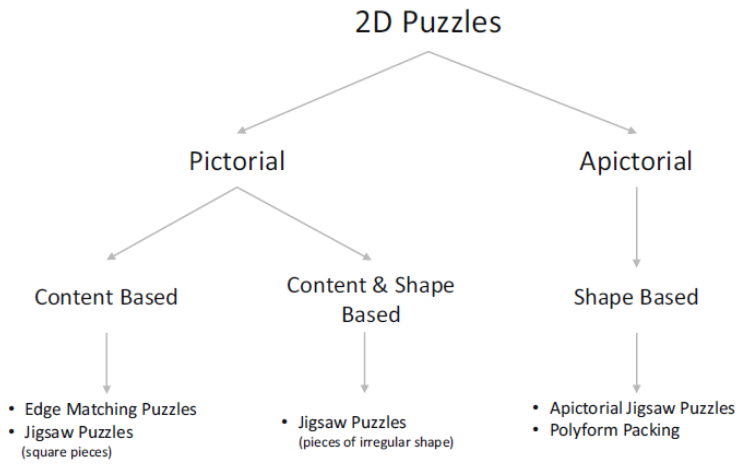


FIGURE 1.1: Different types of jigsaw puzzle.

Solving puzzles can pose significant difficulty, especially when dealing with images that depict nature, buildings, or, more broadly, repetitive designs and patterns. When the compatibility among tiles is unclear, solving the puzzle is classified as NP-complete [DD07]. In recent years, tasks similar to this have garnered significant attention and find application in various problems, such as reconstructing archaeological paintings [MK03] [Bro+08], piecing together shredded documents [ZZH07] [JOF06], as well as in speech recognition [Zha+07], image editing [CAF09], DNA modeling [MB07], and more. Additionally, numerous methods have been employed to address jigsaw puzzles. These methods include using deep neural networks to anticipate suitable positions for the tiles or utilizing manually crafted measures of compatibility along with algorithms for rearranging the puzzle [PPT18] [PPT20] [PSB11] [SHC18]. Each of these methods tackles puzzle-solving with distinct constraints.

In our research, we concentrate on the challenge of reconstructing images from small square, non-overlapping tiles, all of the same size. These tiles must be arranged in a rectangular grid, mirroring the shape and size of the original picture. Additionally, we take into account that each tile has a designated orientation. Unlike physical tiles, the pieces we work with have linear boundaries, lacking additional geometric details, which adds an extra layer of complexity to the rearrangement process. An example can be observed in the following image.

FIGURE 1.2: Example of jigsaw puzzle reconstruction.

Solving a jigsaw puzzle involves two key components: compatibility extraction and tile reordering. Compatibility extraction entails determining, for each tile, which other pieces are likely to be its neighbors based on their color and pattern information. This becomes particularly challenging when puzzle pieces share similar color information.

On the other hand, tiles reordering is the process of finding the optimal arrangement of tiles so that the resulting configuration closely resembles the original image. The primary challenge in tile reordering is the exponential growth in potential solutions as the number of tiles increases.

1.2 A different approach for Puzzle Solving

Currently, the majority of puzzle-solving methods rely on finely crafted compatibility measures to accurately predict whether two tiles are adjacent or not. Our objective is to shift towards learning compatibility through the utilization of deep neural networks' feature extraction capabilities. The neural network's task is to discover similar embeddings for neighboring pieces. These embeddings are then subjected to a similarity function that assesses whether the tiles have been classified similarly.

In addition, we intend to introduce an innovative approach to address this particular problem. The existing methods entail training a network on a large set of puzzles and then utilizing the same network to solve new puzzles. However, we believe this approach is unsuitable for the given task because most puzzles have their unique distribution. Consequently, it's challenging to create a network that can adapt to unforeseen puzzles. Our proposed method, on the other hand, entails training the network on the individual puzzle it is meant to solve. This involves using a self-supervised approach to capture the specific distribution of the current puzzle, offering a more effective solution. By doing so, we eliminate concerns about the network's generalization, focusing instead on enabling the network to learn the specific patterns within the given puzzle and understand its distribution for effective problem-solving.

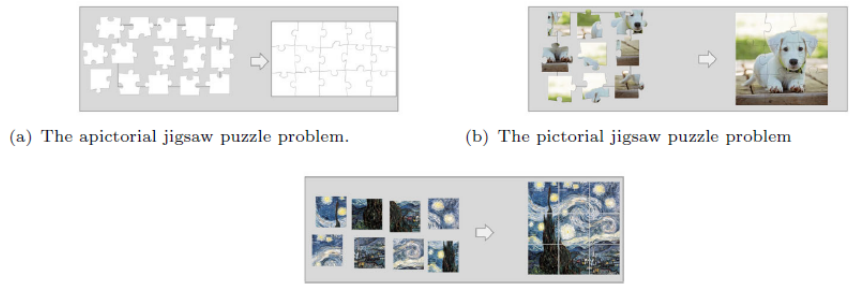
Our approach involves three main steps. First, a deep neural network categorizes each tile in the image, classifying neighboring pieces in a way that enhances their similarity. Second, the embeddings of each tile are paired and input into a correlation similarity function constructing a compatibility matrix, which determines the actual compatibility of the tiles. Notably, our proposed metric avoids relying on manually crafted measures for evaluating the compatibility of adjacent pieces. Finally, a solver is employed to identify the best permutation of tiles, we employ two established solvers from the existing literature:

PSQP by Andalo et al. [ATG16] and relaxation labeling by Khoroshiltseva et al. [Kho+21]. To evaluate the effectiveness of our approach, we conduct tests on publicly available datasets (MIT and McGill). We then compare the performance of our method with other existing techniques in the literature.

Chapter 2

Related Work

Freeman and Garder [FG64] initiated the computational solution to jigsaw puzzles in 1964, marking the first attempt. Subsequently, other methods for solving apictorial puzzles were introduced [BW89] [GMB02] [SS87]. In apictorial puzzles, matching is based solely on shape, while in pictorial puzzles, various types of pieces, including identical rectangular, traditional, and irregularly shaped pieces, are considered, with the primary matching characteristic being chromatic information, here are a few instances of various puzzle types.



(a) The apictorial jigsaw puzzle problem.

(b) The pictorial jigsaw puzzle problem

(c) The square piece jigsaw puzzle problem

FIGURE 2.1: Different jigsaw puzzle problems.

Kosiba et al. [Kos+94] pioneered a method that integrated both jigsaw shape and image information, successfully solving puzzles of 54 pieces using a greedy strategy. This method computed adjacent tiles' compatibility by considering color samples along the edges. Following this, similar methods were proposed [NDH08] [SE06] [YS03], and the research focus shifted towards color-based square-tiles puzzle solvers.

Cho et al. [CAF10] in 2010, introduced a probabilistic pictorial puzzle solver which obtained an approximate reconstruction of the original image using graphical models and a probabilistic function, requiring prior knowledge about the original image layout. To accomplish this, they employed two approaches: firstly, estimating a low-resolution image from a subset of tiles to provide local evidence in the graphical model, and secondly, having the user accurately fixate on certain tiles, referred to as anchors. While this method successfully solved puzzles of up to 432 pieces, it necessitated some level of user intervention.

Pomeranz et al. [PSB11] presented a fully automated square jigsaw puzzle solver in 2010, handling puzzles of up to 3000 pieces. Their approach involves a stepwise process: initially, employing a compatibility function to gauge the affinity between every pair of tiles. Subsequently, it incorporates three modules: positioning, segmentation, and translation. The positioning module strategically places all the tiles on the grid following predetermined logic and considering randomly selected seeds. The segmentation module utilizes the best buddies metric to identify regions that are correctly assembled. Finally, the translation module relocates both regions and tiles on the board, aiming to reconstruct a more optimal solution.

Gallagher et al. [Gal12] proposed a general solution working on square pieces without considering piece orientation or puzzle dimension. It utilized a new compatibility measure based on expected smoothness in gradient distributions across boundaries, allowing for reassembly puzzles.

Sholomon et al. [SDN13] introduced a genetic algorithm capable of solving puzzles up to 22834 pieces, considering tile rotation and puzzle dimensions.

In the field of literature, alternative puzzle-solving strategies have been explored, including those proposed by Andalo et al. [ATG16] and Khoroshiltseva et al.[Kho+21]. Andalo et al. [ATG16] simplified the puzzle-solving problem by framing it as the maximization of a constrained quadratic function, a task that

can be addressed through the gradient ascent approach [Ros60]. The suggested deterministic approach can successfully tackle puzzles comprising up to 3300 pieces, including scenarios with arbitrarily identical rectangular pieces.
Furthermore, Khoroshiltseva et al. [Kho+21] introduced an innovative method leveraging previously mentioned compatibility measures. This method reframes the puzzle-solving task as a consistent labeling problem, involving the maximization of a quadratic function over a probability space. This can be addressed using standard relaxation labeling algorithms.
Sholomon et al. [SDN16] proposed a different approach to compatibility extraction, introducing the first deep neural network capable of predicting with high precision whether two puzzle tiles are neighboring, although they trained their network in a supervised manner.

Chapter 3

Methodology

In this section, we aim to elucidate the employed architecture and outline the workflow undertaken to attain the model used in chapter 4.

3.1 Architecture

In this thesis, we opt to employ a Siamese Neural Network (SNN) to address the problem at hand. Siamese Neural Networks, introduced in 1994, represent an artificial neural network architecture where two identical neural networks, initially perceptrons, evaluate the similarity between two elements [Bro+93]. This architecture is particularly well-suited for scenarios where learning similarity is a critical aspect of the application. Furthermore, it has proven to be scalable and efficient. While Siamese Networks were overlooked for many years, recent advancements in artificial neural network architectures have revitalized their use, especially in the multimedia domain. In this domain, they leverage these improvements to achieve high precision in calculating the similarity between elements. The increasing popularity of Siamese networks in recent years can be attributed to their ability to learn effectively from limited data.

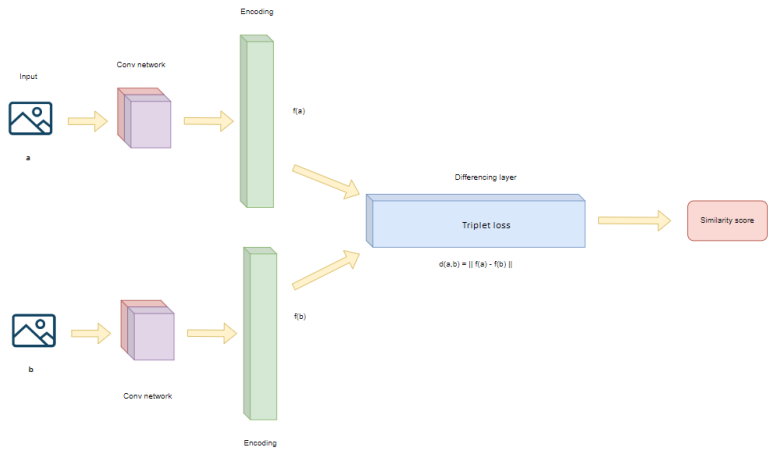


FIGURE 3.1: Siamese Neural Network structure.

Given the chosen architecture, we have to make decisions regarding the feature extractor (backbone) and the fully connected neural network (FNN) to be used. For the feature extractor, we opt for ResNet, which stands for Residual Networks. ResNet represents a category of convolutional neural network (CNN) architecture developed by He et al. [He+16]. This architecture was introduced to address challenges such as vanishing and exploding gradients that commonly arise in deeper networks. The vanishing gradient problem occurs when, in a deep network, the gradients calculated by the loss function diminish to zero due to multiple applications of the chain rule. This leads to the weights not updating, resulting in a lack of learning. ResNets overcome this challenge by allowing gradients to directly flow through skip connections backward from later layers to initial filters.

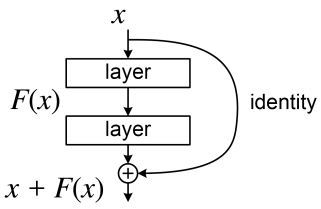


FIGURE 3.2: Residual building block.

Specifically, we choose to utilize the ResNet architecture with 18 layers for this thesis. Given that the objective is to train the network on a single puzzle and

subsequently solve it, for this reason, we conclude that a deeper model would not contribute to enhancing performance.

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

FIGURE 3.3: ResNet 18 architecture.

Instead as FNN, we decide to implement a shallow network composed of one hidden layer where its aim is to create the similarity vector with a size of 512. Regarding the choice of loss function, we decide to attempt two different functions introduced in the following subsections.

3.1.1 Loss functions

3.1.1.1 Triplet Margin Loss

Triplet margin loss is a loss function commonly used in the training of deep neural networks for tasks such as face recognition and image retrieval. It is specifically designed to enforce a desired similarity or dissimilarity between three instances (or triplets) of data points within the training set. In the context

of deep learning for similarity-based tasks, a triplet typically consists of three data points:

- Anchor: The data point for which we want to learn a representation.
- Positive: A data point that is similar or should be close to the anchor. This is typically a sample of the same class or category as the anchor.
- Negative: A data point that is dissimilar or should be far from the anchor. This is usually a sample from a different class or category than the anchor.

The goal of the triplet margin loss is to encourage the network to minimize the distance between the anchor and the positive example while maximizing the distance between the anchor and the negative example by a certain margin. The loss function is computed as follows:

$$L(a, p, n) = \max(d(a, p) - d(a, n) + margin, 0) \tag{3.1}$$

Where a is the anchor point, p is the positive point and n is the negative point. Regarding the distance function, $d(a, p)$ and $d(a, n)$ respectively indicate the distance between the anchor and positive points and the anchor and negative points in the learned embedding space. $margin$ is a hyperparameter that represents the desired minimum difference between the distances of positive and negative pairs.

The loss is zero when the distances $d(a, p)$ and $d(a, n)$ satisfy the margin requirement. If this condition is not met, the loss encourages the network to adjust its parameters to meet the margin criteria.

This loss helps in learning a meaningful and compact representation of the data in the embedding space, where similar instances are close together and dissimilar instances are separated by a margin.

3.1.1.2 Contrastive Loss

The contrastive loss was first introduced in 2005 by Yann Le Cunn et al. [HCL06] is a type of loss function particularly used in the context of learning representations or embeddings. The basic idea behind contrastive loss is to encourage the model to minimize the distance between similar examples in the embedding space while maximizing the distance between dissimilar examples. This is achieved by defining a function that penalizes the model when the distance between similar examples is too large and rewards the model when the distance between dissimilar examples is large. The loss function is computed as follows:

$$L_i = -\log \frac{\exp(z_i^\top z_i'/\tau)}{\sum_{j=0}^{K} \exp(z_i^\top z_j'/\tau)} \tag{3.2}$$

Where all vectors z are extracted using SNN which acts as a feature extractor that maps the images from pixel space to a hypersphere space.
To use the same taxonomy of Triplet Loss Function in this case the anchor is covered by z_i, the positive is covered by z_i' and negatives are covered by $\forall j = 0, \ldots, K \mid j \neq i : z_j'$. The sum in the denominator is computed over one positive and K negative pairs in the same minibatch, intuitively this can be understood as a non-parametric version of $(K+1)$-way softmax classification of z_i to the corresponding z_i'. The hyper-parameter τ is called temperature, it plays an important role in controlling the strength of penalties on hard negative samples (with lower temperature, the contrastive loss tends to punish more on the hard negative samples).

3.2 Workflow

In this section, we will walk through the sequential steps leading to the model utilized in the chapter 4. Along this trajectory, our emphasis has been on a specific image, depicted below, to facilitate the execution of numerous tests within

a condensed time frame.

FIGURE 3.4: Image belonging to MIT dataset.

In order to use this image as a jigsaw puzzle, we have to divide it into numerous pieces, each of which has the same size and the correct orientation, in order to match the c-type puzzle shown in figure 2.1. Furthermore, we decide to try different sizes, from 2×2 to 5×6, where we decide to focus our efforts. Moreover, we opt to cut the image in order to remove constant tiles being that they introduce confusion during the network training process. Below the above image is transformed into a jigsaw puzzle of dimension 5×6 without constant tiles.

FIGURE 3.5: Image from MIT dataset transformed in a jigsaw puzzle problem.

In devising the optimal approach to address this issue, our initial step involves utilizing the ground truth. In this scenario, each tile serves as an anchor, positive or negative, as part of the evaluation process. In addition, the network is trained from scratch using ground truth, as mentioned above, to generate an excellent feature extractor. As the loss function we decide to use during this phase the triplet margin loss is explained in subsection 3.1.1.1.

3.2.1 Baseline approach

In our initial approach, we input a tile of a puzzle into the neural network, and it produces an embedding. The objective is to generate similar embeddings for adjacent tiles. To achieve this, we replace the input layer of FNN with a modified version. This modified layer takes input from the last layer of the backbone, along with a positional encoding representing the tile's position. This positional encoding is treated as a vector composed of 32 floating point. To be clearer, we would in this thesis identify positional encoding with integer values, 1 or -1 corresponds to the left or right position in a horizontal pair, while 2 or -2 represents the top or bottom position in a vertical pair.

The aim is to input the entire set of tiles into the network, each with its corresponding positional encoding. This process results in an output embedding for every combination of tile and positional encoding. Subsequently, to evaluate horizontal and vertical compatibility, we assess the similarity of these embeddings using the Euclidean distance. To achieve this, we consider pairs of embeddings—one from the set forwarded with position value 1 (for horizontal compatibility) or 2 (for vertical compatibility), and the other from the set forwarded with position -1 (for horizontal compatibility) or -2 (for vertical compatibility). Finally, we compute a distance matrix, aiming for low distances between embeddings of adjacent tiles. To make sure that the network does not confuse learning both horizontal and vertical compatibilities at the same time, we decided to define two equal networks, so we were able to divide the learning for the two compatibilities.

To accomplish this objective during the training phase, we implement the triplet margin loss method, as detailed in section 3.1.1.1. As already explained, a, p, and n refer to the anchor, positive, and negative examples, respectively. The selection of these three examples is crucial.

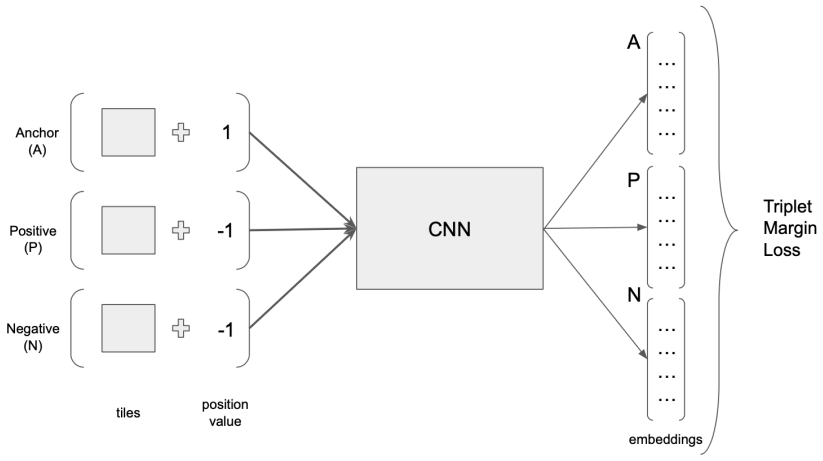


FIGURE 3.6: Example of pipeline for horizontal compatibility.

Once a puzzle is chosen, our approach involves randomly selecting a tile to serve as the anchor example. Depending on whether we are investigating horizontal or vertical compatibility, we pick the tile immediately to the right or at the bottom of the anchor as the positive example. The negative example is chosen randomly from the remaining tiles. For positive and negative examples in horizontal compatibility, we use -1, while in vertical compatibility, we use -2. Each of these examples, along with its assigned position value, is processed through the network, producing embeddings that are then used to calculate the triplet loss. It's important to note that this method assumes knowledge of the pair-wise placement of tiles in the training set. As mentioned earlier, we opt to utilize the specified image illustrated in figure 3.4 from the MIT dataset for both training and validation purposes.

In summary, the entire process begins with two identical neural networks, one used for horizontal and the other used for vertical compatibility. These networks take as input a tile and a corresponding position value. The Convolutional Neural Network (CNN) then extracts features from the image, combines them with the provided positional encoding, and passes the result to the fully connected layer. The output from this network is an embedding, a condensed

representation, which is used to calculate the Euclidean distance. The underlying concept is that two adjacent tiles should exhibit closeness in this embedding space.

In essence, this methodology serves as the foundational framework against which we compare subsequent approaches. Figure 3.7 illustrates an example of anchor, positive, and negative samples. Specifically, for assessing horizontal compatibility (top row), the tile serves as the anchor example and is forwarded with position value 1. The positive example is derived from the tile on the right of the anchor, with distinct position values, and the negative example is chosen at random. Similarly, for evaluating vertical compatibility, we follow a comparable process, utilizing the tile at the bottom of the anchor for a positive example.

FIGURE 3.7: Example of input for both horizontal compatibility (top row) and vertical compatibility (bottom row).

3.2.2 Removing average pooling

As a first test, we decide to train the entire network directly without making any changes to the structure and we obtain the following results:

	Horizontal	Vertical
Accuracy@1	0.760	0.500
Accuracy@2	1.000	0.625
Accuracy@3	1.000	0.708
Accuracy@4	1.000	0.792

TABLE 3.1: Top K accuracy from puzzle showed in image 3.5 using baseline approach.

Based on our findings, we observe that determining horizontal compatibility is more straightforward compared to vertical compatibility, specifically within the context of the single puzzle being analyzed. Consequently, we hypothesize that the network's backbone struggled to retain spatial information. Here, spatial information pertains to intricate details and interconnections within the spatial arrangement or distribution of features in a given context. This notion gains significance in our scenario, given that we are working with images, which inherently possess a spatial structure.

To address this, we decide to eliminate the average pooling layer at the end of the backbone, which tends to average patches of a feature map and create a down-sampled feature map, leading to a loss of spatial context. In its place, we introduce a convolution layer to reduce the number of feature maps (from 512 to 300) before feeding them into the fully connected neural network (FNN). The new results with this configuration are the following:

	Horizontal	Vertical
Accuracy@1	0.680	0.667
Accuracy@2	0.920	0.917
Accuracy@3	1.000	0.958
Accuracy@4	1.000	1.000

TABLE 3.2: Top K accuracy from puzzle showed in figure 3.5 without avg pooling.

From these results, we can deduce that our hunch was correct and that the removal of the average pooling helped maintain spatial information. This is evident in the outcomes, where we experienced a minor decrease in horizontal accuracy, but concurrently witnessed a notable improvement in vertical accuracy.

3.2.3 HPO for horizontal and vertical margin

In this subsection, we present the results of the hyper-parameter optimization (HPO) of the margin in the loss function. Additionally, considering that we have two separate networks to capture both horizontal and vertical similarities, we make the decision to employ two distinct margins. This approach is aimed at enhancing the expressiveness of the network in effectively capturing both types of similarities.

The margin within the triplet margin loss is a pivotal parameter that profoundly influences the learning trajectory of a neural network, particularly in applications such as face recognition, image retrieval, or any task involving learning similarities. Essentially, the margin sets the minimum desired gap between the distances of positive pairs (similar items) and negative pairs (dissimilar items).

The significance of the margin lies in its assurance that the embeddings of similar items (anchors and positive samples) are positioned at a minimum distance closer to each other than the embeddings of dissimilar items (anchors and negative samples). This separation within the embedding space enhances the network's ability to effectively distinguish between similar and dissimilar pairs. Additionally, the margin serves as a control parameter, providing the flexibility to adjust the required degree of dissimilarity for the network to update its weights. A larger margin enforces a more substantial gap between dissimilar pairs, potentially leading to more distinct learned embeddings.

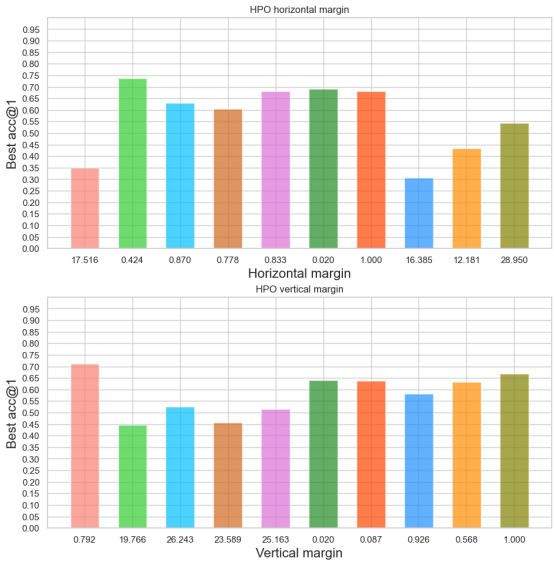


FIGURE 3.8: Margin HPO executed on puzzle showed in figure 3.5.

Thanks to this analysis we have discovered that the best margin value for both has a value less than 1. Moreover the performances of all margins with a value ≤ 1 are greater, showing that a small distance between anchor-positive and anchor-negative brings the model to achieve better performance.

3.2.4 Emphasising specific tile parts

In this subsection, we aim to highlight specific sections of each puzzle piece more prominently. As detailed in section 3.1, our architecture comprises two Siamese networks responsible for assessing horizontal and vertical compatibility, respectively. Focusing on horizontal compatibility, as previously outlined, we introduced a positional encoding in the input layer of the fully connected neural network (fnn) to signify the position of the tile, distinguishing between left and right. In addition to this positional embedding, we choose to amplify the weight assigned by the network to particular sections of the tile. By doing so, the architecture places increased importance on the features present in those specific positions during the compatibility assessment. Here's a visual representation to illustrate this concept.

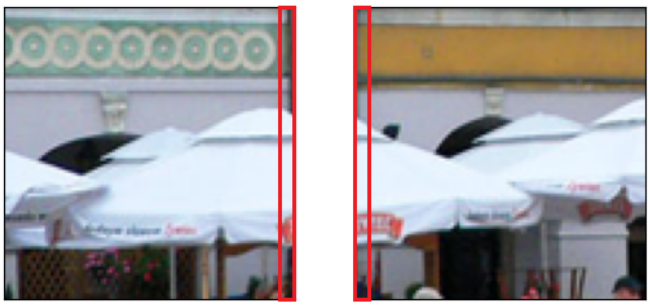


FIGURE 3.9: Example of highlighted portion in an image for horizontal compatibility, right and left respectively.

In the pursuit of this, we first introduce a parameter w, its value determined through hyper-parameter optimization (hpo). However, we later choose to incorporate an additional parameter that undergoes training alongside the neural network. By training this parameter as a net weight, we enhance its generality, making it less dependent on fine-tuning.
Below are the performances achieved using ground truth as a training set.

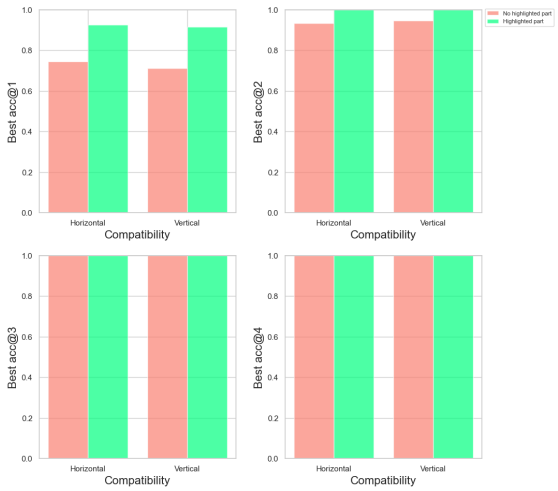


FIGURE 3.10: Bar plots that show the difference in top K accuracy between highlighted specific portion (using model defined in subsection 3.2.4) or not (using model defined in subsection 3.2.3).

3.2.5 Highlighting contours

When a person tries to identify which tile is adjacent to another, he/she typically concentrates on the outlines or contours of the tiles to discern the most likely pairing. Following this human intuition, we aim for the backbone of our network to learn, during training, the extraction of tile contours, enabling it to determine the optimal matches.

As this behavior may not be entirely certain, we opt to enhance the learning process by applying a transformation to all puzzle tiles. Here is an example.

FIGURE 3.11: Example of highlighted contours with respect to the left image.

This transformation involves extracting the existing contours, and the degree to

which these contours are emphasized is determined through hyper-parameter optimization, and the final performance is illustrated in the accompanying bar graph.

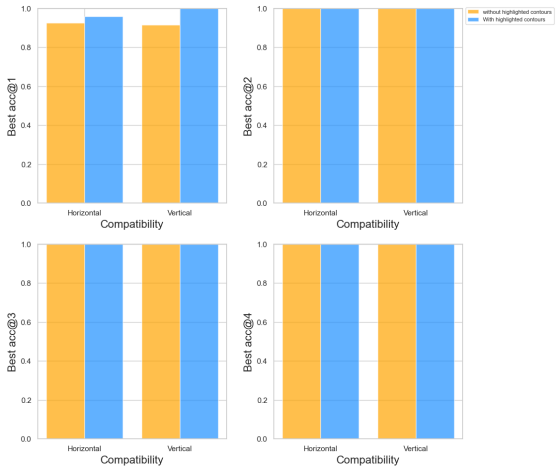


FIGURE 3.12: Bar plots that show the difference in top K accuracy between highlighted contours (using model defined in subsection 3.2.5) or not (using model defined in subsection 3.2.4).

These performances make us realize that the network was not going to extract the contours.

3.2.6 Randomize negative examples

In this subsection, we want to emphasize the importance of randomness in choosing negative examples.
In fact, choosing the right negative examples in Triplet Margin Loss is crucial for the effective training of deep neural networks. This selection directly impacts the learning process and the quality of the learned embeddings. Here are some key reasons highlighting the importance of carefully choosing negative examples:

- Effective Learning Dynamics: Selecting informative negative examples ensures that the model encounters challenging instances during training. If negative examples are too easy, the model might converge quickly but may not learn a robust representation. On the other hand, if negative examples are too hard, the learning process can become unstable. Striking the right balance by choosing negatives that are challenging but not too difficult is essential for effective learning dynamics.

- Avoiding Degenerate Solutions: Poorly chosen negative examples can lead to degenerate solutions where the model learns to minimize the loss without effectively capturing the underlying patterns in the data. The model might collapse the embeddings to trivial solutions, making them less useful for downstream tasks.

- Encouraging Discriminative Embeddings: The purpose of Triplet Margin Loss is to learn embeddings where similar items are close and dissimilar items are far apart. Carefully selecting negative examples that are semantically dissimilar to the anchor encourages the model to create more discriminative embeddings, improving the overall performance of tasks such as classification or retrieval.

- Adapting to Data Distribution: The distribution of the negative examples should reflect the diversity of the data. If the negative examples do not cover the full range of variations present in the dataset, the model may struggle to generalize to unseen examples. Ensuring a representative set of negative examples helps the model adapt to the inherent complexity and diversity of the data.

- Dynamic Selection Strategies: In some cases, employing dynamic or online strategies for negative example mining during training can be beneficial. This involves adaptively selecting negatives based on the current state of the model, which can help in focusing on more challenging examples as the model improves.

In summary, the right choice of negative examples in Triplet Margin Loss is pivotal for effective learning, preventing degenerate solutions, encouraging discriminative embeddings, and adapting to data distribution. It requires thoughtful consideration of the characteristics of the dataset and the learning dynamics of the model.
Therefore, we make the decision to select the negative randomly. By doing so, we ensure adherence to the considerations mentioned earlier, aiming to pinpoint a more favorable local minimum in the loss function.
To be sure of the choice we just made, we decide to train the model using a static negative choice. Static choice in this case means that the negative tile is the tile immediately to the right of the positive in the case of horizontal compatibility and the tile immediately below the positive in the case of vertical compatibility. Below are the results obtained.

	Random negative		Static negative	
	Horizontal	Vertical	Horizontal	Vertical
Accuracy@1	0.96	1	0.4	0.21
Accuracy@2	1	1	0.56	0.29
Accuracy@3	1	1	0.72	0.42
Accuracy@4	1	1	0.88	0.58

TABLE 3.3: Show the difference in top K accuracy between the choice to use either static or random negative.

As evident, opting for a fixed choice of the negative counterpart results in a substantial decline in performance.

3.2.7 Maintenance of spatial information

At this point, we are getting very good results with the ground truth, but convergence is slow. We therefore decide to speed up convergence by removing the tile-down sampling performed by ResNet18. To do this, we modify the backbone of the network by maintaining the size of the tiles and replacing the last conv layer with a Spatial Pyramid Pooling (SPP) [He+15].
SPP is a technique used in deep learning, particularly in the context of convolutional neural networks applied to computer vision tasks such as image classification. It is designed to handle input images of varying sizes and proportions by enabling the network to produce fixed-size feature vectors regardless of the input size.
The main idea behind Spatial Pyramid Pooling is to divide the input feature map into different regions or grids and apply pooling operations independently within each of these regions. The pooling operation is typically max pooling, which selects the maximum value from each region, but average pooling can also be used. The key innovation is that the regions are defined at multiple scales, forming a pyramid structure.
Here are the main steps of Spatial Pyramid Pooling:

- Input Feature Map: Start with a convolutional feature map obtained from the earlier layers of a CNN.
- Divide into Regions: Divide the feature map into a predefined number of levels or layers. For each level, divide the feature map into a grid of spatial bins.
- Pooling Operation: Apply pooling independently within each spatial bin at each level. The pooling can be either max pooling or average pooling.
- Concatenate: Concatenate the results from all the pooled regions at different levels into a single fixed-size vector.

- Flatten: Flatten the concatenated vector to serve as a fixed-size representation of the input feature map.

The advantage of Spatial Pyramid Pooling is that it allows the network to capture information at multiple scales, enabling it to be more robust to variations in object sizes and locations within the input image. This is particularly useful when dealing with images of different dimensions during training and testing.

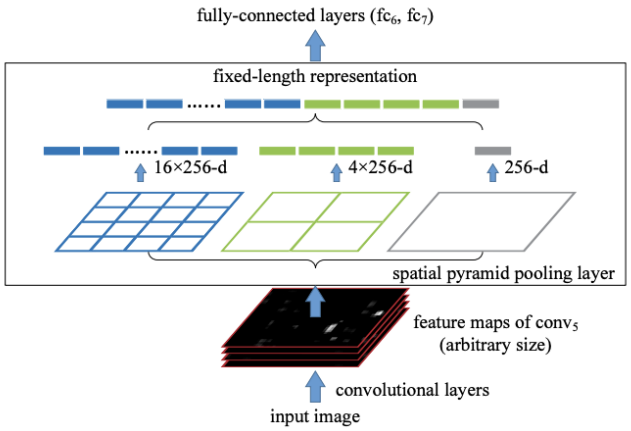


FIGURE 3.13: Spatial Pyramid Pooling.

With this expediency, we are able to achieve the same results highlighted in table 3.3 but with faster convergence of Accuracy@1.

Chapter 4

Experimental Results

In this chapter, our focus is on evaluating the performance of our finalized network, as outlined in subsection 3.2.7. Unlike earlier stages where we relied on ground truth for training, we have now transitioned to using self-supervised learning.

Self-supervised learning is a method where a model learns to predict specific aspects of its input data without explicit supervision. Unlike traditional supervised learning, which relies on labeled datasets, self-supervised learning aims to design tasks that tap into the inherent structure or information present in the data, eliminating the need for external labels.

To implement this approach, we depart from the conventional knowledge of tile placement. Instead, we opted to horizontally or vertically break each tile from the puzzle we want to solve in two pieces, creating anchor and positive pairs for our network. Notably, the negative pair, as elaborated in subsection 3.2.6, is selected randomly from among the other split tiles. This shift in methodology allowed us to leverage the intrinsic information within the data itself, contributing to the overall effectiveness of our self-supervised learning approach.

FIGURE 4.1: Example of a horizontal random split of a tile in two parts.

Furthermore, we decided to exclude constant tiles from our training set due to their tendency to cause confusion in the network training process. Constant tiles are characterized by borders with consistent color information, encompassing both monochromatic pieces and those featuring regular patterns, as depicted in the accompanying image. Moreover, our analysis disclosed that more than 50% of each image in our datasets comprised these constant tiles.

FIGURE 4.2: Examples of constant tiles, monochromatic, or with repetitive patterns.

Moreover, we have consistently employed ResNet18 as the backbone in our experiments. However, in contrast to prior trials, the network was undergoing pretraining using the ImageNet dataset before fine-tuning during the training

process. We decided to block the training after 1500 epochs being that the accuracy metric results are stable. Regarding the loss functions described in section 3.1.1, we decided to continue using the triplet margin loss, since we noticed that both lead the network to the same result.
The datasets that we decided to use for this phase are MIT and McGill.

4.1 Ablation studies

In this section, our aim is to conduct various ablation studies utilizing the self-supervised approach outlined earlier.
An ablation study is a type of experiment in scientific research, where specific components or features of a system are systematically removed or "ablated" to understand their individual contributions to the overall performance. The goal is to analyze and quantify the impact of each component on the system's behavior or performance. In the context of machine learning models, an ablation study involves training and evaluating the model with variations, where certain features, modules, or components are selectively disabled or removed. By observing how the model's performance changes in response to these modifications, researchers can gain insights into the importance of different elements in the system. They provide a systematic and rigorous way to analyze the internal workings of models and contribute to the overall understanding and improvement of machine learning systems.
Taking up this last line our goal is to systematically explore and comprehend the behavior of our model through these experiments.

4.1.1 Random split

The initial experiment we chose to conduct revolves around determining the level of randomness applied during the tile-splitting process.
To clarify, a static split occurs when the tile is precisely divided either horizontally or vertically, resulting in each piece occupying exactly 50% of the original tile's area. However, we can introduce variability by imposing a constraint that sets the minimum percentage of the area ($< 50\%$) for each split relative to the entire tile. This approach injects randomness into the network's training phase, creating a more diverse training set. Consequently, the network is exposed to a greater variety of information about the puzzle, ensuring that when the same

tile is presented multiple times, it will likely be split in different ways, contributing to a more robust learning experience.

In this experiment, we decided to show the best combined (mean between horizontal and vertical accuracy@1) accuracy@1 among all puzzles of the same dataset.

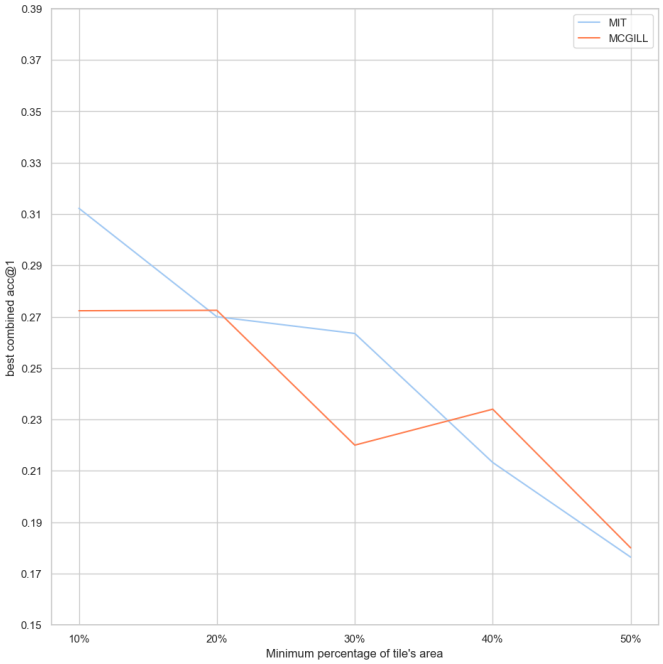


FIGURE 4.3: Ablation study for random split using the mean between horizontal and vertical acc@1 as metric.

By examining this graph, we can infer that decreasing the minimum percentage of area allowed for tile division corresponds to an increase in performance. This observation reinforces our earlier reasoning and leads us to the conclusion that the idea described previously holds true.

4.1.2 Positional encoding

Positional encoding is a technique used in the context of sequence-to-sequence models, particularly in attention-based models like Transformer, to provide information about the relative or absolute positions of tokens in a sequence. Since these models do not inherently understand the order of the input tokens, positional encoding helps inject positional information into the input embeddings. This technique was also used in the paper titled "Representing Scenes as Neural Radiance Fields for View Synthesis" [Mil+], the authors employ a method that involves transforming input coordinates into a higher-dimensional space using sinusoidal functions.
In our case, the purpose of positional encoding is to facilitate the Multilayer Perceptron (MLP) in better approximating a function with higher frequency.
We opted to employ this technique to enable the network to discern specific regions within a tile, directing its attention towards the key areas critical for task resolution, as elucidated in sub-section 3.2.1.
In this ablation, we opted to study three different types of positional encoding:

- Absolute position: use integer values.
- Randomized: use a vector composed of random values between 0 and 1.
- Sinusoidal functions: use a vector formed using sine and cosine functions as in the paper "Attention Is All You Need" [Vas+17].
  The function to define the vector is:

$$\begin{aligned} P(k, 2i) &= \sin(\frac{k}{n^{\frac{2i}{d}}}) \\ P(k, 2i+1) &= \cos(\frac{k}{n^{\frac{2i}{d}}}) \end{aligned} \tag{4.1}$$

  Where:

  - L: length of the input sequence.

- k: position of an object in the input sequence, $0 \leq k < \frac{L}{2}$.
- d: dimension of the output embedding space.
- $P(k, j)$: position function for mapping a position in the input sequence to index (k, j) of the positional matrix.
- n: set to 10000 by the authors of "Attention Is All You Need".
- i: used for mapping to column indices $0 \leq i < \frac{d}{2}$, with a single value of i maps to both sine and cosine functions.

For this thesis, we chose to set the length of each vector equal to 32.
In the above sub-section, we show the best combined (mean between horizontal and vertical accuracy@1) accuracy@1 among all puzzles of the same dataset.

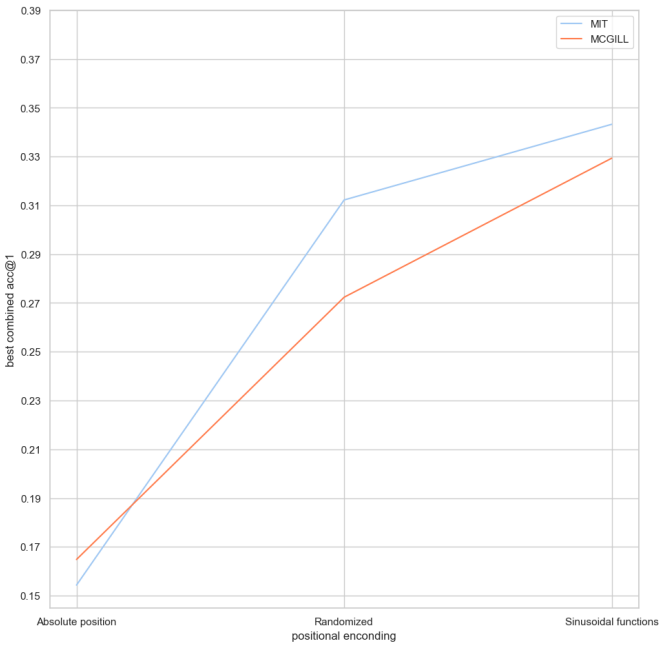


FIGURE 4.4: Ablation study for positional encoding using the mean between horizontal and vertical acc@1 as metric.

From this graph, it can be deduced that using absolute position worsens performance, while using the sine function performs slightly better, leading us to infer that as in the paper "Representing Scenes as Neural Radiance Fields for View

Synthesis," positional encoding makes it easier for the MLP to approximate a function with a higher frequency.

4.1.3 Backbone

In this final ablation, we aim to examine how the network's behavior is affected by employing a more intricate model, such as Resnet34.

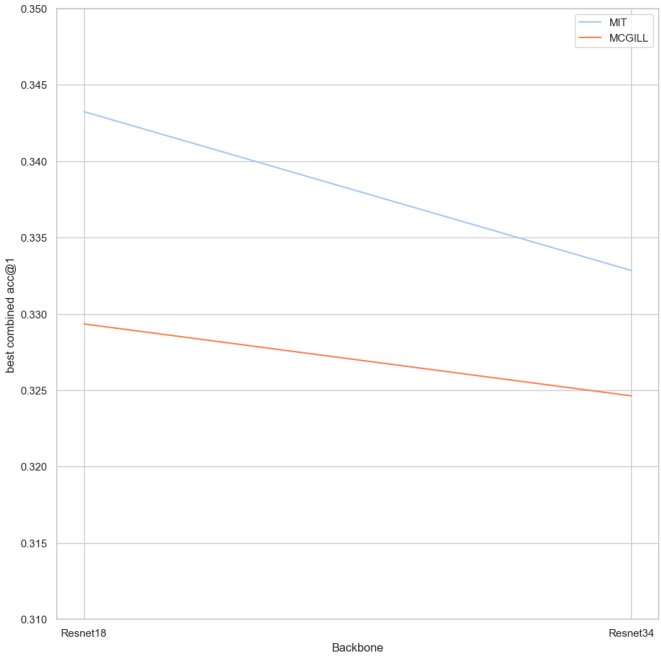


FIGURE 4.5: Ablation study for choosing backbone using the mean between horizontal and vertical acc@1 as metric.

As highlighted in the plot, utilizing a more sophisticated network does not improve performance in the specified task, aligning with our expectations in section 3.1.

4.2 Comparisons

In this section, our goal is to evaluate how well our implementation addresses the specified problem. We have assessed its performance by employing the MIT and McGill datasets as benchmarks. Firstly, we aim to compare two distinct algorithms for reconstructing the puzzle using the compatibility matrix determined by our model. Subsequently, we have chosen to further evaluate our solution by comparing it with some of the most effective hand-crafted compatibility metrics available. Before starting we want to remark as said in the previous section, that this is a self-supervised approach, and to implement this approach we split each tile randomly into two parts in order to capture the intrinsic information from the specific problem in object. In summary, we opted to employ the identical model applied in subsection 4.1. However, we incorporated insights gained from the ablation study by setting the minimum area percentage to 10%. For positional encoding, we chose the sinusoidal function due to its superior performance in the study. Additionally, we retained ResNet18 as the backbone for our model. This is our final model determined during this thesis. Similar to previous practices, we uphold the threshold to halt training after 1500 epochs, given the stability observed in the accuracy metric results. These results showcase the outcomes derived from the network described earlier. Prior to delving into the details, we aim to present the $acc@k$ values for our final model, as determined through the examination of the two datasets.

	Accuracy@1	Accuracy@2	Accuracy@3	Accuracy@4
MIT	44.53%	48.45%	65.12%	75.28%
McGill	43.32%	46.24%	58.76%	73.92%

TABLE 4.1: Results of our model using $acc@k$ as metric

4.2.1 Relaxation Labeling vs PSQP

As described above we now delve into a detailed description of these algorithms, aiming to assess and identify which one better aligns with our specific requirements.

4.2.1.1 Relaxation Labeling

The initial algorithm is Relaxation Labeling, where a consistent labeling problem involves a set of n objects $B = 1, \ldots, n$ and a set of m labels $\Lambda = 1, \ldots, m$. The objective in such problems is to assign a label from Λ to each object in B. At the start, the algorithm initializes with an initial m-dimensional probability vector for each object $i \in B$:

$$\begin{aligned} &p_{i\lambda}^{(0)} = (p_{i1}^{(0)}, \ldots, p_{im}^{(0)})^\top \\ &\text{with } p_{i\lambda}^{(0)} \geq 0 \text{ and } \sum_{\lambda} p_{i\lambda}^{(0)} = 1 \end{aligned} \tag{4.2}$$

Every $p_{i\lambda}^{(0)}$ signifies the probability distribution for object b_i being labeled with λ at time 0. For each object in B, there exists one associated probability distribution. Concatenating all these probability vectors $p_1^{(0)}, \ldots, p_n^{(0)}$ results in the initial weighted labeling assignments $p^{(0)} \in \mathbb{R}^{nm}$. Alternatively, the assignment of labels can be conceptualized as a matrix p within the space IK, defined as:

$$IK = \Lambda^m = \Lambda \times \cdots \times \Lambda \tag{4.3}$$

Where Λ is the standard simplex of $\mathbb{R}^n$.

$$\Lambda^m = \left\{ p \in \mathbb{R}^m \mid p_{i\lambda} \geq 0, \lambda \in \Lambda \ \wedge \ \sum_{\lambda=1}^{m} p_{i\lambda} = 1, \quad i = 1, \ldots, n \right\} \tag{4.4}$$

Each vertex of IK is called an unambiguous labeling assignment [Pel97].
The initial labeling assignment can be grounded in local measures, which capture pertinent features of individual isolated objects. Additionally, contextual information is employed to enhance the accuracy of weak label assignments derived from local measures. This contextual information is represented by a matrix of compatibility coefficients with dimensions $n^2 \times m^2$:

$$R = r_{ij\lambda\mu} \tag{4.5}$$

Where $r_{ij\lambda,\mu}$ rappresents the compatibility between the hypothesis b_i has label λ and b_j has label μ. We can measure the support given by the context to the hypothesis b_i has label λ at time t as [HZ83]:

$$q_{i\lambda}(t) = \sum_{j} \sum_{\mu} r_{ij\lambda\mu} p_{j\mu}(t) \tag{4.6}$$

By properly weighting and combining the support of all labels at all objects, we can also quantify the average support of the assignment, or the so-called average local consistency [HZ83][Kho+21].

$$A(p) = \sum_{i,j} \sum_{\lambda,\mu} r_{ij\lambda\mu} p_{i\lambda} p_{j\mu} \tag{4.7}$$

A labeling assignment p is consistent if $\forall v \in \Lambda^{n \times m}$

$$\sum_{\lambda}^{m} p_{i\lambda} q_{i\lambda} \geq \sum_{\lambda}^{m} v_{i\lambda} q_{i\lambda} \tag{4.8}$$

And if the matrix R is symmetric, then any local maximizer $p \in \Lambda^{n \times m}$ of $A(p)$ is consistent [HZ83].
At each iteration step, the algorithm updates the probability vectors using the heuristic formula, provided by Rosenfeld, Hummel, and Zucker in 1976 [RHZ76]

[Pel97]:

$$p_{i\lambda}(t+1) = \frac{p_{i\lambda}(t)q_{i\lambda}(t)}{\sum_{\mu} p_{i\mu}(t)q_{i\mu}(t)} \quad \forall i, \lambda \tag{4.9}$$

The advantages of this updating rule include the absence of the need to select a step size. Furthermore, the non-negativity and symmetry conditions imposed on the matrix R ensure convergence to a consistent labeling [RHZ76].

To cast the puzzle-solving task as a consistent labeling problem, we designate the set of objects B as the puzzle tiles, and the labels Λ as all the potential positions in the grid. The objective is for each tile in B to be assigned a distinct position from Λ, ensuring that the average local consistency is maximized.

In our situation we deviate from "ideal" compatibility, the algorithm could converge to an inaccurate permutation matrix. Despite the update rule ensuring that p is a stochastic matrix with rows summing to 1, the convergence might lead to a solution where multiple tiles are assigned to the same position, while other positions remain unassigned. This discrepancy occurs because p is not a doubly stochastic matrix.

To address this issue and facilitate convergence to a permutation matrix, we employed the Alternating Projection method [Kho+21]. This process begins after t steps, with t set to 10. Initially, it applies the update rule in equation 4.9 for t steps. Subsequently, it alternates between the update rule in equation 4.9 and the following:

$$p_{i\lambda}(t+1) = \frac{p_{i\lambda}(t)q_{i\lambda}(t)}{\sum_{j} p_{j\lambda}(t)q_{j\lambda}(t)} \quad \forall i, \lambda \tag{4.10}$$

This procedure is applicable because, in the puzzle-solving abstraction, the objects and labels are interchangeable.

4.2.1.2 PSQP

The alternative algorithm, Puzzle Solving by Quadratic Programming (PSQP), shares a similar approach with Relaxation Labeling, relying on maximizing a global compatibility function. However, PSQP introduces extra constraints and

employs the gradient projection method suggested by Rosen to address the maximization problem [Ros60].

By maximizing the global compatibility function, we aim to reach a solution that identifies the optimal arrangement of tiles, considering both the overall compatibility among tiles and a local measure that assesses how well tiles fit into adjacent positions. Similar to the previous approach, we use a 2D grid consisting of N labeled locations $1 \dots N$ and N tiles represented as $t_1 \dots t_N$. The task involves determining a new placement for each tile in a distinct location. This one-to-one relationship between locations and tiles is expressed through a permutation π of the N tiles. We organize this expression into a directed graph denoted as $G = \{V, E = E_H \cup E_V\}$, where the vertices $V = \{1, \dots, N\}$ correspond to the locations. The edge set E encompasses pairs of adjacent locations, with E_H and E_V specifically designating horizontal and vertical neighboring locations, respectively. The graph G is directed because swapping two tiles leads to a change in the overall compatibility value, signifying a directional relationship between the locations. For every pair of tiles (t_i, t_j), where $1 \leq i, j \leq N$ and $i \neq j$, we establish two local compatibility metrics, denoted as $C_{H_{i,j}} \geq 0$ and $C_{V_{i,j}} \geq 0$. These metrics signify the compatibility between the two tiles when placed in locations connected by either a horizontal edge $e \in E_H$ or a vertical edge $e \in E_V$. The global compatibility function of a permutation π is:

$$\epsilon(\pi) = \sum_{(i,j) \in E_H} C_{H\pi(i)\pi(j)} \sum_{(i,j) \in E_V} C_{V\pi(i)\pi(j)} \tag{4.11}$$

Where (i, j) represents the edge linking the neighboring locations i and j, and $\pi(i)$ denotes the tile that has been permuted to occupy location i [ATG16]. Considering all possible permutations π of N elements, the objective is to maximize the function expressed in equation 4.11.

Now, our aim is to rephrase equation 4.11 as a homogeneous quadratic function involving a square matrix.

We can use a permutation matrix to symbolize each permutation π of N elements:

$$P_{ik} = \begin{cases} 1, & \text{if } k = \pi(i) \\ 0, & \text{if } k \neq \pi(i) \end{cases} \tag{4.12}$$

Permutation matrices are a special case of doubly stochastic matrices [Sen06], non-negative matrices in which each row and each column sum up to 1. With this notation, we can rewrite the equation 4.11 as:

$$\epsilon(\pi) = \sum_{(i,j)\in E_H} (P^\top C_H P)_{ij} + \sum_{(i,j)\in E_V} (P^\top C_V P)_{ij} \tag{4.13}$$

$(P^\top CP)_{ij}$, representing the edge $e = (i, j)$, it corresponds to the element at the position i, j within the square matrix $(P^\top CP)$. It's important to note that, for each edge $e = (i, j)$, the expression $P^\top CP$ forms a homogeneous non-negative quadratic function based on the elements of the matrix P [ATG16]. If we arrange the columns $p_1, \ldots, p_N$ from the matrix P with dimensions $N \times N$ into a column vector p of size N^2, we obtain:

$$\epsilon(\pi) = \sum_{(i,j)\in E_H} (p_i^\top C_H p_j) + \sum_{(i,j)\in E_V} (p_i^\top C_V p_j)_{ij} \tag{4.14}$$

We can express this equation in the standard quadratic form $p^\top Ap$, where A is a symmetric non-negative block matrix with dimensions $N^2 \times N^2$. This matrix corresponds to the Hessian of $\epsilon(P)$.

A crucial aspect to note is that every doubly stochastic matrix adheres to N^2 inequality constraints, ensuring that the elements of the column vector p are non-negative. Additionally, there are $2N$ equality constraints, stipulating that the sum of elements in each row and each column of matrix P equals 1.

With these considerations, we are now equipped to present the problem formulation as a quadratic optimization problem and outline the corresponding algorithm for its solution. If we extend the domain of $\epsilon(P)$ for all doubly stochastic

matrices, we can reduce the problem to finding a solution for the quadratic optimization problem:

$$\begin{aligned} \max f(p) &= p^\top A p, \\ \text{s.t.} \quad P\bar{1} &= \bar{1}, \\ P^\top \bar{1} &= \bar{1}, \\ p_k &\geq 0, \quad \forall k = 1, \ldots, N^2 \end{aligned} \tag{4.15}$$

where $\bar{1}$ is an N-column vector of ones.
Given that all diagonal elements of matrix A are zero, the matrix is not positive definite nor positive semi-definite. Consequently, even if the objective function $f(p)$ is positive within the feasible set, it does not necessarily exhibit concavity. This absence of concavity implies no assurance of reaching the global maximum. However, by navigating through the constraints, we can pursue a local maximum for Equation 4.15, representing an approximation to the global maximum.

4.2.1.3 Results

Following this concise overview of the employed algorithms, it is now time to observe how they perform in the context of this specific task.
The following tables are provided to highlight the variation in accuracy when taking into account constant tiles or not. To assess accuracy in the table 4.3 essentially, even if a tile is visually similar to the correct one but placed in the wrong position, we do not categorize it as misplaced.

PSQP		Relaxation Labeling	
MIT	McGill	MIT	McGill
41.52%	40.05%	38.95%	38.47%

TABLE 4.2: Accuracy on puzzle solving with error caused by constant tiles

PSQP		Relaxation Labeling	
MIT	McGill	MIT	McGill
54.85%	51.72%	47.29%	46.66%

TABLE 4.3: Accuracy on puzzle solving without error caused by constant tiles

Referring to the results from the previous tables, it's evident that when disregarding errors stemming from constant tiles, the performance is notably strong. The two puzzle-solving algorithms exhibit nearly identical results, with PSQP showing slightly better outcomes. For a visual illustration, please refer to Figure 4.6, which provides an example of image reassembly using an image from the MIT dataset.

(A) (B)

FIGURE 4.6: Example if image reassembling with tiles of size 100x100, (A) original image, (B) our result using PSQP.

4.2.2 Hand-crafted compatibility metrics

In this section, our objective is to evaluate our model by comparing it to certain manually designed compatibility metrics. The selected methods for comparison all rely on measuring dissimilarity. An approach introduced by Cho et al. [CAF10], calculates the local horizontal compatibility between two tiles, denoted as t_i and t_j, as follows:

$$C_{Hij} \propto \exp\left(-\frac{D_{Hij}}{2\sigma_c^2}\right) \tag{4.16}$$

Where D_{Hij} is the horizontal dissimilarity between two tiles t_i and t_j and the equation is:

$$D_{Hij} = \left(\sum_{k=1}^{\top}\sum_{l=1}^{3}(|t_i(k,T,l) - t_j(k,1,l)|)^p\right)^{\frac{q}{p}} \tag{4.17}$$

where T represents the dimension of a tile, the up script of the inner summation is the number of color channels, t_i and t_j are $T \times T \times 3$ matrices representing square tiles, and the color difference is computed in the normalized color space. Instead σ_c is defined as the difference between the lowest and the second lowest D_{Hij}, for $1 \leq j \leq N$. The parameters p and q are set to 2. The vertical dissimilarity D_{Vij} and the local vertical compatibility C_{Vij} are estimated similarly.

Another approach introduced by Pomeranz et al. [PSB11], employs the same equation 4.17 with a notable distinction in the parameterization. They observed that the parameters p and q used by Cho et al. were linked to the L_2 norm, and by experimenting with different values of the norm L_p^q, they suggested using $p = \frac{3}{10}$ and $q = \frac{1}{16}$ for improved results.

The last approach was introduced by Gallagher [Gal12] and it is an alternative approach to jigsaw puzzle solving, specifically through a metric known as Mahalanobis Gradient Compatibility. This metric addresses changes in intensity gradients by considering the covariance between color channels and utilizing the Mahalanobis distance. Essentially, the goal is to ensure that the boundaries

between adjacent tiles exhibit a similar distribution of gradients to those on either side of the boundary. To implement this, one starts by defining an array of gradients, denoted as G_{iL}, consisting of T rows (corresponding to the pixel dimension of the tile) and 3 columns (representing each color channel). G_{iL} signifies the alterations in intensity along the right side of the tile t_i, positioned on the left side of the pair. The objective is to calculate the average distribution of these gradients specifically on the right side of the tile t_i.

$$\begin{aligned} \mu_{iL}(c) &= \frac{1}{T}\sum_{k=1}^{\top} G_{iL}(k,c) \\ G_{iL}(k,c) &= t_i(k,T,C) - t_i(k,T-1,c) \end{aligned} \tag{4.18}$$

In each color channel, μ_{iL} represents the average disparity in the final two columns of tile t_i. The 3×3 covariance matrix S_{iL} is derived from the gradients in G_{iL}, encapsulating the interplay among gradients near the edge of the tile across color channels. Subsequently, the horizontal dissimilarity between tiles t_i and t_j is approximated by:

$$D_{LR}(t_i,t_j) = \sum_{k=1}^{\top}(G_{ijLR}(k) - \mu_{iL})S_{iL}^{-1}(G_{ijLR}(k) - \mu_{iL})^{\top} \tag{4.19}$$

Here, $G_{ijLR}(p,c)$ denotes the gradient originating from the right side of the piece t_i and extending to the left side of piece t_j, specifically at row position k. This gradient is characterized by:

$$G_{ijLR}(k,c) = t_j(k,1,c) - t_i(k,T,c) \tag{4.20}$$

Because the assessment of the connection between pieces t_i and t_j relies on the distributions derived from the t_i side of the boundary, the dissimilarity $D_{LR}(t_i,t_j)$ is asymmetric. The horizontal symmetric dissimilarity indicates with D_{Hij} is determined as the sum between $D_{LR}(x_i,x_j)$ and $D_{RL}(x_j,x_i)$.

Finally, we convert the just obtained dissimilarities into the horizontal compatibility between tiles t_i and t_j as follows

$$C_{Hij} \propto \exp\left(-\frac{D_{Hij}}{K_{minH}(i)}\right) \tag{4.21}$$

where $K_{minH}(i)$ is the K-min value of the dissimilarity between all other pieces to piece i. As suggested by Gallagher [Gal12] we set $K = 2$. Naturally, the equation above is properly adapted to compute the vertical dissimilarity and the local vertical compatibility.

Regardless of the methods employed, after obtaining horizontal and vertical compatibilities for each tile, we reset the compatibility values to zero for all non-best-buddy matches and assigned a value of 1 for the compatibility of any two best buddies. This approach, known as the "best buddies concept", was introduced by Pomeranz et al. [PSB11].

	With constant		Without constant	
Method	MIT	McGill	MIT	McGill
Ours	41.52%	40.05%	54.85%	51.72%
Pomeranz et al. [PSB11]	84.33%	82.50%	85.27%	86.05%
Gallagher [Gal12]	90.83%	90.21%	92.12%	91.38%

TABLE 4.4: Accuracy of various methods, considering both scenarios with and without errors induced by constant tiles.

In Table 4.4, we conduct a comparative analysis between our proposed method, which utilizes PSQP for puzzle reconstruction, and several manually designed compatibility metrics discussed in this section. It becomes clear that when accounting for errors stemming from constant tiles, our method notably differs in performance from the other metrics. Even after removing constant tiles, the

disparity lessens but remains significant. Consequently, our approach still falls short of competing with handcrafted methods.

Chapter 5

Conclusions

This thesis focuses on the challenge of reconstructing images from square tiles that do not overlap, arranging them in a rectangular grid matching the original image's shape and size. We specifically deal with tiles featuring linear boundaries and set orientations. While existing literature covers various approaches to puzzle solving, most methods either directly tackle puzzle solving by searching for optimal permutations or utilize manually crafted compatibility measures in conjunction with advanced solvers. Only a limited number of works address puzzle solving by learning compatibility in a supervised manner. In our research, we developed a neural network that operates in a self-supervised environment and effectively extracts compatibility training. The network in object is trained only on the individual puzzle it is meant to solve. Although our results are promising, they still fall short of matching the performance of the top hand-crafted compatibility measures.

At the outset of this detailed exploration, we selected a specific architectural approach to address the problem. Following this initial decision, we delved into finding the optimal configuration for the chosen network. Consequently, we opted to operate within a supervised framework. This method allowed us to enhance the performance of our approach significantly.

Following this initial investigation, we transitioned to a self-supervised approach to direct our focus toward the core problem. During this phase, we conducted ablation studies to thoroughly examine and measure the influence

of various components on the system's behavior. These analyses enabled us to implement minor adjustments to our model.
In our final step, we employed our model to compare the performance of two solvers for image reassembly: relaxation labeling and PSQP. We discovered that PSQP outperforms relaxation labeling in this context. Additionally, as a concluding evaluation, we compared our model to specific manually designed compatibility metrics.
Presently, our solution hasn't quite reached the pinnacle of performance seen in the finest hand-crafted compatibility metrics. Nevertheless, it delivers quite satisfactory results. Our primary challenge lies in handling constant tiles. We've observed that excluding errors caused by these tiles leads to notably higher outcomes. Moreover, we introduced a model capable of extracting tile compatibility for jigsaw puzzle solving. This model operates within a self-supervised framework, learning intricate patterns only in the puzzle to be solved and discerning their distribution to facilitate effective problem-solving.

Future works. As demonstrated, once we successfully tackle the management of constant tiles, it becomes feasible to develop a self-supervised neural network-based compatibility metric for jigsaw puzzle solving. Such a metric has the potential to surpass all existing hand-crafted metrics documented in the literature. Here are some possible improvements:

- Training the deep neural network to be agnostic to constant tiles.
- Replace the backbone of the network with a modern architecture such as the Vision Transformer.

Bibliography

[ATG16] Fernanda A Andalo, Gabriel Taubin, and Siome Goldenstein. "PSQP: Puzzle solving by quadratic programming". In: *IEEE transactions on pattern analysis and machine intelligence* 39.2 (2016), pp. 385–396.

[Bro+08] Benedict J Brown et al. "A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings". In: *ACM transactions on graphics (TOG)* 27.3 (2008), pp. 1–9.

[Bro+93] Jane Bromley et al. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems* 6 (1993).

[BW89] BG Burdea and Haim J Wolfson. "Solving jigsaw puzzles by a robot". In: *IEEE Transactions on robotics and automation* 5.6 (1989), pp. 752–764.

[CAF09] Taeg Sang Cho, Shai Avidan, and William T Freeman. "The patch transform". In: *IEEE transactions on pattern analysis and machine intelligence* 32.8 (2009), pp. 1489–1501.

[CAF10] Taeg Sang Cho, Shai Avidan, and William T Freeman. "A probabilistic image jigsaw puzzle solver". In: *2010 IEEE Computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 183–190.

[DD07] Erik D Demaine and Martin L Demaine. "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity". In: *Graphs and Combinatorics* 23.Suppl 1 (2007), pp. 195–208.

[FG64] Herbert Freeman and L Garder. "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition". In: *IEEE Transactions on Electronic Computers* 2 (1964), pp. 118–127.

[Gal12] Andrew C Gallagher. "Jigsaw puzzles with pieces of unknown orientation". In: *2012 IEEE Conference on computer vision and pattern recognition*. IEEE. 2012, pp. 382–389.

[GMB02] David Goldberg, Christopher Malon, and Marshall Bern. "A global approach to automatic solution of jigsaw puzzles". In: *Proceedings of the eighteenth annual symposium on Computational geometry*. 2002, pp. 82–87.

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[He+15] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[HZ83] Robert A Hummel and Steven W Zucker. "On the foundations of relaxation labeling processes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 3 (1983), pp. 267–287.

[JOF06] Edson Justino, Luiz S Oliveira, and Cinthia Freitas. "Reconstructing shredded documents through feature matching". In: *Forensic science international* 160.2-3 (2006), pp. 140–147.

[Kho+21] Marina Khoroshiltseva et al. "Jigsaw puzzle solving as a consistent labeling problem". In: *Computer Analysis of Images and Patterns: 19th International Conference, CAIP 2021, Virtual Event, September 28–30, 2021, Proceedings, Part II 19*. Springer. 2021, pp. 392–402.

[Kos+94] David A Kosiba et al. "An automatic jigsaw puzzle solver". In: *Proceedings of 12th International conference on pattern recognition*. Vol. 1. IEEE. 1994, pp. 616–618.

[MB07] William Marande and Gertraud Burger. "Mitochondrial DNA as a genomic jigsaw puzzle". In: *Science* 318.5849 (2007), pp. 415–415.

[Mil+] B Mildenhall et al. "Representing scenes as neural radiance fields for view synthesis., 2021, 65". In: *DOI: https://doi. org/10.1145/3503250* (), pp. 99–106.

[MK03] Jonah C McBride and Benjamin B Kimia. "Archaeological fragment reconstruction using curve-matching". In: *2003 Conference on Computer Vision and Pattern Recognition Workshop*. Vol. 1. IEEE. 2003, pp. 3–3.

[NDH08] Ture R Nielsen, Peter Drewsen, and Klaus Hansen. "Solving jigsaw puzzles using image features". In: *Pattern Recognition Letters* 29.14 (2008), pp. 1924–1933.

[Pel97] Marcello Pelillo. "The dynamics of nonlinear relaxation labeling processes". In: *Journal of Mathematical Imaging and Vision* 7 (1997), pp. 309–323.

[PPT18] Marie-Morgane Paumard, David Picard, and Hedi Tabia. "Image reassembly combining deep learning and shortest path problem". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 153–167.

[PPT20] Marie-Morgane Paumard, David Picard, and Hedi Tabia. "Deepzzle: Solving visual jigsaw puzzles with deep learning and shortest path optimization". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 3569–3581.

[PSB11] Dolev Pomeranz, Michal Shemesh, and Ohad Ben-Shahar. "A fully automated greedy square jigsaw puzzle solver". In: *CVPR 2011*. IEEE. 2011, pp. 9–16.

[RHZ76] Azriel Rosenfeld, Robert A Hummel, and Steven W Zucker. "Scene labeling by relaxation operations". In: *IEEE Transactions on Systems, Man, and Cybernetics* 6 (1976), pp. 420–433.

[Ros60] Jo Bo Rosen. "The gradient projection method for nonlinear programming. Part I. Linear constraints". In: *Journal of the society for industrial and applied mathematics* 8.1 (1960), pp. 181–217.

[SDN13] Dror Sholomon, Omid David, and Nathan S Netanyahu. "A genetic algorithm-based solver for very large jigsaw puzzles". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 1767–1774.

[SDN16] Dror Sholomon, Omid E David, and Nathan S Netanyahu. "DNN-buddies: A deep neural network-based estimation metric for the jigsaw puzzle problem". In: *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25*. Springer. 2016, pp. 170–178.

[SE06] Mahmut Samil Sagiroglu and Aytül Erçil. "A texture based matching approach for automated assembly of puzzles". In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 3. IEEE. 2006, pp. 1036–1041.

[Sen06] Eugene Seneta. *Non-negative matrices and Markov chains*. Springer Science & Business Media, 2006.

[SHC18] Kilho Son, James Hays, and David B Cooper. "Solving square jigsaw puzzle by hierarchical loop constraints". In: *IEEE transactions on pattern analysis and machine intelligence* 41.9 (2018), pp. 2222–2235.

[SS87] Jacob T Schwartz and Micha Sharir. "Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves". In: *The International Journal of Robotics Research* 6.2 (1987), pp. 29–44.

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[YS03] Feng-Hui Yao and Gui-Feng Shao. "A shape and image merging technique to solve jigsaw puzzles". In: *Pattern Recognition Letters* 24.12 (2003), pp. 1819–1835.

[Zha+07] Yu-Xiang Zhao et al. "A puzzle solver and its application in speech descrambling". In: *WSEAS International Conference on Computer Engineering and Applications*. 2007, pp. 171–176.

[ZZH07] Liangjia Zhu, Zongtan Zhou, and Dewen Hu. "Globally consistent reconstruction of ripped-up documents". In: *IEEE Transactions on pattern analysis and machine intelligence* 30.1 (2007), pp. 1–13.