STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 1, 2024
DOI: 10.24193/subbi.2024.1.01

MATCHING APICTORIAL PUZZLE PIECES USING DEEP LEARNING

RALUCA-DIANA CHIŞ

Abstract. Finding matches between puzzle pieces is a difficult problem relevant to applications that involve restoring broken objects. The main difficulty comes from the similarity of the puzzle pieces and the very small difference between a pair of pieces that almost match and one that does. The proposed solution is based on deep learning models and has two steps: firstly, the pieces are segmented from images with a U-Net model; then, matches are found with a Siamese Neural Network. To reach our goal, we created our own dataset, containing 462 images and just as many masks. With these masks, we built 3318 pairs of images, half of them representing pieces that fit together and half that do not. Our most relevant result is estimating correctly for 290 out of 332 pairs whether they match.

1. Introduction

The aim of this paper is to present the design and implementation of a solution for the complicated problem of matching *apictorial* puzzle pieces. The term *apictorial* is used in this work with the meaning that the pairs of puzzle pieces are found only by looking at the shape of the pieces and neglecting the model on top of them, as we aim for a robust solution, that could suit pieces with any type of patterns. This problem is a complicated one, due to the fact that the puzzle pieces are very resembling and it is hard to differentiate between them. This requires a very well-defined puzzle piece edge extraction, that does not compromise their shapes at all, according to [5]. Moreover, the search space is considerably large, considering that for each side of a piece, the program has to compute as many comparisons as images in the dataset. Normally, the number of comparisons should have been multiplied by 4 (the

Received by the editors: 23 October 2023.

2010 *Mathematics Subject Classification.* 68T45, 68U10.

1998 *CR Categories and Descriptors.* I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding – *Shape*; I.4.6 [**Image Processing and Computer Vision**]: Segmentation – *Edge and feature detection*.

Key words and phrases. U-Net, Siamese architecture, Edge-matching, Puzzle Pieces.

number of possible rotations of a piece), but for this research it was considered that all the pieces are oriented as in their final position from the puzzle.

This research topic revolves around finding pairs of puzzle pieces, that complement each other perfectly. This has great applications in the reassembly process of fragmented objects, which is a very time-consuming task for humans and of high interest in archaeology, according to [17] and [15]. The main difference between solving a puzzle and restoring an object is that the former can be treated as a two-dimensional problem. Willis et al. [20] associate the reconstruction of ancient artefacts with solving a *real-world geometric puzzle*. In the same paper, the authors present puzzle piece matching as a more regular version of the problem of artefact piece matching, since puzzles have a more predictable shape and were not exposed to erosion prior to the reconstruction. Besides this, it is mentioned that the edges of jigsaw pieces are more simple to separate, because the corners of each side are easily identifiable.

The main contribution of the current study refers to the creation of an automated pipeline designed for pairwise matching of puzzle pieces. The process is done in two steps. Initially, images of puzzle pieces are segmented to extract the shape of each piece. For training this model, an original dataset was created, consisting of images of real puzzle pieces and corresponding masks. The second step of the pipeline involves the creation of a deep learning model for matching pairs of puzzle pieces according to their shape, which to our knowledge has never been tried before. A Siamese neural network performs edge matching for all segmented pieces and the matching score is computed for a series of generated pairs of puzzle pieces. Our most relevant result gives an accuracy of 87.34% for the edge-matching task on the test set, which means that for the majority of the puzzle pairs, the model managed to correctly predict if they match or not.

The current paper is structured in five sections as follows. The second section details the existing literature in the field. Section 3 presents the development of the segmentation and the edge-matching model. The fourth section describes the environment in which the experiments were carried out, what results we achieved and how they compare to the outcomes of other papers. Finally, conclusions are drawn and possibilities for improvement in the future are mentioned.

2. Related work

In this section we will review the most recent research papers on techniques for finding matches between different pieces, using images of them. Since we have established that our aim is not to solve a puzzle, we will only briefly address the parts of the papers related to generating puzzle solutions. The

methods presented in what follows expose both two and three-dimensional approaches and are focused on image processing techniques. Some of them are not completely automatic and require a specialist for constant feedback. One can easily notice a common pattern for all papers exposed here. They all follow a similar structure: photos or scans of the pieces (puzzle pieces or other types) are taken, followed by instance segmentation or contour curve extraction, and using the resulting features, edge matching between pieces is performed. The results of the papers presented below will be presented in Section 4.5 and they will be compared to the ones of the current research.

As far as the data used in the most relevant papers on edge-matching are concerned, they consist of images of torn documents [2], patches of papyrus or ostraca fragments images [18] and [14], images of unconventional puzzle pieces [6] and [7], 3D scans of broken objects [1], and synthetically generated three-dimensional objects [5]. Regarding the segmentation task, the authors propose methods that differ slightly from each other. Grim et al. [5] use Bezier curves with selected control points, while in [6] images were segmented using active contours [9] and smoothed using a naïve spline algorithm. In [7] the edge curvature is computed with integral invariants and Alagrami et al. [1] proposed a graph-based method for extracting the breaking curves of an object, with which the sides of the object were identified.

For the task of matching the edges, the approach proposed in [8] and in [1] uses the Iterative Closest Points (ICP) algorithm. Authors of [6] and [5] divided each boundary curve into a set of bivertex axes with the extended Euclidean signature method. All sets of axes were compared to each other and a confidence score on their matching was computed. In [2] the features lists for each corner detected were joined with the Minkowski Sum.

In a recent survey addressing the solving of Jigsaw puzzles [13], emphasis is placed on the utilization of deep learning techniques exclusively for methodologies employing square or rectangular patches extracted from images. This particular perspective proves valuable for discerning the relationship between two fragments belonging to the same object [18] or predicting the permutation of image patches to compose a comprehensive image [11]. However, our objective diverges from this approach. Our model seeks puzzle pieces that complement each other in shape rather than similarity, as the synergy of two matching puzzle pieces lies in their complementarity. To our knowledge, there are no previous experiments applying a Siamese network for comparing the shapes of two puzzle pieces.

We focus specifically on the shape of the pieces in the images. In the context of patches, continuity in terms of the patterns in the image is sought. This represents the main difference between our edge-matching model and the

Siamese neural network from [14]. Moreover, in the architecture proposed by Ostertag et al., the two inputted images are passed through four convolutional blocks and then 4 patches with a width of 10 pixels are extracted from the feature map of each branch. Each patch is subtracted from its opposite belonging to the second branch, the results are concatenated and the output consists of a 5 probabilities linear vector, each of them representing the similarity of each side. The disadvantage of this method that we overcome with our proposed architecture is that by creating those 10-pixel wide patches the overall context is lost and important chunks might be missing for some puzzle pieces.

3. Methodology

The manner in which we have approached the problem of matching puzzle pieces from images consists of two phases: the development of a model for segmenting puzzle pieces from images and the development of a deep-learning model for finding matching pairs of pieces.

3.1. **Segmentation.** The model for image segmentation has a very large influence on the success of the attempted approach, as the predictions generated by it will be the network input for edge-matching. The higher the quality of the segmentations, the better the piece-matching results will be. In particular, the edge of the pieces is preferable to be as well defined as possible, but the difficulty arises when the shadow of a piece appears in the image, which misleads the model about its edge.

We have decided to use for this task the model called U-NET, due to its effectiveness in capturing fine-grained details. The authors of this model [19] prove the usefulness and the very good performance of their network architecture, by applying it to three segmentation tasks: segmentation of neuronal structures in electron microscopic recordings, segmentation of cells on a polyacrylamide substrate recorded by phase contrast microscopy, and HeLa cells on a flat glass recorded by differential interference contrast microscopy. The results of the U-NET outperform the state-of-the-art (up until 2015) on these tasks.

This U-NET architecture is based on using fewer images than usual for training neural networks while exploiting the given images more efficiently. The authors use data augmentation techniques to achieve this performance, by elastically deforming images. In this way, the model can learn invariance to different types of deformations, which in their case of biomedical segmentation was very useful, because deformation in tissue is very common. For our case, data augmentation is also useful because images of puzzle pieces can vary greatly in terms of object position, brightness, colour, and so on.

The architecture of the network comprises a downsampling of the feature map, followed by an upsampling of these features. The model encodes an image, extracts features using multiple convolutional layers. Then, the decoder upsamples the features using the transpose convolution and concatenates them using a process, called skip connection. Through this process, the model skips some of the network layers and feeds the output of one layer as the input to the next layers. The output of the network is a segmentation mask. The pixels of this mask have values ranging from 0 to 1 and for increasing the chances of the edge-matching model, a threshold of value 0.4 was applied, meaning that all pixels under this value were converted to 0 and 1, respectively. The value of the threshold was set experimentally.

3.2. **Edge matching.** What comes after segmenting the puzzle pieces is finding pairs that match, depending on the shape of their sides. For this matter, we chose a deep-learning approach, following a Siamese architecture. The idea of this architecture dates back to 1993 [3], when a group of researchers proposed the use of two identical neural networks merged at the output. Their goal was to check if two signatures written on a touch-sensitive pad are identical. They obtained impressive results on test data, the model being able to detect a real signature in 95.5% of the cases and forgeries in 80%.

Siamese networks are used for measuring the similarity or the dissimilarity between two images, by comparing the feature vectors outputted by the twin networks. Our architecture is presented in Fig. 1. The network comprises two identical branches with shared weights, meaning that the parameters (weights and biases) of these branches are the same. Each branch is made out of three convolutional blocks. Each of them is composed of 2 convolutional layers with 64, 128, and 256 filters and kernel sizes of 11x11, 5x5, and 3x3 respectively, one batch normalization layer, one ReLu activation layer, and lastly, a max-pooling layer, which reduces the image size by 2.

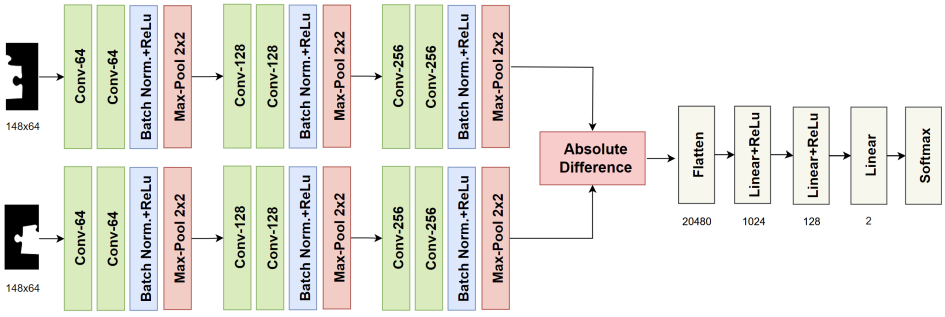

FIGURE 1. Edge-matching network architecture

At this point in the network, most of the Siamese architectures propose computing a loss between the two feature vectors, usually a contrastive loss, as in [4]. We tried out this method too, but the results were disappointing. The best model we obtained with this approach was able to make correct estimations only in half of the cases. Moreover, the Euclidian distance between one image and 50 others from the test set, was the same for 25 of them and equal to 0.26116937. This value for the distance would mean that the images are compatible, but when we reviewed them, we discovered both suitable images and completely not suitable ones. These facts led us to the conclusion that this path cannot be successful, since the model was focusing on finding if the two input images are similar or not and this was not what we specifically needed.

Inspired by [18] and [14], we decided to subtract the two feature vectors element by element. The result was flattened and we obtained a one-dimensional feature vector of size 20408. As visible in Fig. 1, the vector is afterwards passed through a couple of linear transformations. After the first linear transformation, we tried adding a dropout layer, with a probability of 0.5, to prevent the model from overfitting. In some situations, this layer helped, but not always, as we will discuss in the next section. Lastly, the Softmax layer computes the probability of the two images forming a match or not. Further on, we used the Cross-Entropy loss from PyTorch [16], to compute the distance between the probabilities outputted by the model and the true ones.

4. Experimental results

Based on the models presented above, experiments were conducted for both tasks. In the following, the most relevant experiments will be presented, discussed and compared, together with the dataset and the methodology used in their evaluation. The code used for experiments is available here[1].

4.1. **Data.** For this study, a dataset was created taking into account the use of various puzzle piece sizes, the need for having a minimum of several hundred images for training, testing and validation, and a reduced difficulty for the pre-processing and mask generation step. The dataset is available here[2].

The first step in creating the dataset was to photograph seven puzzles with different numbers of pieces and different piece sizes, ranging from 1.6x1.8 cm to 4.9x4.1 cm. All the pieces had the classic shape of puzzle pieces, namely four sides, each with an indent or an outdent. The exception was the edge pieces, which had one or two straight sides. The pieces were placed face down, on a black background, to have a bigger contrast between the piece

[1]https://github.com/RalucaChis/PiecePerfect

[2]https://www.kaggle.com/datasets/ralucachiss/pieceperfect-dataset/

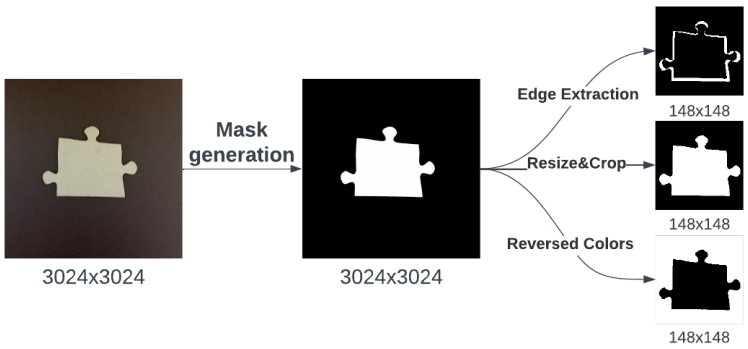

FIGURE 2. Puzzle piece images preprocessing pipeline: mask generation, mask resize and crop, mask with reversed colours and an only-edge mask

and the background, which helped enormously the binarization process. The photographs were taken with an iPhone, version XS, with a focal length of 26 mm, aperture ratio of f/1.8, and image size of 3024x3024. In total, we took 462 pictures, belonging to 7 different puzzles. The quality of the images was good enough for our task, but only very late in the research a possible problem in the data collection occurred to us. Since we did not use a fixed setup for taking the pictures (a camera setup [8] or a photocopier [7]), the distances between the camera and the pieces may be different by a few centimetres, thus affecting the ratio between the size of the piece and the image.

To generate the masks, images were first opened in grayscale mode and then converted into binary images with thresholding, separating foreground (objects of interest) and background. Experiments revealed that a threshold value of 140 (with pixels ranging from 0 to 255) was optimal.

The generated masks have white pixels for puzzle pieces (value 1) and black pixels (value 0) for the background. Since the masks had some black gaps inside the pieces, we made use of the "floodFill" method from OpenCV to fill the gaps. FloodFill is a technique for changing the colour or value of a connected region of pixels in an image, starting from a seeding point. The filling process began from the left top corner of the image, by converting to white the whole background. The colours of the retrieved image were inverted and then we applied a logical OR operation on the resulting image and the original one. The obtained masks have very good quality and correspond completely to the shape of the pieces (Fig. 2).

The obtained dataset consists of 462 pictures of puzzle pieces, suitable for training a segmentation model. However, the model was limited to identifying

puzzle pieces without an image and a simple background. Therefore, we generated five additional variants for each image, randomly selecting backgrounds from a set of 13 and foregrounds from a set of 19. This augmentation resulted in a dataset of 2772 images (original 462 + 5*462 augmented), significantly improving segmentation performance on test images.

The initial images have a size of 3024x3024 pixels. Since working with very large images is difficult, we chose to resize the images to 256x256 pixels. After performing several experiments with unsatisfactory numeric results on the edge-matching model, we found that reducing the background area by cropping the images improved model performance. For this, we cropped the images, keeping 5 pixels on each side between the piece and the edge of the image. To maintain uniform mask sizes without losing quality through resizing, we added black padding to each side, resulting in images standardized to the largest size (148x148 pixels). For future reference, we will call this set *DS1*.

Furthermore, we thought it might help if, when analyzing a pair of masks, the second image had the colours reversed. In this way, the side of the first piece would be white, and the outside of the pair side of the second image would still be white, and thus the model would more easily identify the compatibility of the two pieces. So we generated a set of images with the colours reversed (*DS2*). Also, in order to simplify the work of the model, we generated a set of images containing only the edges of the pieces (*DS3*). To obtain them, we resized the masks to 135x135 pixels, added black padding to the images on all sides until we obtained pictures of 148x148 pixels with the piece in the centre, and then applied a logical XOR operation between the initial mask and the transformed one. Fig. 2 contains an example mask from each of the three resulting sets.

For efficient image management, we scanned each puzzle solved and numbered each piece. The numbers on each puzzle were manually translated into a matrix, which we used to create the pairs of pieces. Thus, for each piece we considered the neighbouring pieces in the 4 directions (top, bottom, left, right) as compatible, and the neighbours on the diagonal (top-left, top-right, bottom-left, bottom-right) as incompatible.

We started from the premise that all pieces are oriented in the final position in the puzzle, so between any two puzzle pieces there are 4 possible matches, plus the case where they do not match at all. An example of the 4 matching cases can be seen in Fig. 3.

The compatibility of two pieces is determined by how well two of their sides complement each other. Thus, to encourage the model to pay more attention to the edges of the pieces, we chose to train the model with the images cut in half for all experiments. Following the pre-processing procedures applied to

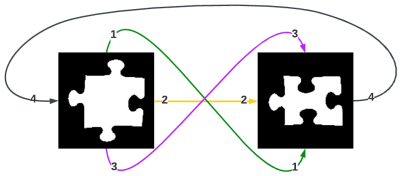

FIGURE 3. Matching Possibilities: 1&3 are top-down matches (the top of a piece matches the bottom of the other and vice-versa) and 2&4 are left-right matches.

the masks, in each of them the piece is in the centre of the image, so there is no risk that a half of the image does not contain any part of the piece. Prior to cropping, the images were rotated 90°, 180° or 270°, according to their match-case, so that the right side of the first image (first from left to right) matches the left side of the second image. Since we rotated and cropped the images, the task of differentiating the four types of match is almost impossible, so we chose to classify the pairs as binary, match or non-match. With this strategy, we obtained a total of 3318 entries, of which 1618 represent matches and 1700 non-matches.

4.2. **Evaluation methods.** In order to evaluate the segmentation model, we split the image dataset into three sets: for training (70%), validation (20%) and testing (10%). One of the evaluation metrics used is L1 loss, which is also known as the absolute error loss. It is defined as the absolute difference between the predicted and actual values, as follows: $L1 = |y_{actual} - y_{predicted}|$. The mean absolute error (MAE), which is computed as the mean of L1 values, will be used for evaluating the mean value of pixels, that have not been correctly predicted (with values from 0 to 1, where 1 states that all pixels have been wrongly classified and 0 the opposite).

Moreover, the Dice coefficient was used to estimate how well the mask overlaps the input image. The Dice coefficient is computed as twice the intersection of the predicted and ground truth masks, divided by the sum of their areas: Dice coefficient = (2 * Intersection) / (Sum of areas). It takes values in the range of 0 and 1, where 1 indicates a perfect overlap, while 0 indicates no overlap.

For the edge-matching task, the main criterion applied for evaluating the model is the accuracy of the predictions. The pairs discussed in Section 4.1 were divided into three sets: training (70% of them), validation (20%) and testing (10%). The accuracy of this task is simply computed by dividing the number of correctly estimated pairs from the test set, by the total number of pairs. For an improved perspective, we also computed the F1 score. This evaluation method is computed as a harmonic mean of precision and recall,

with the following formula: $F1 = 2 * (precision * recall)/(precision + recall)$. The F1 score ranges from 0 to 1, where 1 represents the best possible value and 0 is the worst.

In addition to these classical evaluation methods, we considered it important to analyze the performance of the model in the context of finding the ideal pair of a part. We chose from the test set the pieces that had a pair and analysed the fit between one piece and all the others. If the true match is among the top 10 pieces with a very high probability of being a match, we will consider it a success. For future reference, we will call this evaluation *top_10*.

4.3. **Segmentation results.** The segmentation part of the problem did not impose as many difficulties as the edge-matching task, so a small number of experiments was adequate. After a few tries, we noticed that the model behaved best when using the Adam optimizer and a batch size of 16. The difference between using a learning rate of 0.003 and 0.0003 was minimal. After applying the second value, the mean absolute error (MAE) obtained was 0.003490, and the Dice coefficient value on average was 0.9751, confirming that the model learns the proposed task very well.

The results of both experiments are visible in Table 1. A mean value as high as 0.97 out of 1 for the Dice coefficient suggests that the masks generated by the segmentation model overlap almost perfectly with the ground truth. This result is further supported by the very low value achieved for the MAE score, 0.0034. The progress of the L1 loss during training supports the good performance of the model (Fig. 4). The values range from 0.0896 to 0.0092 for training and from 0.5906 to 0.0123 for validation. The trainings have been conducted for 30 epochs, as we noticed that the loss stabilizes and fluctuates very little after the first 10 epochs.

4.4. **Edge matching results.** The edge-matching task was difficult, so several experiments were needed to come up with at least a satisfactory solution. Table 2 exposes the numerical results and we are going to detail each of the experiments in what follows. All experiments were done with a batch size of value 64, a learning rate of 0.001 and with the Adam optimizer [10]. These specifications have been established experimentally. The batches are computed before training and images are randomized. For all experiments, the images were halved. As we noticed that accuracy and loss stabilize after 30 epochs, we did all our experiments with this number of epochs. All the experiments described in the next chapter are done on Google Colaboratory, using GPU Nvidia T4, with 16 GB.

Inspired by the architecture shown in [18], where the authors present a model with an accuracy of 79% on matching papyrus fragments, we tried

TABLE 1. Experimental results for image segmentation with U-Net.

Dataset	**Batch Size**	**LR**	**Optimizer**	**MAE**	**Dice coeff.**
DS1	16	0.003	Adam	0.004968	0.9649
DS1	16	0.0003	Adam	**0.003490**	**0.9751**

TABLE 2. Experimental results for edge-matching (columns *Img1* and *Img2* state the dataset used as input for the first and second branches respectively; *Drp* stands for *Dropout Layer*)

Id	**Img1**	**Img2**	**Drp.**	**LR**	**Opt.**	**Acc. %**	**F1**	***top_10* %**
1	*DS1*	*DS1*	NO	0.001	Adam	51.2048	0	0
2	*DS1*	*DS1*	YES	0.001	Adam	83.4337	82.9721	10.17
3	*DS1*	*DS1*	NO	0.001	Adam	84.6385	84.9557	12.34
4	*DS1*	*DS2*	YES	0.001	Adam	80.4216	79.4952	**17.36**
5	*DS3*	*DS3*	YES	0.001	Adam	**87.3493**	**85.6164**	8.982

flattening the feature vectors outputted by the two identical branches and only afterwards perform the absolute difference between the two (experiment with id *1*). The model performed poorly, predicting that almost all pairs are not matching, which lead to very low results for all three performance measures. The behaviour of the model is unexpected and worth investigating in future.

The second experiment (id *2*) is based on the architecture described in Section 3.2, having as input the preprocessed puzzle pieces images, rotated and cut in half, as described in Section 4.1. Although this approach obtained very good results on all evaluation methods, it was outperformed with more than one percent on each metric by the experiment in which we removed the dropout layer from the neural network (experiment with id *3*). The reason for this lies in the fact that the model used had a rather simple design, which was not prone to overfitting, so the dropout layer could be omitted. This was not the case for the last two experiments. The fourth experiment performed better when we used the dropout layer and the same for the fifth. This happened thanks to the ability of the dropout layer to reduce the sensitivity of a model, which helped the model focus on the most salient features.

The experiment with id *4* was very similar to the one with id *2*, with the exception that for experiment *4* the second inputted image had its colours inverted (white became black and vice-versa). While the accuracy and the F1 score value were considerably lower than in the approach with id *3*, the model managed to find the perfect match of a piece among the top 10 most suitable piece images in 29 out of 167 cases. On the other side, for the experiment *5*

we used as input the images containing only the edge of the puzzle pieces. In this case, the model obtained the best accuracy, 87.3493%, and the highest F1 score, 85.6164. However, it performed the worst on the *top_10* evaluation, with only 8.9820% of matches identified. This discrepancy can be explained by the fact that the model did a little overfitting, and when evaluating *top_10*, it obtained unrealistically high probabilities for many parts (in one of the cases the probabilities were 1.4511e-07 non-match and 1.0000e+00 match) and it was difficult to find the true pair.

Regarding the evolution of the cross-entropy loss at each epoch for the training and validation sets, the results stabilize somewhere after 30 epochs (Fig. 4). They fluctuate between 0.693 and 0.4101 for training and between 0.6933 and 0.4367 for the validation process. As expected after discussing the outcomes of the experiment *5*, the loss in this case was the lowest, for both training and validation, while the approach with id *1* had the highest loss values.

4.5. **Discussion.** The results presented in the previous section can be summed up to achieving an MAE score as low as 0.0034 for the segmentation task and accuracy as high as 87.34% for the edge-matching model. Even though these are very good results from our perspective, some issues remain, which will be discussed at length below.

Regarding the segmentation task, as mentioned, good results were achieved on the test set. For situations where only one piece appears in an image, and the contrast between the piece and the background is high enough, the model manages to do the segmentation correctly. In a more complicated situation, however, it fails to estimate the mask accurately enough so that the mask can be used for further edge-matching. There is certainly room for improvement in the model, especially on the part of the diversity of the data used as input, but from the perspective of the purpose of this paper, the current capability of the mask estimation model is satisfactory.

As for the results obtained for the evaluation *top_10*, an example of what they look like can be seen in Fig. 5. Despite the actual pair not being within the top 10, all featured images depict pieces remarkably similar to the sought pair, each possessing a framing-compatible hole. Distinguishing the true pair is challenging, even for a human, due to minimal differences between pieces. Notably, with a smaller image set, finding the actual match becomes more probable. In our experiments, the task was challenging, with a test set comprising 332 pairs. Another attribute that influences the model results for edge-matching is the ratio between the piece and the image sizes. Some images were taken at a shorter distance between the camera and the piece, and others at a longer distance. Thus, the model may learn to classify as non-match two

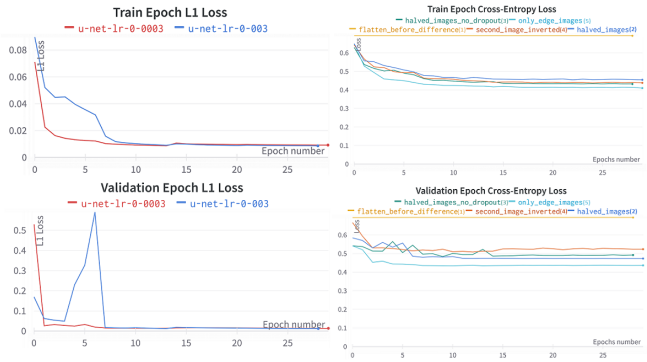

FIGURE 4. Epoch loss for training and validation on segmentation (left column) and on edge-matching (right column)

pieces that in reality match, but the images do not overlap perfectly because of the different size ratios. To investigate the extent of this issue, we constructed a confusion matrix (Fig. 6) for the model trained using the *DS3* dataset (experiment id *5*). False negatives account for 11.14% of the total predictions, but only 11.9% of the incorrect predictions are false positives, revealing that the model is more prone to generating false negatives than false positives.

For the specific situation of a puzzle, where the solution is guaranteed to be unique, the process can be continued with the assembly of the puzzle. The compatibility of a piece with the top 10 possible pairs is very high, with small differences between them, which is why, for the purpose of automatic puzzle assembly, we consider it appropriate to find tuples of 4 pieces that match two by two, and then match them all together. For what we set out to do, this step is not necessary.

Comparing the results presented in this paper with those obtained by other researchers is difficult. One reason is that in some papers the authors used

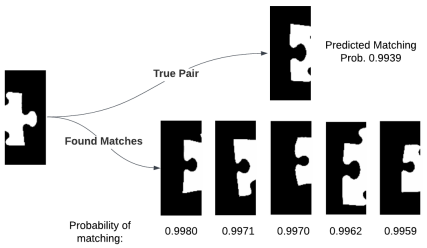

FIGURE 5. Example of very probable matches found for a piece, yet none of them is the true pair.

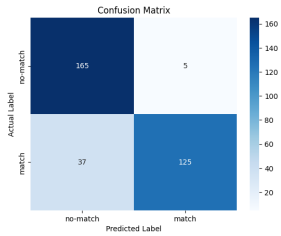

FIGURE 6. Confusion matrix for the edge-matching model trained with *DS3*

image patches ([18], [14]), as discussed in Section 2. In [18], the highest accuracy obtained for predicting if two patches belong to the same image is 79%, while in a similar situation, we obtained 87.3493%. Paper [14] exposes no numerical results done on a test set, but presents a 96% accuracy achieved on the validation set. Our best accuracy on the validation is only 89.15%, but our dataset had just 2322 entries for training and 664 for validation, while in the mentioned paper a set of 6000 entries was used for training and 1000 for validation. An overfit on the training data cannot be excluded in their case, since the authors of [14] mentioned poor results in reconstructing an image from patches, despite the high accuracy.

Another difficulty comes from the fact that some works exhibit working with three-dimensional models of artefacts [8] or 3D puzzles [5], and our solving is strictly based on 2D images. While [8] lacks numerical results, in [5] the model manages to solve a 3D 18-piece puzzle successfully. However, when tested on a broken ostrich eggshell, only 11 out of 15 shell pieces are placed correctly.

As for the papers [6], [7], [12] and [2], they come closest to our work in terms of the proposed problem, because they use 2D puzzle pieces, i.e. pieces of documents, as input data. What distinguishes our work is the use of deep learning techniques, whose help is difficult to estimate in comparison with image processing techniques. This happens especially because in the previously mentioned works the methodology for evaluating the experiments is to measure the time taken to reassemble the puzzle, i.e. the document, while this was not a main focus of our work. Also, in the case of [2], the reassembled documents are cut into only 4 parts, and the puzzle pieces used in [6] and [12] have very different shapes from each other, much more irregular than the pieces of a normal jigsaw puzzle. These aspects favour the success of the model. In terms of results, Makridis et al [12] present the assembly of a puzzle, without mentioning the time needed for the whole process, and in [2] the best image preprocessing time was 0.394s, and the best joining time was 0.336s. The results provided in [6] refer to the solving of a 48-piece puzzle in 58 minutes and a 67-piece one in 31 minutes. The same 48-piece puzzle was solved in 3 minutes with the approach from [7], which is a great improvement. However, the approach from [7] is not as robust, since the authors mentioned a higher assembly time in comparison with [6], for the same 67-piece puzzle.

5. Conclusions and future work

The aim of this paper is to develop a solution to the problem of matching puzzle pieces from images, using deep learning methods. Solving this problem is the first step in the digital approach to the difficult task of restoring artefacts (e.g. putting together ostraca pieces). The solution proposed consists of two

phases: segmenting the puzzle pieces, followed by finding matching pairs for each side of a puzzle piece.

Regarding segmentation, the method applied is the U-Net model, trained with 462 images with puzzle pieces and related masks. The main result obtained in this respect is a score for the dice coefficient of 0.9751 for the test set, which represents a very good result given that the highest possible value is 1 for perfect overlap.

The edge-matching task has been addressed with a Siamese architecture model. Two masks are processed at the same time by two identical networks, and then the absolute difference between the two feature vectors is computed. The result is flattened, linearized, and at the end, the output consists of two probabilities: match or non-match. The main result obtained is an accuracy as high as 87.3493% and an F1 score of 85.6164, for a number of 332 pairs of masks used in the test. This result was achieved when the masks used to feed the network were processed so that they contained only the contours of the pieces. Reversing the colours of the second mask also proved effective, so the model managed to find for 17.36% of cases among the best 10 predictions (the most compatible images), the ideal match of the piece in the initial image.

In the future, we plan to improve the image set with more puzzle pieces in different contexts, to help the segmentation model identify the contours of the pieces in any situation. Also, for the edge-matching model, the results of the *top_10* evaluation need improvement, which could be done by developing a more complex model and diversifying the dataset. The differences between the puzzle pieces are small, at least for the puzzles we used, which made learning the matches difficult.

The current study contributes to the exploration of artefact reconstruction, obtaining good results on both segmentation and edge-matching of images with puzzle pieces.

References

[1] Alagrami, A., Palmieri, L., Aslan, S., Pelillo, M., and Vascon, S. Reassembling broken objects using breaking curves, 06 2023.

[2] Biswas, A., Bhowmick, P., and Bhattacharya, B. Reconstruction of torn documents using contour maps. In *IEEE International Conference on Image Processing 2005* (2005), vol. 3, pp. III–517.

[3] Bromley, J., Bentz, J., Bottou, L., Guyon, I., Lecun, Y., Moore, C., Sackinger, E., and Shah, R. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence 7* (08 1993), 25.

[4] Dey, S., Dutta, A., Toledo, J. I., Ghosh, S. K., Llados, J., and Pal, U. Signet: Convolutional siamese network for writer independent offline signature verification, 2017.

[5] Grim, A., O'Connor, T., Olver, P., Shakiban, C., Slechta, R., and Thompson, R. Automatic reassembly of three-dimensional jigsaw puzzles. *International Journal of Image and Graphics 16* (04 2016), 1650009.
[6] Hoff, D., and Olver, P. Automatic solution of jigsaw puzzles. *Journal of Mathematical Imaging and Vision 49* (05 2014).
[7] Illig, P., Thompson, R., and Yu, Q. Application of integral invariants to apictorial jigsaw puzzle assembly. *Journal of Mathematical Imaging and Vision 65* (2021), 340–353.
[8] Jampy, F., Hostein, A., Eric, F., Laligant, O., and Truchetet, F. 3d puzzle reconstruction for archeological fragments. *Proceedings of SPIE - The International Society for Optical Engineering 9393* (03 2015).
[9] Kass Michael, Witkin Andrew, T. D. Snakes: Active contour models. *International Journal of Computer Vision 1* (1988), 321–331.
[10] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014).
[11] Li, R., Liu, S., Wang, G., Liu, G., and Zeng, B. Jigsawgan: Auxiliary learning for solving jigsaw puzzles with generative adversarial networks. *IEEE Transactions on Image Processing 31* (2022), 513–524.
[12] Makridis, M., and Papamarkos, N. A new technique for solving a jigsaw puzzle. In *2006 International Conference on Image Processing* (2006), pp. 2001–2004.
[13] Markaki, S., and Panagiotakis, C. Jigsaw puzzle solving techniques and applications: a survey. *The Visual Computer 39*, 10 (2023), 4405–4421.
[14] Ostertag, C., and Marie, B.-A. Matching ostraca fragments using a siamese neural network. *Pattern Recognition Letters 131* (2020), 336–340.
[15] Papaioannou, G., Schreck, T., Andreadis, A., Mavridis, P., Gregor, R., Sipiran, I., and Vardis, K. From reassembly to object completion: A complete systems pipeline. *ACM Journal on Computing and Cultural Heritage 10* (2017), 8:1–8:22.
[16] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: Crossentropyloss. `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`, 2021.
[17] Pintus, R., Pal, K., Yang, Y., Weyrich, T., Gobbetti, E., and Rushmeier, H. A survey of geometric analysis in cultural heritage. *Computer Graphics Forum 35*, 1 (2016), 4–31.
[18] Pirrone, A., Aimar, M. B., and Journet, N. Papy-s-net: A siamese network to match papyrus fragments. In *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing* (New York, NY, USA, 2019), HIP '19, Association for Computing Machinery, pp. 78–83.
[19] Ronneberger, O., P.Fischer, and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015), vol. 9351 of *LNCS*, Springer, pp. 234–241.
[20] Willis, A. R., and Cooper, D. B. Computational reconstruction of ancient artifacts. *IEEE Signal Processing Magazine 25*, 4 (2008), 65–83.

Department of Computer-Science, Faculty of Mathematics and Computer Science, Babeş,-Bolyai University, Cluj-Napoca, Romania
Email address: `raluca.chis@ubbcluj.ro`