Determining Jigsaw Puzzle State from an Image based on Deep Learning

Ijaz Ahmad[1], Suk-seung Hwang[2], and Seokjoo Shin[*]
[1,*]Dept. of Computer Engineering, [2]Dept. of Electronic Engineering
Chosun University
Gwangju, 61452 South Korea
[1]ahmadijaz@chosun.kr, [*]sjshin@chosun.ac.kr (Corresponding author)

***Abstract*—The intriguing nature of jigsaw puzzle has captured the attention of researchers for many years. In this paper, we propose a deep learning model to determine different states of jigsaw puzzle from an image. We represent the task as a classification problem where each state of the puzzle is considered as a class. For this purpose, we have proposed a method to generate a dataset that can efficiently represent the jigsaw puzzle states space. The proposed model has 93% accuracy on the test dataset. In addition, we have shown that when the tiles size changes the model is still able to recognize 83% of the states. Though, genetic algorithms (GA) have been successful in solving larger puzzles, they require hand-crafted sophisticated compatibility scores. The computation and memory requirement to store the piecewise compatibility measure increases with the size of the puzzle. As an application, we have shown that the proposed method can be used as a fitness function of GA based jigsaw puzzle solver without using any compatibility measure.**

Keywords—jigsaw puzzle, deep learning, genetic algorithm

I. INTRODUCTION

A jigsaw puzzle consists of multiple non-overlapping blocks of an image and the goal is to reassemble all the blocks to reconstruct the original image. Computational jigsaw puzzle assembly has applications in the field of image editing, recovery of shredded documents or photographs, art-piece recovery from shards in archaeology etc. [1]. In the recent years, automatic jigsaw solvers have been vastly improved in terms of number of tiles the puzzle has, tiles size, solution accuracy and amount of manual labor. For the size of the puzzle: [2] proposed a probabilistic approach to solve a puzzle of 432 pieces, [3]'s solver could handle up to 3000 pieces, [4], [5] genetic algorithm based solver to solve larger puzzle of size 22,834 pieces. On the other hand, for the variant of the puzzle: [6] proposed a method to handle puzzle with unknown piece orientation and location, and [7] handled puzzles with missing pieces. [8] proposed a neural network to predict the adjacent pieces and applied shortest path optimization to assemble the puzzle. [9] proposed to use graph connection Laplacian to determine the edge relationships for solving jigsaw puzzles. [1] proposed a deep learning based compatibility measure between two pieces. The metric improves the existing jigsaw puzzle solver accuracy. A somewhat similar approach has been adopted in [10] for solving a real world problem i.e., Portuguese tile panels reconstruction. [11] proposed a GAN-based architecture to solve jigsaw puzzles with eroded boundaries. [12] proposed a GAN-based architecture that utilizes both edge and semantic information to solve jigsaw puzzles efficiently.

In general an automatic jigsaw puzzle solver consists of two steps: a compatibility measure of adjacent tiles and a strategy to reassemble the compatible tiles. The compatibility measure has been widely studied, e.g., [2] compares five different compatibility measures, among which the dissimilarity function is the most reliable one. For reassembly strategy, greedy algorithms become problematic in case of local optima [1]. To overcome the problem, genetic algorithm (GA) based jigsaw puzzle solver has been proposed in [4]. The GAs have been able to solve larger and harder puzzles; however, they require hand-crafted sophisticated compatibility measures [1]. The compatibility measure for a puzzle of size $N \times M$ with unknown location requires $R(N \times M)^2$ size matrix to store all of the pairwise compatibilities for all pieces. $R = 4$ is the possible position of a tile to be placed in relation to another tile e.g., top, right, bottom or left.

In this paper, we propose a deep learning model to identify a jigsaw puzzle state from a given image. We have posed the problem as a classification problem where each state is a separate class. For this purpose, we propose a method to generate a dataset that can efficiently represent different states of the jigsaw puzzle. Each state is recognized by a score based on the number of tiles being swapped and their original locations. The trained model is able to recognize majority of the jigsaw puzzle states. The misclassified states are the ones where two or more than two tiles have the same texture and are visually imperceptible. Further, we evaluated the efficiency of the model to recognize states when the size of the tiles changes. As an application of the proposed method, we have used the trained model as a fitness function of GA based jigsaw puzzle solver. We assumed a puzzle with unknown tiles position but known dimensions.

II. PROPOSED METHOD

A. Jigsaw Puzzle

We define jigsaw puzzle state as two or more than two tiles have swapped their positions as opposed to their appearance in the original image. Numerically, the state can be represented as a sum of differences between each tile in the puzzle and their neighbors (i.e., for two tiles to be neighbors, their Euclidean distance should be 1). The state representation of a jigsaw puzzle with $r \times c$ tiles t, where each tile is indexed by $t_{i,j} = c \times i + j$, is given as

$$\mathrm{S} = \sum_{i=0}^{r} \sum_{j=0}^{c-2} d\left(t_{i,j}, t_{i,j+1}\right) + d\left(t_{j,i}, t_{j+1,i}\right) \tag{1}$$

where,

$$d(x,y) = \begin{cases} 1, & |x-y| \in \{1, c\} \\ 0, & Otherwise \end{cases}$$

It is an adjacency function that measures if two tiles are adjacent in the original image based on their absolute

difference. The difference value should be equal to 1 or c for two tiles to be horizontally or vertically adjacent, respectively. The score S in (1) not only depends on the number of tiles being shuffled but also on their positions they have been swapped from. Let's suppose, only two tiles have swapped their positions. Then, the score obtained is as: for a corner piece being swapped with another corner piece or with its neighbor, then $S = 4$. However, if a corner piece is swapped with a piece that is in the same row or column then $S = 5$. Similarly, $S = 6$ for a corner piece being swapped with a piece that is neither in the same row nor in the same column and so on. The score can have a minimum value of 4 and maximum value of $r^2 + c^2 - (r + c)$.

B. Automatic Jigsaw Puzzle Solver

An automatic jigsaw puzzle solver consists of two parts: a measure of piecewise compatibility of adjacent blocks and a strategy to reassemble the blocks in correct order as they would have appeared in the original image. For reassembly strategy, greedy algorithms becomes problematic in case of local optima. To overcome the problem, [4] has proposed genetic algorithm (GA) based jigsaw puzzle solver. The pseudocode for GA is given in Algorithm 1.

Algorithm 1 Pseudocode for Genetic Algorithm

1. **Generate** initial population P of n random chromosomes
2. **while**(stopping GA criteria is false) **Repeat**
3. **Evaluate** P using the fitness function
4. $new\_population = NULL$
5. copy b best chromosomes to $new\_population$
6. copy m mutated chromosome to $new\_population$
7. **while** $size(new\_population) \leq n$ **do**
8. $parent1 = select\ chromosome$
9. $parent2 = select\ chromosome$
10. $child = crossover(parent1, parent2)$
11. add child to $new\_population$
13. $P = new\_population$

The first step of the algorithm is to generate an initial population P with n random candidates solutions called chromosomes. In line 2, the criteria is the number of generations or how many times the algorithm will repeat the subsequent steps. Next, the entire population is evaluated based on the fitness function, which decides the selection of a particular chromosome to reproduce and pass onto next generation. The new population is generated by the crossover and mutation process operations. The whole process is repeated for a given number of times or when the desired fitness function value has been achieved.

C. Model

We use ConvNet of the VGG16 [13] model as a feature extractor. Our model consists of a sequence of convolutions followed by batch-normalizations, and max pooling layers. On top of the feature extractor is two dense layers with 512 nodes in the first layer and 22 in the second with softmax activation. The full architecture is shown in Table I. The pixel values are processed to have value between 0 and 1.

(a) (b) (c)

Fig. 1. Examples from the dataset to represent jigsaw puzzle states. (a) label is j_2, Score = 5. (b) label is j_{11}, Score = 14. (c) label is j_{21}, Score = 24.

TABLE I. Architecture of the proposed model. Conv3: 3×3 convolution, BN: Batch normalization.

Layer	Shape	Parameters #
Input	224×224×3	0
Conv3+Conv3+BN	224×224×64	39k
Max Pooling2D	112×112×64	-
Conv3+Conv3+BN	112×112×128	221k
Max Pooling2D	56×56×128	-
Conv3+Conv3+Conv3+BN	56×56×256	1.4M
Max Pooling2D	28×28×256	-
Conv3+Conv3+Conv3+BN	28×28×512	5.9M
Max Pooling2D	14×14×512	-
Conv3+Conv3+Conv3+BN	14×14×512	7M
Max Pooling2D	7×7×512	-
Fully Connected	512	12M
Fully Connected	22	11k

TABLE II. Accuracy of the model on different size of tiles.

Training	Test	
	56x56	28x28
0.95	0.93	0.83

III. Results and Discussion

In this section, we first present our experimental setup. Then, we show the performance of the proposed model to determine a jigsaw puzzle state. Finally, we present an application of the method to be used as a fitness function of genetic algorithm to solve a jigsaw puzzle.

A. Determining Jigsaw Puzzle State

For this purpose, we have collected 10k color images of size 224×224. Next, we have performed the following steps to generate our dataset:

- Divide the images into 4x4 tiles each of size 56x56.
- Generate 10k random indices corresponding to each score value (1), e.g., in our case $S \in \{4, 5, \cdots, 24\}$, as $I = \{I_1, I_2, \cdots, I_{21}\}$. Each element of I has 10k entries and corresponds to a score, i.e., each entry of I_1 has score 4, I_2 has score 5, and so on.
- To generate labelled data, shuffle a copy of the original images using the indices matrix from the previous step.

The dataset consists of 22 classes each with 10k images. The classes labels are $J = \{j_0, j_1, \cdots, j_{21}\}$, where j_0 is the solved state of the puzzle and the subsequent classes represents states with score equal to the subscript of the label plus 3, e.g., j_1 is the puzzle state where $S = 4$. The examples from the dataset are shown in Fig. 1. The dataset split is: 95% as training, 2.5% as validation and 2.5% as test datasets. We trained the model in a supervised manner using stochastic gradient descend (SGD) with momentum 0.9 for 100 epochs. The initial learning rate was set to 0.1 and then reduced by a factor of 10 after 30 epochs.

In the first experiment, the model was trained and evaluated on the same size of tiles, i.e., 56×56. In the second experiment, the trained model was evaluated on smaller tiles size of 28×28. We show the accuracy of the model on the train, validation and test dataset in Table II. The model has correctly determine the jigsaw puzzle state 93% of the times in the test dataset. Fig. 2. shows the confusion matrices to represent counts from actual and predicted classes. It can be seen that the model has misclassified the classes corresponding to higher scores with the classes closed by, e.g., class 16 with 17. The reason for this is that the images have regions where the texture is uniform. Two or more tiles from such regions are visually indifferentiable. As for the second

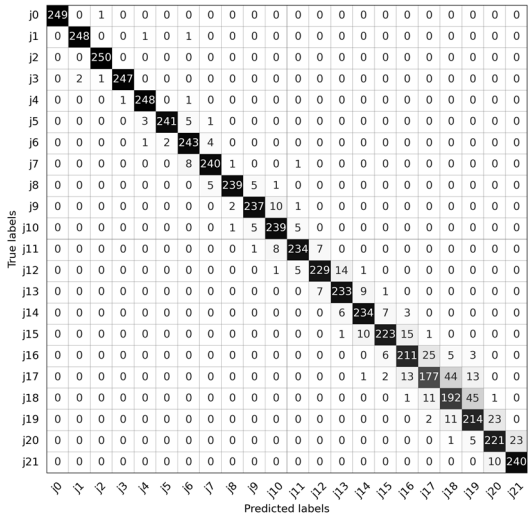

Fig. 2. Confusion matrices to count actual and predicted classes.

experiment, when the block size is reduced to 28×28, the model is still able to determine 83% of the jigsaw puzzle states. Note that it is necessary to resize the images to meet the input image size requirement of the trained model. We conjecture that the accuracy may improve in this case with the model being trained on mixed sized tiles.

B. Application – Jigsaw Puzzle Solver based on GA

As an application, we have replaced the fitness function of GA with the proposed jigsaw state recognizer in Algorithm 1. We have generated 40 chromosomes as the initial population and set the number of generations to 30. We have chosen 30% of the top population to be in the successor generation and set the mutation rate to 5%. We show an evolution of the solver how it approaches to the solution state in Fig. 3. The figure shows the average fitness score of the entire population of each generation and score of the best solution in the population. It can be seen that with each generation the solver gets closer to the solution. For visual analysis, we show an example in Fig. 4. The starting state of the puzzle has score $S = 24$. The initial number of generation was 30 but the algorithm converges to the solution in 11 generations. For this experiment, we took 30 images randomly from the test dataset. The GA based solver was able to solve the puzzle 85% of the times. The performance may improve by tuning the algorithm parameters, e.g. increasing population size and/or number of generations.

IV. Conclusion and Future Work

We proposed a deep learning model to determine jigsaw puzzle states from an image. The model was able to identify 93% of the states in the test dataset. In addition, the model was able to recognize the puzzle states from different size tiled images. As an application, we replaced the fitness function of genetic algorithm based jigsaw puzzle solver with the trained model, and we were able to solve 16 tiles puzzle.

In future, we are interested in extending our method to recognize and solve puzzles with larger number of tiles. For example, the proposed model can be used in the crossover operator to identify good parts in a solution and pass them into next generation.

Acknowledgment

This research is supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07048338).

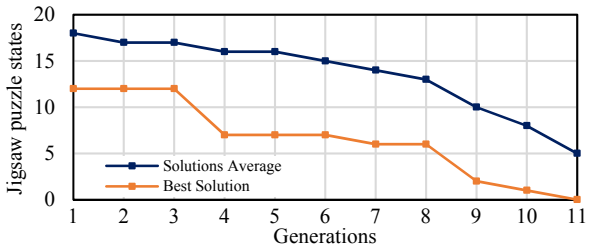

Fig. 3. The evolution of the genetic algorithm to find a solution.

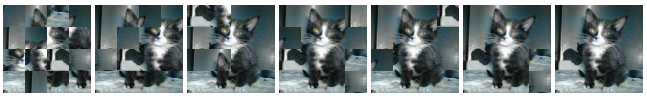

Fig. 4. As genetic algorithm converges to the solution. The original puzzle and the best solution in the generation 1, 4, 7, 9, 10, and 11 (left to right).

References

[1] D. Sholomon, E. David, and N. S. Netanyahu, "DNN-Buddies: A Deep Neural Network-Based Estimation Metric for the Jigsaw Puzzle Problem," *ArXiv171108762 Cs Stat*, vol. 9887, pp. 170–178, 2016, doi: 10.1007/978-3-319-44781-0_21.

[2] T. S. Cho, S. Avidan, and W. T. Freeman, "A probabilistic image jigsaw puzzle solver," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, Jun. 2010, pp. 183–190. doi: 10.1109/CVPR.2010.5540212.

[3] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," in *CVPR 2011*, Colorado Springs, CO, USA, Jun. 2011, pp. 9–16. doi: 10.1109/CVPR.2011.5995331.

[4] D. Sholomon, O. David, and N. S. Netanyahu, "A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, Jun. 2013, pp. 1767–1774. doi: 10.1109/CVPR.2013.231.

[5] D. Sholomon, O. E. David, and N. S. Netanyahu, "Genetic algorithm-based solver for very large multiple jigsaw puzzles of unknown dimensions and piece orientation," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, Vancouver BC Canada, Jul. 2014, pp. 1191–1198. doi: 10.1145/2576768.2598289.

[6] A. C. Gallagher, "Jigsaw puzzles with pieces of unknown orientation," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, Jun. 2012, pp. 382–389. doi: 10.1109/CVPR.2012.6247699.

[7] G. Paikin and A. Tal, "Solving multiple square jigsaw puzzles with missing pieces," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 4832–4839. doi: 10.1109/CVPR.2015.7299116.

[8] M.-M. Paumard, D. Picard, and H. Tabia, "Deepzzle: Solving Visual Jigsaw Puzzles With Deep Learning and Shortest Path Optimization," *IEEE Trans. Image Process.*, vol. 29, pp. 3569–3581, 2020, doi: 10.1109/TIP.2019.2963378.

[9] V. Huroyan, G. Lerman, and H.-T. Wu, "Solving Jigsaw Puzzles by the Graph Connection Laplacian," *SIAM J. Imaging Sci.*, vol. 13, no. 4, pp. 1717–1753, Jan. 2020, doi: 10.1137/19M1290760.

[10] D. Rika, D. Sholomon, E. (Omid) David, and N. S. Netanyahu, "A novel hybrid scheme using genetic algorithms and deep learning for the reconstruction of portuguese tile panels," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Prague Czech Republic, Jul. 2019, pp. 1319–1327. doi: 10.1145/3321707.3321821.

[11] D. Bridger, D. Danon, and A. Tal, "Solving Jigsaw Puzzles With Eroded Boundaries," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 3523–3532. doi: 10.1109/CVPR42600.2020.00358.

[12] R. Li, S. Liu, G. Wang, G. Liu, and B. Zeng, "JigsawGAN: Auxiliary Learning for Solving Jigsaw Puzzles With Generative Adversarial Networks," *IEEE Trans. Image Process.*, vol. 31, pp. 513–524, 2022, doi: 10.1109/TIP.2021.3120052.

[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv14091556 Cs*, Apr. 2015, Accessed: Mar. 08, 2021. [Online]. Available: http://arxiv.org/abs/1409.1556