Jigsaws and How to Solve Them

James Ngai
Carnegie-Mellon University
jdngai@andrew.cmu.edu

Lawrence Feng
Carnegie-Mellon University
lawrencf@andrew.cmu.edu

Eray Can Elumar
Carnegie-Mellon University
eelumar@andrew.cmu.edu

Shreya Terupally
Carnegie-Mellon University
sterupal@andrew.cmu.edu

Abstract

The reassembly of fragmented images, akin to solving a jigsaw puzzle, presents a challenging problem in computer vision and machine learning due to its visual, semantic, and spatial nature. In addition, the number of permutations of a jigsaw puzzle grows tremendously as the number of pieces grows. In this paper, we take inspiration from *AlphaGo's* variation of a Monte Carlo tree search. We use policy gradient methods to train our policy network, assigning probabilities to state-action pairs, and use supervised learning to train our value network, assigning values to states, both of which are deep convolutional neural networks. In addition, we leverage the episodic and exploratory nature of reinforcement learning and reframe the jigsaw puzzle as a sequential swapping game. The aim is to demonstrate the application of reinforcement learning to an image manipulation task.

1 Introduction

Image manipulation is a broad category of problems studied in computer vision and machine learning, including image denoising, translation, and transformation. Image manipulation can be used for solving problems in medicine, perception, and more. Here, we focus on the assembly of jigsaw puzzles as our image manipulation task. We wish to demonstrate the application of reinforcement learning to such an image manipulation task.

A jigsaw puzzle is a tiling puzzle that requires the assembly of interlocking and tessellating pieces. Each piece has a small part of a picture on it, and thus learnable, local features; when complete, a jigsaw puzzle produces a complete picture with emergent global properties. This problem is challenging because it requires the model to recognize local and global patterns, understand spatial relationships, make sense of fragmented information, and act with a tremendous search space. Solving the jigsaw puzzle problem using machine learning is an area of research that explores techniques for effectively integrating information from different parts of an input to improve overall understanding and performance in tasks such as image segmentation, object recognition, or scene understanding. Earlier reassembly algorithms utilized shape [11], contour, or color [8] of the pieces to match adjacent pieces. With the rapid advancements in machine learning, numerous approaches have been used for jigsaw puzzle reassembly such as convolutional neural network (CNN) based methods [6], generative adversarial network-based methods [4], and reinforcement learning [2].

We generate 3×3 jigsaw puzzles from the MNIST dataset. Though MNIST images are 28×28, we form our jigsaw puzzle using nine 9×9 pieces. The digits in the MNIST dataset are centered, and so the loss of some edge pixels is irrelevant. CNNs underlie our puzzle-solving architecture. We are inspired by *AlphaGo's* Monte Carlo tree search (MCTS) variant, which utilizes value and policy networks to reduce the depth and breadth of the tree search, respectively.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

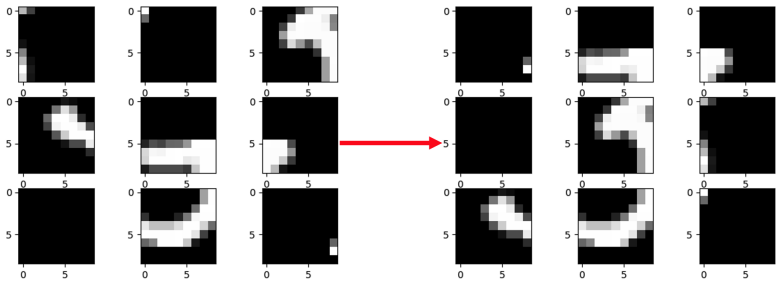

Figure 1: The jigsaw reassembly problem

2 Literature Review

In the domain of reassembly of images altered by fragmentation and permutation, the current landscape predominantly manifests a convergence between CNN methodologies and Reinforcement Learning (RL) paradigms. Pioneering strides have been made by Le et al. [3] and Paumard et al. [5], wherein the utilization of ConvNets has been instrumental in accurately discerning and repositioning image fragments, thereby effectuating image reassembly.

However, with the development of deep reinforcement learning, leveraging the seminal work of Sutton et al. [10], we have seen the application of reinforcement learning to countless tasks, particularly games.

Recent advancements have witnessed the assimilation of the AlphaZero algorithm [7] into the reassembly framework, as elucidated by Gras [2]. On the other hand, Song et al. [9] utilize Deep Q-Networks instead, another recent development in the field of reinforcement learning. These studies underscore the burgeoning interest in employing RL-based methodologies for the precise reconstruction of fragmented images.

In this vein, our current work draws significant inspiration from the foundational contributions of Sutton et al. [10] and Silver et al. [7], while also building upon the advancements put forth by [2]. Our baseline implementation is crafted by integrating key insights from these seminal works, striving to harness the collective potential of RL and CNNs for optimal image reassembly in fragmented and permuted scenarios.

3 Model Description and Baseline Selection

In this paper, we assume that an image is divided into an $m \times m$ grid of same-sized fragments. Letting I denote the image, we use f_i to denote the i^{th} fragment of the image, and we use $I_r(t)$ to denote the current state of the reconstructed image. We assume that at $t = 0$, $I_r(t = 0)$ is a blank (zero-valued) image and that the fragments of the image are shuffled using a permutation function P. In each round, an action $a_t = (f_i, j)$ can be taken. The action $a_t = (f_i, j)$ means placing fragment f_i at j^{th} position. When the fragment f_i is placed at j^{th} position, the j^{th} position of the reconstructed image $I_r(t)$ is updated with this fragment. The final goal of this model is to reconstruct the original image this way such that $I_r(m^2) = I$, or equivalently to find the inverse permutation function P^{-1} to reshuffle the fragments back to its original position. An example shuffled image and its reconstruction is given in Fig. 1

We can measure the performance of any model performing this image reassembly task by simply assessing the percentage of images that are reassembled completely correctly.

We have identified two baselines for this project. The first baseline [1] utilizes a convolutional neural network architecture that represents the general approach that has been dominantly used for this task, and the second baseline we have identified [2] uses reinforcement learning in its implementation that represents the recent advancements in this field. While convolutional networks can solve the problem for 2×2 or 3×3, exponential size of possible rearrangements necessitates the use of reinforcement learning for larger grid sizes.

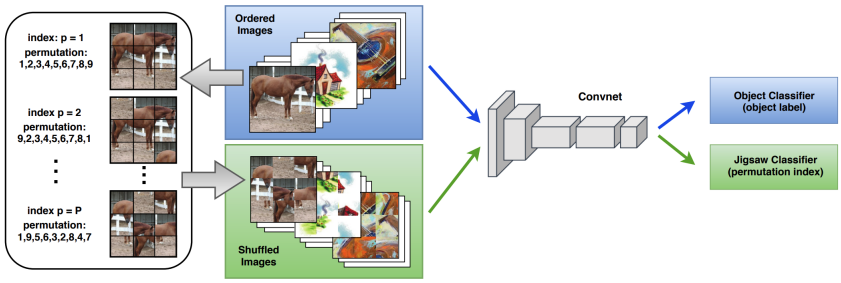

Figure 2: The model overview of the baseline 1

3.1 The Baseline Model 1

The first baseline model is a dual objective system that adds jigsaw puzzle solving as a secondary objective to improve accuracy on an image classification task. In this model, the original image and a shuffled image are fed as input, and the network aims to classify the image using the original image, and tries to reassemble the original image from the shuffled image. The overview of this network is given in Fig. 2. The loss function is defined as the linear combination of the cross-entropy losses of both objectives.

For training, the model uses a pre-trained convolutional network such as ResNet or AlexNet, removes the last fully connected layer, and substitutes it with the new object and classification layers.

3.2 The Baseline Model 2

This baseline model uses a model-based reinforcement learning technique with Monte Carlo Tree Search to search to estimate the value of the reassembly states and find the optimal trajectory that correctly reassembles the image. The overview of the architecture of this baseline is given in Fig. 3.

To guide our reinforcement learning model to the optimal solution, we use a reward function r_t which gives a reward of $\frac{1}{m^2}$ is the fragment is placed in its correct position, and zero otherwise. This can be written as:

$$r(t) \begin{cases} \frac{1}{m^2}, \text{ if } a_t \in \mathcal{A}^* \\ 0, \text{ if } a_t \notin \mathcal{A}^* \end{cases} \tag{1}$$

where $\mathcal{A}^* = \{(f_1, P^{-1}(1)), (f_2, P^{-1}(2)), \cdots, (f_{m^2}, P^{-1}(m^2))\}$ is the set of optimal actions (actions that place the fragment in its correct position). The final reward of the reconstruction is the total number of correctly identified pieces which can be written as $R = \sum_{t=1}^{m^2} r(t)$.

The baseline model consists of two networks; the Policy Network (PN) and the Value Network (VN). The policy network takes the current state of the assembled puzzle, and a fragment as input, and computes the move probability for the fragment for each of the remaining spaces. The current state of the assembled puzzle consists of fragments that are already placed into the puzzle, and blank image for the places that don't have a fragment placed on it. This current state of the assembled puzzle is fed into the Feature Extraction Network 1 which consists of convolutional layers, and a remaining fragment is fed into the Feature Extraction Network 2 which also consists of convolutional layers to extract features in these images. Then, the Combination Layer which consists of a bilinear layer is used to capture the spacial covariances among the features of both images. The output of this network is flattened and a softmax activation is used to get the move probabilities of the fragment to any empty space in the current puzzle.

Let θ denote the parameters of our PN. Let ρ denote the performance (the expected reward) of the given policy. Let α correspond to the learning rate. We update the parameters of the policy network using the *policy gradient* approach.

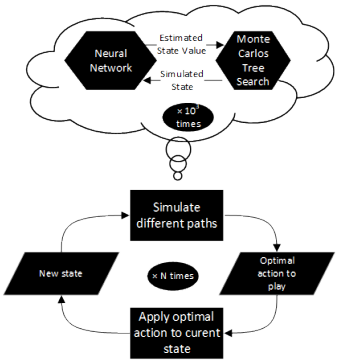

Figure 3: The model overview of baseline 2

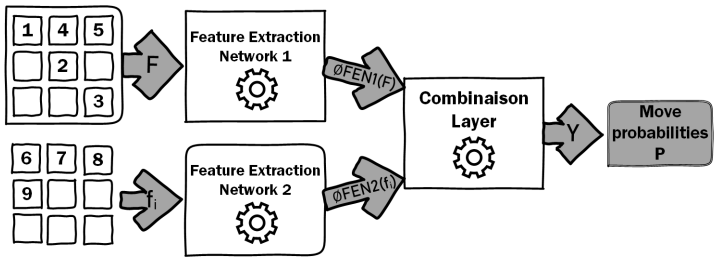

Figure 4: The policy network

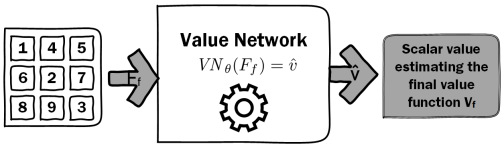

Figure 5: The value network

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta} \tag{2}$$

The architecture of the PN is given in Fig. 4.

The VN is simply a convolutional network that takes the final reassembled image $I_r(m^2)$ as input and outputs a scalar score that tells how similar reconstructed image is to the original image. The architecture of the VN is given in Fig. 5.

In this framework, first MCTS randomly simulates different paths of actions and uses the PN to estimate the value of these different paths, and the optimal action is estimated from these paths. Then, this optimal action is applied, and the same process starts again until assigning every fragment to a position. Also, after each simulation the final state value estimated by the VN is backpropagated. This lets the algorithm reassemble the puzzles without having the ground truth after the training phase.

4 Baseline Implementation

We implemented the Baseline model 1 with the following training parameters:

- Dataset: PACS
- Batch size: 128
- Pretrained network: resnet18

- Image Augmentations: color jitter, random horizontal flip, random grayscale
- Learning rate: 0.001

After training the network with these parameters, the accuracy we obtained is 82.14%.

5 FINAL(edit name) Baseline Implementation

Here introduce general model architecture

5.1 Policy Network

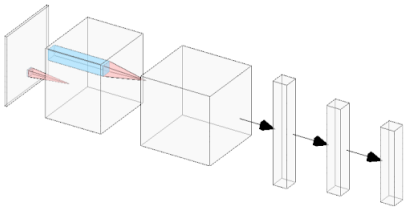

Figure 6: Convolutional Neural Network

Architecture Ablation Summary

Model	Hidden Layers	Parameters
1layerCNN_1Dense	$1 \times$ {CNN, Maxpooling, Batchnorm}, 1 Dense	Kernel Size (3)
2layerCNN_1Dense	$2 \times$ {CNN, Maxpooling, Batchnorm}, 1 Dense	Kernel Size (3,5)
3layerCNN_1Dense	$3 \times$ {CNN, Maxpooling, Batchnorm}, 1 Dense	Kernel Size (3,5,9)
1layerCNN_2Dense	$1 \times$ {CNN, Maxpooling, Batchnorm}, 2 Dense	Kernel Size (3)
2layerCNN_2Dense	$2 \times$ {CNN, Maxpooling, Batchnorm}, 2 Dense	Kernel Size (3,5)
3layerCNN_2Dense	$3 \times$ {CNN, Maxpooling, Batchnorm}, 2 Dense	Kernel Size (3,5,9)

Table 1: Policy Network Architecture Ablations

The highest performing model architecture was the 3layerCNN with 2 Dense layers.

5.2 Value Network

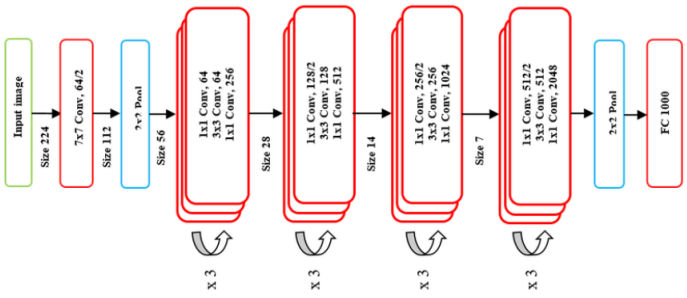

Figure 7: Resnet Architecture

Architecture Ablation Summary

The highest performing model architecture was the 1Resnet18_2Dense.

Model	Hidden Layers	Parameters
3layerCNN_2Dense	3 × {CNN, Maxpooling, Batchnorm}, 2 Dense	Kernel Size (3,5,9)
1Resnet18_2Dense	3 × {Resnet18, 2 Linear, Softmax}	Resnet

Table 2: Value Network Architecture Ablations

5.3 Monte Carlo Tree Search

6 Training

6.1 Loss Function

7 Proposed Extensions

The team has identified several clear ideas for extending the baseline into the final project. Within the domain of JigSaw Reassembly, our research aims to harness the power of reinforcement learning for the sequential reassembly of images from disordered fragments. Building upon the work of Gras [2], who introduced a rewarding mechanism assigning $1/n$ for correctly placed pieces, our team has identified pivotal areas for further enhancement in this paradigm.

Our critical examination revealed certain constraints within the prescribed reward structure, which overly supervised the RL agent's learning process. Additionally, the lack of incentivization for the reconstruction of semantically equivalent images hindered the effectiveness of the reward function. To surmount these limitations, our team is actively developing a novel reward scheme that evaluates the semantic similarity between the original and reassembled images.

Another limitation we encountered was the mismatch between reinforcement learning and the non-sequential nature of the game proposed in Gras [2]. In response, we are proposing a substantial paradigm shift by modifying the fundamental gameplay to better align with RL's sequential learning strengths. Our approach involves adapting the RL agent to engage in an image reassembly game allowing the swapping of any puzzle piece with its adjacent counterpart. This revised gameplay facilitates a sequence of permissible moves for image reconstruction, aiming to minimize the RL agent's steps and achieve mathematical optimality in reassembly.

The objective is to enable the RL agent to learn and execute optimal strategies in reconstructing fragmented images, thereby contributing to enhanced generalizability and efficiency in this domain.

Acknowledgments and Disclosure of Funding

We thank our TAs Dheeraj Pai and Rucha Manoj Kulkarni for their guidance and support on this project.

References

[1] Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019.

[2] Johan Gras. Puzzle reassembly using model based reinforcement learning. 2019. URL `https://johan-gras.github.io/projects/puzzlereassembly/`. `https://johan-gras.github.io/projects/puzzlereassembly/`.

[3] Canyu Le and Xin Li. Jigsawnet: Shredded image reassembly using convolutional neural network and loop-based composition. *arXiv preprint arXiv:1809.04137*, 2018.

[4] Ru Li, Shuaicheng Liu, Guangfu Wang, Guanghui Liu, and Bing Zeng. Jigsawgan: Auxiliary learning for solving jigsaw puzzles with generative adversarial networks. *IEEE Transactions on Image Processing*, 31:513–524, 2021.

[5] M. M. Paumard, D. Picard, and H. Tabia. Image reassembly combining deep learning and shortest path problem, 2018.

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

[7] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961.

[8] Kilho Son, James Hays, and David B Cooper. Solving square jigsaw puzzles with loop constraints. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 32–46. Springer, 2014.

[9] Xingke Song, Jiahuan Jin, Chenglin Yao, Shihe Wang, Jianfeng Ren, and Ruibin Bai. Siamese-discriminant deep reinforcement learning for solving jigsaw puzzles with large eroded gaps. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(2):2303–2311, Jun. 2023. doi: 10.1609/aaai.v37i2.25325. URL `https://ojs.aaai.org/index.php/AAAI/article/view/25325`.

[10] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL `https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf`.

[11] Kang Zhang, Wuyi Yu, Mary Manhein, Warren Waggenspack, and Xin Li. 3d fragment reassembly using integrated template guidance and fracture-region matching. In *Proceedings of the IEEE international conference on computer vision*, pages 2138–2146, 2015.