Puzzle reassembly using deep reinforcement learning with adversarial model

Loïc Bachelot
University of Cergy-Pontoise
loic.bachelot@etu.u-cergy.fr

Supervised by Marie-Morgane Paumard
and David Picard
marie-morgane.paumard@ensea.fr
david.picard@ensea.fr

Abstract

The goal of this paper is to propose a method to automatically re-assemble archaeological artifacts from fragments to help archaeologists in their work. We suggest that, fragments reassembly can be simplified to a puzzle-reconstruction problem, which may be solved by computer vision algorithms. In this paper we discuss the reassembly from 9 fragments. Given an unordered set of fragments, we try to merge them back to the original image. The main contributions of this work is the comparison between the four following methods: two discriminator-based reassembly methods using deep learning to predict the probability of the positions for each available fragments; a Monte Carlo Tree Search (MCTS) based method inspired by AlphaZero to reconstruct the image; and a combined method based on the MCTS using a discriminator as the value prediction. The results show the benefits of using both a discriminator and an MCTS algorithm.

1. Introduction

Automatic puzzle reconstruction is a very large problem depending on the type of puzzle. In this paper, we only consider visual puzzles from 2D images to 3D objects, but we expect our work to be extended to non-vision puzzles such as DNA reassembly. It is especially interesting to use automatic puzzle reassembly in archaeology as broken objects usually come with more than a thousand pieces, with missing fragments and eroded borders. Reassembling such puzzles is taking years to humans whereas it can be done automatically by a computer. This grants the archaeologists the time to focus on the interpretation of artifacts.

During the past years, computer vision has improved with the use of deep learning, helping the puzzle-solving methods to achieve better results.

In this paper, we simplify the reconstruction of archaeological puzzle into the reconstruction of 2D images. We split the images into 9 squared fragments. We aim to predict the exact position of every fragment on the final image. We add erosion to our fragments to make the task more difficult by adding space between the fragments. Using erosion is useful to unsure that the method cannot rely on the information contained in the borders to perform the reconstruction. It is important to us because we aim at the reconstruction of archaeological puzzles, potentially very old and in bad shape. The last constrain is an iterative reconstruction, we add the fragments one by one to the finale image. We have in Figure 1 an example of a reconstruction from the MNIST dataset.

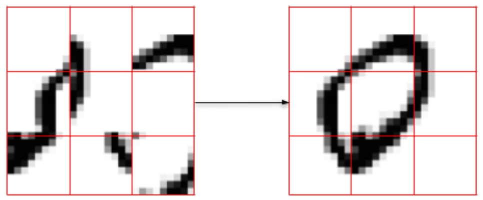
Figure 1. Image reconstruction from 9 unorganized fragments. On the left, a MNIST image is divided in 9 shuffled pieces. On the right side, it is perfectly reassembled.

The reassembly task is characteristic of reinforcement learning problems. The first method we present is a MCTS-based method, inspired by AlphaZero. However these methods require supervision during the training in the form of the puzzle solution. We tested two discriminator-learning-based techniques to get ride of this requirement. These three methods have been tested independently on the MNIST dataset. Finally, we merged the methods to obtain our proposition of a MCTS trained with a discriminator.This last technique has been tested on MNIST, and also on real car images with eroded fragments.

The paper is organized as follow: in section 2, we present related work on puzzle solving and on adversarial models. We give a brief overview of the Monte Carlo Tree Search though AlphaZero. Next, we detail the three different methods in the section 3 starting with the MCTS/DQN. Then, we continue with the discriminative value estimation

(DVE) and we finish with the MCTS using a discriminative value estimation. In section 4, we present the experiences on MNIST comparing the three techniques, before going further with the setups and analyze the MCTS-DVE on another datasets and using a different experience setup. We conclude by discussing the different results.

2. Related work

Our proposed methods are based on some previous work in three different areas. First we discuss puzzle reassembly techniques. Then, we give an overview of adversarial learning to predict the value function. We finish with a brief section on Monte Carlo Tree Search with AlphaZero.

2.1. Puzzle reconstruction

Visual puzzle reconstruction method can be divided into two categories, namely precise reassembly which aim to perfectly align the fragments, and global positioning which aim to place the fragments next to each other. For the first category, [9], [6] and [13] propose a method to automatically reassemble archaeological puzzles, aiming for a perfect alignment. But these techniques mainly rely on the borders — the shape of the fragments, to solve the problem.

On the other hand, [5], [1], [8] simplify the problem as we do, trying to solve jigsaw puzzles with missing fragments, and fragments of different size. They have the advantage on not relying on the shape of the fragments anymore, but they still rely on information contained by the borders to solve it. This information might be missing in our case of antic puzzles. Moreover, these methods require a costly human preparation in order to succeed.

We also look at the deep learning approach from Doersch et al. in [3]. In this paper the authors are not interested in puzzle-solving but they aim to get the network understand semantic of the images. To do so, they train the network to predict the relative position on two fragments based on the features. They use this as a pre-training task for their CNN because of the possibility to auto-generate the ground truth. This task is helping the network understand the context of an image, and therefore it can have better results for classification tasks. Noroozi and Favaro [10] compare the 9 fragments of the jigsaw puzzle in the same time, basing their work on [3].

Paumard et al. [11] propose a deep neural network specifically to solve jigsaw puzzles, based on the work of Doersch et al. [3]. In this work, fragments are cropped in an image leaving gaps of 1/2 of the fragment size between the fragments to be sure the network cannot rely on the borders to predict the position. They also have missing pieces and pieces coming from other puzzles. The technique is composed of two steps: first they compare every fragments to the central one using CNNs to extract the features and they use a full bilinear product with the Kronecker product of the features. The last step is a classification between the 8 positions around the central fragment and an extra 9th class for fragments not related to the puzzle. To reconstruct the image from this probability matrix they cast the problem as a shortest path problem. In order to train the model to reconstruct puzzles, they are using a dataset provided by the Metropolitan Museum of Art (MET) composed of 14,000 images.

In our work we aim to solve the same problem as in [11], but instead of comparing the fragments together, we compare the fragments to the current image reconstruction and we add the fragments step by step.

2.2. Adversarial learning and value function estimator

In this part we are introducing some notions of adversarial learning and value function estimator.

The Actor-Critic algorithms proposed by Vijay R. Konda and John N. Tsitsiklis [7] is a base of our discriminator based technique. Instead of using an actor and train it directly with the ground truth to train the network, they are using another network called critic. This method aim to train the actor using the value function estimated by the critic network.

In our case, we combine the actor-critic idea with the ideas proposed by Ian J. Goodfellow [4]. In this article, they propose a two networks model: a generator G to capture the data distribution and a discriminator D to evaluate if the input is coming from the generator or from natural data. Then, the network G is aiming to maximize the error of D.

In our model, the generator is reassembling the puzzle and the discriminator is comparing it to real images. The difference is we do not train the generator with the error of the discriminator but we use its value. This Generative Adversarial Network technique is very efficient because it can be trained without having the ground truth (puzzle solution). Because it is a indirect supervised method through the training dataset of the discriminator, we can train the generator to generate lookalike results to the eyes of the discriminator. In our case we will use real images to train the discriminator and the generator will aim to reconstruct puzzles such that they look like real images. Moreover, we do not need the ground truth for the puzzle we are trying to reconstruct, we only need similar images. The last major difference, in most of the GAN used, the input of the generator is random noise, and the generator is completely imagining the result. In our case, the input will be the fragments of the puzzle.

2.3. Monte Carlo Tree Search

Solving puzzles can also be seen as searching for the best sequence of fragment-position pair in a tree. This is why studying the Monte Carlo Tree Search [2] method is very interesting. We focused on the AlphaZero [12] version because of its use of a value estimation to explore the tree. For further details, refer to Gras's research project: "Puzzle Reassembly using Model Based Reinforcement Learning"

3. Proposed method

In this section, we introduce briefly the MCTS method before focusing on the two discriminator-value-estimator-based techniques and finishing by the merged method with the MCTS and the DVE.

3.1. Puzzle reassembly using MCTS/DQN

The MCTS is an oriented tree exploration. In our case, every level of the tree in composed by 81 branches representing all the fragment-position pairs. Every exploration is scored by a value estimator. The value estimator is predicting the quality of the reassembly using the ground truth: It is predicting how many fragments are well positioned.

After an exploration phase, the network choose the most promising option and position the fragment on the image before starting again a level deeper in the tree.

For the details on this method, refer to Gras's paper: Puzzle Reassembly using Model Based Reinforcement Learning.

3.2. Puzzle reassembly using discriminative value estimation

The idea of this technique is to train the network responsible of the puzzle reconstruction through a classifier. The classifier is trying to make the difference between real images and images coming from the generator. Then the prediction of the discriminator is used to train the generator. The more accurate the image is, the higher is the score. The generator is trying to predict the output of the discriminator for every fragment-position pair possible.

3.2.1 Generator

The generator is the network reconstructing an image. It takes as an input the fragments f_i and an image in reconstruction F (empty at the beginning). Both are going through some feature extractor networks. Then, they are combined with a bilinear product:

$$P_{ic} = F^T W_c f_i,$$

where P_{ic} is the probability of the fragment i to be in the position c and W_c is the weight matrix of shape 9 by the dimension of the encoding for f_i by the dimension of the encoding for F_i.

The feature extractors are problem dependent. We did not use the same CNN to work on MNIST and to work on bigger images. They are here to encode the semantic of the image. The generator is trying to predict the output of the

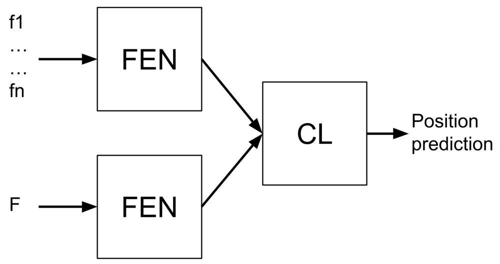

Figure 2. The generator architecture: the two FEN are the different feature extractors for the f_i and the F_i. They are problem dependent and can go from a simple convolution to a VGG16 depending on the image. The combination layer CL is a bilinear product.

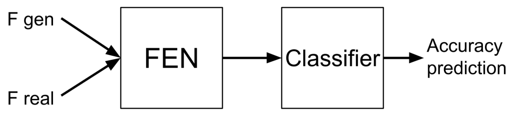

Figure 3. Discriminator architecture: the FEN network in the feature extractor. It is important to adapt it to the problem to avoid basing the classification on none relevant features. The classifier part of the network is a simple multi-layer perceptron. $F\ gen$ are images coming from the generator and $F\ real$ are images coming from the training dataset

discriminator for the generated image for each fragment-position pair. The output is a 9 by 9 matrix for the 9 fragments and the 9 positions. As the image is more and more reconstructed, some fragments are not available anymore, just like some positions. These options in the matrix are not considered anymore. Then, we use the probability matrix to add a fragment to the image and repeat the operation. There is no interactions between the fragments. The network cannot predict the position of a fragment with respect to another. It is predicting the position in the reconstructed image. Therefor, the first step is very difficult because the network has no other information than the fragments themselves.

3.2.2 Discriminator

The discriminator is a classifier trained to classify whether an image is coming from the generator or if it is a real image. It predicts a score between 0 for a generated image to 1 for a real one.

The discriminator is very problem dependent and it needs to stay simple enough to avoid over-fitting (a good image could be badly rated just because it is not part of the discriminator training dataset). However, it also needs to be accurate enough to give good indications to the generator. During the training, the good reconstructions from the generator are getting a high score (close to 1) because they are visually similar a real image and the bad reconstructions are easy to recognize and get a bad score. The training dataset of the discriminator is very important. This is what the generator will try to get the puzzles look like. If the training dataset is too far from the puzzle we are trying to reconstruct with the generator, the discriminator will base his classification on irrelevant features and the generator will never output good reconstructions.

3.2.3 Training

The training loop is composed by 4 independent steps:

- The episode generation: the generator is taking as an input the different fragments and a empty image of the size of 9 fragments. We set the values to -0.1 on the empty image to avoid confusion with black fragments. Then the generator is predicting the score for each fragment-position pair. We take the highest score and add the corresponding fragment in the good position in the image. We repeat the operation until there is no more fragments. We store every steps in a memory: action chosen and the fi, Fi inputs. We use a random exploration at the beginning of the training. It means we do not always pick the highest score. This exploration is disappearing through time.

- The evaluation of the episodes: We are picking some episodes in the memory and get them evaluate by the discriminator. With this evaluation we produce some training data for the generator. Because we can only evaluate one reconstruction — one fragment-position choice, the target of the generator is a vector of size $number\_of\_fragments \cdot number\_of\_positions$. All the values are 0 except the one where we have the score.

- The training of the discriminator: The training data are composed by 50% of real images and 50% of images coming from the memory, generated by the generator. The real images are coming from a different dataset than the one we are cropping our fragments from. The real images have a target of 1 and the generated a target of 0.

- The training of the generator: In the same time, we train the generator with the scored episodes from the discriminator. Therefor, the loss is a MSE multiplied by the label.

We defined two different training methods, each with its own advantages. The first one is scoring every step of the reconstruction independently. The discriminator is also trained to recognize if a partial reconstruction is coming from a real image or from the generator. The second is scoring only the final reconstruction. The first method is better for partial reconstructions because we have individual score for each step. So in case of missing fragment the system knows better how to act. Moreover, it allows the system to have a better average well positioned fragments. But putting a score to an image with one fragment does not have much sens. And we noticed the discriminator can base his classification on non relevant features for the first steps, such as the two first fragments are not touching. This would lead the generator to try to put the two first fragments not together and probably adding more difficulty.

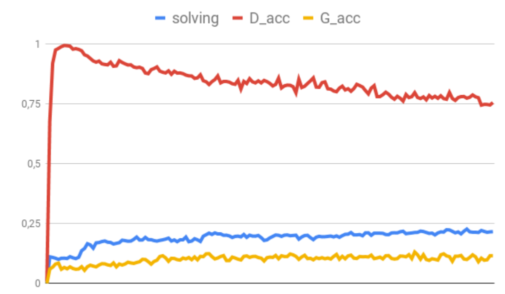

Figure 4. Training dynamic: there is a first part where both the accuracy of the discriminator (accD) and of the number of well positioned fragments (solving) are increasing. The accuracy of the generator is quite stable for the complete training because it is trying to predict the discriminator's output but it keeps changing. The number of well positioned fragments is stabilizing as the accuracy of the discriminator is too low to really help the system do better.

This is why we tried the second technique. Scoring only he final reconstruction and propagating the score it as a reward for the previous steps. The problems with this method is the difficulty to get a perfect reconstruction. Indeed, the probability is only of 1/362880, but if we consider that the discriminator is not good at the beginning, we can say that images with 6 fragments well positioned will get a good score, we reduce the probability of a good image to 1/60480.

It is important to train both the discriminator and the generator in the same time. As the generator is trying to predict the output of the discriminator and not maximizing its score, if the discriminator is too good too fast, the generator is always predicting 0. By adjusting the learning speed of both, the generator is lowering the score of the bad reconstructions to 0 but the better ones are chosen and the score gets higher.

We can see on the Figure 4, the general dynamic of the training. The important thing to observe here is the limitation of the method. The lower the discriminator accuracy is, the less the general accuracy (solving) is increasing. We can also notice that the generator accuracy is stable.

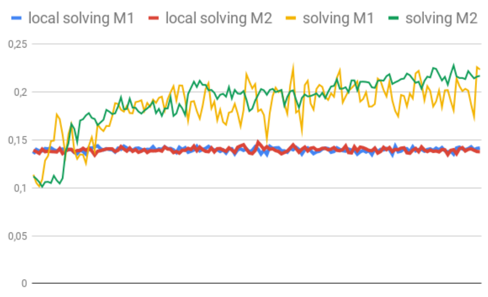

Figure 5. Solving: the solving metric is the percentage of perfectly position fragments (solving metric) and the relative position (local solving). The results are divided by method, the method 1 is the step by step score and the method 2 is the final score.

3.3. DVE-MCTS based DQN

We merged the two previous methods into one. We keep the MCTS architecture but instead of using a value estimator to estimate the potential gain based on the number of well positioned fragments, we use a discriminator as we saw in the DVE section. We use a discriminator exactly like the one of the method 2, only scoring the final images. The value in now propagated along the branches of the tree.

Combining the two techniques, we keep all of the advantages of the MCTS but we do not need any ground truth to train the network. It is helping the network to be less sensitive to missing pieces, unusual shapes as long as we present similar objects to the discriminator.

4. Experiences

First, we present the results on the MNIST dataset for all the techniques presented before, and we finish by some results on an other dataset using the combined method.

4.1. MNIST

We tested on the MNIST dataset because the images are small enough for us to train the complete network, including the FEN, and have relevant results to orient our work.

4.1.1 MCTS

With the MCTS we reach 70% accuracy for the fragments well positioned metric. For more information, refer to Gras's research project: Puzzle Reassembly using Model Based Reinforcement Learning.

4.1.2 Discriminator value estimator

On the figure 5, we can see that both of the methods provide similar results for the two metrics. The local solving metric is measuring the accuracy of the position of a fragment compare to the surrounding, and the solving metric is the percentage of well positioned fragments. We still can notice the better stability of the method 2. This can be explained by the fact that the generator is playing the fragment-position with the higher chance to be well rated by the discriminator directly, even if it is not making sense for the final image. For example, maybe playing two fragment next to each other for the first two is very easy to recognize for the discriminator and thus not selected by the generator. Where in the other hand the order for the second method does not matter because only the final image is evaluated.

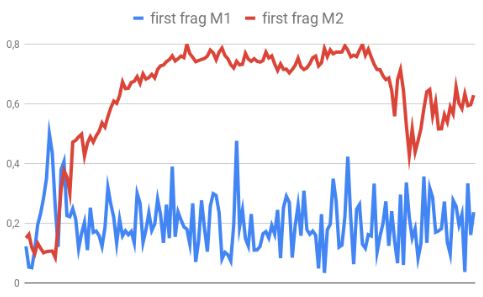

Figure 6. This graph is representing the accuracy of the position for the first fragment-position pair positioned.

We directly notice the difference between the two methods. But surprisingly, it it the method with the final scoring that gets the better results. This can be explained by the fact that in the method 1, the generator is playing what will instantly get a better score through the discriminator, so it is playing what the discriminator cannot classify. This is why playing randomly is possibly a good answer to the problem, because classifying an image with only one fragment is almost impossible. Whereas for the method 2 we can see the results are a lot better than random, even going up to 80% accuracy. We can notice it is decreasing in the end. The reason of this gap can be explained by a change of strategy of the network in choosing the first fragment by taking more risks. For example for the first part, it is choosing easier moves to start with and finishing by the more complex. In that case the image would be categorized based on the errors in the end. But after some time, some other moves have good scores to start. In the same time we do not notice any unusual change in any other metric. the overall solving is still increasing and none of the discriminator of generator is having a brutal change on the loss or accuracy.

With this results, we proved that it is possible to use a discriminative value estimator to supervise the training of a network in the puzzle reconstruction problem. But we also saw that compared to the MCTS the results are not very good with a limit at 22% of fragments well positioned average against 70%.

The images in 7 are some pathological cases where the

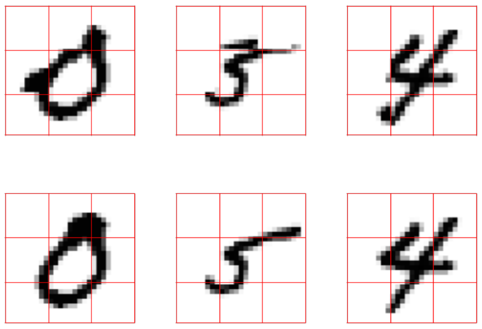

Figure 7. Cases of high values estimated by the discriminator. Top row: reconstruction , bottom row: perfect image.

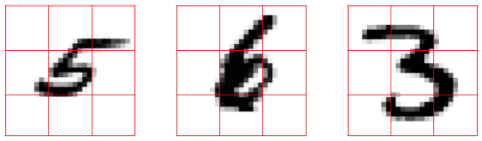

Figure 8. Perfect reconstructions

reconstructions are scored higher than 0.6 by the discriminator whereas they are not perfect. the network still reconstruct adjacent shapes in all of these cases, event correct numbers for the 0 and the 4. The problem is for the 5 where the network reconstruct half a 3 and half a 5. The images in 8 are examples of perfect reconstructions. It is interesting to see that the network in learning how to reconstruct every number and not only the easiest like the ones or zeros.

4.1.3 DVE-MCTS

On the MNIST dataset, the results are similar to what we had with the MCTS, but without using the ground truth for the training. We reach 70% accuracy for the well positioned fragments.

4.1.4 Comparison

To conclude on the experiences with the MNIST dataset, we can see on the graph 9 that the difference between the algorithms is not very important for the perfect reconstruction metric. Noticing the gap between the solving metric and the perfect reconstruction for the DVE algorithms, we can conclude that the reconstructions are either perfect or completely wrong. This can come from the discriminator: it is only saying if it is perfect or completely wrong. But, it can also come from the lack of information for the first steps, so the network is only able to reconstruct perfectly if by chance the first steps were good, but if not, the probabilities are all going to be 0 so it is picking almost randomly.

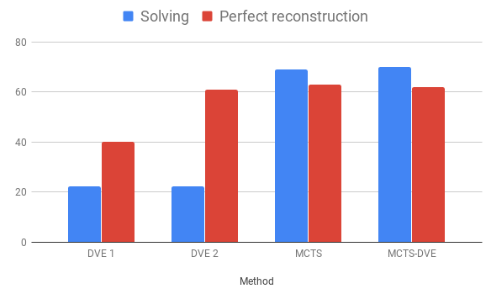

Figure 9. Solving is the accuracy of fragments well positioned, and perfect reconstruction is the ration of perfect reconstructions. The results are obtained on MNIST

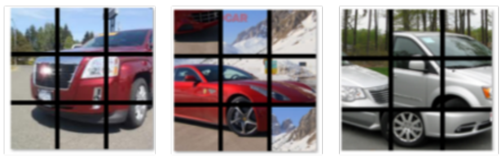

Figure 10. Promising results on the car dataset.

The second option is verified because we do not have this appearing with the MCTS-DVE method.

4.2. Car dataset with MCTS-DVE

After combining the methods, we tested our algorithm on a car dataset using a VGG16 pre-trained on ImageNet as a feature extractor. We show in 10 some promising reconstructions. On this dataset, with a really short training time, we achieve 20% accuracy for the well positioned fragments and we have some perfect reconstructions.

5. Conclusion

In this paper, we proved that an adversarial reinforcement learning technique can be efficient in puzzle solving problems through the two discriminator-based methods. First, we showed that the step by step reassembling performs poorly. We suppose that the issues with positioning the first fragment can be solved by using the relative position instead of the exact position, in such a way that the network cannot be wrong on the first pick. The work on the MCTS showed that it is a very promising track and combining the MCTS with a value estimation by a discriminator is also working. Balancing the training and the performance of the discriminator is the key to the convergence to a good solution. The next step is scaling to bigger puzzles using the MET dataset to train and test the combined algorithm MCTS-DVE.

References

[1] F. Andalo, G. Taubin, and S. Goldenstein. Puzzle solving by quadratic programming. *IEEE TPAMI 39*, 2017. 2

[2] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game ai. In *AIIDE*, 2008. 2

[3] C. Doersch, A. Gupta, and A. Efros. Unsupervised visual representation learning by context prediction. *ICCV*, 2015. 2

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

[5] Z. Hammoudeh and C. Pollett. Clustering-based, fully automated mixed-bag jigsaw puzzle solving. *Computer Analysis of Images and Patterns*, 2017. 2

[6] F. Jampy, A. Hostein, E. Fauvet, O. Laligant, and F. Truchetet. 3d puzzle reconstruction for archeological fragments. *3DIPM*, 2015. 2

[7] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000. 2

[8] C. Lifang, D. Cao, and Y. Liu. A new intelligent jigsaw puzzle algorithm base on mixed similarity and symbol matrix. *IJPRAI*, 2018. 2

[9] J. McBride and B. Kimia. Archaeological fragment reconstruction using curvematching. *CVPRW*, 2003. 2

[10] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV*, 2015. 2

[11] M.-M. Paumard, D. Picard, and H. Tabia. Image reassembly combining deep learning and shortest path problem. *ECCV*, 2018. 2

[12] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 2

[13] L. Zhu, Z. Zhou, J. Zhang, and D. Hu. A partial curve matching method for automatic reassembly of 2d fragments. *ICI LNCIS 345*, 2006. 2