Using Vision Transformer to Solve Jigsaw Puzzles

Wu Yuyang
Tsinghua University
yy-wu23@mails.tsinghua.edu.cn

Abstract

Jigsaw puzzle is important in many applications. In computer vision field, it is also a pretext task that can help models learn good word embedding. We use methods like Jigsaw-ViT [4] to solve the problem with patch size 64 *and puzzle scale* 5×5*. Compared to previous work, we scale up patch size, embed the images by ResNet-18, modify the output layer, remove some unnecessary structures of the model, and explore the performance of this approach on larger patch sizes.*

1. Introduction

Jigsaw puzzle is important in many applications, including image editing [5], biology [11], archaeology [3] and recovering shredded documents or photographs [10] to name a few.

In the computer vision field, Jigsaw puzzle as a pretext task can help the model learn good word embedding [7] in self-supervised learning, which is also of certain significance.

Earlier methods used custom distance functions to judge the similarity of two puzzle pieces and used various algorithms to optimize the process of finding the optimal solution. Recent AI methods use methods such as CNN and GAN to calculate the distance between images.

We use methods like Jigsaw-ViT [4] to solve the problem with patch size 64 and puzzle scale 5×5. Compared to previous work, we scale up patch size, embed the images by ResNet-18, modify the output layer, remove some unnecessary structures of the model, and explore the performance of this approach on larger patch sizes.

2. Related Work

Some earlier work investigated solutions for Jigsaw Puzzles with many pieces (e.g., 300 pieces), using algorithms including random optimization [14], linear programming [16], and combinatorial optimization [8], [15]. The common characteristic is the use of simple distance functions to measure the similarity between adjacent puzzle pieces, without true understanding of the image semantics, and the dependence on the image cropping, which often leads to suboptimal results, such as placing the sky at the bottom of the image. For example, [14] measures the similarity of the pixels on the boundaries of two puzzle pieces, which requires no gaps between them, but the performance is still not satisfactory; [8] uses their proposed Mahalanobis Gradient Compatibility (MGC) that describes the local gradients near the boundary of a puzzle piece, which also relies on the continuity between puzzle pieces.

In 2015, [7] used CNNs to solve the problem of predicting the relative position of two 3×3 Jigsaw puzzle fragments, which can help neural networks learn a sense of spatial semantics. This method and subsequent ones do not require the puzzle pieces to have continuous boundaries.

The subsequent work can be roughly categorized into two types: focus on methods for reassembly, such as [12], which builds a graph of all possible permutations and solves for the shortest path. These methods mostly still handle the 3×3 case and have limitations for larger dimensions; focus on measurement, such as [2], which uses GANs to generate the missing boundaries of puzzle pieces and uses the discriminator to judge the similarity.

The latest work, such as [13], uses a method similar to AlphaZero's Monte Carlo Tree Search, only calculating the similarity of all adjacent pieces at the end to evaluate the goodness of the result; [4] uses Vision Transformer to directly obtain the correct arrangement in an end-to-end manner, which is also the method this paper aims to study.

3. Methods

Here is the subsection on problem formulation:

3.1. Problem Formulation

The problem we aim to solve is one with a fixed patch size and a fixed number of patches.

Viewed in reverse, this problem involves taking an image and cutting it into $L = n \times n$ patches of size $p \times p$, where n is a fixed value. Our task is to take these L $p \times p$ patches and

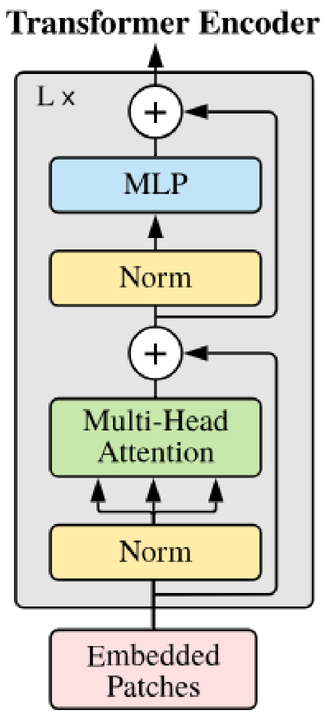

Figure 1. A layer of transformer layer of ViT.

find the correct arrangement that reconstructs the original image. Note that n and p are fixed in our formulation.

In the experiments that follow, p is generally set to 64, and n is generally set to either 4 or 5.

3.2. Model Architecture

We use a similar approach to Jigsaw-ViT [4].

Patch Embedding. In the original Jigsaw-ViT paper, the patch size was 16, so they used a simple linear projection as the patch embedding. However, since we want to solve the Jigsaw problem with $p = 64$, we use a ResNet-18 [9] as the embedding layer.

Transformer. We use the same transformer architecture as Jigsaw-ViT. See Figure 1 for more details.

Output Layer. Firstly, we use an MLP to transform each feature vector into logits of size L, representing the position of the permutation. Then, we calculate the maximum weight matching of the $L \times L$ logits to obtain the correct permutation.

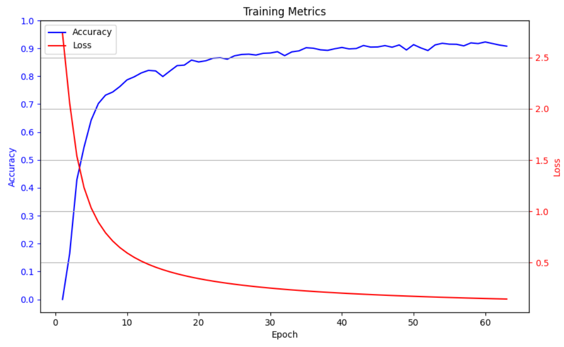

Figure 2. The loss and accuracy of the training (No maximum weight matching).

No position embedding and no [CLS] token. We find that there is no need to remember the position of the patches, since the problem itself is to find the correct order of the patches. Similarly, we find no need to create a [CLS] token. Experiments show that these two changes do not reduce the accuracy.

4. Experiments

Here are the additional subsections on the environment, datasets, results, and ablation study:

4.1. Environment

We use PyTorch to implement the model. The model has 21M parameters. We use a single 12GB RTX 4070 GPU.

4.2. Datasets

We use 20K images from the ImageNet-1k dataset [6], with 16K for training and 4K for validation.

We ensure that the height and width of these images are at least $p \times n$. Afterwards, we randomly crop the images to a size of $(p \times n) \times (p \times n)$ and divide them into $n \times n$ patches.

4.3. Results

After training for 63 epochs, the model can achieve 94.6% accuracy on the validation set.

Figure 2 shows the curve of loss and accuracy during the training. In the figure, the evaluation does not contain maximum weight matching.

4.4. Ablation Study

Positional encoding and [CLS] token. We test a model that uses sine-cosine positional encoding and adds a [CLS] token at the beginning. In the case of $p = 64$ and $n = 5$, this model achieved 76.4% accuracy after 12 epochs, which is not significantly different from our model's 76.7%.

Maximum weight matching. When the evaluation does not contain maximum weight matching, the accuracy drops down to 92.2%, showing that the maximum weight matching is important.

Dataset preprocess. We tested the model's performance when only cropping the top-left corner of the dataset. The results show that after training for 63 epochs, the model can only reach an accuracy of 86%, and only 61% accuracy on the DIV2K dataset. This indicates that the preprocessing of the dataset is crucial.

4.5. Performance

Generalization. We also evaluated our model on the DIV2K [1] dataset. We also randomly cropped all images to process them. The results show that the model can achieve an accuracy of 88.3% on the entire DIV2K dataset. Since DIV2K is a completely different dataset, this can indicate that the model has good generalization capability for natural images.

Handwriting Images. Our model does not behave well for handwriting images. Figure 3 is an example.

We conclude two reasons: Firstly, the model needs colorful and distinguishable patches. But handwriting images contain same patches and few colors. Secondly, the model does not train on handwriting images, so it does not have strong generalization.

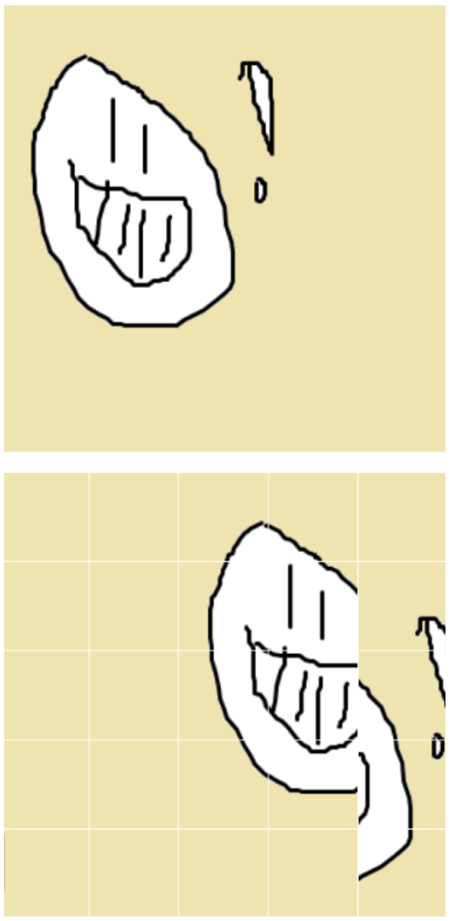
Figure 3. An example of handwriting images.

5. Conclusion

5.1. Summary

We have constructed an end-to-end model based on ViT that can solve jigsaw puzzles given the individual pieces.

While we have only implemented the problem with fixed patch size and puzzle scale, the transformer architecture makes it easy to extend to variable puzzle scales.

5.2. Shortcomings

Our approach is unable to handle changes in the patch size.

Our approach also struggles with images that are far from the training dataset, such as handwritten drawings, as seen in our experiments where the model failed to perform well on such inputs.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. 3

[2] Dov Bridger, Dov Danon, and Ayellet Tal. Solving jigsaw puzzles with eroded boundaries. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3526–3535, 2020. 1

[3] Benedict J Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings. *ACM transactions on graphics (TOG)*, 27(3):1–9, 2008. 1

[4] Yingyi Chen, Xi Shen, Yahui Liu, Qinghua Tao, and Johan AK Suykens. Jigsaw-vit: Learning jigsaw puzzles in vision transformer. *Pattern Recognition Letters*, 166:53–60, 2023. 1, 2

[5] Taeg Sang Cho, Shai Avidan, and William T Freeman. The patch transform. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1489–1501, 2009. 1

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[7] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 1

[8] Andrew C Gallagher. Jigsaw puzzles with pieces of unknown orientation. In *2012 IEEE Conference on computer vision and pattern recognition*, pages 382–389. IEEE, 2012. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[10] Hairong Liu, Shengjiao Cao, and Shuicheng Yan. Automated assembly of shredded pieces from multiple photos. *IEEE transactions on multimedia*, 13(5):1154–1162, 2011. 1

[11] William Marande and Gertraud Burger. Mitochondrial dna as a genomic jigsaw puzzle. *Science*, 318(5849):415–415, 2007. 1

[12] 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. 1

[13] Marie-Morgane Paumard, Hedi Tabia, and David Picard. Alphazzle: Jigsaw puzzle solver with deep monte-carlo tree search. *arXiv preprint arXiv:2302.00384*, 2023. 1

[14] Dror Sholomon, Omid David, and Nathan S Netanyahu. A genetic algorithm-based solver for very large jigsaw puzzles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1767–1774, 2013. 1

[15] 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. 1

[16] Rui Yu, Chris Russell, and Lourdes Agapito. Solving jigsaw puzzles with linear programming. *arXiv preprint arXiv:1511.04472*, 2015. 1