Neural Perspective to Jigsaw Puzzle Solving

Viveka Kulharia*, Arnab Ghosh*, Nikhil Patil*, Piyush Rai

Department of Computer Science, IIT Kanpur
Kanpur, India
{vivekakulharia,arnabghosh93,nikhilpatil123}@gmail.com
piyush@cse.iitk.ac.in

Abstract. Understanding, predicting the overall configuration from its constituent parts is an important challenge in Machine Learning and Artificial Intelligence. Agents trained to reason about subparts can then be combined to form complicated reasoning systems. Jigsaw Puzzle solving is an avenue in which the different components have to be correctly placed in order to create a plausible image from its component parts. We analyze this problem using a neural approach since ultimately it will be easier for neural networks to communicate with each other and such subnetworks will help in advancing the research of neural networks in Computer Vision. We present models based on Multilayer Perceptron, CNNs and LSTMs alongside a Markov Random Field based baseline which works on Energy Based Configurations. We perform experiments on a large scale based on the ImageNet dataset which is orders of magnitude larger than the previous evaluations of algorithms developed for the task.

1 Introduction

The problem that we are addressing in this work is the task of Jigsaw Puzzle Solving. It involves reassembling the pieces of the image into a coherent and plausible image. To accomplish this task we use images from the large scale image recognition task ILSVRC 2015 dataset, more popularly known as the ImageNet Dataset [6] and divide the image into rectangular grids, assign the different image parts the identities $(1, 2, \ldots, n)$ in a row major format, jumble the image parts to get x_1, x_2, ..., x_n parts in row major format corresponding to some permutation of $1, 2, \ldots, n$. The task is to predict the permutation (one out of $n!$) that produced the resulting jumbled image, as shown in figure 1 & figure 2.

Models developed for this problem can be useful for image reconstruction and also for collating and analyzing the results from different MRI scans. Many real life scenarios can be reduced to a task of recombination or reassembly of component parts which correlates well with Jigsaw Puzzle Solving. Jigsaw

* Equal Contribution

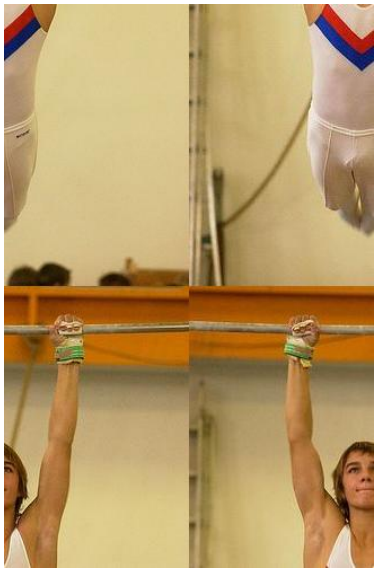

Fig. 1. Jigsaw puzzle (Source: ILSVRC2015).

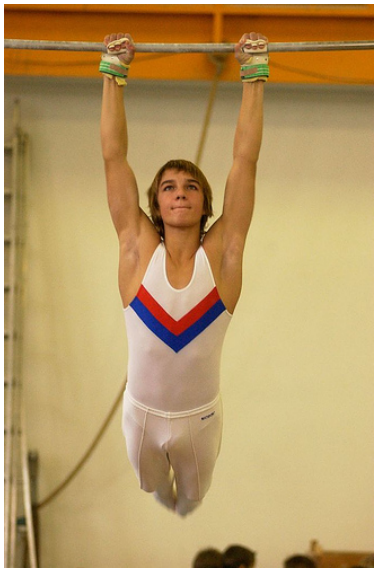

Fig. 2. Solved puzzle (Source: ILSVRC2015).

Puzzles and the models for solving them have been useful in reassembling fossil/archaeological relics, reassembling shredded documents, DNA/RNA modeling and Image based CAPTCHA construction. Hence solving Jigsaw puzzle efficiently can be a useful for all these applications and has the potential to advance the state of the art systems. Completing jigsaw puzzle is a challenging problem and requires expertise even for humans. It is also known that this is an NP-complete problem [5] and hence very difficult if the number of pieces to be combined into an image is huge.

It is a hard problem since the number of choices are quite huge ($n!$) if there are n parts into which the original image is divided. Most of the existing methods employ $\Omega(n^2)$ [10] comparisons along the sides to check whether the images align and are adjacent in the original image which poses problems during run time one of which is that even if boundaries match, the two parts may not align in adjacent positions.

Most of the work that preceded were based on heuristics and considered ad hoc models with lots of design choices rather than models which are capable of learning. Moreover, since the methods were not based on learning most of the previous methods used small scale datasets but our models are based on Deep Neural Networks which gains from available large scale image databases. Moving forward, neural networks created for the task can help in several similar problems and applications. Further, with gaining traction of research on multiagent neural

networks which cooperate to solve a given task [27] [8], neural models will be useful for interacting with other neural networks and hence aid in solving larger tasks using the networks designed for the current task.

Our analysis of the problem led us to explore models based on Multilayer Perceptrons, Convolution Neural Networks and Recurrent Neural Networks (LSTMs). To compare the methods with the traditionally used methods we also came up with a Markov Random Field based method which takes into account energies of various configurations to evaluate a reassembly of the individual parts. Our results gave us some interesting insights. The improved performance of LSTMs as compared to that of the CNNs and Multilayered Perceptrons show the importance of structure to be incorporated into deep neural networks. Since the different pieces of the original image were provided at different timesteps, the model didn't have to figure out the boundaries of the different parts of images. Our proposed model with LSTMs perform exceedingly well, almost matches the performance of carefully designed state of the art methods.

2 Related Work

Puzzle assembly is an actively pursued problem in Computer Science and Artificial Intelligence, initially introduced by [9] and proved by [5] that it is an NP-Complete problem. Several early works have focused on emulating humans and trying to match the edges maximally in order to produce a plausible configuration. [1], [4], [30] and [26] analyzed the problem when the pieces are square with aligned boundaries and the pieces are in the correct configuration, with just the correct position of the puzzle pieces to be inferred. With the knowledge of the resulting shapes and the original shape of the puzzle, the problem reduces to inferring the optimal permutation among all the possible permutations of each piece given an identity.

Majority of the previous work model it as a maximization of a predefined compatibility measure such as Mahalanobis Gradient Compatibility (MGC) [10] with the compatibility metric defined in an implicit or an explicit manner. Modeling of the problem with the maximizing objective can lead us to solutions which can be solved using optimization techniques such as belief propagation [4], particle filtering [30], greedy methods [20], constrained quadratic minimization [2] and genetic algorithms [23].

Some of the most influential recent works include [26] which uses a novel growing consensus based method which overcomes the challenges faced by just edge based compatibility scores. Another influential work [10] analyzes the more general version of the problem in which the orientation of the pieces is not known beforehand and used a novel tree based assembly method using Kruskal's algorithm.

Neural Networks have been performing exceptionally well even in tasks which humans find challenging as shown by recent successes of Deep Neural Network based methods in Go [25], Visual Question Answering [3], Abstract Reasoning Diagram Generation [11], Text to Image synthesis [21] among several others and hence a neural network based method was due for the task of Jigsaw Puzzle Solving. Among the neural models our model also closely relates to [7] which tries to predict the relative configuration of image patches.

A recent work [19] analyzes the task of solving jigsaw puzzles using a deep unsupervised learning setting with CNNs and further goes on to show that models developed for such tasks can help in learning better feature representations. Our work while having similarities also has notable differences, we provide a modeling with LSTMs which can work on arbitrary number of pieces and also is scalable as compared to CNNs as when the sizes of the individual pieces are not sufficient enough a pretrained CNN faces trouble extracting features from it.

Our finetuned CNN model is based upon recent work [14] which finetunes model trained on ImageNet dataset and uses it for a different task of style recognition in Flickr images which thus demonstrates transfer learning. Recurrent Neural Networks have shown tremendous promise for sequence based models and has become the de facto standard for language modeling tasks. Recently variants such as Pointer Networks [29] and [28] have shown the application of LSTM based methods for combinatorially hard optimization problems and a similar modeling approach has been used in our LSTM based approach.

3 Models

3.1 Convolutional Neural Network

Convolutional Neural Network are very efficient in extracting good features from images. It contains filters which move axially on an input image to get features and pooling which makes the object position invariant which are helpful in image classification/object detection. We used pre-trained Convolutional Neural Network (CNN) on ImageNet dataset. The CNN, BVLC CaffeNet, is slight modification of AlexNet (shown in figure 3) in that the order of pooling and normalization layers is switched. We changed the final layer of the model to match our labeling scheme and loss function, and fine-tuned the model on the dataset. While finetuning, we ensured that learning rate of the final layer was largest and very small for lower layers. We used two different modifications on BVLC CaffeNet both different in only the loss function used on the final layer.

Softmax loss We used 24 classes for each of the 4! possibilities (as n = 4). Each of the 24 possibilities was mapped to one of the classes while training. The second last layer had 24 elements in its output. While testing, the element with the maximum value among 24 elements of the output of second last layer

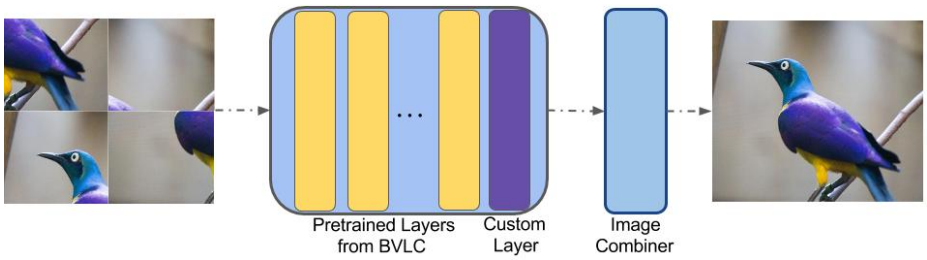

Fig. 3. Our Finetuned CNN Model alongside the Image Combiner which takes the generated vector of size $n!$ (here n=4) as shown above as well as component images to produce the output image.

was used to map to corresponding permutation (e.g. while testing figure 1, 24^{th} element had highest value which was mapped to label: 4, 3, 2, 1).

Mean Squared Error We had 4 outputs (as n = 4) from the second last layer of BVLC CaffeNet (as opposed to 1000 for 1000 classes in original network used for IMAGENET). While using the Mean Squared Error (MSE) for figure 1, the model minimized the sum of squared difference between each of the four elements of the output of second last layer and each of the four elements of the label: 4, 3, 2, 1. During test time, the prediction was made greedily by using the output of second last layer. The label was obtained by assigning 4 to the element with highest value, 3 to second highest value, 2 to third highest value and 1 to least value in the output and thus getting the mapping to labels.

In MSE, labels are assigned greedily which doesn't capture the way labels are used while training i.e. classification is done using regression which causes MSE to perform poorly than Softmax loss. Due to this superior performance of Softmax loss, we decided to only use it in the subsequent models.

3.2 Multilayer Perceptron

Feedforward Networks based models have shown promise in several avenues such as image generation [12], language modeling [16] among numerous others. Hence we modeled the problem with a 4-layer Multilayer Perceptron network with Rectified Linear Units [18] as the chosen non-linearity since it has been shown by several models [18] as the best non-linearity among tanh, sigmoid and ReLU. To model the problem using a MLP we concatenated the vectors for each of the component images (x_1, x_2, x_3, …, x_n) in the same order that we received into a single vector and passed the vector through the feedforward network. The loss function used for modeling the feedforward process was a softmax loss function because of the superior results obtained from this loss function in the CNN based model. One of the primary reasons for its failure is the additional overhead incurred by the network to figure out the different pieces

of the vector (corresponding to the component images) from the vector passed to the Feedforward Network.

3.3 Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM) is a very powerful model as it is a modified version of the Recurrent Neural Network which has been shown to be Turing Complete[24]. It is used to model sequences as it has recurrent connections in comparison to the feed forward-only connections in an MLP and CNN. The model is shown in the figure 4.

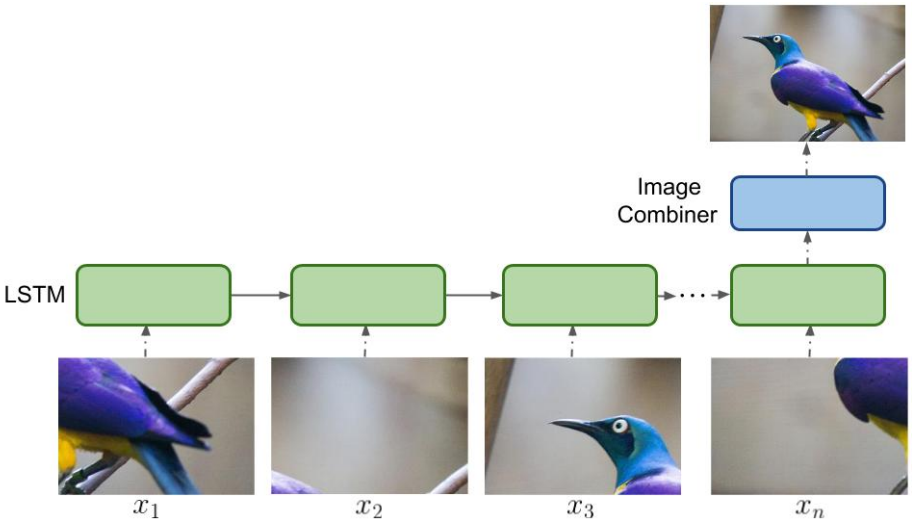

Fig. 4. LSTM model where Image Combiner takes the generated vector of size $n!$ (here n=4) as shown above as well as component images to produce the output image.

We took vectors for each of the component images (x_1, x_2, x_3, ..., x_n) as different timesteps of the LSTM [13]. Our LSTM model was a many-to-one model since we have several timesteps and a single output at the end of these steps which while training was a one-hot representation of the permutation that generated the current jumbled image as softmax loss is used.

The different image parts at the different timesteps allowed the model to recognize the different constituents of the image and hence was the major difference from the MLP model. We hypothesize that it worked better than the CNN model because the CNN was only finetuned on the images of our model but the learning rate of the earlier layers were very less and hence the earlier convolution filters faced a difficult time finding out the edges in the jumbled up images.

3.4 Markov Random Field

Markov Random Field (MRF) do not need edge orientation and are very useful for image analysis. We chose to create two compatibility functions which use the known pixels [17]. Here are the two compatibility functions:

1. Square of difference between pixel values on the edges
2. Square of difference between pixel values perpendicular to edge up to a certain width weighted by Gaussian of certain peakiness (defined by variance).

$$E(X) = \sum_{i \in W} N(i; 0, 1)(P(i) - P(-i))^2 \quad (1)$$

where $E(X)$ is the energy of the edge, W is the width, $N(i; 0, 1)$ is the probability density of i given by standard normal distribution and $P(i)$ is the pixel value at i distance away from edge, perpendicular to it. The first compatibility function is specific case of this function with width being 1.

4 Features

Raw Pixels These features were obtained using the images' pixels. The pixels were obtained after resizing images of size (256 x 256 x 3) to images of size (227 x 227 x 3) which was further processed to mean normalize the data for efficient processing by neural networks and other machine learning algorithms.

Alexnet Features [15] started the renaissance of Deep Learning by using the deep convolution neural network and surpassing the other methods by a huge margin. Subsequently CNNs became the de facto standard for image based features in most computer vision problems. Similar to a long range of previous work such as [3] etc. we use penultimate layer of the Alexnet (having dimensionality 4096) for high quality features.

5 Experimental Setup

Dataset We used the test dataset provided for the task of "Object detection" in IMAGENET Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)[22]. The dataset contained 51294 images. One image was anomalous as it was of $1 \times 1 \times 3$ pixels. So, we used 51293 images. Each of these images was divided into four quadrants and jigsaw was created by using their 6 unique random permutations (out of $4! = 24$ possible permutations). So, we had 301758 (50293×6) images to train our model on with 3000 (500×6) images for validation and remaining 3000 (500×6) images to test it.

Labeling scheme Consider the figure 1 used for training. The labeling scheme used for it is 4, 3, 2, 1 (in row major form) as the correct position for top left portion of the image is 4 (bottom right in correct image), top right is 3 (bottom left in correct image), bottom left is 2 (top right in correct image) and bottom right is 1 (top left in correct image) to obtain figure 2.

Training Details

1. **CNN** The CNN model was trained with the only difference being the final layer depending on whether the regression loss (4 output units) or the Softmax Loss Function (24 output units) were used. The learning rate of the second last layer was set more than any other layer as we were fine-tuning the model and assumed that other layers were already trained.
2. **MLP** We used a Multilayer Perceptron which was 4 layers deep and had 1000 hidden units in each hidden layer, dropout of 0.5 was used for this model for training.
3. **LSTM** We used a variation of the LSTM which is a sequence to label model with 1000 hidden units and the final layer producing 24 outputs corresponding to the Softmax loss function. A dropout of 0.5 was used for this model for training.
4. **MRF** For each of the images divided into four quadrants, we pre-computed the compatibility along each of the 24 edges of possible neighbors. It was then used to find the best possible permutation of the quadrant which maximizes the compatibility.

Evaluation Metrics True positive is defined as the number of test samples for which the model correctly predicts the composite image, we define accuracy as the ratio of true positive and total test samples.

6 Results and Analysis

Table 1. Results for MLP with different features.

Features	Accuracy
Raw features	7%
Alexnet features	10%

As illustrated, Table 1 shows the results from the Feedforward based Multi-Layer Perceptron. We can see the Alexnet features perform better than the raw pixel level features which is expected as Alexnet is trained on the images from the ImageNet dataset and provides high quality feature information even about its constituent parts. Features based on raw pixels also perform decently and are useful features in case of pieces being very small and the pretrained Alexnet can't find good representations of the pieces of the image.

As illustrated in Table 2, the results from the Finetuned CNN model consistently outperforms the results from the feedforward network hence emphasizing

Table 2. Result for Finetuned CNN with different loss.

Loss Function	Accuracy
Mean Squared error	2.81%
Softmax loss	46.33%

the superior quality of detectors present in the Convolution layers. Another interesting insight to be noted here is that the mean squared loss is extremely poor which shows that it is inappropriate to model categorical values with a regression based model. A notable caveat to the model is that if the pieces are very small then the Finetuned CNN model won't be able to get good features from the jumbled up image and won't perform as well.

Table 3. Results for LSTM with different features.

Features	Accuracy
Raw features	79%
Alexnet features	82%

As illustrated in Table 3, LSTMs outperform all of the results from the finetuned CNN based models. The superior performance as compared to the CNN and MLP based methods can be explained by the fact that the structure of the problem is better harnessed by the LSTMs by modeling the different pieces of the puzzle as different timesteps of the LSTM. It is a much more general approach than the finetuned CNN based model since the performance of raw pixel based features is not far from the best results obtained using the Alexnet features and hence can also be applicable when the pieces are small and the number of pieces are arbitrary.

Table 4. Results (as accuracy) for MRF based approach.

Width / Variance	0.5	1
1	87.47%	87.47%
2	87.273%	86.81%
5	87.271%	86.58%
10	87.271%	86.58%
20	87.271%	86.58%

As illustrated in Table 4, the Markov Random Field based methods which analyzes the pairwise compatibility scores among adjacent pieces of the puzzle and trying to reduce the overall dissimilarity among the pieces of the puzzle produces good results but with a runtime overhead.

7 Limitation of current models

For future work, we present the limitations of our methods. One limitation is that they can't scale very well with the number of cuts made in image to create the Jigsaw problem. If the image is divided into n parts,

1. In case of CNN, the number of labels would grow to $n!$. So, a lot of images would be required to train. We can generate more images but the training time would exponentially increase.
2. In case of MRF, if we go by current method, $2n(n-1)$ number of comparisons would be needed for pre-computation. These pre-computations would be used $n!$ times for getting the configuration with minimum energy. If we adopt greedy strategy to get the result, the accuracy would decrease.
3. In case of LSTM, the problem of increased number of classes would be there. One more problem that would be present is that we are using ImageNet trained CNN to get features to be the input of LSTM. If we create a lot of pieces of the images, the CNN won't give good quality features for each of the pieces.

8 Conclusion and Future work

We have shown some of the baseline methods used for modeling of the problem of Jigsaw Puzzle Solving with Deep Neural Networks and we have illustrated that exploiting the structure available in the problem actually helps us in getting the best accuracy. Modeling of the problem using LSTMs by reducing the problem to a sequence to label problem is a good approach for small sized models. Future work can reduce the complexity of the softmax and use a 2 dimensional Spatial-LSTM based modeling which can figure out the relative positions independently in the rows and the columns. Autoencoder based approaches can also be used but it might lead to lossy reconstruction. Modeling of relative patches once a partial solution has been reconstructed can also be used based on [7]. It needs to be noted that using MRF features as input to Neural Nets can't help as edge pixels don't capture much information about image parts and any information that can be helpful is already well utilized by the structure of MRF model, so no need for deep-nets.

References

1. Naif Alajlan. Solving square jigsaw puzzles using dynamic programming and the hungarian procedure. *American Journal of Applied Sciences*, 6(11):1941, 2009.
2. Fernanda A Andaló, Gabriel Taubin, and Sione Goldenstein. Solving image puzzles with a simple quadratic programming formulation. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 63–70. IEEE, 2012.
3. Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

4. Taeg Sang Cho, Shai Avidan, and William T Freeman. A probabilistic image jigsaw puzzle solver. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 183–190. IEEE, 2010.
5. Erik D Demaine and Martin L Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23(1):195–208, 2007.
6. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
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.
8. Jakob N Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016.
9. Herbert Freeman and L Garder. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers*, (2):118–127, 1964.
10. Andrew C Gallagher. Jigsaw puzzles with pieces of unknown orientation. In *Computer and Robot Vision (CRV), 2012 International Conference on*, pages 382–389. IEEE, 2012.
11. Arnab Ghosh, Viveka Kulharia, Amitabha Mukerjee, Vinay Namboodiri, and Mohit Bansal. Contextual rnn-gans for abstract reasoning diagram generation. *arXiv preprint arXiv:1609.09444*, 2016.
12. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
13. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
14. Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. Recognizing image style. *arXiv preprint arXiv:1311.3715*, 2013.
15. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
16. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
17. Debajyoti Mondal, Yang Wang, and Stephane Durocher. Robust solvers for square jigsaw puzzles. In *Computer and Robot Vision (CRV), 2013 International Conference on*, pages 249–256. IEEE, 2013.
18. Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.
19. Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *arXiv preprint arXiv:1603.09246*, 2016.
20. Dolev Pomeranz, Michal Shemesh, and Ohad Ben-Shahar. A fully automated greedy square jigsaw puzzle solver. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 9–16. IEEE, 2011.

21. Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.
22. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
23. Dror Sholomon, Olivier David, and Nathan S Netanyahu. A genetic algorithm-based solver for very large jigsaw puzzles. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1767–1774. IEEE, 2013.
24. Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 440–449. ACM, 1992.
25. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
26. Kilho Son, James Hays, David B Cooper, et al. Solving small-piece jigsaw puzzles by growing consensus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1193–1201, 2016.
27. Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *arXiv preprint arXiv:1605.07736*, 2016.
28. Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.
29. Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.
30. Xingwei Yang, Nagesh Adluru, and Longin Jan Latecki. Particle filter with state permutations for solving image jigsaw puzzles. In *Computer and Robot Vision (CRV), 2011 International Conference on*, pages 2873–2880. IEEE, 2011.