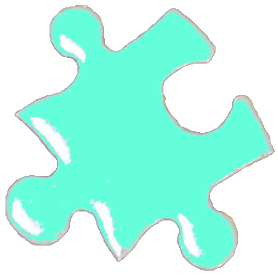

Puzzle Boon

Jigsaw Puzzle Assist

CSI4900 Honours Project

Tiffany Maynard
300047456

Winter 2021

Supervisor: WonSook Lee

Tiffany Maynard
300047456

Table of Contents

Table of Figures

Tiffany Maynard
300047456

Introduction

For my honours project, I chose to make a tool to assist with solving jigsaw puzzles.

I got interested in jigsaw puzzles as a means to distract myself from the pandemic and relax. With tougher puzzles, sometimes I felt like I had puzzle block, kind of like writer's block, and I was stuck because I couldn't find any pieces to fit in with the part I had finished so far. I realized it would be great if there was a tool that could help me find the next few pieces to break the stalemate.

Some puzzles are tough because they have so many pieces and others because they have indistinct colours or non-traditional piece shapes. I have one puzzle that has 18,000 pieces and is almost 2 metres by 3 metres in size(Figure 1). Since I will need to dedicate the entire floor of one room to completing it

Figure 1: An 18,000 piece puzzle

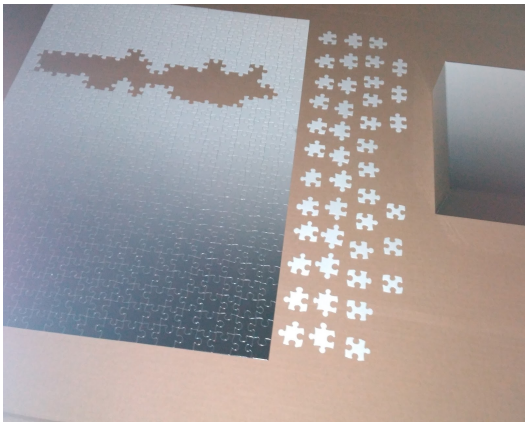

Figure 2: A gradient puzzle where the colour varies only slightly

because I don't have any tables quite so large, I have not been brave enough to attempt it yet. It would be a boon to have a helpful tool. I also have other tough puzzles like a crypt puzzle where all pieces are the same colour and have unusual circular shapes, and the gradient puzzle where the colour varies only slightly (Figure 2).

Tiffany Maynard
300047456

Methodology

Puzzle Boon will suggest possible matches based on three photos – one of the full puzzle image(Figure 3) from the puzzle box, one of the puzzle in progress (Figure 4), and one of the remaining puzzle pieces (Figure 5).

Figure 3: Full puzzle image

Figure 4: Puzzle in progress

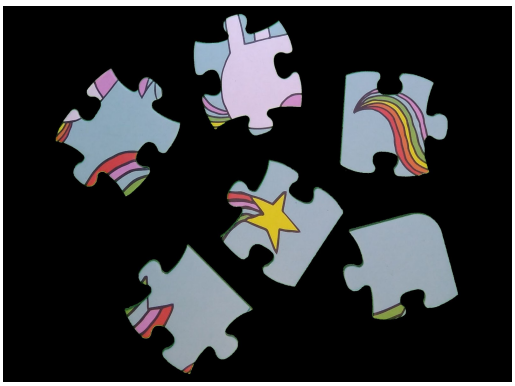

Figure 5: Remaining puzzle pieces

With this information, the tool will analyse the images and provide suggested matches showing which remaining piece will fit in which part of the puzzle of progressFigure 6.

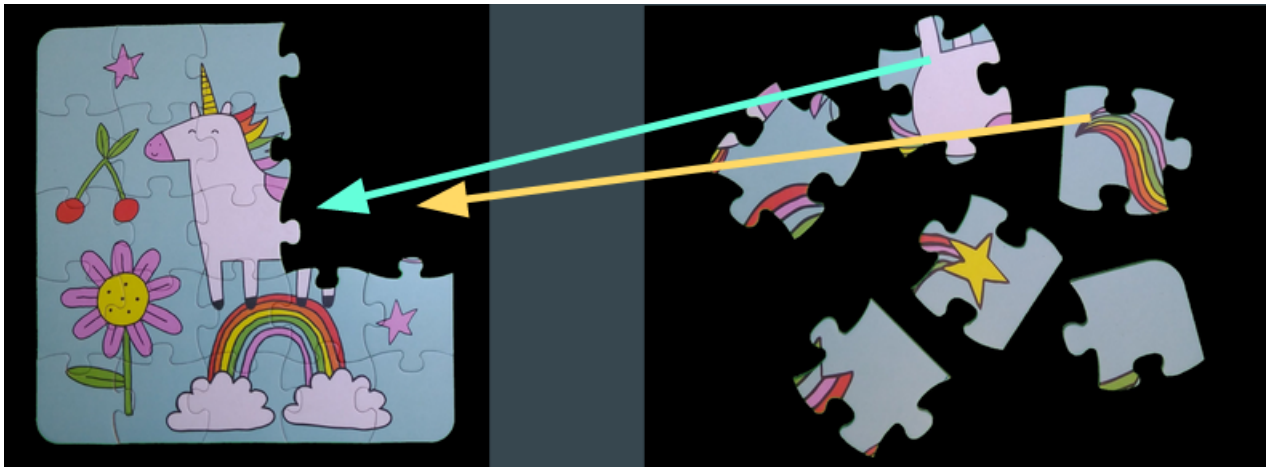

Figure 6: Provide suggestions for possible matches

Puzzle Boon needed a way to recognize which items in a photo were puzzle pieces, a way to determine likely matches between the blockers and the puzzle in progress and a way to clearly communicate the possible matches to the user.

Solutions

My solution needed several components, including creating a dataset of puzzle piece images, a trained machine learning model, the means to locate puzzle piece corners and re-orient pieces, some matching strategies, and a way to communicate the match results.

Initial Exploration

To start, to find puzzle pieces in a photo, I first tried traditional computer vision tools and algorithms using OpenCV. OpenCV is an open source computer vision library and it includes a huge number of functions to analyse and manipulate images and video. The traditional approaches I tried seemed to require a lot of manual intervention and they weren't always effective at isolating the puzzle pieces correctly, depending on the lighting and background of the image. As a result, I decided I would try to use deep learning to train a model to detect puzzle pieces. Deep learning has an advantage because it can be trained with task-specific data by even a naive user like me, while the algorithms used in traditional image processing are not something I am able to customize to be task-specific without significant domain knowledge. For example, here you can see my attempts at using Sobel and Canny image filters, where they found edges both inside and outside of each puzzle piece. Results were also indistinct with GrabCut when starting from a bounding box around the piece. Similarly, the Hough transform gave false edges and the goodFeaturesToTrack corner detector found way too many corners.

These disappointing results led to my decision to use deep learning to find puzzle pieces in an image. I read about the different types of visual perception tasks that deep learning can handle including Image Classification, Semantic Segmentation, Object Detection, and Instance Segmentation and realized I would need to do instance segmentation, since my images would contain many instances of the same object, in this case puzzle pieces rather than the dogs shown on the slide, and I needed to know the distinct location of each piece.

For instance segmentation, Mask R-CNN was chosen. Mask R-CNN (Region Based Convolutional Neural Network) is one available deep neural network architecture that aims to solve instance segmentation problems, so I decided to try it. Models based on Mask R-CNN have already been trained to recognize many things like people, cars, and dogs. Unfortunately Mask R-CNN was not trained to recognize puzzle pieces, and so it does not have the facility to recognize them built-in, so I needed to develop my own dataset of puzzle pieces to train a model to recognize them.

Training a model from scratch can take a very long time and require a really large amount of data, so luckily, I was able to leverage an existing trained model by using transfer learning. Transfer learning uses a model trained on one task and re-purposes some of it's findings it for a new task. I began with pre-trained weights that were already good at determining patterns and shapes in images of common objects and used these as the starting point for my puzzle piece training.

Creating a Dataset of Puzzle Piece Images

To develop my dataset, I took lots of pictures of puzzle pieces from many different puzzles on many different coloured backgrounds. I labelled my images with bounding boxes around each puzzle piece using a free tool called Label Studio. The LabelStudio interface made it easy to keep track of which images had been annotated with bounding boxes and allowed for easy corrections and adjustments. You can see here on the right a sample image with the bounding boxes drawn for each piece.

Once I had my set of images and their bounding boxes, I isolated the pieces from the background using the background colour within each bounding box as a filter and I used the GrabCut algorithm in Open CV to isolate each individual piece and create a mask of the puzzle piece. A mask is a black and white image of the same dimensions as the original image. Each of the pixels in the mask is either black (0) or white (1). The mask can be used as a filter to keep only the relevant part of an image or to make the contrast more obvious for finding edges and contours.

To use GrabCut, you designate parts of the image that are definite or probable foreground and background. This can be done using a mask or a bounding box, with everything outside the bounding box being taken as definite background. A graph of connected components is constructed with this information and then after the specified number of iterations to refine the graph connections, a “cut” is done to separate the foreground from the background components. I used these masks to get the precise location of the whole puzzle piece and its outline and exported this information to the COCO format.

COCO, or Common Objects in Context, is an open source object recognition dataset that contains hundreds of thousands of images with millions of objects already labeled. Since it is widely-recognized, the format used to store the annotations is well supported by many deep learning implementations. I chose to use this format for my puzzle piece dataset because it is easily interpreted by the deep learning implementation I ended up choosing, so no loading customization was required.

Since more data is always better when training a deep learning model, I decided to augment my dataset by creating a set of puzzle piece images with transparent backgrounds and generating a new set of images by pasting each piece in a random location on each of the new backgrounds. I also created the corresponding mask for each image at the same time. I ended up creating more than 1,000 pieces and used 17 background patterns and as a result I produced more than 18,000 new images and masks.

My dataset augmentation greatly increased the number of images of puzzle pieces I had available to train my model.

Train model to locate puzzle pieces in an image

Deep learning models are best trained on machines with GPUs with lots of RAM, so I looked for a cost-effective way to use a good GPU. I found Paperspace Gradient, which did not require a credit card to sign up for the free tier, and had a few free GPU configurations available out of the box. Other options available for compute included colab and sagemaker, but I stuck with gradient since it met my needs.

With my free GPU ready, first I tried using the Tensorflow implementation of Mask R-CNN but I could only make it work on an older version of Tensorflow and training on even a small batch of images took a very long time. Because the compute from Paperspace Gradient that I used was free, it was limited and auto-shutdown after 6 hours, so I was not able to finish and save my results with this implementation. Thankfully, I discovered that there was a more up-to-date implementation of Mask R-CNN available in Detectron2 which uses PyTorch. Detectron2 is a modular object detection library from Facebook. Detectron2 cut the training time down significantly, and I was able to use my augmented dataset to train a reasonably effective model in less than an hour, well within the 6 hour free limit on Paperspace Gradient.

Locate corners and orient pieces

Now that I had a way to reliably find the location of each puzzle piece in a photo, I needed to tackle the problem of determining likely matches between these pieces and the puzzle in progress. First, I worked on locating the corners of each piece. The corner locations make it possible to determine the four distinct sides of each piece and make it easier to match the contours. I also worked on re-orienting the pieces that were on an angle. For standard puzzles, once an image is oriented horizontally or vertically, only 4 possible image orientations need to be checked when trying to match the image content of a piece.

Various matching strategies

I then explored some matching strategies. One strategy, matching based on image, divides up the full puzzle image and the puzzle in progress into grids and searches within each grid section not present in the puzzle in progress for a match among the blockers using template matching.

Another strategy considered matches based on the shape of the contours of the puzzle in progress and the blockers. For puzzles that are all one colour, or very limited in colour, like the silver crypt puzzle and the gradient puzzle mentioned at the start – contour matching is really the only viable option.

It’s likely that a combination of the template matching and contour matching strategies would yield superior results.

Communicate results of matching

As a final step, I needed to find a way to communicate the possible matches found. I decided that colour coding the edges that matched the blockers and the puzzle in progress image would be a good way to highlight the matches. It would also be good to allow the user to choose to ask for only a small number of matches, if they wanted to avoid too many spoilers.

Results

Here are some inference results from running the Mask R-CNN model trained to recognize puzzle pieces. As you can see, the coloured masks provide pretty good coverage for each puzzle piece.

And here are some results from template matching. Most pieces were matched correctly, except for piece B. The matching is done using a grayscale version of the images and the section of the piece used for matching piece B contained no distinguishing features. In addition, template matching does not take into account the contours of the pieces, so it is not surprising that piece B was mismatched.

Conclusion

To conclude, I learned a lot about image manipulation and machine learning while working on this project and I am still in awe of the amount of free resources available, including libraries, free compute, and tutorials for how to accomplish almost anything.

Future Work

Future work could include gathering more images of puzzle pieces to improve the accuracy of the piece -detection module, refinement of the matching strategies, and getting user feedback on how to communicate match results more intuitively.

Tiffany Maynard
300047456

References and Tools Used

OpenCV https://opencv.org/

Label Studio https://labelstud.io/

Mask R-CNN https://github.com/matterport/Mask_RCNN

Detectron2 https://github.com/facebookresearch/detectron2

Paperspace Gradient https://gradient.paperspace.com/

fast.ai Practical Deep Learning for Coders https://course.fast.ai/

Stanford CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.stanford.edu/