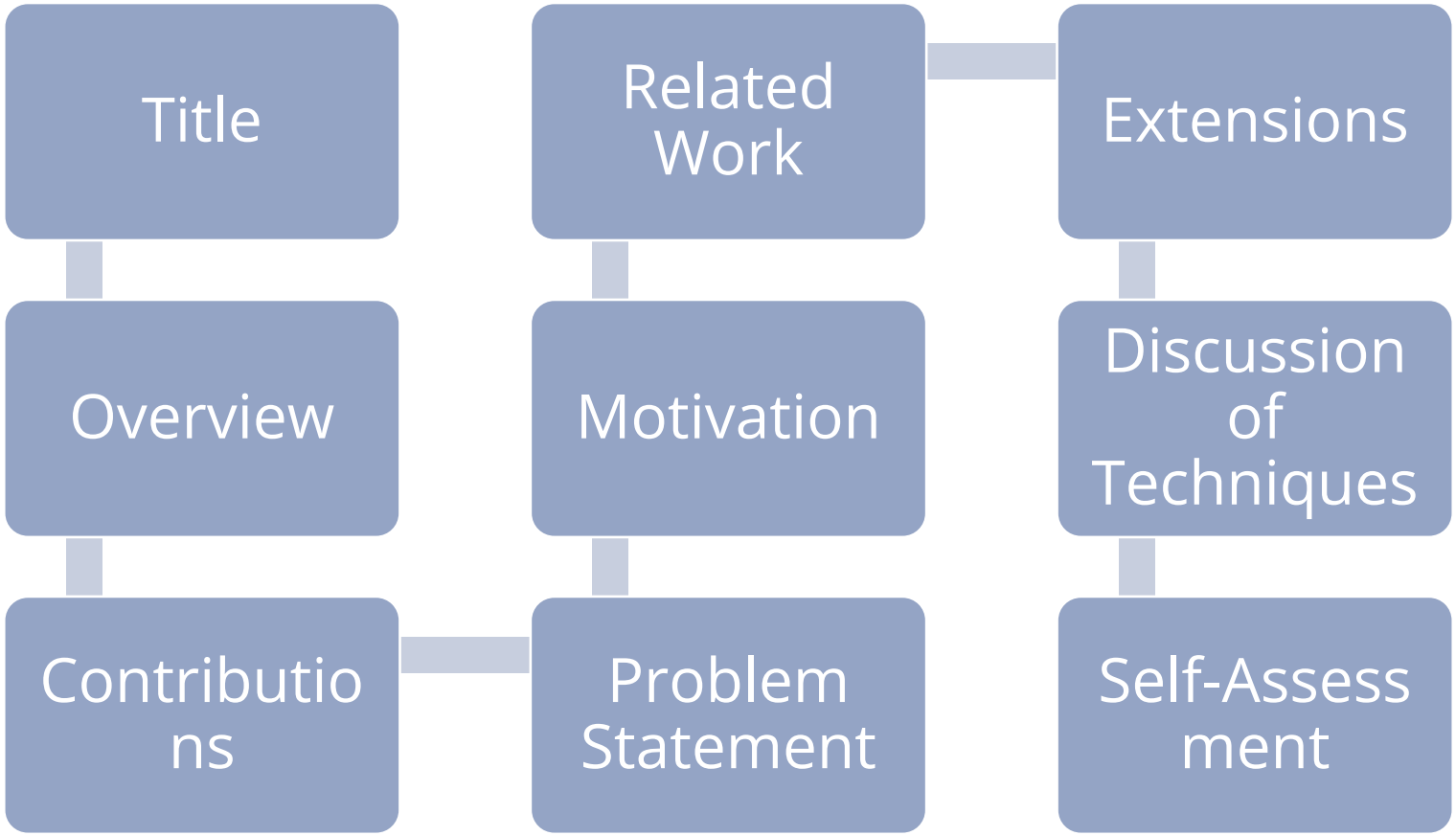
A STUDY OF CLASSICAL AND DEEP-LEARNING APPROACHES TO THE AGE-OLD ART OF JIGSAW SOLVING

Sanchit Jindal (200020120),

Hardik Rajpal (200050048),

Amruta Parulekar (20D070009)

OVERVIEW OF PRESENTATION



CONTRIBUTIONS OF MEMBERS

Member	Sanchit	Hardik	Amruta
Quantitative	100%	100%	100%
Qualitative	Studied and implemented a greedy solution. Aided with testing and fine tuning of various models.	Studied and implemented the linear programming solution. Aided with improvements/modifications to DL model using the paper.	Generated dataset. Studied and implemented the deep learning model. Made modifications to improve performance to said model.

PROBLEM STATEMENT

To take rectangular parts of an unknown rectangular image and figure out how to put them together to reconstruct the original image.

By any means

MOTIVATION

While our motivation was not too much, fortunately, it also was not the ideal value of zero.

We found the problem statement to be closely related to the topic of image-stitching discussed in class.

And we'll go to any length to savour a sliver of simpler

5. Motivation: Not too much. 0-1 slide. Ideally zero.

RELATED WORK ("ORIGINAL" PAPERS)

1. https://www.cs.bgu.ac.il/~ben-shahar/Publications/2011-Pomeranz_Shemesh_and_Ben_Shahar-A_Fully_Automated_Greedy_Square_Jigsaw_Puzzle_Solver.pdf
2. https://arxiv.org/pdf/1511.04472.pdf
3. https://arxiv.org/abs/1603.09246

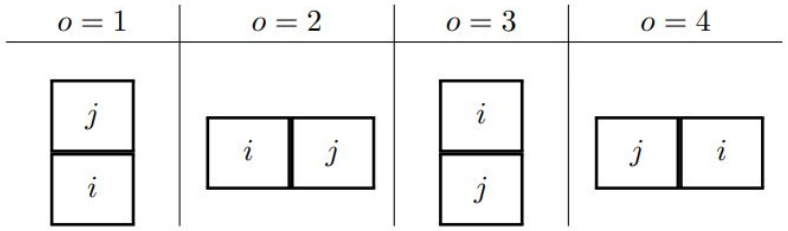
EXTENSIONS TO ORIGINAL PAPERS

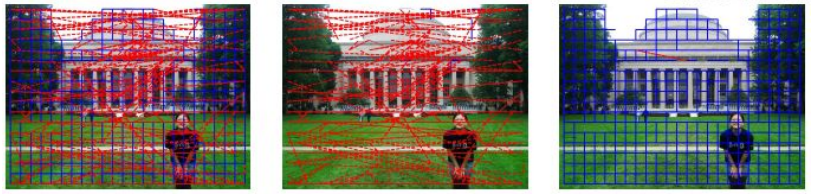
1. Extensions that exist already include:
    1. Accounting for rotation of the image segments. (https://par.nsf.gov/servlets/purl/10200913)
2. Extensions that we attempted during the project include:
    1. Appending to the input of the DL model with vectors important to the linear programming approach.
    2. Making the loss function penalize repetition of indices in the outputted order vector, and also tried a cross-entropy loss function.
    3. We trained models for blurred and low-resolution images as well.

DISCUSSION OF TECHNIQUES

We now discuss the various methods we studied and their evaluation.

LINEAR PROGRAMMING

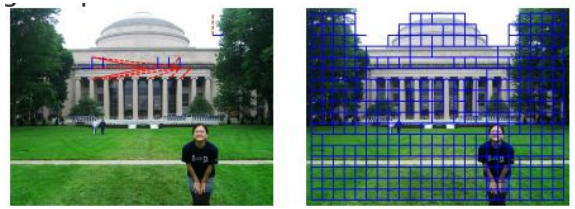




Initial confident pairwise matches $A^{(0)}$

Rejected matches $R^{(0)}$

Confident matches consistent with first LP $A^{(0)} - R^{(0)}$



(c) New matches introduced in second LP $A^{(1)} - A^{(0)}$

(d) Confident matches consistent with second LP $A^{(1)} - R^{(1)}$

The key idea is to rely on the similarity of edges of adjacent pieces and dissimilarity between non-adjacent ones,.

We first compute the mahalanobis distances between all possible pairs of edges (verticals and horizontals separately).

We then compose an expression linear in x and y (vectors of real x and y coordinates of the segments) that represents the cost of a given ordering of a subset of the segments.

We run a standard linear programming solver to obtain solutions.

This is done iteratively to cover the entire set of segments.

LINEAR PROGRAMMING

Performance:

The method, as expected, works with an accuracy close to 100%. It fails when the key idea, the similarity between adjacent edges and dissimilarity between non-adjacent ones, is bent.

Extension:

The original implementation checked similarity of edges by considering a single row of pixels. To accommodate for a variation of the problem statement where the edges of the segments have been blanked to a margin, we updated the implementation to evaluate the similarity over a margin of edges.

~~Setting the margin for similarity higher than the blanked~~ edges' width results in an accuracy close to 100 %

LINEAR PROGRAMMING

Original

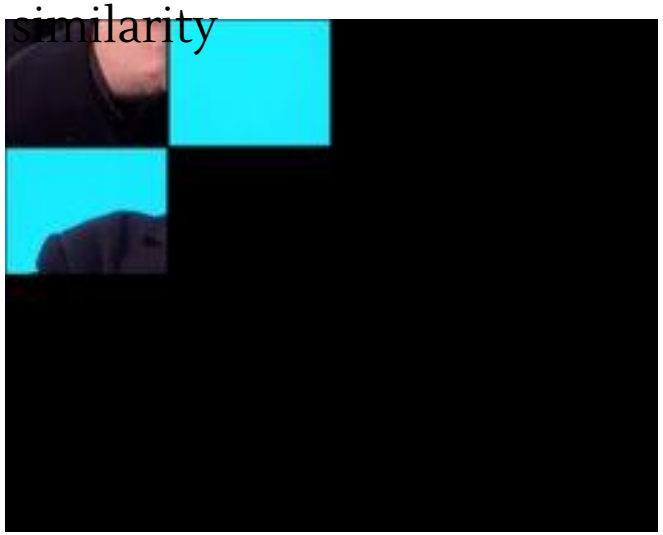
Shuffled

Shuffled with blanked edges

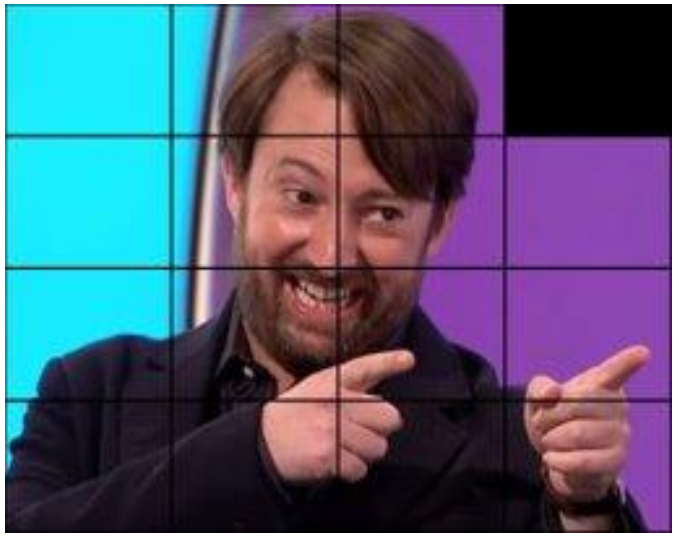
LINEAR PROGRAMMING

Solution without edge blanking or margin in similarity

Solution with edge blanking but without margin in similarity

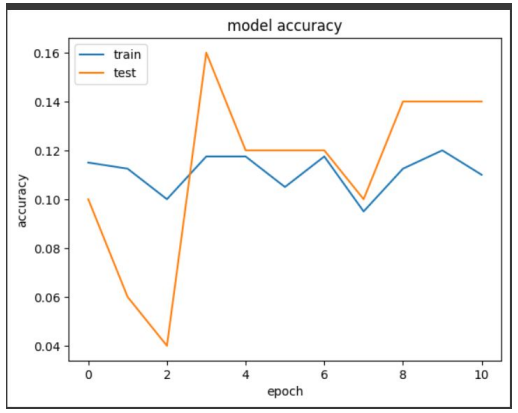


Solution with edge blanking and margin in similarity

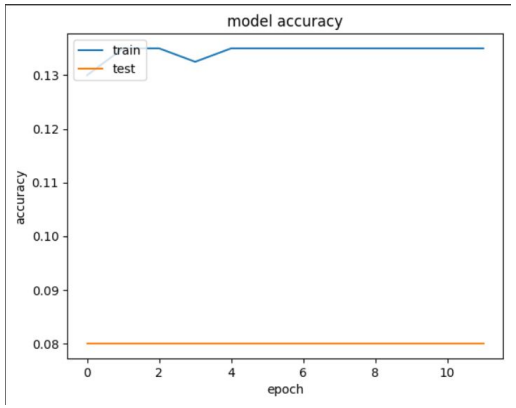


LINEAR PROGRAMMING

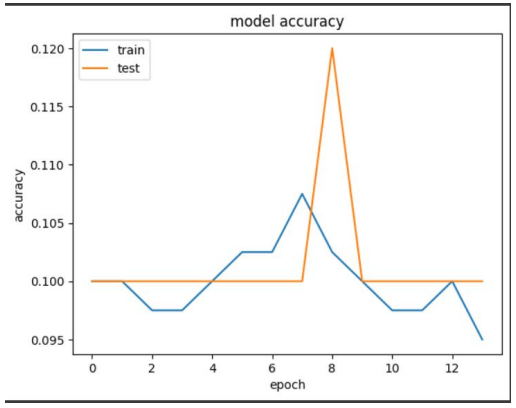
Experimental Results



Absolute Error (accuracy 14%)



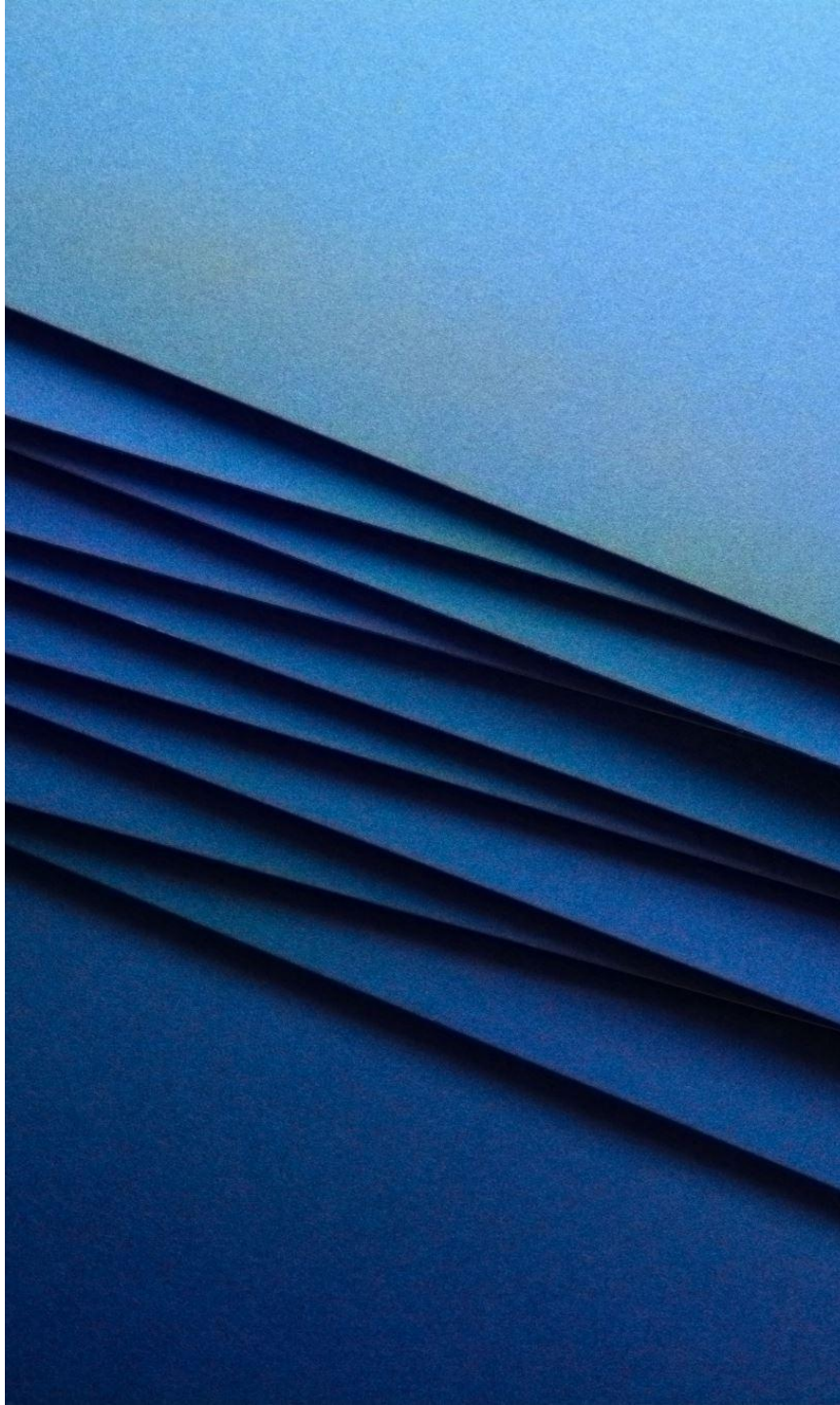
Squared Logarthmic error (accuracy 14%)



Absolute Percentage error (accuracy 4%

GREEDY APPROACH

- The Greedy approach paper calculates the distance between each edge of each piece with each corresponding edge (that is up edge with the down edge of others ....)
- Then it creates a best piece for each piece that is the pieces that have the least distance and should be most probably be the together
- Then a basic wave collapse algorithm is called that will seed a piece and then try to fill its neighbours with a good matching piece
- In this step it does not care that all the edges of each piece match only expanding one piece matches



GREEDY APPROACH

- After filling up the board, It segments the puzzle, that is it tries to find a big collection of pieces that seems to make a correct region
- After filling up the board, It segments the puzzle, that is it tries to find a big collection of pieces that seem to be all correct, Then it removes the other pieces
- And moves the biggest segment to the middle of the board
- Using this segment, it again runs the wave collapse algorithm and in this way, it tries to get bigger and bigger correct regions

RESULTS GREEDY ALGORITHM

The Result after placing all the Pieces in the image (using a 8x8 example)

Then the best segment out of the result is chosen which is being shown through the example,
The Black parts were originally identified wrong and so were not part of the segment

After this the algorithm will again start with this picture as the original seed and will most probably choose the correct options

GREEDY APPROACH

Observations and Comments

The Algorithm asks to seed each piece into the board and then run the algorithm and then selects the best output, This seems to be very cost inefficient approach for large number of pieces

As we manually have to define the distance between the edges which need to be compared, using irregular shaped pieces is not possible,

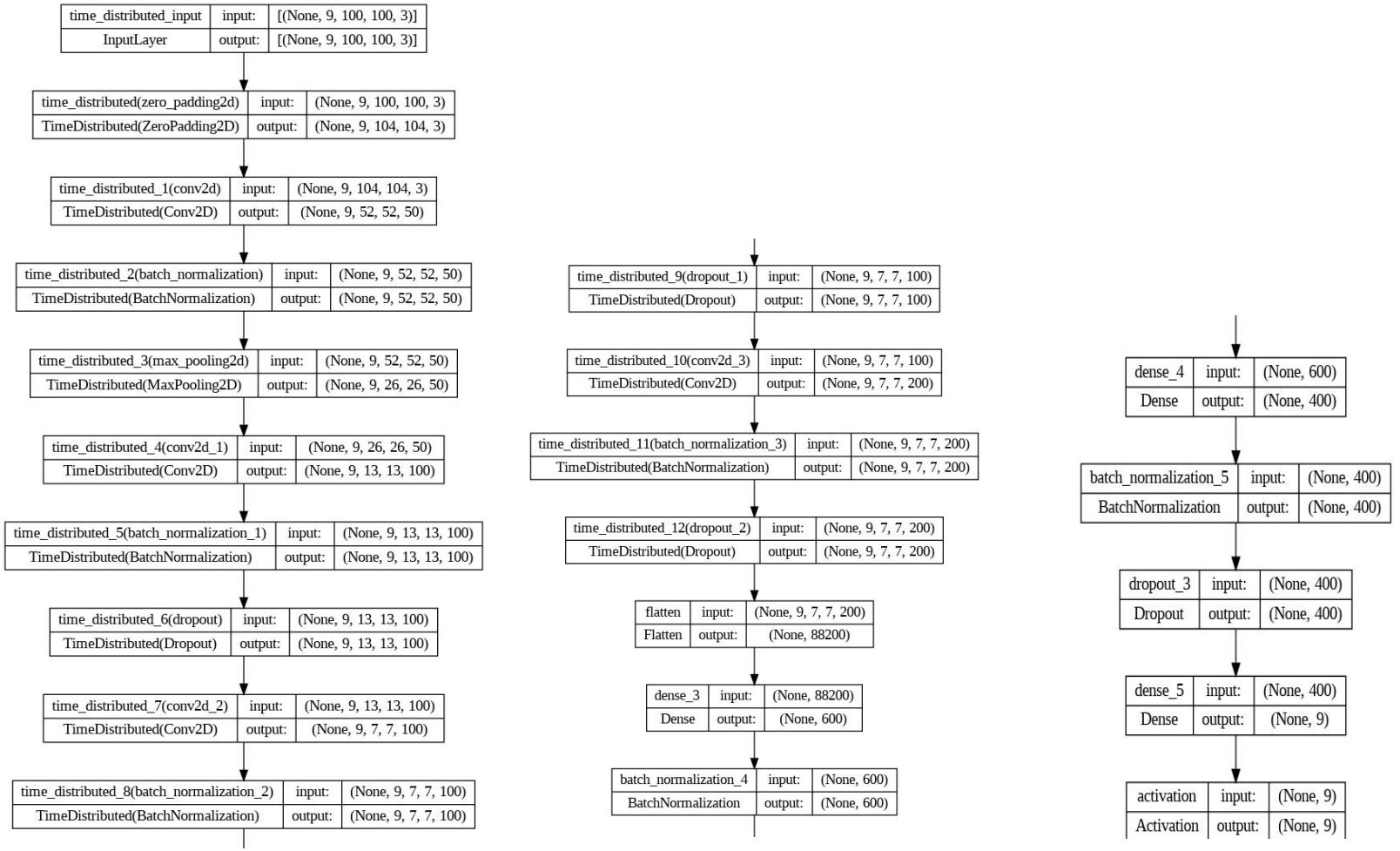
Also, this approach is very susceptible to noise as the distance (which the difference between the edges) is very important factor to select the best neighbouring pieces and noise makes this selection not feasible

DEEP LEARNING MODEL

Data:

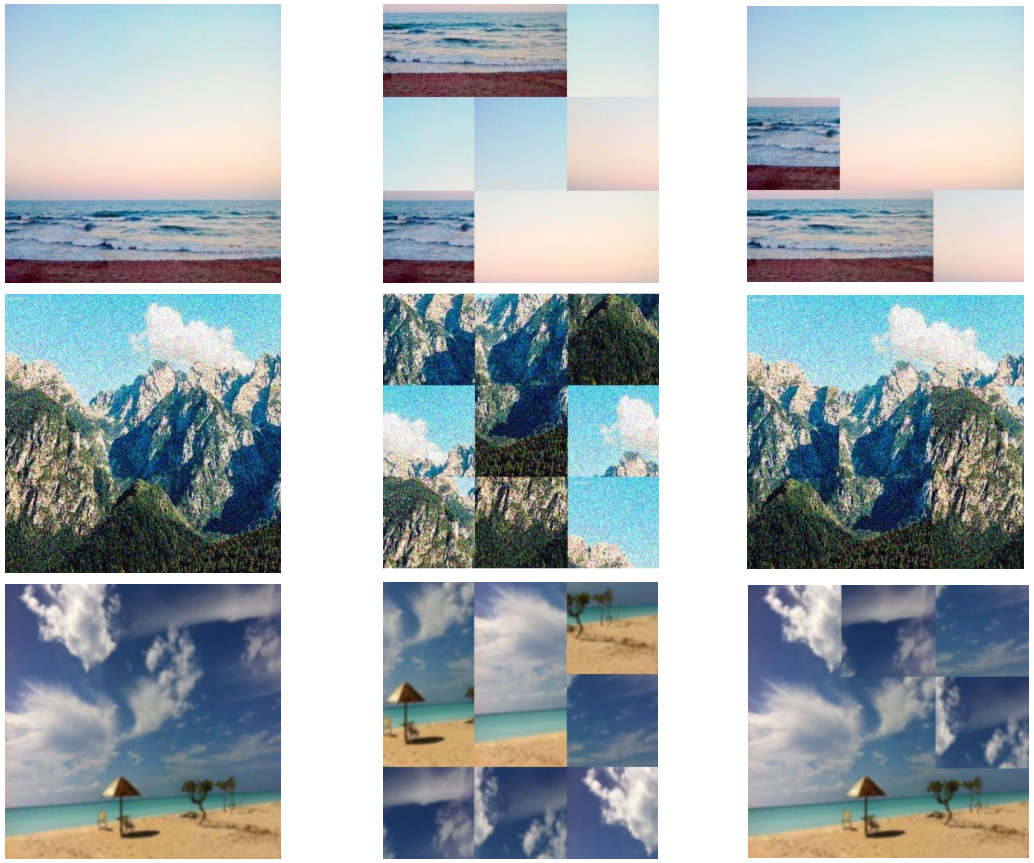
300x300 landscape images. 1000 images were normal, 1000 images were gaussian blurred and 1000 images had white gaussian noise added

Model Architecture:



DEEP LEARNING MODEL

Original Shuffled Solved

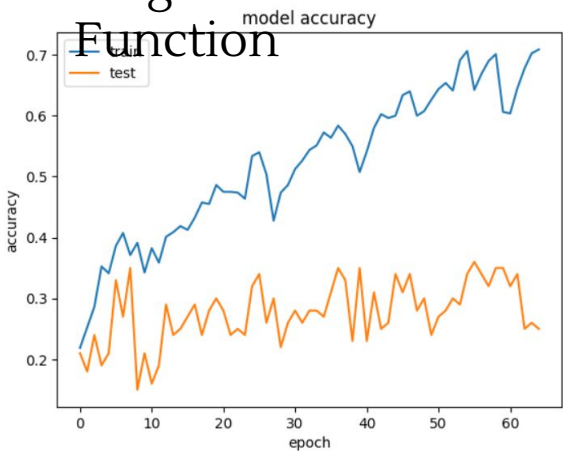


CORRUPTED IMAGE ADAPTATION

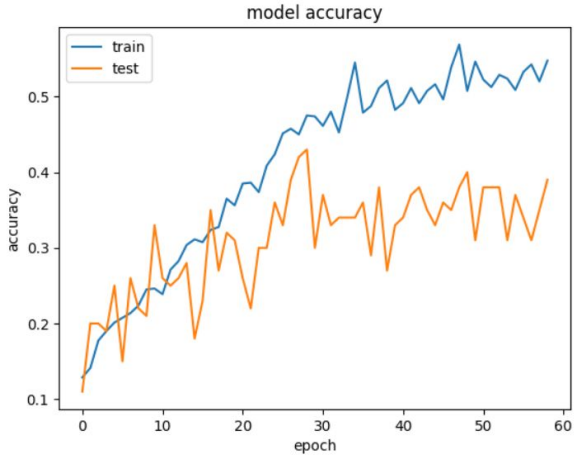
The model was retrained on blurry images and images with additive white gaussian noise added. This would make the model robust against corrupted images.
These images were generated by us.
Performance: The testing accuracy of the model improved to 34 %. Visual performance was considerably better too

DEEP LEARNING MODEL

Original Loss Function



Custom Loss Function

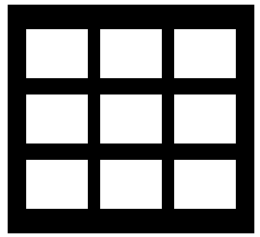


Original

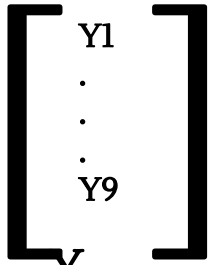
Shuffled

Solved

CUSTOM LOSS FUNCTION



X



Y

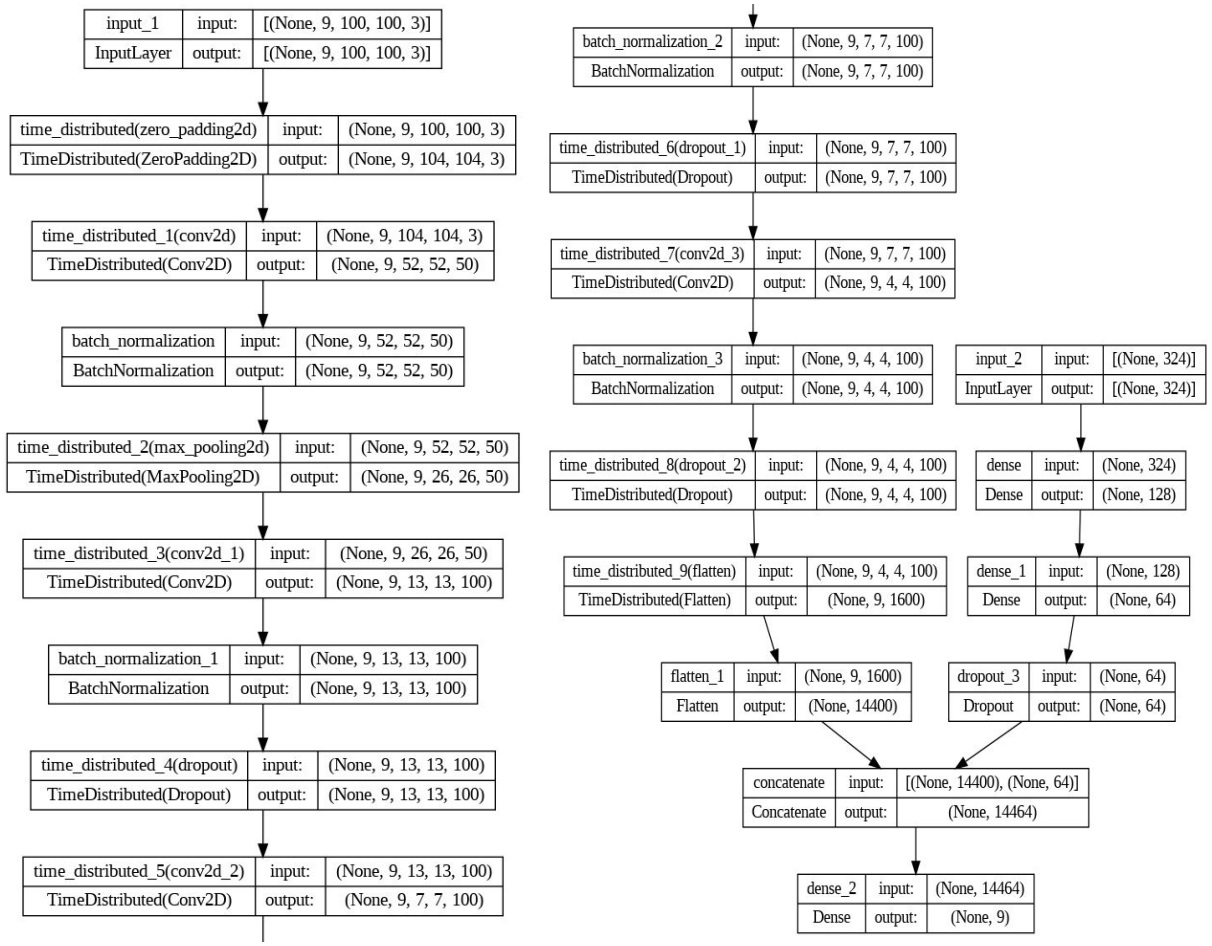
Consider each Yi as the classification label for the puzzle pieces.

The original loss function took the mean squared error between the actual and predicted label vectors.

We used binary cross entropy as the loss metric for each Yi.

Performance: This model trained for longer with a significant increment in validation loss. It also gave better visual results.

DEEP LEARNING MODEL



MGC DISTANCE ADAPTATION

The MGC distances obtained from the Linear Programming Model were used in the Deep Learning Model to combine both techniques.

Performance: The performance was not good, and the validation accuracy barely crossed 10%.

SELF-ASSESSMENT

While we did study and implement three different approaches to jigsaw solving (as planned), we had to rethink our choice of methods due to our limited understanding of two of the three methods we had initially decided on and time constraints.

THANKS