Deep Learning and Neural Networks Final Project

Murad Abu-Gosh, Mohammad Khayyo

Summary:

The problem we are trying to solve is finding duplicate images of pieces that appear in more than one image by comparing each piece to all other pieces. In addition, we try to find the piece in another capture/image.

In order to solve this problem, we've tried two different approaches:

The first approach was using VGG-16 first few layers for feature extraction, together with t-Distributed Stochastic Neighbor Embedding (t-SNE), which we used to reduce dimensions of the data and visualize it, then find the closest two images and check similarities by eyes.

The second approach was using Graph Neural Network (GNN). Three models were used: outdoor and indoor models, trained on MegaDepth and ScanNet datasets, respectively, and custom model with no pre-trained weights.

The results we arrived at when training the custom model were incoherent. We've had better results and conclusions when using t-SNE and the pretrained models on our images.

Introduction:

The Dead Sea Scrolls were discovered in different caves in the Judean Desert and other sites in the area in the 50s. The discovery of the scrolls is considered one of the most important archaeological findings in the Land of Israel. The scrolls were torn and became small scraps of paper, and hard to decipher and read. The archeologists brought these pieces to the museum and took photos of them.

Our graduation project consists of working on the scrolls, trying to match each piece to the scroll it belongs to. One of the tasks that were given to us by the Museum of Israel was to find the pieces that appear in more than one photo, that were taken in the 50s. After some research, we found a few approaches that may assist in doing the task. And so, we've decided to try those approaches on the images that we were given in hope that we find pieces that appear more than once, or at least, find pieces that look similar to one another.

Previous work:

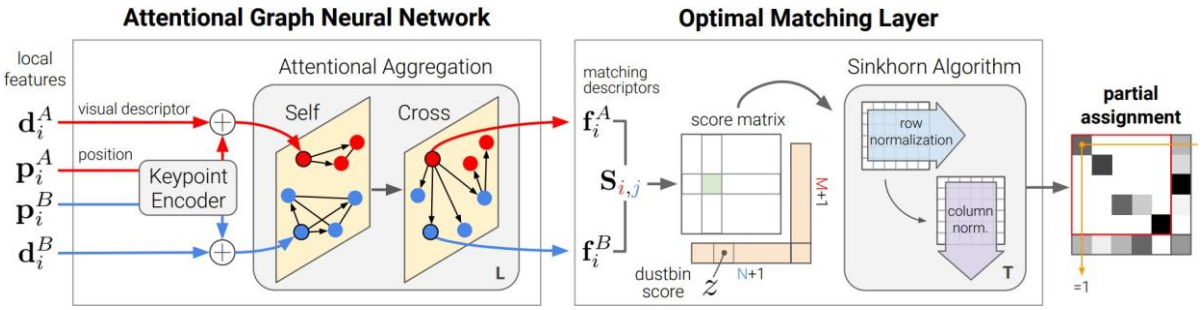
- SuperGlue: Learning Feature Matching with Graph Neural Networks
  We are using their pretrained models to match different pieces and find pieces inside the photos containing various pieces.
- Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data

Data:

The data we were given by the Museum were captured photos of pieces of the Dead Sea Scrolls (107 images, dimensions 1280x720) that were taken in the 50s, in addition to their masks. To work with the data, we had to apply some pre-processing.

We applied segmentation to the images, separating the pieces from the background. After that, we applied various algorithms like Watershed, in order to separate each piece from the other as best as possible. Then each piece was separated into a unique image file, with a black background. The pieces were rotated so that all pieces were aligned the same way. We applied padding to the images, so that all images have the same dimensions as the largest image (4076 pieces inside assets/data/). Then we took a list of the images' names and created a text file, with each line containing the files names in pairs, to match each piece with another (assets/pieces_pairs_names.txt). In addition, we made pairs of each original photo with every piece it contains, to check if we can find pieces in the photos (assets/pieces_to_pictures_pairs_names.txt).

Methods:



We use SuperGLUE by Magic Leap, which is a Graph Neural Network combined with an Optimal Matching layer that is trained to perform matching on two sets of sparse image features. SuperGlue operates as a "middle-end," performing context aggregation, matching, and filtering in a single end-to-end architecture. We were provided two pre-trained weights files: an indoor model trained on ScanNet data, and an outdoor model trained on MegaDepth data.

The SuperGlue architecture: "SuperGlue is made up of two major components: the attentional graph neural network, and the optimal matching layer. The first component uses a keypoint encoder to map keypoint positions p and their visual descriptors d into a single vector, and then uses alternating self- and cross-attention layers (repeated L times) to create more powerful representations f. The optimal matching layer creates an M by N score matrix, augments it with dustbins, then finds the optimal partial assignment using the Sinkhorn algorithm (for T iterations)."

In SuperGLUE, the color of the lines indicate the confidence level of matching, with red color being highest confidence level, and blue being lowest confidence level. Image augmentations were applied during training, in addition to adding random keypoints to the image.

After running the SG models on our images, files with the extension (.npz) are created. We read these files and from them we can find pairs with the highest number of matches found.

We've also used (auto-encoder + t-SNE) as an alternative approach to using GNN.

RESULTS:

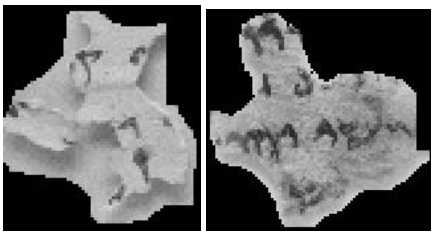
We ran our experiments on 4 different models: (auto-encoder + t-SNE), SuperGLUE indoor, SuperGLUE outdoor, SuperGLUE custom.

The custom SG model was trained on our custom dataset. The model had the following hyperparameters to perform ablation study: image size, keypoint threshold, Sinkhorn iterations, match threshold, NMS (Non-Maximum Suppression) radius, match threshold, learning rate, and number of epochs.

Evaluation: Due to time constraints and limited knowledge of Graph Neural Networks, we were not able to evaluate the SuperGLUE models' performance on our custom dataset. The reason is because we need three key ground truth matrices for each pair: intrinsics matrix of image1, intrinsics matrix of image2, and matrix of the relative pose extrinsics. Due to the fact that our dataset is custom, we were not provided with such matrices, and we had no time to research how to obtain the required information. We've decided to use loss as a metric to measure the "improvements" on our custom model. As for the pre-trained models, we've decided to run inference on all pairs, and pick the results with the highest number of matches found.

(Autoencoder + t-SNE) results:

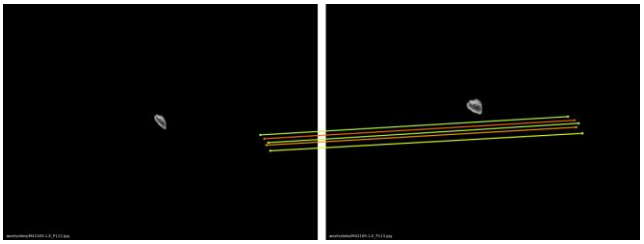
We found the closest pairs and checked with our eyes, but no pairs were identical. They had similar shape and color, but nothing to indicate that they are the same piece. Example:



SuperGlue Custom dataset (100 epochs, learning_rate = 0.001):

Input Size	KeyPoint threshold	match threshold	Sinkhorn iterations	NMS radius	Loss
640x480	0.005	0.2	20	4	0.59368173795
1280x720	0.005	0.2	20	4	0.58212364905
1280x720	**0.2**	0.2	20	4	0.58847993481
1280x720	0.005	0.2	**100**	4	0.57190559816
1280x720	0.005	**0.3**	20	4	0.57390134103
1280x720	0.005	0.2	20	**8**	0.58892987908
1280x720	0.005	**0.001**	20	4	0.56578333077
1280x720	0.005	0.2	20	**16**	0.56391640682
1280x720	0.005	**0.3**	**100**	**16**	0.56386571683

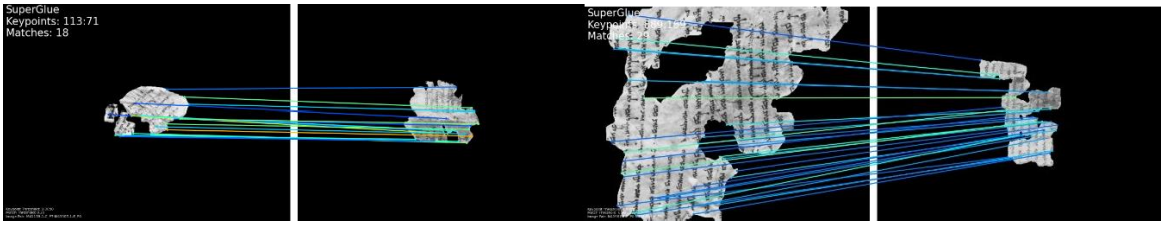
The custom model was not able to match the keypoints. Example:



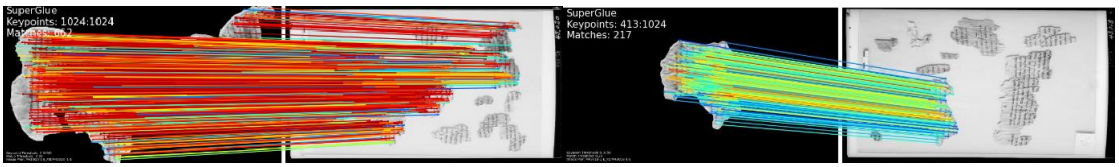
SuperGlue Pretrained (indoor + outdoor):

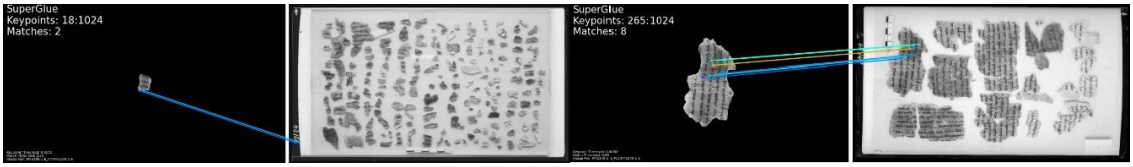
The pretrained models had no difference when matching pairs of our custom dataset. Both gave similar results:

Piece-to-Piece: When trying to find matches between two pieces, the models were able to find the highest between two identical image files (as expected). However, looking at other results, the model found matching keypoints between two different pieces, but with similar “shape”.



Piece-to-photo: When trying to see if pretrained models can recognize a piece inside a photo, we found out that the models are able to find big pieces inside photos that contain a relatively small number of pieces. In contrast, the models had worse results trying to find small pieces inside photos containing large numbers of pieces.





Error analysis:

(Autoencoder + t-SNE): We assume that the method finds pieces with similar shape and colors but does not take into consideration the text written on the piece.

SuperGlue (custom dataset): The model failed to give any good results that we could compare with the pretrained models. Most likely due to the nature of our custom dataset being relatively small and no ground truth provided with it.

SuperGlue (indoor + outdoor): The model fails when trying to find small pieces inside photos with large number of pieces in it. In order to get better results, I recommend using dataset of photos containing big pieces in small numbers.

Conclusions:

We've tested 4 models on our custom dataset, and looking at the results, we concluded that using the pretrained SuperGlue models seems to be the best option compared to the rest.

Using the t-SNE model, we found no identical pieces, only similar ones in shape and color. We have no way to know for certain, but as of right now, we assume that the results indicate that no piece exists in more than one photo. Only the Museum of Israel can confirm or deny that statement.

Looking at the table for custom SuperGlue training, we see that the following helped achieve slightly better results in terms of loss: using higher resolution images, higher Sinkhorn iterations. Changing other hyperparameters (increasing, decreasing) resulted in incoherent results. Due to missing data and time constraints, measuring the accuracy of the model was not possible. For future studies, we recommend looking into finding the "ground truths" for the custom dataset if accuracy measurement is needed.

As for the pretrained SuperGlue models, we found that they were able to find similarities between pieces that do indeed look identical from afar. As for identifying a piece inside photos with other pieces, it worked quite well with large pieces, but not so well with smaller ones. In addition to using larger pieces, we recommend trying images with higher resolutions if possible.