SuperGlue: Learning Feature Matching with Graph Neural Networks

Amir I. Arfan, Tiantian Wang, Jerry Lai

A report presented for fulfillment of CS 294-082

Electrical Engineering and Computer Sciences
University of California, Berkeley
Dec.16, 2021

1 Introduction

3D localization and pose estimation are crucial tasks in robotics and autonomous driving. A key component is to perform local feature matching between a pair of images, which is challenged by illumination changes, occlusion, blur in the 2D data. SuperGlue[1] is a neural network that achieves robust feature matching by performing partial assignment between two set of local features.

In the original publication of SuperGlue, it is used in 3 experiments: estimating homographies, indoor matching, and outdoor matching. This paper focuses on the first experiment, estimating homographies. In this experiment, the model is trained on pairs of images from a subset from the Paris Dataset [2], then tested on a different subset from the Paris Dataset and on the Oxford Dataset [2].

1.1 Machine Learner and Dataset Properties

1.1.1 Input and Prediction labels

The SuperGlue machine learner trains on pairs of images in the Paris Dataset. For each training sample, the machine learner takes as input 2 set of keypoints (one for each image) and 2 "descriptor" tensors describing the keypoints. The descriptors and keypoints are produced by the SuperPoint network.

The label that the machine learner needs to predict is a list of index pairs, representing matchings between keypoints.

The part of the machine learner that is examined by this paper does not include the final layer which produces the matchings. Instead, it stops at assigning a score to each keypoint pair, to be used in matching.

To succeed, the machine learner must correctly predict the matching labels.

1.1.2 Annotations

The annotations included points that matched on objects that appeared in multiple photos. The dataset also contained pairs of an identical image warped with different homographies, for which the provided matching points were guaranteed to be accurate.

1.1.3 Accuracy metric and original performance

In the published paper for SuperGlue, the accuracy metric used was AUC when predicting homographies. To convert point matches to homographies, the authors tried both RANSAC to make robust homographies, and DLT, a least-squares solution that is not robust.

It showed that superglue achieved superior AUC's for both RANSAC and DLT. The results are shown in table 1.

Method	RANSAC AUC	DLT AUC
SuperGlue	53.67	65.85
Nearest neighbors	39.47	0
Nearest neighbors + mutual	42.45	0.24
Nearest neighbors + PointCN	43.02	45.40
Nearest neighbors + OANet	44.55	52.29

Table 1: AUC results

In our experiments, instead of recovering homographies, we used simple accuracy in predicting matching point pairs.

1.1.4 Structure Diagram

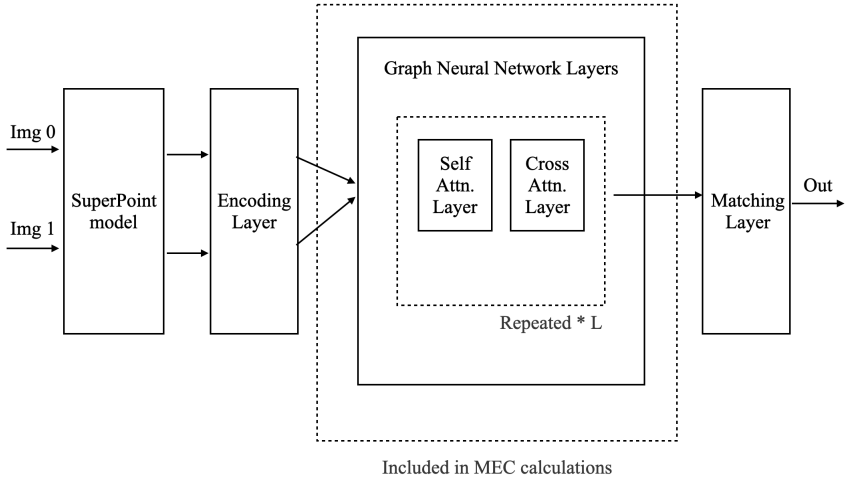

Figure 1: Model Diagram

In the default configuration of SuperGlue, L (the number of Attention Layer pairs) is 9.

2 Memory Equivalent Capacity

2.1 MEC of Dataset

As input, the neural network does not take in images; instead, it takes in the output from the SuperPoint [3] neural network.

This input comes in pairs of images. Each image in the pair is represented by a size 3×512 tensor of keypoint encodings.

This is then passed through an MLP encoder, which outputs a pair of 256×512 size tensors for the main graph neural network.

2.1.1 Methodology

To calculate the MEC of the Paris Dataset, which was used for training, we used these tensors provided by the encoder.

Before calculating the MEC, we passed each tensor through a 8×8 2D Maxpool layer. This is due to the constraints of computer power; without the Maxpool layer, there would be over 5×10^5 features. After the Maxpool, each datapoint had 4096 features.

Subsequently, we transformed the dataset from a feature matching problem into a classification problem by constructing a classification dataset. Each image pair in the original dataset had a list of indices of ground-truth matching points between the two images. Each pair of matching points is its own entry in the classification dataset. Thus, each entry in the classification dataset has the following data:

- Features:
  - All of the features from the entire image pair from the original dataset
  - The index of the first point of the matching pair

- Labels:
  The index of the other point of the matching pair

Additionally, the indices were capped at 150. Any point pair where either of the points had an index of above 149 was omitted. This number was chosen because it was higher than the number of point in most images, but kept the final dataset to a reasonable size.

Finally, the processed dataset had a size of 217584 rows, 4097 feature columns and 1 label column. 4096 of the feature columns had floats from the pooled tensor described above; 1 has an index integer between 0 and 149. The label column has an index integer between 0 and 149. This dataset was given as input to Brainome.

2.1.2 Results

Brainome return an Estimated MEC of 76614 bits for neural network, with an expected generalization of 20.37 bits/bit.

The capacity progression of an ideal machine learner is as shown in figure 2. The shape of the curve shows that the model generalizes reasonably well, albeit not perfectly.

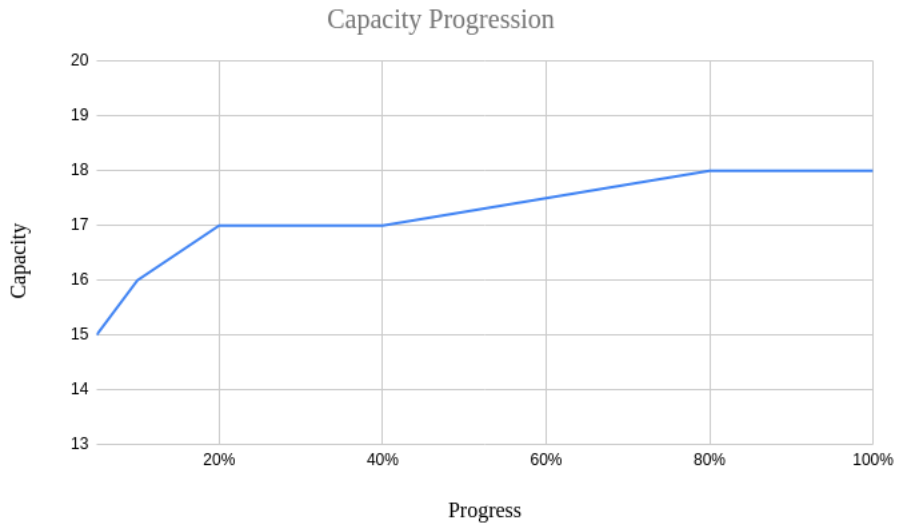

Figure 2: Capacity Progression

2.2 MEC of Current Neural Network

The structure of the neural network is shown in figure 1.

The size hyperparameters of the model are:

- SuperPoint model: 3×512 outputs per image.
- Keypoint Encoding Layer:
  - 3×512 inputs per image for input layer size 3.
  - Layer sizes of $[32, 64, 128, 256$ (output)].
  - 256×512 outputs per image.
- Each Attention Layer (out of $2 \times$ L layers):
- Input layer size 256
  - Input layer size 256.
  - Attention (ignored for MEC)

– MLP: Layer sizes of $[512, 512, 256$ (output)$]$.

In the calculation of the MEC for the Neural Network, we omitted the encoder, as these are only filters and do not do the decision making.

Therefore, the MEC calculations only include the L pairs of graph neural networks, meaning $2 \times L$ self/cross attention layers end-to-end each containing a $[512, 512, 256]$ MLP.

The information passed through the layers is bottlenecked at input start of the encoder. There, the encoder takes, for each image, a 3×512 size "descriptor" tensor that is the output of the SuperPoint model. Since each training sample has two images, that is $2 \times 3 \times 512 = 3072$ outputs provided by SuperPoint. Therefore, no layer in the SuperGlue network can have over 3072 bits of MEC.

The first layer (input 256, output 512), with $256 \times 512 = 79872$ parameters, has far more parameters than the previous bottleneck; so its capacity is 3072 bits.

The second and third MLP layer of the first attention layer, with sizes of 512 and 256, respectively, both have 512 bits of memory capacity, as that is the number of outputs of the first layer. This means 4096 bits for the first attention layer.

Every MLP layer in the subsequent attention layers has 256 bits of memory capacity, as the third layer is now the bottleneck. This means 768 bits for each subsequent attention layer.

Thus, the MEC in bits for a network with L layers is :

$$MEC \text{ (bits)} = 4096 + (2L - 1) \times 768$$

In the default configuration, $L = 19$, so the MEC is **17152**.

3 Training Process and Results

3.1 Training Configuration and Methodology

The official training code is not available through the model's GitHub repository; thus, our training implementation is implemented using Python. The implementation is created with the intention of being repeatable and requires minimal change to work. All random aspects are fixed using seeds. Additionally all the training and testing runs produced logs which are stored as text-files, and the classification metrics are extracted and stored as Numpy files, which can be loaded and compared in Python.

The training machine specifications are given by:

- Lambdalabs Cloud Computer
- RTX 6000 (25 GiB VRAM)
- EPYC™ 7502P (2.50 GHz) 6 vCPUS
- 46 GiB RAM

The training was performed on the Paris dataset [2], specifically the following queries: *Eiffel Tower Paris, Moulin Rouge Paris, La Defense Paris , Paris (General) ,Louvre Paris, Hotel des Invalides Paris, Arc de Triomphe Paris, Sacre Coeur Paris, and Pompidou Paris.* For validation, during the training process, the *Pantheon Paris* query was used. For independent testing, the queries *Notre Dame Paris* and *Musee d'Orsay Paris* were used, also the Oxford dataset was used [2].

The model was trained on different configurations, where for each configuration, the number of layers in the Graph Neural Network (GNN) was reduced to reduce the model's Memory Equivalent Capacity (MEC), the configurations used for the GNN are presented in table 2.

Config #	No. of Self Attentional Layers	No. of Cross Attentional Layers	Training Time
Config 0	9	9	$\approx$ 8 hours
Config 1	7	7	$\approx$ 6 hours
Config 2	5	5	$\approx$ 5 hours
Config 3	3	3	$\approx$ 4 hours
Config 4	2	2	$\approx$ 3 hours
Config 5	1	1	$\approx$ 2.5 hours

Table 2: Configurations for Training and Testing

3.2 Results and Analysis

3.2.1 Accuracy Training

The model was trained for accuracy using the different configurations from table 2. The accuracy progression of each configuration for the training data is shown in figure 3 and table 3.

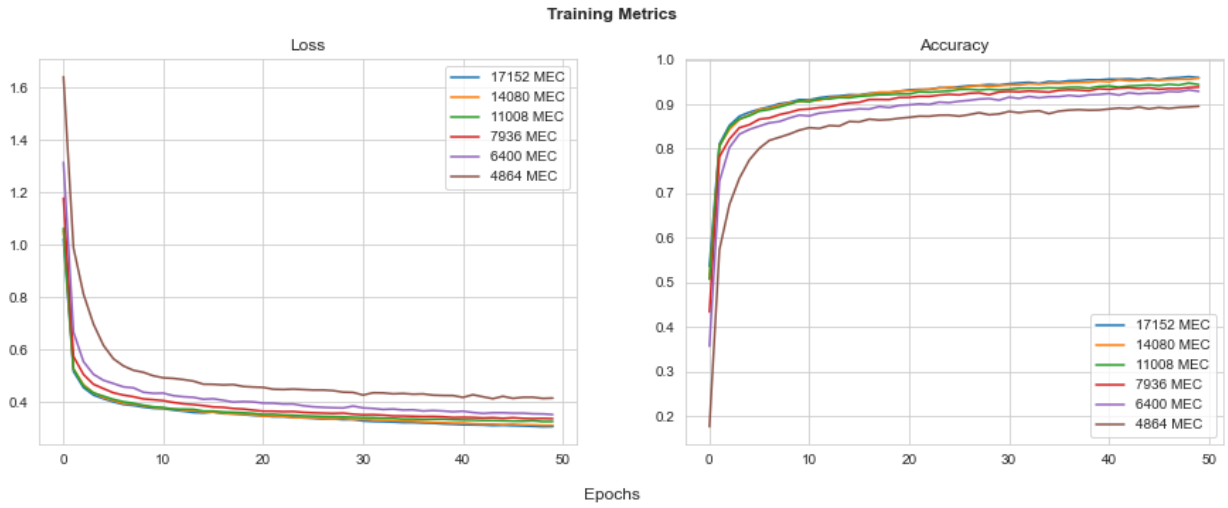

Figure 3: Training Loss and Accuracy per epoch with different model MEC

Training Maximum Accuracy Results	
MEC (Config #)	Accuracy (%)
17152 (Config 0)	96.19%
14080 (Config 1)	95.87%
11008 (Config 2)	94.77%
7936 (Config 3)	93.85%
6400 (Config 4)	93.21%
4864 (Config 5)	89.56%

Table 3: Maximum Accuracy Achieved Per Config

The results show that the machine learner is **not** able to achieve 100% memorization. We hypothesize that the amount of MEC of the model is lower than the amount estimated by Brainome leaves it unable to memorize the dataset. One could increase the amount of GNN layers even further to achieve higher model MEC, and we assume that one could thus confidently achieve memorization. Increasing the number of epochs would help as well. This would, however lead to a further increase in training time but should be achievable, we did not train with a higher amount of GNN layers due to the long training time and amount of memory required.

The accuracy results for the validation dataset can be seen in figure 4. The reduction of MEC of the model seems to affect the validation accuracy and loss in a more variational manner, which might be an indicator of the memorization capabilities of the machine learner, which will be further investigated under training for generalization.

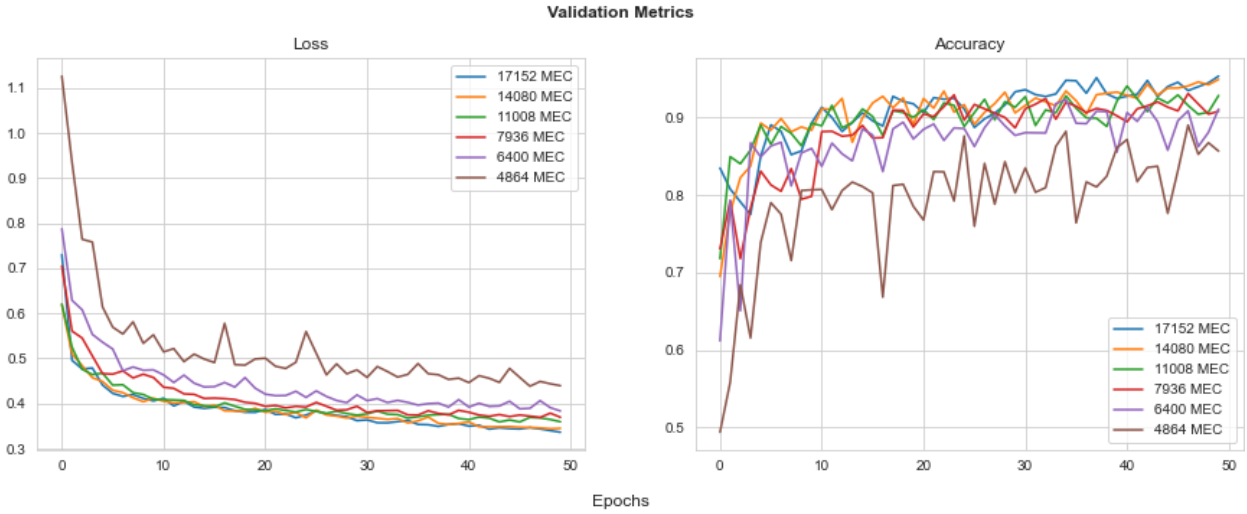

Figure 4: Validation Loss and Accuracy Per Epoch with different model MEC

3.2.2 Generalization Training

The training results when training for generalization is presented in figure 5. Where quantization is inversely proportional to MEC. One can observe that the validation and training curve follow each other quite closely. This might be due to the nature of the problem and the datasets used. The problem is classifying pairs of images, and these pairs stem from the same location, Paris. Which might indicate that the training and validation data are correlated.

Nonetheless the point N_{approx} indicates where the difference between the training and validation is low enough without falling off the accuracy cliff. This point refers to config 4 (see table 2) with a MEC of 6400. The expected generalization is calculated from equation 1. We can see from the generalization figure 6, how the generalization curve proceeds with increasing MEC. The corresponding N_{approx} generalization value as shown in the figure is **1.75**. Since $G \geq 1$ indicates that the model **might** not be overfitting. A generalization value of ≤ 1 indicates a definitive overfit, the larger the generalization value is the less chance of overfitting. From figure 6 we can see that the higher the MEC is for the model the lower the value of G. Which indicates that the model might be overfitting for larger MEC values. These results will be verified on an independent test dataset.

$$G = \frac{\#Correct\ Classifications}{MEC} \tag{1}$$

3.2.3 Testing

For testing the models evaluated the two subsets of the Paris dataset which were not included in the training or validation, specifically the *Notre Dame Paris* and *Musee d'Orsay Paris* queries. The testing results are shown in figure 7 and table 4. The test results show that config 1 is the one with highest test set accuracy. Our chosen N_{approx} is not the one with the highest test set accuracy but performs better than config 3 despite the fact that config 3 has a larger MEC, which indicate that the generalization plays a role. We hypothesize that config 0 and config 1 are the ones with highest test set accuracy due to the similarity of the training and test set data. Despite having lower MEC config 4 (N_{approx}) has similar accuracy to config 2, which also emphasizes the importance of generalization.

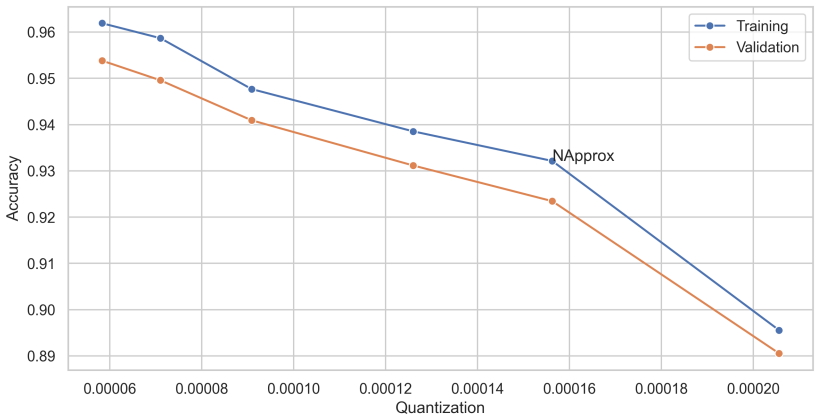

Figure 5: Quantization - Accuracy Graph

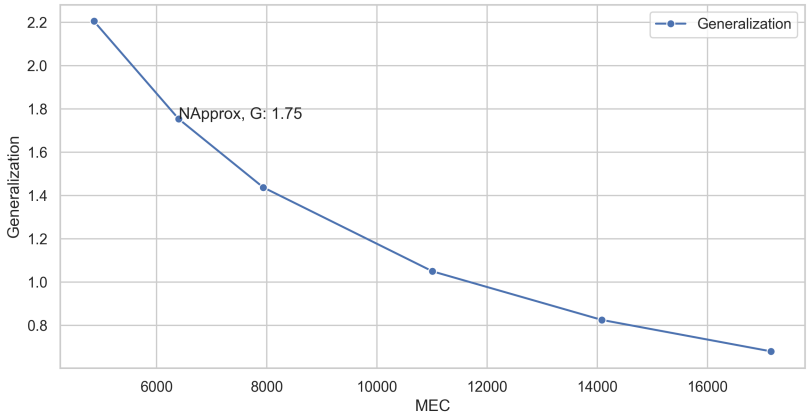

Figure 6: Generalization values with increasing MEC

The testing was also conductde on the Oxford dataset [2], this can be seen in figure 8 and table 5. We can see the same pattern here, but with closer results of config 4 despite the reduction in MEC. Again emphasizing the importance of generalization, but higher MEC with lower generalization are performing better. We hypothesise that this is still due to the similarity of the problem which is to classify landmarks, but needs further investigation on different test datasets.

We do not necessarily need to try a different machine learner as the accuracy is quite satisfactory with the results of Sarlin et al. [1] in mind. One could of course test out a different machine learner and compare the accuracy/MEC ratios.

We are quite **confident** in the results as they have been evaluated through our training script, all the logs are produced from the training procedure and show the progression of the loss and accuracy. Albeit the generalization did not hold, but it shows significance compared to different configurations, and further investigation could lead to making the choice of model. One could perhaps go with the model with higher generalization and accept the trade-off in accuracy with the benefit of gaining stability and perhaps more useful model with different datasets, this however needs further investigation.

4 Conclusion

The keypoints are encoded with both the visual features and the positions. The neural network can be considered as a multiplex graph with two type of undirected edges, where the nodes are the keypoints

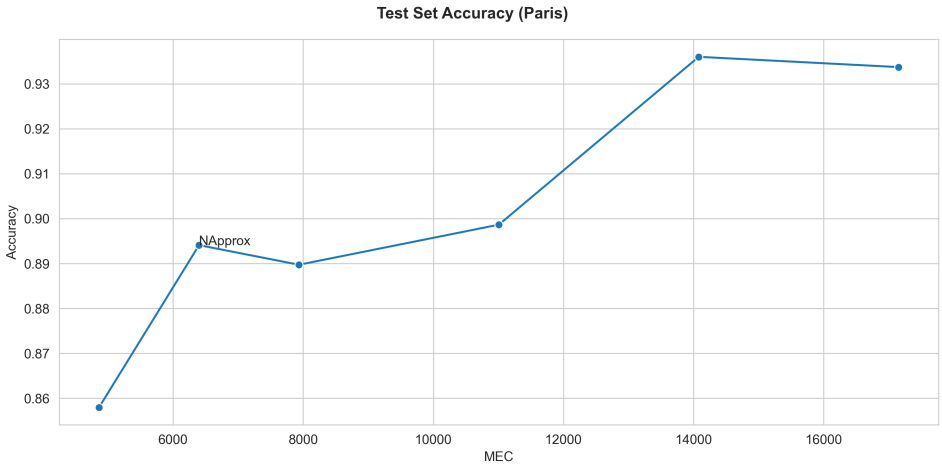

Figure 7: Test Set Accuracy with different model MEC on the Paris test subset

Test Set Accuracy Results (Paris)	
MEC (Config #)	Accuracy (%)
17152 (Config 0)	93.37%
14080 (Config 1)	93.60%
11008 (Config 2)	89.86%
7936 (Config 3)	88.97%
6400 (Config 4 N_{approx})	89.41%
4864 (Config 5)	85.79%

Table 4: Accuracy Achieved Per Config for Test Data (Paris)

of both images. Intra-image edges connect one keypoint to all other keypoints within the same image; inter-image edges connect one keypoint to all keypoints from the other image. An update is aggregated simultaneously across all edges. Therefore, the network is able to capture not only co-occurrence or adjacency within the image but also global cues given by the other image. These are the application specific measures. Inter-image matching itself has a flavor of cross-validation. We observe that as the layer increases, self-attention narrows down to focus on neighbouring regions and cross-attentions gradually reduces the candidate set. This is exactly what we anticipate a repeatable and reproducible model would look like.

The model has leveraged a dustbin to hold unmatched keypoints, which is a common technique in graph neural networks. The optimization of the score matrix can therefore be solved by optimal transport [4]. The probabilistic matching framework is a *real* soft matching, rather than a relaxation of hard matching since it is entropy-regularized.

5 Future Work

When we combine SuperGlue with SuperPoint [3], the result ultimately surpasses other widely-used feature matchers. It perform very well even under the condition as illumination changes, repeated pattern and large viewpoint. The validation accuracy is satisfactory. However, the dataset has limited diversity in terms of landmark notations and types.

Other measures that could potentially boost the model performance:

Doubly Stochastic Matrix Instead of incorporating all the feature points of the image pair into a single hyper-graph, we could generate two hyper-graphs [5] each containing features from one image. The edges represent the intra-image relation. The similarity of the two hyper-graph is computed using global transformation before zoomed in to match hyper-edges (intra-image relation). In this case, the computation cost (projection and etc.) for edge update could be

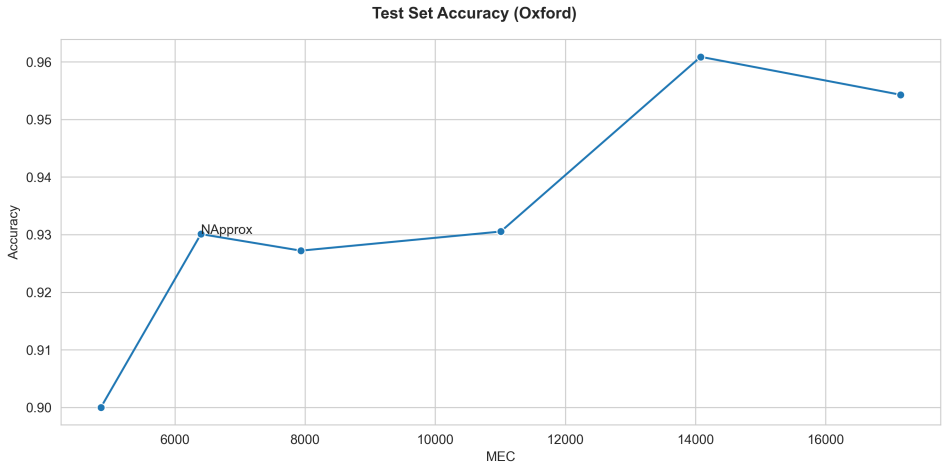

Figure 8: Test Set Accuracy with different model MEC on the Oxford test data

Test Set Accuracy Results (Oxford)	
MEC (Config #)	Accuracy (%)
17152 (Config 0)	95.42%
14080 (Config 1)	96.09%
11008 (Config 2)	93.05%
7936 (Config 3)	92.72%
6400 (Config 4 N_{approx})	93.00%
4864 (Config 5)	90.00%

Table 5: Accuracy Achieved Per Config for Test Data (Oxford)

reduced. Moreover, dustbin is no longer needed, therefore avoid the ambiguity in the score assigned to unmatched keypoints.

Aggregation We could aggregate the local features with Fisher vectors [6] or VLAD [7] to produce new global features. With these compact global features, the training time might shrink and the matching process could greatly speed up during cross-attention update. As [8] points out, deep learning based features (SIFT [9]) are performing differently from shallow features when working with aggregation. Using sum pooling to aggregate deep learning features on the last layer is shown to achieve better accuracy than max pooling.

Depth-based Sampling Current implementation of the model is sampling uniformly at per 10 pixels, with no discrimination among feature points. This could lead to a sparser sampling density in far regions than that of close regions. The low resolution will result in a lower accuracy in far regions, which cannot be revealed by the matching accuracy of the entire image. Moreover, objects at far regions occupy fewer pixels and are smaller and blurry. We could take advantage of the depth information of the images when sampling to get a suitable matching density. Then perform adaptive feature refinement [10] to get discriminative features. The accuracy at far regions will increase and consequently the overall accuracy will be improved. This measure would be extremely meaningful if to apply the model to autonomous driving and robotics where detection of objects at far regions is crucial.

Contribution

Amir I. Arfan

The work of section 3. Implemented the training, visualization and parsing scripts. Performed training and testing, and parsed the results needed from the logs produced. Visualized the training and testing results. Interpreted the results gained from the training/testing process.

Tiantian Wang

Team meeting organization. Synchronized the background knowledge of 3D localization and graph neural network among team members. Literature review. Model overview. Model evaluation and investigation. Explored future work.

Jerry Lai

Model introduction, dataset introduction, Brainome analysis, data preprocessing, MEC calculations and related things.

Fixed training, logging code and ran it.

References

[1] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," 2020.

[2] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Revisiting oxford and paris: Large-scale image retrieval benchmarking," 2018.

[3] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," 2018.

[4] G. Peyré, M. Cuturi, *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[5] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[6] G. Lev, G. Sadeh, B. Klein, and L. Wolf, "Rnn fisher vectors for action recognition and image annotation," in *European Conference on Computer Vision*, pp. 833–850, Springer, 2016.

[7] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3304–3311, IEEE, 2010.

[8] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proceedings of the IEEE international conference on computer vision*, pp. 1269–1277, 2015.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[10] H. Zhang, X. Ye, S. Chen, Z. Wang, H. Li, and W. Ouyang, "The farther the better : Balanced stereo matching via depth-based sampling and adaptive feature refinement," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.