Improving odometry with SOTA feature matching model

Ting-Wei Hsu[1], Guo-Wei Hong[2], and Tzu-Hung Chang[3]

[1] Department of Computer Science and Information Engineering, National Taiwan University
[2] Graduate Institute of Electrical Engineering, National Taiwan University
[3] Department of Computer Science and Information Engineering, National Taiwan University

Abstract. Visual odometry utilizes the power of feature matching to recognize the keypoints in consecutive frames for further estimation of the camera motion. In this report, we dive into the comparative analysis of three prominent techniques employed in the feature-finding/matching stage: Superglue, Lightglue, and LOFTR.

Keywords: feature matching · visual odometry.

1 Introduction

Visual odometry, as a fundamental aspect of computer vision, plays a key role in robotic systems and the autonomous navigation. One of the critical stages in visual odometry is feature-finding and matching, which identify keypoints in images and matches them between consecutive frames. This stage is crucial for the accuracy and efficiency in estimating the motion of a camera or sensor. In this report, we dive into the comparative analysis of three prominent techniques employed in the feature-finding/matching stage: Superglue, Lightglue, and LOFTR in the feature-finding/matching stage.

2 Related Works

2.1 Feature finding

Local feature descriptors play a key role in understanding the segments of the image. In visual odometry, we need to identify the same points in successive video frames to estimate the relative motion of the camera. SIFT[6], scale-invariant feature transform, stands as a pioneering method in feature finding in its age, recognized for its robustness in identifying keypoints and descriptors across various scales and orientations. Its ability to withstand variations in illumination and viewpoint makes it a stalwart choice in traditional computer vision applications.

However, in a real-time application, SIFT is too computationally intensive for practical usage. This is where FAST[8] should be introduced: the key idea that FAST comes in handy is the acceleration of the identification of corner. Unlike the SIFT builds a sequence of scale/orientation-free filter,

SIFT streamlines this process by examining the neighbors around the corner candidate, which significantly accelerates the corner detection process.

On top of that, ORB[9] is also a realtime-detection of features. Combining FAST and BRIEF[1], ORB is adept at feature finding in diverse conditions. Its binary descriptor facilitates efficient matching, making it a compelling choice for applications requiring a balance between speed and robustness.

On the other hand, Superpoint[2] leverages a deep neural network architecture, to predict keypoints and descriptors directly from image data. By eschewing the traditional multi-step approach, Superpoint streamlines the feature extraction process, demonstrating its efficacy in real-time applications.

2.2 Sparse feature matching

Sparse feature matching builds upon the foundation laid by feature finding, where distinctive points and descriptors are identified in the images first. After the keypoints are found, sparse feature matching matches feature points by comparing their descriptors.

SuperGlue[10], a learning based methodology, combined feature detection with the optimal transport problem. The cost function of this optimization is predicted by a Graph Neural Network using self and cross attention technique by Transformer[12] on the keypoints and descriptors. After that, the optimal transport problem is solved by Sinkhorn algorithm[11], which is a differentiable version of the Hungarian algorithm[7] famous in bipartite matching. This clever approach enables SuperGlue to produce reliable and robust sparse feature matches, making it well-suited for applications like visual odometry.

LightGlue[4] is a subsequent work of SuperGlue. Based on SuperGlue, LightGlue inherit the main ideas of SuperGlue, but further excels in accuracy and latency. LightGue focuses on refining the matching process. Instead of directly applying the Sinkhorn algorithm to the confidence map, LightGue employs a pruning mechanism to eliminate less probable matches. This process is iteratively repeated multiple times. LightGue's iterative refinement approach aims to optimize matching precision by iteratively improving the reliability of selected keypoints, making it a valuable addition to the sparse feature matching toolkit.

2.3 Dense feature matching

Another approach of feature matching is detector-free methods. Detector-free methods remove the feature detector phase and directly produce dense descriptors or dense feature matches. The idea of dense features matching dates back to SIFT Flow.[5]

A contemporary exemplar in this domain is D2-Net[3], a dense feature matching method that sidesteps the need for explicit feature detection. Instead, D2-Net adopts a learned approach, generating dense descriptors for each pixel in an image. This departure from conventional approaches results in a performance that is comparable or even superior in challenging conditions, such as day-night comparisons and instances where the camera experiences shaking during shuttering.

LoFTR is another detector-free method. Inspired by SuperGlue, LoFTR applies Transformer[12] with self and cross attention layers to process the dense local features extracted from the convolutional backbone. Also LoFTR adopts multi-resolution approach, addressing the intricacies of feature matching from a unique perspective. Similar to D2-Net, the strength of LoFTR lies in its ability to handle scenarios where traditional feature detectors may falter, especially in environments with significant viewpoint changes or variations in lighting conditions. By directly addressing dense feature matching, LoFTR exhibits promising performance in challenging situations.

3 Methodology

3.1 SuperGlue

SuperGlue starts by receiving keypoints and descriptors generated by methods like SIFT or Superpoint. Subsequently, SuperGlue embeds these keypoints and descriptors into an attention network through positional encoding. The attention network employs a Graph Neural Network (GNN) to establish confidences of each predicted matching. Finally the matching is fine-tuned using the Sinkhorn algorithm on the confidence matrix.

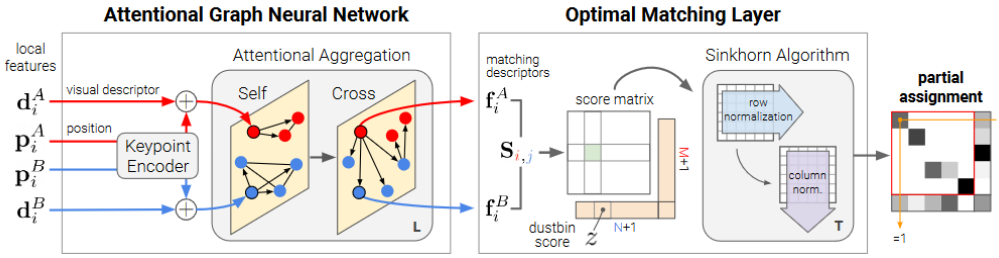

Fig. 1. The flowchart of SuperGlue

3.2 LightGlue

Similar to SuperGlue, LightGlue starts by receiving keypoints and descriptors generated by methods like SIFT or Superpoint and then generate the confidence of each predicted matching using GNN and attention. After that, a classifier determines if there are enough points with high confidence in current round, if not, a pruning and the next round of attention mechanism is applied again.

3.3 LoFTR

LOFTR initiates its feature matching process with a multi-resolution analysis. Instead of exclusively considering the original high-resolution image, LOFTR

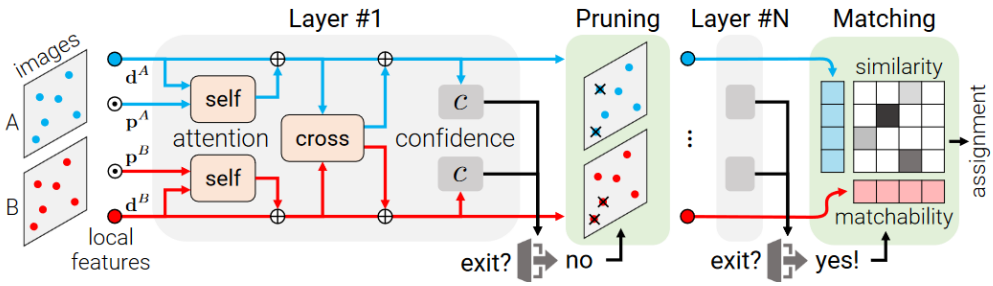

Fig. 2. The flowchart of LightGlue

starts by examining lower resolution versions obtained through down-sampling. After that, incorporation of a pixel-wise attention mechanism is used. On the light of attention, LOFTR can effectively capture long-range dependencies and relationships between pixels, enhancing the precision of feature matching across the entire image. After producing initial dense feature matches based on the analysis of lower-resolution images, LOFTR refines these matches by incorporating information from the high-resolution version of the image.

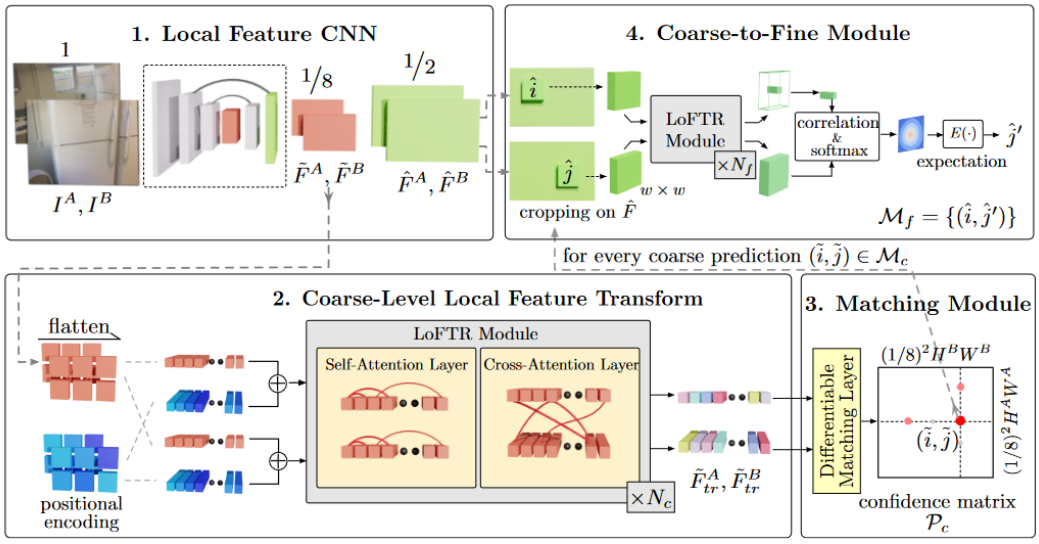

Fig. 3. The flowchart of LoFTR

3.4 Visual odometry

After we obtain the keypoints correspondence of consecutive video frames. We use them to find out the relative motion of the camera in these frames. In

three consecutive frames, we find out the first relative motion between the first one and the second one, and the second relative motion between the second one of the third one. After that, we need to scale the second relative motion to obtain a reasonable concatenation of the relative motions. To cope with the scale problem, we establish two point clouds using the keypoints which appears in all three frames. Then we compare the relative distances in the two point clouds to estimate the scale.

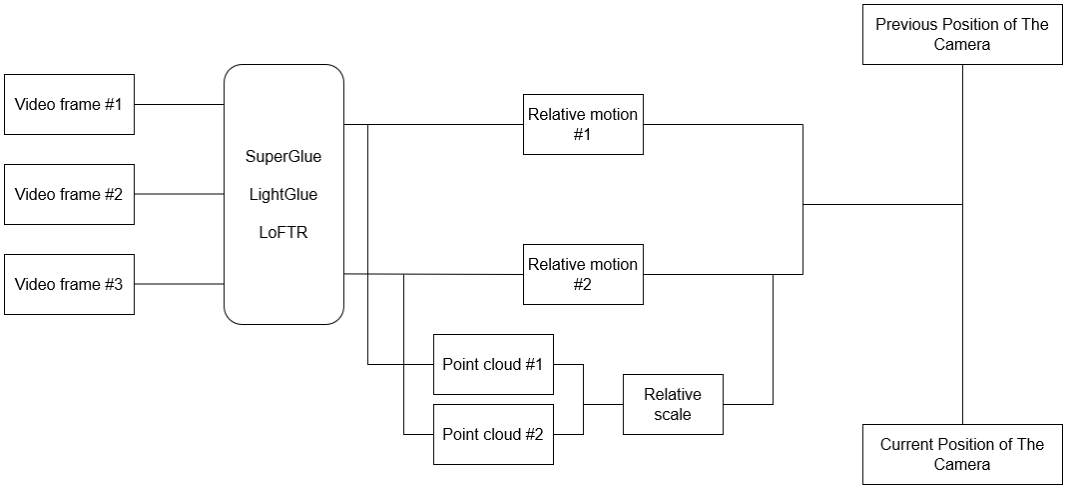

Fig. 4. The flowchart of our Odometry

4 Result

We conducted experiments on two datasets. The first dataset, as provided in homework 3, consists of 507 RGB images with dimensions of 640x360 pixels each. The depicted scene involves circumnavigating a specific area within the campus of National Taiwan University. The second dataset is derived from The KITTI Vision Benchmark Suite, specifically the raw data from the sequence labeled "2011_09_26_drive_0019." This dataset comprises 487 grayscale images with dimensions of 1392x512 pixels each. The original video has a duration of 48 seconds and captures a scenario where the camera initially moves along a straight road and eventually comes to a stop at a turning point. In the following sections, we will examine the results obtained from these two datasets separately. The experimental environment employed Ubuntu 22.04 as the operating system, with an NVIDIA GeForce RTX 3060 GPU.

4.1 NTU Dataset

To begin with the comparison of speed, our results are organized in Table 1. It is evident that the fastest speed is achieved by the original method

(Brute-force descriptor matcher). This can be attributed to the traditional approach employed by the original method, which is not based on deep learning models, thereby ensuring a relatively high speed. Following closely is LightGlue, an improvement upon SuperGlue, exhibiting a notably faster speed compared to SuperGlue. On the contrary, the slowest performance is observed in LoFTR. This can be attributed to LoFTR's utilization of a Semi-Sparse matching approach, resulting in significantly reduced speed when compared to other methods. For real-time display applications, either the original method or LightGlue may be considered more favorable choices. From a 2D perspective, a comparative analysis

Method	Origin	LightGlues	SuperGlue	LoFTR
Time (s)	95	162	633	1495

Table 1. Comparison of Processing Time for Various Methods on the NTU Dataset

of three models and the original method is presented, as illustrated in Figure 5 below. The original video depicts circumnavigation around a rectangular area, resulting in an expected outcome of a loop with rectangular characteristics. It is noticeable that, in the original method, there is a discernible misalignment at the beginning and end, indicating a slight discrepancy. In the LightGlue approach, the disparities at the commencement and conclusion are significantly reduced compared to the original method. Importantly, both the SuperGlue and LoFTR methods exhibit seamless alignment at the video's initiation and termination.

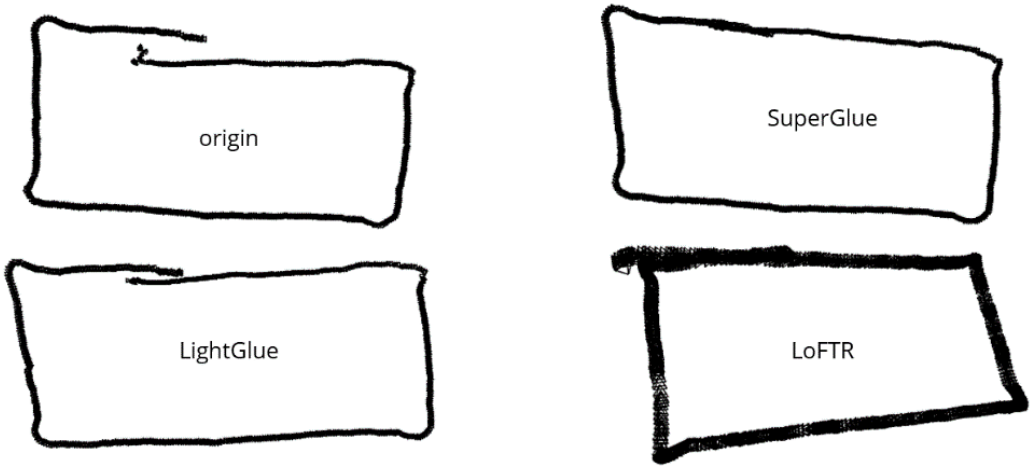

Fig. 5. 2D Odometry Results of Four Methods on the NTU Dataset

Finally, a comparison of vertical errors from a side-view perspective is conducted, as depicted in the Figure 6 below. Since the original video

moves within the same plane, there should not be drastic variations in the vertical direction. It is evident that, regardless of the method employed—origin, LightGlue or SuperGlue—the vertical errors at the head and tail are considerable. However, the LoFTR method demonstrates a substantial reduction in these errors, resulting in the overall trajectory appearing nearly flat. This indicates the effectiveness of the LoFTR approach in minimizing vertical discrepancies, providing a trajectory that closely adheres to the same plane throughout the movement.

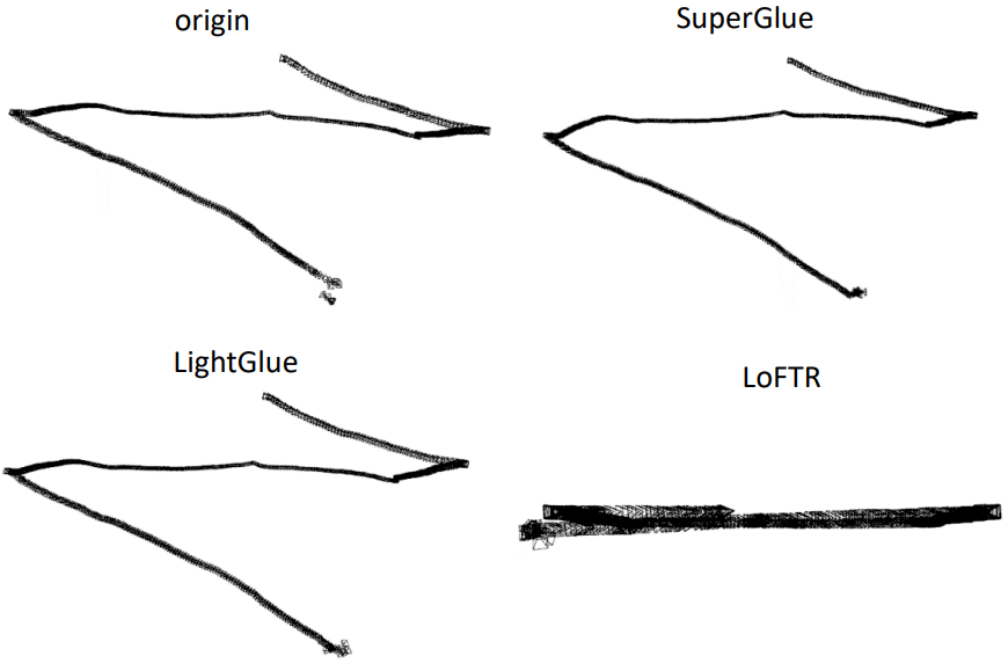

Fig. 6. Visualized Odometry Results of Four Methods on the NTU Dataset

4.2 KITTI Dataset

Firstly, in terms of speed comparison, the results are organized in Table 2. Notably, LightGlue exhibits unexpectedly faster processing speed than the original method. This phenomenon is hypothesized to stem from the absence of constraints on the number of SIFT feature points, potentially leading to an excessive selection of features and subsequently causing a computationally intensive process. Additionally, the parallel computing capabilities of LightGlue utilizing GPU acceleration contribute to surpassing the speed of the original method. The remaining results align closely with those observed in the previous analysis.

Method	Origin	LightGlues	SuperGlue	LoFTR
Time (s)	241	87	584	10000

Table 2. Comparison of Processing Time for Various Methods on the KITTI Dataset

Subsequently, a comparison of odometry visualization results is presented in the Figure 7 below. It can be observed that all four methods successfully depict straight trajectories with a turning point at the end. Due to a brief stop at the turning point, there is a slight divergence caused by moving objects, such as vehicles and pedestrians, persisting in the original images. In the original method, these deviations concentrate primarily at the endpoint. Conversely, the LightGlue method enhances the concentration of these deviations, resulting in more continuous trajectories compared to the original method. The SuperGlue and LoFTR methods disperse these deviations across various spatial positions. However, the removal of points with excessive displacement may potentially enhance the performance of SuperGlue and LoFTR for scenarios involving brief pauses in movement.

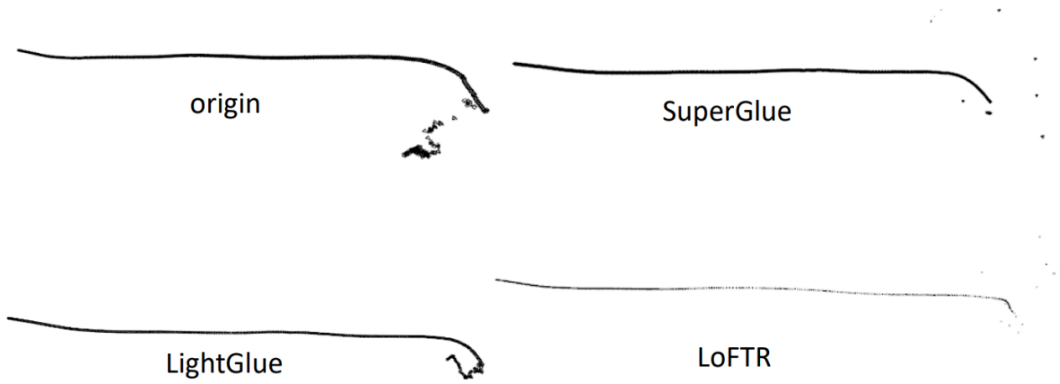

Fig. 7. Visualized Odometry Results of Four Methods on the KITTI Dataset

5 Conclusion

Based on the experimental results, the current State-of-the-Art (SOTA) model, LightGlue, outperforms traditional methods significantly, showcasing superior performance. However, there is still room for improvement, particularly in terms of computational time. Currently, only LightGlue competes with traditional methods in terms of speed, while the semi-sparse model lags significantly behind in terms of computational efficiency.

In our experiments, we encountered challenges when using an RTX3050 for LoFTR execution, as its memory usage proved demanding. A more capable RTX3060 was required to handle LoFTR's memory requirements. This concern is noteworthy

for the application of visual odometry, especially in mobile devices, which often contend with limited memory resources.

The visual odometry method taught in class is designed for scenarios where keypoints are found. As a result, it is primarily suitable for use with sparse models, which inherently possess keypoints. During our implementation process, we faced difficulties with the semi-sparse model, where keypoints for image matching are determined based on two images. However, when performing triangulation, one of the images requires keypoints from two different matching instances unique to the semi-sparse model. We explored various methods to address this challenge, such as considering points with closer distances between the two instances as the same point (S). Unfortunately, the results were unsatisfactory. Another approach involved using new keypoints to match with the previous image using the Brute-Force Matcher for a second round of matching, and then utilizing the results for triangulation. This alternative approach yielded significantly better results.

To effectively integrate the semi-sparse model into visual odometry, a different odometry process or a complete redesign of the entire process may be necessary to achieve optimal results. This is an area that warrants exploration and expansion in future research, with a focus on optimizing keypoint triangulation for semi-sparse models in visual odometry applications.

References

1. Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Computer Vision – ECCV 2010, volume 6314, pages 778–792, 09 2010.
2. Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 337–33712, 2017.
3. Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.
4. Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In ICCV, 2023.
5. Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(5):978–994, may 2011.
6. David G Lowe. Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision, volume 2, pages 1150–1157. Ieee, 1999.
7. James Munkres. Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics, 5(1):32–38, 1957.
8. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, Computer Vision – ECCV 2006, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
9. Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In 2011 International Conference on Computer Vision, pages 2564–2571, 2011.

10. Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4937–4946, 2019.
11. Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. Pacific Journal of Mathematics, 21:343–348, 1967.
12. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.