Redefined Relationformer with Improved Matching for Line Detection

Chen Yu (Erioe) Liu
Boston University
erioe@bu.edu

Yuanhao Shen
Boston University
shenyh11@bu.edu

Abstract

Line detection or graph generation plays a crucial role in various computer vision tasks. Although previous research has made significant progress in modeling inter-object relationships, it still faces challenges in endpoint prediction and matching stability. To address these issues, our aim is to reproduce and validate Relationformer and its variants. We further propose a distributed-based approach that replaces the traditional Dirac delta-based box prediction to improve node localization accuracy. Furthermore, inspired by General Focal Loss [8] and D-EIM [7], we redesign the loss function combined with data augmentation techniques to reduce the impact of negative samples and enhance the robustness of the model. Experimental results demonstrate that our RED-Relationformer significantly improves correctness, completeness, and APLS scores, as shown in Figure 1.

GitHub: *https://github.com/chenyu020816/Redefined-Relationformer-with-Improved-Matching-for-Line-Detection.git*

1. Introduction

Line detection or Graph generation in images plays a crucial role in a variety of applications, such as the automatic extraction of roads, railways, and rivers, which supports hydrography change analysis and mining resource prediction [1]. Beyond geospatial analysis, line detection is also widely used in medical image processing, particularly to extract blood vessel networks in retinal images or angiograms, critical for the early diagnosis of conditions such as diabetic retinopathy and cardiovascular disease [10, 13].

However, existing models often struggle to generalize across different scenarios and frequently fail to accurately capture the position and connectivity of line segments, limiting their reliability and applicability. Through an analysis of previous methods, we identify two primary issues.

First, for DETR-based models [13, 16, 18], positive and negative samples are assigned by matching predicted nodes to ground truth nodes using the Hungarian algorithm. While this approach is suitable for traditional object detection, it poses challenges for graph-based line detection. In this setting, the placement of nodes in ground truth graphs is often influenced by the annotator's subjective labeling habits. As a result, predicted nodes that correctly lie on the line may not match any ground truth node and are mistakenly treated as negative samples. A more detailed analysis of this issue is presented in Section 3.2.

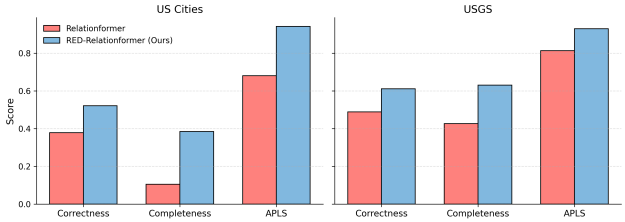


Figure 1. Quantitative comparison between Relationformer and RED-Relationformer. The plot illustrates performance across three graph-level metrics: Correctness, Completeness, and APLS.

Second, the prevailing approach to node localization along line segments relies on an inflexible Dirac delta distribution, which may fail to represent appropriate node positions and can hinder optimization during training.

To address these challenges, we first reproduce Relationformer [13] and evaluate it on its original dataset. Building on this foundation, we incorporate insights from D-FINE [12], D-EIM [7], and General Focal Loss [8] to introduce two key improvements:

- Combine Relationformer with fine-grained distribution refinement to enhance node localization.
- Propose various training strategies and augmentations to improve positive sample matching and accelerate convergence.

2. Related Work

2.1. Transformer in Computer Vision

Transformer [14] was originally developed for natural language processing (NLP) tasks, where it revolution-

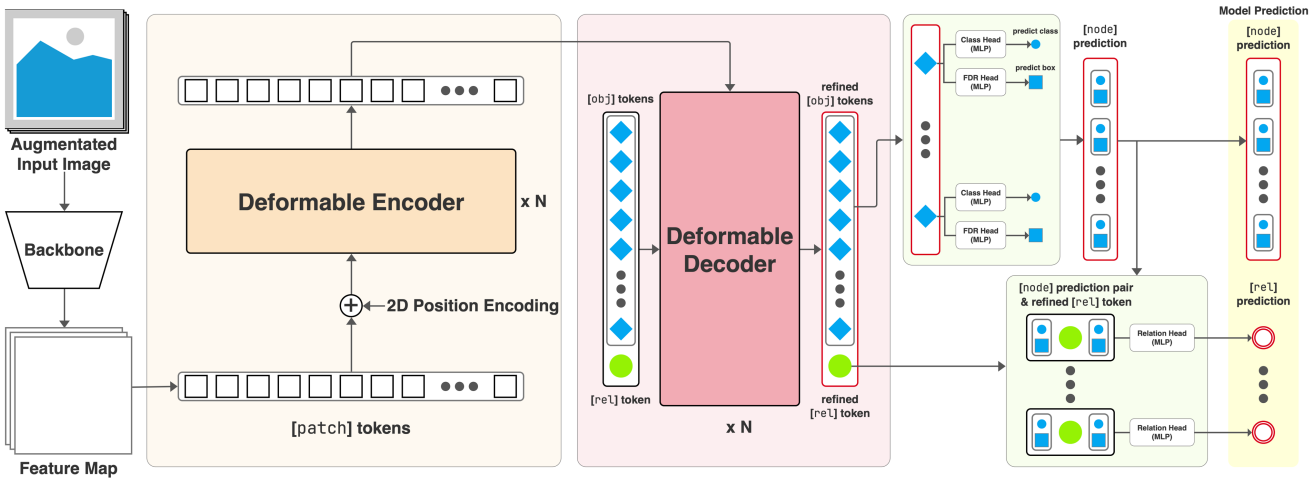


Figure 2. Overview of the proposed RED-Relationformer. The input image is first processed through an augmentation pipeline determined by different training strategies. It is then passed through a CNN backbone to extract a $H \times W \times D$ feature map, which is flattened into $H \times W$ patch tokens and combined with 2D positional encodings before being fed into a Deformable Encoder. The Decoder takes as input a set of object tokens and a relation token. Through deformable cross-attention, it outputs refined object and relation tokens. Each refined object token is passed to a node prediction module that predicts the class and location of the corresponding node. The predicted nodes and refined relation token are then fed into a relation prediction module. All node pairs are constructed and, together with the relation token, are passed through an MLP to predict whether a relation (i.e., edge) exists between each node pair. The final outputs are the predicted nodes and their relations.

ized the field through its self-attention mechanism and ability to model long-range dependencies. In recent years, Transformer-based architectures have made significant strides in computer vision (CV) as well.

The Detection Transformer (DETR) [2] was the first to successfully introduce Transformers into object detection, leveraging self-attention to directly model set-based outputs and enabling end-to-end prediction without the need for post-processing steps like non-maximum suppression. Following DETR, various vision-specific Transformer models such as Vision Transformer (ViT) [4] and Swin Transformer [9] demonstrated strong performance on classification tasks, highlighting the generalization capability of Transformers across different CV domains.

2.2. Line Detection

Line detection or graph generation methods can generally be categorized into two major approaches: segment-based and graph-based. Segment-based approaches predict, for each pixel, whether it belongs to a specific line category, typically producing a dense segmentation map. In contrast, graph-based approaches aim to predict explicit topological structures by identifying nodes and edges that define line segments in the image.

Compared to segment-based methods, graph-based approaches have recently gained attention due to their ability to better represent the relational structure between line elements, enabling more interpretable and flexible representations.

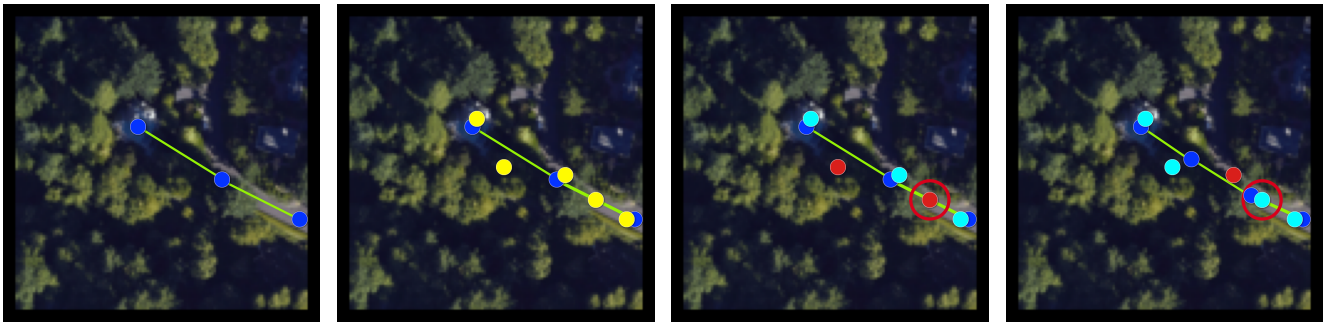
Classical algorithms such as the Hough Transform and Line Segment Detector (LSD) [11] estimate line parameters based on local gradient statistics. However, these methods often suffer from fragmented outputs and are sensitive to noise or occlusions. Recent learning-based methods attempt to overcome these limitations. Sat2Graph [6], for example, encodes road graphs into tensor representations and performs structured prediction. The Wireframe parser [17] utilizes a Hough-space voting mechanism to aggregate local evidence for line structures.

More recently, Relationformer [13] adapts the DETR architecture to formulate line detection as a graph prediction problem. By leveraging a transformer-based design, it directly predicts nodes and their connectivity, offering a fully end-to-end trainable framework for structured line detection.

2.3. Distribution-based Detection

Recent advances in object detection have explored distribution-based representations for bounding box regression, which aim to better capture localization uncertainty and spatial ambiguity. Unlike traditional detectors that directly regress box coordinates using point estimates, distribution-based methods treat each coordinate (e.g., left, top, right, bottom) as a discrete or continuous probability distribution.

General Focal Loss [8] reformulates the regression of the



(a) Ground Truth (b) Model Prediction (c) Matching (Same GT) (d) Matching (Different GT)

Figure 3. From left to right: (1) Ground truth graph; (2) model prediction, where blue dots represent ground truth nodes and yellow dots represent predicted nodes; (3) matching results, with light blue dots indicating positive samples and red dots indicating negative samples; (4) different ground truth annotations but the same model prediction results in different positive and negative sample assignments.

bounding box as a classification task on a set of discrete bins and introduces Distribution Focal Loss (DFL) to supervise the distribution learning process. This design improved overall detection performance. Similarly, probabilistic variants of YOLO, such as Gaussian YOLO [3], model box coordinates as Gaussian distributions, enabling the network to jointly predict the mean and variance of localization. D-FINE [12] redefines the bounding box regression task by modeling object locations as fine-grained probability distributions. D-EIM [7] further optimizes the convergence speed of D-FINE by adopting a dense one-to-one matching strategy, enhancing the quality of positive sample matching.

3. Methods

In this section, we first analyze the core limitations of Relationformer and formally define the problem in Sections 3.2 . We then propose **RED-Relationformer** (**RED**efined **Relationformer** with Improved Matching) Figure 2, which addresses two main challenges observed in existing models. Inspired by D-FINE [12] and DEIM [7], RED-Relationformer introduces two key components, detailed in Sections 3.3 and 3.4 .

3.1. Original Relationformer

Relationformer [13] is one of the state-of-the-art models for graph generation. It is built upon the Deformable DETR framework [18], which itself extends the DETR (DEtection TRansformer) architecture widely adopted in object detection tasks.

The core idea of Relationformer is to treat node detection as an object detection problem. In addition to predicting individual nodes, the model introduces relation tokens to capture the connectivity between nodes. For each predicted node, the model simultaneously evaluates its relationships with other predicted nodes to determine the existence of edges. As a result, the model produces a set of predicted nodes along with their corresponding edges, forming a complete graph structure.

3.2. Problem Statement

The Relationformer [13] builds on the DETR [2] architecture by adapting the object prediction module to a node prediction module for line detection tasks. Similar to DETR, the model computes loss by matching predicted nodes to ground truth nodes using the Hungarian algorithm, which determines positive and negative samples accordingly.

While this matching strategy is suited for object detection, it presents unique challenges in the context of line detection. As illustrated in Figure 3, (a) shows the original ground truth node positions, (b) shows the model's initial predicted nodes, and (c) shows the matching result during loss computation. In (c), light blue dots indicate positive samples and red points indicate negative samples. Notably, (d) demonstrates that under different ground truth annotations, the same model prediction may result in a different assignment of positive and negative samples.

This issue arises because, in line-based annotations, there can be multiple valid nodes to compose the same graph. Annotators may label nodes at slightly different positions, all of which are semantically correct. Consequently, some predicted nodes that lie correctly on the line may not match any ground truth node and are incorrectly classified as negative sample. This mismatch can hinder model convergence speed and negatively impact prediction performance.

3.3. Improved Matching Strategy

DEIM [7] highlights a key limitation in DETR-based frameworks: the number of input object tokens is typically much larger than the number of ground truth objects in each image. During training, only a small subset of tokens are matched to ground truth objects and treated as positive samples, while the remaining unmatched tokens become neg-

ative samples. Since the training signal is dominated by the positive samples, the large imbalance slows down the learning process, as the majority of tokens contribute little to gradient updates. This results in slower convergence and limited optimization efficiency.

To address this, DEIM proposes a dynamic augmentation strategy that increases the number of ground truth objects per image during training. By applying aggressive augmentations and Mixup in the early stages of training, the number of positive samples increases, accelerating convergence and improving overall performance.

Inspired by this strategy and motivated by our analysis in Section 3.2, where we show that varying ground truth node placements can significantly affect convergence, we propose a new graph-based augmentation technique. Specifically, we randomly insert or delete nodes between existing node pairs while preserving the original graph structure. This simulates multiple plausible annotations for the same image and encourages the model to generalize across different labeling patterns.

In our training schedule, we follow a staged augmentation approach similar to DEIM. During the first 60% of training, we apply all augmentations including Mixup. From 60% to 80%, we disable Mixup but retain the other augmentations. In the final 20% of training, all augmentations are turned off to allow the model to refine its predictions on clean data.

3.4. Fine-grained Distribution Refinement

Distribution-based detection [3, 8, 12] was introduced to address the uncertainty in object localization, especially in scenarios involving occlusion or imprecise annotations. Instead of predicting a single bounding box location—which implicitly assumes a Dirac delta distribution—these methods predict a probability distribution over potential object locations, thereby modeling spatial uncertainty more effectively.

D-FINE [12], built upon the DETR architecture [2], proposes Fine-grained Distribution Refinement (FDR), where each decoder layer predicts a distribution over the offsets of bounding box edges relative to the center point. These outputs are then progressively refined across decoder layers to improve localization accuracy.

Inspired by this approach, we adapt the idea of distribution-based prediction to the node prediction task in Relationformer. As illustrated in Figure 2, the refined object tokens from each decoder layer are passed into a node prediction module. The first decoder layer produces an initial node prediction, which is then propagated to the next decoder layer. Each subsequent decoder refines the node position by predicting offset distributions relative to the initial node using an FDR head. These refined features are fed through the decoder stack, allowing for progressively more accurate node localization. The predictions of the final nodes are generated in the last decoder layer, as shown in Figure 4.

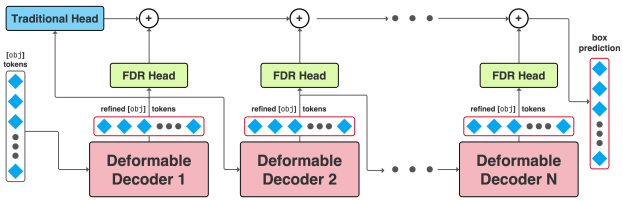


Figure 4. Fine-grained Distribution Refinement (FDR)

4. Experiments

4.1. Datasets

For training and evaluation, we use two datasets: the 20 US Cities dataset [15] and the USGS Historical Topographic Maps.

The 20 US Cities dataset includes high-resolution satellite imagery and annotated road network graphs covering 180 metropolitan regions across the United States. Figure 5a illustrates examples from the dataset. For each region, the dataset provides aligned satellite images, binary road network masks, and graph-based representations of topological structures in both JSON and pickle formats.

The USGS Historical Topographic Maps consist of a large collection of historical maps of the United States, annotated with features such as railroads, waterlines, scarp lines, and thrust fault lines. Figure 5b shows examples of this data. Both datasets enable fine-grained evaluation of line detection and graph reconstruction tasks in real-world scenarios.

(a) 20 US Cities dataset (b) USGS dataset

Figure 5. Examples from the two datasets. Left of each: original map image; Right: corresponding annotation.

Data Preprocessing

For data preprocessing, we follow the same procedure as Relationformer [13], where each large image is divided into smaller multiple overlapping patches. Additionally, the graphs are pruned based on large angular deviations between connected points, as illustrated in Figure 6.

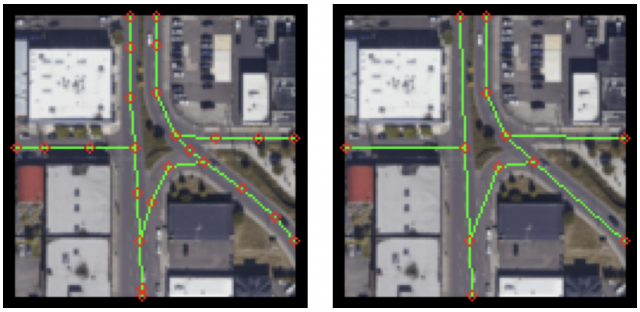


Figure 6. Data pruning example: the left image shows the original annotations, and the right shows the result after removing graph nodes with adjacent edge angles greater than 160 degrees.

4.2. Evaluation Metics

To evaluate the quality of the predicted graph, we used the following three metrics: Correctness, Completeness, and Average Path Length Similarity (APLS) [5]. These metrics respectively measure the precision, coverage, and connectivity of the predicted graph in comparison to the ground truth.

Correctness

Correctness measures the proportion of predicted edges that also exist in the ground truth graph. It evaluates the precision of the edge predictions. A higher correctness indicates that the predicted connections are mostly valid and do not include many false positives.

$$\text{Correctness} = \frac{|\text{Predicted Edges} \cap \text{Ground Truth Edges}|}{|\text{Predicted Edges}|} \tag{1}$$

Completeness

Completeness measures the proportion of ground truth edges that are successfully predicted. It evaluates the recall or coverage of the model in reconstructing the true graph structure. A higher completeness means that most of the important edges in the ground truth have been predicted.

$$\text{Completeness} = \frac{|\text{Predicted Edges} \cap \text{Ground Truth Edges}|}{|\text{Ground Truth Edges}|} \tag{2}$$

Average Path Length Similarity (APLS)

APLS evaluates connectivity similarity between predicted and ground truth graphs by comparing the shortest path lengths between node pairs. An APLS value close to 1 indicates that the predicted graph preserves the overall topological and spatial structure of the ground truth.

$$\text{APLS} = 1 - \frac{1}{|P|} \sum_{(s,t)\in P} \frac{|d_{pred}(s,t) - d_{gt}(s,t)|}{d_{gt}(s,t) + \epsilon} \tag{3}$$

where P is a set of node pairs sampled from the ground truth graph, $d_{pred}(s,t)$, $d_{gt}(s,t)$ are the shortest path lengths between nodes s and t in the predicted and ground truth graphs.

4.3. Ablation Study

Implementation Details

The original paper trains the model for 100 epochs. However, due to the larger dataset and limited GPU resources in our setting, we conduct the ablation study using 50 epochs on the USGS dataset. Based on the results, we identify the best-performing hyperparameter configuration under the 50-epoch setting and use it to train a final model for 100 epochs. This final model is then compared with the original relationformer trained for 100 epochs. All other parameters, including the dimensions and number of encoder and decoder layers, follow the original implementation.

Improving Matching

In this section, we evaluate our proposed improvements to the matching strategy. We compare different augmentation methods, augmentation scheduler, random node insertion, and learning rate settings on the USGS dataset.

- **Augmentation Strategy:** We compare three different augmentation strategies: (1) no augmentation, following the original Relationformer setting; (2) basic augmentations, including horizontal and vertical flips and cutout; and (3) a combination of basic augmentations with Mixup.
  As shown in Table 1, applying augmentation improves both the Completeness and APLS scores. Furthermore, adding Mixup leads to additional gains in Correctness and APLS, achieving the best overall performance. Based on these results, we adopt the combination of basic augmentation and Mixup in our final training setup.

Table 1. Comparison between different strategy

Strategy	Correctness	Completeness	APLS
W/O Aug	0.609	0.612	0.848
+ Aug	0.602	**0.672**	0.863
+ MixUp	**0.639**	0.652	**0.895**

- **Augmentation Scheduler:** Inspired by DEIM, which adopts different augmentation strategies across training stages—strong augmentations in early and mid phases, weaker ones in the later phases, and disabling augmentation during the final epochs—we compare this staged augmentation schedule with two alternatives: applying basic augmentation throughout training, and applying Mixup throughout.
  As shown in Table 3, using Mixup consistently throughout training yields the best performance. Interestingly,

Table 2. Comparison of Line Detection Performance on US Cities and USGS Datasets

Graph-level Metrics				
Datasets	Model	Correctness	Completeness	APLS
US Cities	Relationformer	0.379	0.105	0.681
	RED-Relationformer (Ours)	0.522 (+38%)	0.385 (+367%)	0.942 (+38%)
USGS	Relationformer	0.489	0.427	0.814
	RED-Relationformer (Ours)	0.611 (+25%)	0.631 (+48%)	0.930 (+14%)
Node/Edge-level AP/AR Metrics				
Datasets	Model	Node AP/AR	Edge AP/AR	
US Cities	Relationformer	0.807 / 0.875	0.798 / 0.861	
	RED-Relationformer (Ours)	0.452 / 0.570 (-44% / -35%)	0.296 / 0.473 (-63% / -45%)	
USGS	Relationformer	0.583 / 0.697	0.577 / 0.683	
	RED-Relationformer (Ours)	0.483 / 0.610 (-17% / -12%)	0.400 / 0.595 (-31% / -13%)	

always applying basic augmentations leads to a drop in performance, suggesting that augmentation strength and scheduling can significantly affect model convergence.

Table 3. Comparison between different scheduler

Scheduler	Correctness	Completeness	APLS
Medium Aug	0.639	0.652	0.895
Full Aug	0.545	0.564	0.765
Full MixUp	**0.688**	**0.699**	**0.910**

- **Randomly Add Nodes:** To address the issue discussed in Section 3.2 regarding the variability of ground truth node placement and its impact on training, we introduce an augmentation strategy that randomly adds nodes between existing pairs of nodes. These new nodes are then reconnected to form updated graph.
  This experiment compares the effect of applying this node-adding strategy, with and without an additional pruning step after augmentation. As shown in Table 4, adding nodes improves overall model performance. Moreover, applying pruning after node insertion yields similar results in most metrics, but notably improves the Completeness score. We hypothesize that pruning simplifies the ground truth graph structure, making it easier for the model to align its predictions with the ground truth.

Table 4. Comparison between adding nodes

Add Nodes	Correctness	Completeness	APLS
W/O Adding	0.607	0.612	0.840
+ Adding	**0.639**	0.652	**0.895**
+ Pruning	0.635	**0.673**	0.883

- **Learning Rate:** The original Relationformer adopts a learning rate of 2e-3. However, since we introduce additional augmentations, the training becomes more challenging. To investigate the impact of learning rate under these new settings, we experiment with the original value as well as both higher and lower alternatives. As shown in Table 5, a moderate learning rate of 2e-4 achieves the best APLS score.

Table 5. Comparison between different learning rate

Learning Rate	Correctness	Completeness	APLS
1e-3 (Large)	0.287	0.239	0.874
2e-4 (Origin)	0.639	0.652	**0.895**
1e-4 (Small)	**0.644**	**0.656**	0.859

4.4. Comparison with Relationformer

Based on the findings in Section 4.3, we configure our RED-Relationformer using the best-performing hyper-parameters and training strategies. We compare its performance with the original Relationformer, trained under its default settings, on both the US Cities and USGS datasets for 100 epochs.

As shown in Table 2, RED-Relationformer achieves consistently better performance across all three graph evaluation metrics. On the US Cities dataset, our model outperforms Relationformer significantly, with particularly notable gains in the Completeness score—showing up to a threefold improvement. Similarly, on the USGS dataset, RED-Relationformer yields higher scores across all metrics, with a substantial increase in Completeness as well.

Figure 7 presents qualitative results on samples from the US Cities and USGS datasets. The top row shows pre-

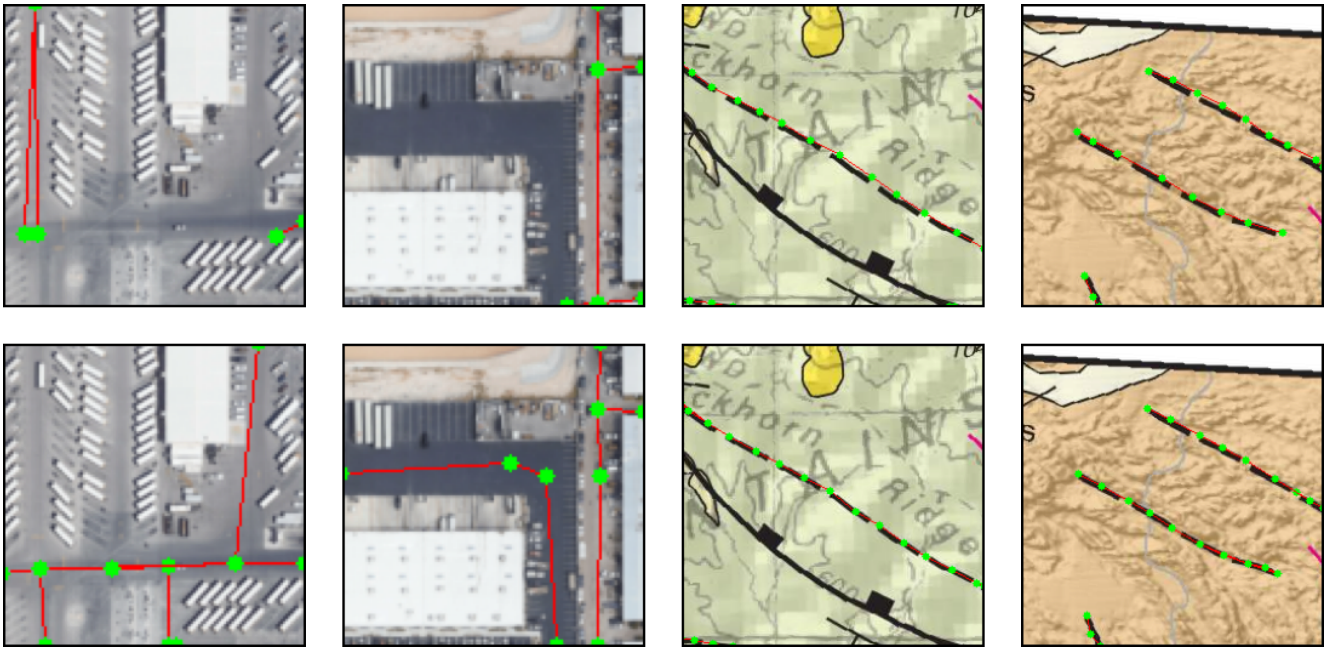


Figure 7. Qualitative comparison of prediction results. The first two columns show samples from the US Cities dataset, and the last two columns are from the USGS dataset. The top row presents predictions from the original Relationformer, while the bottom row shows results from our RED-Relationformer. RED-Relationformer captures more complete and accurate line structures, especially in challenging cases where Relationformer fails.

dictions from the original Relationformer, while the bottom row shows outputs from RED-Relationformer. As illustrated, our model produces predictions that more closely align with the underlying line structures. Moreover, in cases where the original Relationformer fails to capture the correct graph topology, RED-Relationformer is able to successfully reconstruct the line-based graph structure.

However, we observed that when applying the Improving Matching strategy during training, our RED-Relationformer tends to produce a larger number of predicted nodes. As a result, the model generates more complex graph structures during inference. While this leads to significant improvements in Correctness, Completeness, and APLS scores, we also noticed that the resulting graphs are often overly dense and noisy compared to those produced by the original Relationformer.

Although these dense predictions still capture the overall structure of the target graph, we argue that such results are not necessarily better in terms of practical utility or clarity. Importantly, this issue is not fully reflected in the standard evaluation metrics mentioned above. To better assess the quality of the predicted graphs, we treat nodes and edges as detection targets (similar to bounding boxes) and evaluate them using standard object detection metrics—Average Precision (AP) and Average Recall (AR). As shown in Table 2, RED-Relationformer exhibits lower AP and AR scores for both nodes and edges across the two datasets. The decline is particularly notable in precision, which aligns with our observation that RED-Relationformer tends to over-predict, capturing most of the ground truth but also introducing a higher number of false positives.

5. Conclusion

Line detection and graph generation play a vital role in various computer vision applications. While existing methods such as Sat2Graph and Relationformer have demonstrated promising performance, we focused on addressing the limitations of Relationformer. Inspired by D-FINE and DEIM, we proposed RED-Relationformer, which aims to improve training convergence and prediction accuracy by enhancing the matching strategy and incorporating distribution-based refinement.

Our experimental results show that RED-Relationformer achieves significantly better Correctness, Completeness, and APLS scores compared to the original Relationformer. However, we also identified new challenges introduced by our model—particularly, the tendency to produce overly complex graph predictions that are not fully captured by current evaluation metrics. These findings suggest that future research should consider the development of new loss functions and evaluation protocols that better reflect the structural quality of predicted graphs, enabling more accurate assessment of graph generation models.

References

[1] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. RoadTracer: Automatic Extraction of Road Networks from Aerial Images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4720–4728. 1

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. 2, 3, 4

[3] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving, 2019. 3, 4

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 2

[5] Adam Van Etten, Dave Lindenbaum, and Todd M. Bacastow. Spacenet: A remote sensing dataset and challenge series, 2019. 5

[6] Songtao He, Favyen Bastani, Satvat Jagwani, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Mohamed M. Elshrif, Samuel Madden, and Mohammad Amin Sadeghi. Sat2Graph: Road Graph Extraction Through Graph-Tensor Encoding. In *Computer Vision – ECCV 2020*, pages 51–67. Springer International Publishing. 2

[7] Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. DEIM: DETR with improved matching for fast convergence. 1, 3

[8] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. 1, 2, 4

[9] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 2

[10] Roman Naeem, David Hagerman, Lennart Svensson, and Fredrik Kahl. Trexplorer: Recurrent DETR for Topologically Correct Tree Centerline Tracking. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*, pages 744–754. Springer Nature Switzerland. 1

[11] Rémi Pautrat, Daniel Barath, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. DeepLSD: Line segment detection and refinement with deep image gradients. 2

[12] Yansong Peng, Hebei Li, Peixi Wu, Yueyi Zhang, Xiaoyan Sun, and Feng Wu. D-FINE: Redefine regression task in DETRs as fine-grained distribution refinement. 1, 3, 4

[13] Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Sahand Sharifzadeh, Georgios Kaissis, Volker Tresp, and Bjoern Menze. Relationformer: A unified framework for image-to-graph generation. In *Computer Vision – ECCV 2022*, pages 422–439. Springer Nature Switzerland. 1, 2, 3, 4

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1

[15] Xiaotong Xu, Zhenjie Zheng, Zijian Hu, Kairui Feng, and Wei Ma. A unified dataset for the city-scale traffic assignment model in 20 u.s. cities. 11(1):325. Publisher: Nature Publishing Group. 4

[16] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection, 2024. 1

[17] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. 2

[18] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2021. 1, 3