Evaluating large language-vision models on geographic language understanding

Anonymous COLING 2025 submission

Abstract

Geographic language understanding (GLU) tasks ask models to map from text to maps. Geographical complex description parsing (GCDP) is a GLU task where models must assign a set of map coordinates to an unnamed location described by text such as "*. . . between the towns of Adrano and S. Maria di Licodia, 32 kilometres northwest of Catania*". In GCDP, the input is both the text describing the unnamed location and the geometries of the other locations named in the description (e.g., the geometries of *Adrano*, *S. Maria di Licodia*, *Catania*), and the output is the geometry of the unnamed location. In this paper, we convert a GCDP corpus into an `image+text→image` benchmark to evaluate recent large language-vision models on this complex task. The models show weak performance, with analysis showing a lack of understanding of even simpler tasks like recognizing regions by color.

1 Introduction

The goal of geographic language understanding (GLU) is to develop models that can map from descriptions of locations in text to the corresponding locations on a map. A commonly studied GLU task is geoparsing, which asks models to map mentions of locations in text to their geographical geometries, typically by linking mentions to entries in a toponym database like GeoNames[1] (Gritta et al., 2018; Zhang and Bethard, 2023). A more complex GLU task is geographical complex description parsing (GCDP; Laparra and Bethard, 2020), where the input is a description of a geographical region and a list of reference geometries (sets of coordinates), and the goal is to predict the geometry of the region described. For example, the text *"a town and comune in the Metropolitan City of Catania, Sicily, southern Italy. . . located between the towns of Adrano and S.Maria di Licodia, 32 kilometres (20 mi) northwest of Catania."* describes a location that is not explicitly named. The goal is to approximate the geometry of the location using the description as input along with the geometries of the reference locations: *Catania*, *Sicily*, *Italy*, *Adrano*, etc. It is typically impossible to predict the precise target geometry given only the input text and reference geometries, but an approximately correct geometry is enough for many applications.

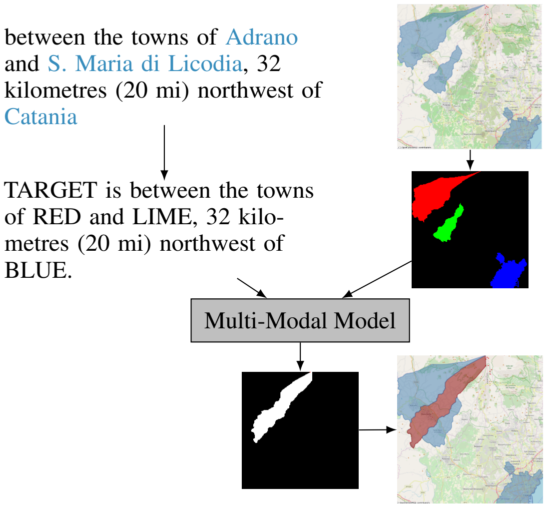


Figure 1: Framing geographic geometry prediction as a multimodal `image+text→image` problem. The toponyms in the left text and geometries in the right figure of the first row are translated into color words in text and a corresponding figure of colored geometries on a black background in the second row. A model should take the modified text and the image with colored regions as input, and generate an image with a white region (third row). That image can then be translated into the corresponding target geometry.

Laparra and Bethard (2020) propose a grammar-based baseline that achieves 22.1 F1 on this task by parsing the descriptions into spatial operators (functions) whose composition yields the target geometry. We estimate that humans can achieve about 35 F1 on this task (see appendix A), thus there is room for improvement with machine learning methods. However, a major challenge for machine learning methods is that the reference geometries, be they polygons or linestrings, are represented by

[1]http://www.geonames.org

a variable number of coordinates, ranging from just a few to over a million. Although methods exist to convert geometries to machine-learned embeddings (Mai et al., 2022), to date there is no clear way to get language models to output geometries.

We consider an alternative to predicting coordinate sets: convert geometries to bitmap images and apply multi-modal language-vision models (LVM), as shown in Figure 1. Our contributions are [2]:

- We propose a strategy to convert GCDP into an `image+text→image` problem and evaluate two LVMs designed to work in this setting.
- Due to the high difficulty of the task, we develop 4 variants of the dataset, each designed to be simpler than the GCDP task and individually analyze a different required skill to solve the task.
- We find that although the current models show some ability to solve the task, their failures stem from lack of understanding of simpler tasks like recognizing regions by color.

2 Related Work

Traditionally, geographic language understanding has focused on identifying geographical entities following a named entity recognition approach (Karagoz et al., 2016; Magge et al., 2018) and linking such named entities to a reference knowledge base such GeoNames (Karagoz et al., 2016; Magge et al., 2018; Zhang and Bethard, 2023; Zhang et al., 2024). To broaden the scope of research beyond geolocations explicitly named in the text, Laparra and Bethard (2020) proposed the GCDP task where an unnamed geolocation is linked to its geometry based on a description of the unnamed geolocation and the geometries of nearby named geolocations.

Many recent language-vision models target tasks without an image output: text-image reasoning, text-image classification, visual question answering, etc. (LXMERT, Tan and Bansal 2019; CLIP, Radford et al. 2021; LLaVA, Liu et al. 2023; Qwen-VL, Bai et al. 2023; etc.). **These models are not ideal for an `image+text→image` task** as they lack a decoder that can decode an output image from the latent-space representation of the input text and image. We choose CLIP as a baseline in this paper because of its popularity and success on several multi-modal tasks, and train a decoder that can decode the CLIP latent space into an image.

A few works take image and text as input and directly output images or masks. Instruct-Pix2Pix (Brooks et al., 2023) is a diffusion-based model that can generate an edit of the original input image based on text instructions. This model is not good at isolating specified objects, making it difficult to locate reference geometries in our task. LISA (Lai et al., 2023), built from the popular LLaVA (Liu et al., 2023) LVM, is LoRA-finetuned (Hu et al., 2022) to generate segmentation masks given an image and a text description of the target. The downstream task of LISA is most similar to our task, so we choose it as another baseline.

3 Datasets

We use the Laparra and Bethard (2020) GCDP corpus, derived semi-automatically from Wikipedia and OpenStreetMap, that contains 360,187 uncurated training examples and 1,000 manually curated test examples. We select 67,293 and 1,000 examples for training and development, respectively, where all the reference locations in the description have associated geometries (see Appendix B). We use the same test set as Laparra and Bethard (2020).

In the following sections, we first introduce our translation of the GCDP problem into an `image+text→image` problem, then introduce our dataset variants, shown in Figure 2, that allow the study of different capabilities of LVMs. See Appendix B for more dataset generation details.

3.1 IMAGE

Obtaining an image-based dataset from GCDP data requires decisions of which part of the world map to show in the image and how to show links between toponyms in the text and geometries in the image.

Decide boundary: To create an image, we must first select a small region of the map, as using the entire map would result in most locations being smaller than a single pixel. A good region for GCDP should completely include the target geometry, represent such geometry with a sufficient number of pixels and include at least a portion of every reference geometry. However, the target geometry is not known at prediction time and thus should not be used when selecting the boundary. We thus use a heuristic: set the boundary to 100 km in each cardinal direction from the geometric median of the centroids of the reference geometries.

Link reference geometries and toponyms: We create a pixel grid with $N \times N$ pixels representing the selected boundary. For the input image,

[2] Code and data will be available.

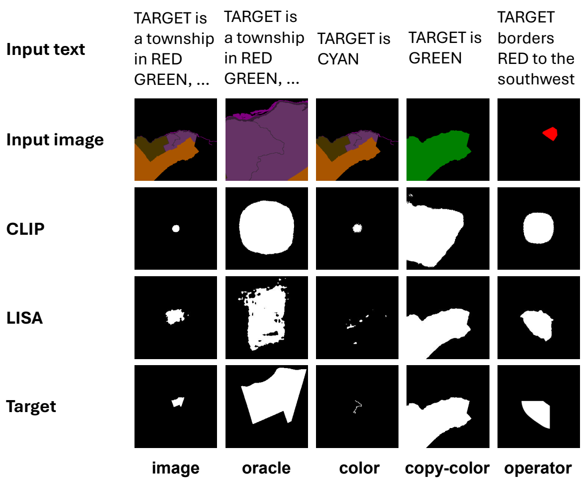


Figure 2: One example (column) from each of the datasets: IMAGE, ORACLE, COLOR, COPY-COLOR and OPERATOR. The last row shows the output target geometry, and the first two rows show the input text and input images containing the reference geometries. A color mentioned in the input text refers to the region in the input image with such color. When two regions overlap the image shows the composition of their colors. The CLIP and LISA rows show the predictions of the respective models as white regions.

we overlay the grid with the reference geometries, assigning a different color to each, calculating the average of the colors in RGB space when geometries overlap. For the output image, we overlay the grid with only the target geometry in white. To link the reference geometries in the image with the reference toponyms in the text, we replace each toponym in the text with the color name of corresponding geometry. The middle of Figure 1 visualizes this and the preceding step.

3.2 Image Oracle Boundary (ORACLE)

The method to decide the boundary in section 3.1 may result in wide boundaries where the target geometry is represented with just a few pixels. To better understand how this size affects LVMs, we develop an oracle version of the image-based dataset where we use the target geometry (hence an oracle) to select a narrower boundary. We start from the envelope covering the target geometry and extend it until it touches at least one point of all reference geometries. We add 10 kilometers in all 4 cardinal directions to ensure that the images include a portion of all geometries.

3.3 Image Colored Region Identification

For the `image+text→image` approach to work, it is essential that the models are able both to relate textual mentions of colors to those colors in the images, and to differentiate objects of a given color from the other objects in the image. However, this is not possible to analyze in detail in the GCDP dataset, due to the complexity of the task. Therefore, we generate two datasets where the text simply states the color of the target in the input image, e.g., "*TARGET is RED*".

COLOR The input image contains the target geometry and all reference geometries, and the output image is the same as image dataset. This dataset tests whether the models are able to differentiate objects of the color mentioned in the text from objects of other colors.

COPY-COLOR The input image contains only the geometry of the target image, and the output image is the same as the image dataset. This dataset tests whether the models are able to recognize a region in an image based on color.

3.4 Image Single Operator (OPERATOR)

The `image+text→image` approach requires the models to interpret the spatial relationships described in the text and to perform the corresponding calculations to obtain the target geometry. Most input descriptions in the Laparra and Bethard (2020) dataset correspond to calculations involving multiple spatial relationships. To understand whether models can understand the simpler case of a single spatial relation (e.g. "between") in isolation, we generate a synthetic dataset. The descriptions include only a single spatial relation using the grammar defined by Laparra and Bethard (2020) and their deterministic implementation of the spatial operators. A description in this dataset looks like "*TARGET is 50 km Southwest of RED*". See appendix B.4 for a list of the operators.

4 Metrics

We apply the following **image-based metrics** to our analysis. Let the area of the target region be S_t, the area of the predicted region be S_p, the area of $S_t \cap S_p$ be S_I, we evaluate the performance of the models using the per pixel precision P, recall R, and $F1$ score.

$$P = \frac{S_I}{S_t} \qquad R = \frac{S_I}{S_p} \qquad F1 = \frac{2PR}{P+R}$$

We include results using the **polygon-based metrics** proposed by Laparra and Bethard (2020) calculating the overlap between the predicted and target geometries. For this evaluation, the predicted images must be translated back to a set of coordinates.

	RANDOM			CLIP			LISA			GRAMMAR		
dataset	P	R	F1	P	R	F1	P	R	F1	P	R	F1
IMAGE	1.3	50	2.5	12.7	34.1	18.5	15.9	31.2	21.1	-	-	-
ORACLE	10.7	50	17.7	21.0	20.4	20.7	35.9	42.6	39.0	-	-	-
COLOR	19.8	50	28.3	38.2	56.1	45.5	56.1	61.5	58.7	-	-	-
COPY COLOR	20.2	50	28.8	43.7	59.4	50.3	73.5	82.1	77.5	-	-	-
OPERATOR	12.4	50	19.9	23.9	26.1	25.0	89.7	89.8	89.8	-	-	-
GCDP	-	-	-	7.2	27.0	11.3	9.1	38.7	14.7	17.2	31.0	22.1

Table 1: Performance of CLIP and LISA models on the five image-based datasets and on GCDP. The results for IMAGE, ORACLE, COLOR COPY COLOR and OPERATOR are calculated using the **image-based metrics**. The models are compared on these 5 datasets with the random BASELINE. The results for GCDP are calculated using the **polygon-based metrics** and the models are compared with GRAMMAR based model of Laparra and Bethard (2020).

5 Baselines

As explained in Section 2, we test two different LVMs: CLIP and LISA. See Appendix C for why GPT-4o fails on this task; we do not include it in the tested LVMs. See Appendix D for model implementation details. We compare these models on the polygon-based evaluation with the grammar-based model by Laparra and Bethard (2020) and on the image-based evaluation with a random baseline.

Random We report the theoretical results of a random baseline which has a 50% of chance to predict each pixel as part of the target.

CLIP (Radford et al., 2021) To predict the target region, we use CLIP as an encoder to extract text and image features and feed the fused features to a decoder (Appendix D) to generate the target region.

LISA (Lai et al., 2023) We finetune the model as in the LISA paper: the text decoder is trained to generate a special `<SEG>` token and the image decoder is trained to generate a segmentation mask (target region) from the special tokens.

6 Results and Discussion

The last row of Table 1 shows that on GCDP in terms of F1, the grammar-based baseline of Laparra and Bethard (2020) outperforms both CLIP and LISA, though both of those models outperform the random baseline. Performance on the remaining rows, the image-based datasets, provides some insight into why these models have difficulties solving GCDP. We observe that:

The size of the target regions significantly influences the model performance. F1 score of the CLIP model increases 2.2 points and the LISA performance nearly doubles when using the ORACLE boundaries where the target regions are a larger portion of the image. Both models also perform better in the COPY COLOR, COLOR, and OPERATOR datasets where the target region area is also generally larger. This suggests that in real-world settings where an oracle boundary is not available, finding a good boundary is key for model performance.

Segmentation-based pre-training helps to understand spatial relations in text. There is a huge gap between CLIP and LISA on the OPERATOR dataset. LISA is better able to understand spatial relations in text and reason over the image accordingly. This may also explain LISA's better performance on IMAGE and ORACLE datasets.

Segmentation-based pre-training helps to capture the shape of the target regions better. As shown in the third row of Figure 2, CLIP generates mostly circle-like shapes in the middle of the image. While this guarantees some recall of the prediction, the overall precision of CLIP is low. LISA captures the shape of geometries better. LISA can get a near-perfect target shape in the COPY COLOR dataset and a very close guess when predicting the result on OPERATOR. This is also verified by the high performance of LISA on these two datasets.

Colors are more difficult to understand than shapes. Understanding colors is crucial for the model to capture the relationship between the input text and image. The task not only requires the model to relate the color words to colors in the image, but also requires the model to understand how different colors mix when there are overlapping regions. Our results show that when this kind of color understanding is required, models tends to perform poorly. This is indicated by the low performance of both models on the COLOR, ORACLE, and IMAGE datasets. This suggests that more work is needed to infuse color knowledge into LVMs, and that it may be worth exploring ways of representing geometries in images that do not rely on color.

Limitations

While the paper gives insights into the potential and limitations of two multi-modal language-vision models for GCDP, it is unable to evaluate the full range of multi-modal language-vision models, as most are not designed with the necessary inputs and outputs for `image+text→image` tasks. Further investigation is required into algorithms that can take language-vision models that were not designed for `image+text→image` tasks and alter their inputs and outputs so that they can be used in them.

Intended Use and Ethical Concerns

The data and models we developed in this paper is intended to be used on GCDP tasks. We do not foresee immediate ethical concerns of our work. However, we acknowledge that as we use LVMs in our experiments, the models may generate unexpected images if not properly used by an user or not used on this task.

References

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18392–18402.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2018. Whatś missing in geographical parsing? *Language Resources and Evaluation*, 52(2):603–623.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Pinar Karagoz, Halit Oguztuzun, Ruket Cakici, Ozer Ozdikis, Kezban Dilek Onal, and Meryem Sagcan. 2016. Extracting Location Information from Crowd-sourced Social Network Data. In Cristina Capineri, Muki Haklay, Haosheng Huang, Vyron Antoniou, Juhani Kettunen, Frank Ostermann, and Ross Purves, editors, *European Handbook of Crowdsourced Geographic Information*, pages 195–204. Ubiquity Press.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. 2023. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*.

Egoitz Laparra and Steven Bethard. 2020. A dataset and evaluation framework for complex geographical description parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 936–948, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc.

Arjun Magge, Davy Weissenbacher, Abeed Sarker, Matthew Scotch, and Graciela Gonzalez-Hernandez. 2018. Deep neural networks and distant supervision for geographic location mention extraction. *Bioinformatics (Oxford, England)*, 34(13):i565–i573.

Gengchen Mai, Krzysztof Janowicz, Yingjie Hu, Song Gao, Bo Yan, Rui Zhu, Ling Cai, and Ni Lao. 2022. A review of location encoding for geoai: methods and applications. *International Journal of Geographical Information Science*, 36(4):639–673.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Zeyu Zhang and Steven Bethard. 2023. Improving toponym resolution with better candidate generation, transformer-based reranking, and two-stage resolution. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 48–60, Toronto, Canada. Association for Computational Linguistics.

Zeyu Zhang, Egoitz Laparra, and Steven Bethard. 2024. Improving toponym resolution by predicting attributes to constrain geographical ontology entries. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 35–44, Mexico City, Mexico. Association for Computational Linguistics.

A Human Performance on GCDP

Laparra and Bethard (2020) never evaluated humans on the geographical complex description parsing (GCDP) task, so to get a rough approximation of the difficulty of the task, one author randomly sampled 10 geographical descriptions and manually annotated polygons. The annotator followed the guideline that the predicted polygon should be based solely on the text and the polygons for the reference locations; no external knowledge about the shape of geographical regions was to be used. For each description, the annotator wrote Python code that took the OpenStreetMap ids of the reference polygons as input, and used the shapely library to generate an output polygon. Due to the complexity of translating the text description into appropriate shapely calls, a single description often required more than 10 minutes of writing code.

Comparing the human-annotated examples to the target polygons in the dataset, average precision was 22.0 and average recall was 74.4, for an average F1 of 34.0. Recall was higher than precision because most descriptions are not specific enough to pinpoint the exact location described. For example, the target location *Gylen Castle* is described as *a ruined castle, or tower house, at the south end of the island of Kerrera*, but the castle is tiny in comparison to the southern part of the island. Comparing the human performance to the model performance in Table 1, where automatic approaches achieve at most 17.2 precision, 38.7 recall, and 22.1 F1, there is still substantial room for improvement between the best models and human performance.

B Dataset Generation

The corpus by Laparra and Bethard (2020) contains 360,187 uncurated examples and a test set of 1,000 manually curated examples. In our experiments, we use the same test set. For training, we use the uncurated portion of the corpus, however this portion does not guarantee that all the locations mentioned in the descriptions have a mapping to their corresponding geometry. We run named-entity recognition on the uncurated examples to obtain all the location mentions, and check if the recognized locations are linked to a geometry. We keep only those descriptions that have all the recognized locations linked. As a result, we obtain 68,293 examples from which we use 67,293 as training set and 1,000 as development set.

All images in our image-based datasets have the size of 224×224. This is the default input size of the CLIP model. LISA does not have constraint on the input image size.

B.1 Image-based Dataset

Below we detail the steps we follow for the conversion of this dataset:

Generate boundary: From the reference geometries, we first discard the geometry with the largest area to avoid boundaries that are too wide for the target geometry. The geometry is discarded only to calculate the boundary but it is included in the resulting image. Then, we calculate the geometric median (which is more robust to outliers than the centroid) of the centroids of the remaining reference geometries. We set the boundary to 100 km in each cardinal direction from this centroid and obtain the coordinates of each of the 4 corners of this boundary

This process ensures that all boundaries are 200 km $\times$ 200 km, guaranteeing that the distances mentioned in the text will have the same relation to the distances in the images regardless of the size of the reference (or target) geometries. For example, if a description mentions a distance of "100 km" between two locations and it is represented as 40 pixels in the resulting image, a mention of "50 km" in another description will be represented with 20 pixels in the corresponding image.

Generate images: Once we calculate the boundary for the images, we apply postgis' *st_asraster*[3] function to obtain a bitmap representation of the geometries. The function creates a pixel grid with $N \times N$ pixels where each corner corresponds to each coordinate of the boundary. The function overlays this grid on a geometry and calculates if each pixel intersects with the geometry, assigning 1 if true and 0 otherwise.

After obtaining a bitmap image (raster) for each geometry, each pixel grid is translated into a RGB format, using a different color for each geometry. Then, all these RGB pixel grids are joined in a single image. Where two or more colors overlap in one pixel, we calculate the average. E.g. for a pixel where (255, 0, 0) and (0, 255, 0) overlap, we assign (128, 128, 0) in the final image. In the case of the target image, only one geometry will be part

[3] `https://postgis.net/docs/RT_ST_AsRaster.html`

of it and we use the white color (255, 255, 255) to represent it.

Update the descriptions: The last step consists of replacing all the mentions of each location in the description with the name of the color used in the previous step to represent the corresponding geometry. For example, if the location *Adrano* is assigned the color (255, 0, 0), then all instances of *Adrano* in the text will be replaced with the string "RED".

B.2 Image Oracle Boundary (ORACLE

To find the oracle boundary, we start initially from the envelope covering the target geometry and extend it until it touches at least one point of all reference geometries. Finally, we extend the boundary 10 kilometers in all 4 cardinal directions to ensure that the images include a portion of all geometries. Once the boundary is obtained, the reference and target images are generated as described in 3.1.

Unlike the dataset described in 3.1 where all images correspond to the same spatial extent, the boundary of the images in this version may cover different extents in each case, which does not guarantee the correspondence between the distance units described in the text and the distance in pixels of the images. To solve this problem, we automatically modify the spatial units mentioned in the text by scaling them appropriately. First, for each case, we calculate the ratio between the number of pixels of the width of the images and the width in kilometers of the boundary. Then we identify by a simple regular expression all mentions of distance units, e.g. "100 KM", extract the quantity and multiply it by the ratio calculated in the previous step. Finally, we modify the text with the result of this calculation rounded to the nearest integer.

B.3 Image Colored Region Identification

To generate each example of the COLOR and COPY-COLOR datasets, we randomly select one of the reference geometries and obtain a target image containing only that geometry. The description in this case will be simply "TARGET is COLOR", where COLOR corresponds to the color assigned to the selected geometry. Finally, in each of the datasets, we follow a different strategy to generate the reference image:

COLOR The reference image contains all reference geometries including the one selected for the target image.

COPY-COLOR The reference image contains only the geometry selected for the target image.

B.4 Image Single Operator (OPERATOR)

The GCDP paper defined 5 spatial relations of geometries and implemented corresponding polygon operators. We refer the reader to their paper for the complete details, but briefly:

- `Between(r1, r2)`: The target is between reference location `r1` and `r2`.
- `Intersection(r1, r2, ...)`: The target region is the intersection of all reference locations `r1`, `r2`, ... in the arguments.
- `Union(r1, r2, ...)`: The target region is the union of reference locations `r1`, `r2`, ... in the arguments.
- `Adjacent(r1)`: The target location shares some part of the border with the reference location `r1`.
- `Distance(r1, D, u, geocardinal)`: The distance between the target and the reference `r1` is `D`. `u` is the unit of the distance (e.g. kilometer, mile). `geocardinal` (e.g. "north of") is optional and when given it refers to the direction from the reference `r1` to the target.

We use these operators to generate a single-operator understanding dataset. Each example in this dataset is generated following the steps:

1. One of the Laparra and Bethard (2020) operators is randomly selected and the values of the arguments are also randomly selected. For example, if the operator takes a geocardinal as argument, its values are selected among the possible values None, North, Northeast, and so on.
2. Select a pattern defined in the grammar for the operator selected in the previous step and complete it with the selected values. For example, a possible pattern could be "TARGET is [Distance] [Unit] [Cardinal] of [Reference]" which could be completed as "TARGET is 50 km Southwest of REFERENCE".
3. Generate the necessary reference geometries randomly. For instance, we generate a geometry for the REFERENCE in the previous example.
4. Apply the corresponding operator with the values of the arguments obtained in step 1 and the references in step 3, to obtain a new geometry that corresponds to the TARGET of the description generated in step 2.

After this process, the reference and target images are generated as explained in Section 3.1, assigning a random color to the reference geometry, white to the target, and updating the description accordingly.

C GPT-4o

We do not consider GPT-4o as a baseline because while GPT-4o allows image and text input, it cannot directly output masks/images, as is required by the GCDP task.

GPT-4o may be able to create images by generating code. However, we find that this method does not work for GCDP. It is difficult even for humans to write code to describe the target region in an image. Figure 3 shows an example of GPT-4o output. The model writes code that turns the reference image into grey-scale, without solving the task.

D Model Implementation and Training Details

D.1 CLIP

We used the largest pretrained CLIP model from OpenAI [4] as the encoder. The original CLIP model limits the maximum input text token length to be 77. To input longer texts in our dataset, we replace the original input projection layer (length of 77) with a 512-long projection layer. We initialize the first 77 elements of this layer with the pretrained CLIP weights and initialize the remaining of this layer randomly.

The code and structure of the CLIP decoder is shown in Figure 4 and Figure 5 respectively. The CLIP text and image encoder each output a 768-length feature vector. We concatenated them as the input of the decoder. The decoder is a stack of transposed convolution layers (also known as deconvolution layers), the details of the dimension of each layer is shown in Figure 4. The decoder is randomly initialized without pretraining.

We use a combination of the DICE loss and per-pixel binary cross-entropy (BCE) loss as the training objective:

$$\mathcal{L}_{\text{CLIP}} = \lambda_1 \mathcal{L}_{\text{DICE}} + \lambda_2 \mathcal{L}_{\text{BCE}}$$

DICE loss is a commonly used loss function in image segmentation tasks, which is defined as:

$$\mathcal{L}_{\text{DICE}} = 1 - \sum_{i,j} \frac{2 p_{ij} y_{ij} + 1}{p_{ij} + y_{ij} + 1}$$

where $p_{ij} \in [0, 1]$ is the i-th row, j-th column element value of the prediction image and $y_{ij} \in$

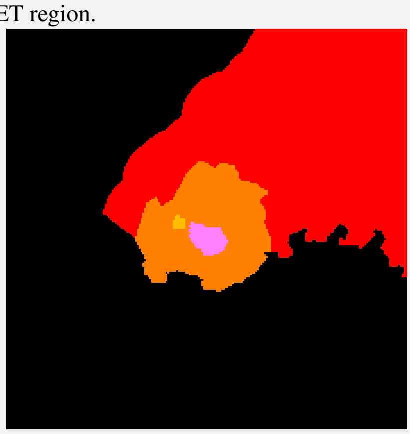
[4] https://huggingface.co/openai/clip-vit-large-patch14

GPT-4o Prompt

Your task is to generate an image of the TARGET region described by the text:

TARGET is a village in the RED province of GREEN. It is a part of the municipality of BLUE, and lies about 19 km east of YELLOW.

The given image represents a 100 km by 100 km region. The colors in the text corresponds to the colored regions in the given image. When regions overlaps, the overlapping region color is the average color of all regions in RGB space. Response with a black-and-white image of the TARGET region.



GPT-4o Response

```
from PIL import Image

# Load the image provided by the user
image_path = "img.png"
image = Image.open(image_path)

# Convert the image to grayscale (black-and-white
) to highlight the TARGET region
grayscale_image = image.convert("L")

# Save the processed image for further use
processed_image_path = "img_grayscale.png"
grayscale_image.save(processed_image_path)
```

Result Image

Figure 3: GPT-4o simply converts the image to grayscale instead of following the instructions for GCDP.

	Strict					Relaxed					Coverage
model	P	R	F_1	P_{x2}	R_{x2}	P	R	F_1	P_{x2}	R_{x2}	%
GRAMMAR	0.172	0.310	0.221	0.272	0.381	0.213	0.276	0.240	0.358	0.365	52.8%
CLIP	0.072	0.270	0.113	0.169	0.456	0.134	0.261	0.177	0.288	0.451	100%
LISA	0.091	0.387	0.147	0.205	0.566	0.155	0.367	0.218	0.315	0.560	87.5%

Table 2: Comparison of LISA and CLIP with the grammar-based baseline proposed by Laparra and Bethard (2020).

CLIP Decoder Structure

```
Sequential(
    (0): ConvTranspose2d(1536, 512,
                        kernel_size=(7, 7),
                        stride=(1, 1),
                        bias=False)
    (1): BatchNorm2d(512, eps=1e-05,
                    momentum=0.1,
                    affine=True,
                    track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(512, 256,
                        kernel_size=(4, 4),
                        stride=(2, 2),
                        padding=(1, 1),
                        bias=False)
    (4): BatchNorm2d(256, eps=1e-05,
                    momentum=0.1,
                    affine=True,
                    track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(256, 128,
                        kernel_size=(4, 4),
                        stride=(2, 2),
                        padding=(1, 1),
                        bias=False)
    (7): BatchNorm2d(128, eps=1e-05,
                    momentum=0.1,
                    affine=True,
                    track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(128, 64,
                        kernel_size=(4, 4),
                        stride=(2, 2),
                        padding=(1, 1),
                        bias=False)
    (10): BatchNorm2d(64, eps=1e-05,
                     momentum=0.1,
                     affine=True,
                     track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 32,
                         kernel_size=(4, 4),
                         stride=(2, 2),
                         padding=(1, 1),
                         bias=False)
    (13): BatchNorm2d(32, eps=1e-05,
                     momentum=0.1,
                     affine=True,
                     track_running_stats=True)
    (14): ReLU(inplace=True)
    (15): ConvTranspose2d(32, 1,
                         kernel_size=(4, 4),
                         stride=(2, 2),
                         padding=(1, 1),
                         bias=False)
)
```

Figure 4: Code of the CLIP decoder

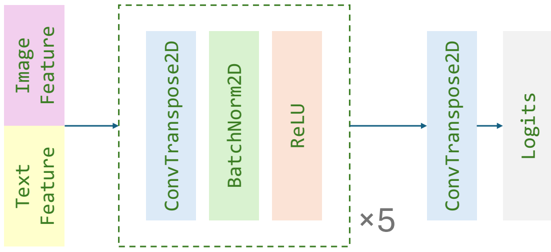


Figure 5: CLIP decoder takes the concatenated text and image feature vectors as input and construct the output image with 2D transposed convolution layers.

$\{0, 1\}$ is the i-th row, j-th column element value of the ground truth image. The DICE loss is useful for unbalanced datasets such as ours where the target region makes up a small proportion of the output image, and we want to prefer predicting the target region to predicting every pixel as negative. In our experiments, λ_1 are λ_2 are set to 1.

During training, we set the learning rate to 0.0002 and finetune the model for 5 epochs. We used one Nvidia A100 GPU for finetuning, and finetuning each task takes about 3 hours.

D.2 LISA

We follow the setting of the original LISA work and used the pretrained LISA-7B-v1 weights offered by the LISA paper authors in our experiments.

As the language encoder generates a sentence which should contain a special token representing the prediction, we follow the original paper and add an extra text generation loss besides the DICE loss and the BCE loss:

$$\mathcal{L}_{\text{LISA}} = \lambda_1 \mathcal{L}_{\text{DICE}} + \lambda_2 \mathcal{L}_{\text{BCE}} + \lambda_3 \mathcal{L}_{\text{txt}}$$

$\mathcal{L}_{\text{txt}}$ is the cross-entropy loss between the language model predicted word and the teacher-forcing label. In our experiments, λ_1, λ_2 are λ_3 are set to 1.

We set the learning rate to 0.00003 and finetune the model for 1 epoch. We used 4 Nvidia A100 GPUs for finetuning, and finetuning each task takes about 13 hours.

E GCDP Evaluation

Table 2 shows the performance of CLIP and LISA on GCDP using all the metrics proposed by Laparra and Bethard (2020), not just the strict metric as reported in the main text. These metrics are run with two different criteria. The *strict* version calculates the exact overlap between the predicted and the target geometries. In the *relaxed* version, the metric calculates the overlap between the predicted geometry and the oriented envelope of the target geometry, i.e. the minimum rectangle that encloses the geometry. In addition, the P_{x2} column shows the results of precision when the target geometry is scaled by a factor of 2. Similarly, the R_{x2} column shows the results of recall when the predicted geometry is scaled by a factor of 2. Laparra and Bethard (2020) proposed these alternatives to give some credit to predictions that are close to the target geometries but do not overlap. Finally, The *coverage* column shows the percentage of cases where the model is able to predict a geometry.