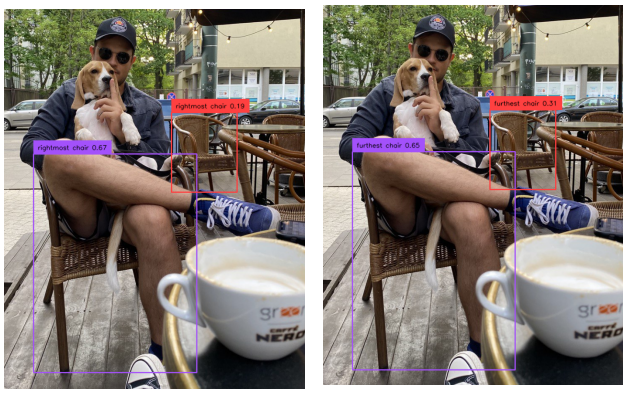
GDINOSAUR: **G**rounding **DINO** with **S**patial **A**wareness for **U**nderstanding **REC**

Je-Ching Liao, Billy Mazotti, Jake Sansom, Aryan Sharma, Aditya Vasudevan
{jeching, bmazotti, jhsansom, aryanraj, adivasu}@umich.edu

***Abstract*—Referring Expression Comprehension (REC) Object Detection (OD) is the task of identifying image regions corresponding to natural language captions such as "the red shoe". Current REC OD systems such as Grounding DINO [1] struggle with referring expressions that emphasize *spatial* understanding (e.g., "the rightmost object"). To address this shortcoming, we first curate spatial and non-spatial 2D datasets emphasizing spatial and non-spatial referring expressions respectively derived from the RefCOCO/+/g datasets [2], [3]. We then generate RefCOCO-3DS, a custom synthetic 3D dataset emphasizing spatial 3D referring expressions. Finally, we explore several methods for fine-tuning Grounding DINO using our curated datasets and employing Elastic Weight Consolidation (EWC) to improve its spatial understanding in a continual learning manner. As a result, the curated datasets improved model performance on both spatial and non-spatial portions of the RefCOCO dataset. Paired with EWC, the model sees an increase of 15% in mean average precision. Our code for training and evaluating GDINOSAUR is available at https://github.com/jhsansom/OpenGroundingDino and our code for generating RefCOCO-3DS is available at https://github.com/BillyMazotti/RefCOCO-3DS.**



(a) Rightmost Chair (b) Furthest Chair

Fig. 1: Given left and right images and the prompts "rightmost chair" and"furthest chair", Grounding DINO wrongly detects the leftmost chair and the nearest chair with the highest scores respectively.

Fig. 2: GPT-4 failing to understand the query and while also producing a factually incorrect response

I. Introduction

Many machine learning models have been developed to align text and images together. *Open-vocabulary object detection* models are a subset of object detection machine learning models that aim to detect objects in an image corresponding to an object mentioned in a given text prompt regardless of whether the model had been trained to detect that specific object. Some of these models are trained for REC, a type of open-vocabulary object detection that uses unstructured natural language to describe the object or region of interest in an image. For example, in Fig. 1, "rightmost chair" and "furthest chair" are referring expressions. While Grounding DINO [1], a deep learning model trained for REC, identifies several of the chairs in the image, it fails to successfully locate the correct chair corresponding to the spatial cues of "rightmost" and "furthest", giving highest predictive scores to the "leftmost" and "nearest" chair. This deficiency is noticed in other state-of-the-art models such GPT-4 as seen in Fig. 2 where the model correctly recognizes the "black king" while struggling to recognize the spatial relationship between it and its neighboring pieces.

REC requires *compositional* understanding, the task of composing multiple attributes such as "the glass" and "furthest to the right". The capabilities of connectionist (and consequently deep learning) systems on tasks that require compositional generalization are subjects of longstanding debate [4]. For instance, it may be possible to build a deep learning system that easily detects shoes or to build a system that easily detects red objects, however designing such a system to identify novel compositions (e.g., "red shoes") in a zero-shot manner is challenging. In particular, existing systems seem to struggle with compositionality over *spatial features*, as demonstrated in Fig. 1 and Fig. 2.

In this project, we aim to reinforce Grounding DINO [1]

and improve spatial expression performance when detecting objects through our model GDINOSAUR (**G**rounding **DINO** with **S**patial **A**wareness for **U**nderstanding **R**EC).

Solving the problem of compositional generalization for spatial features is particularly interesting due to possible applications in robotics. For instance, when communicating with a human interlocutor, a robot may need to understand phrases such as "the rightmost chair" or "the plate at the bottom". To act on this information, such a robot would need to integrate both linguistic information (from the interlocutor's referring expression) and visual information (from its own perceptual devices). Thus, by solving the problem of compositional generalization over spatial features (or at least by making progress toward a solution), we hope to make progress toward the ultimate goal of developing useful and capable robots.

Additionally, compositional generalization over spatial features could play a major role in the technology that assists visually impaired people. For instance, in response to a query such as "Are there napkins nearby?", a useful model would understand the location of nearby napkins relative to other objects and to the user. It would then be able to direct the user to find the napkins. Thus, we believe this project could make progress toward the long-term goal of developing assistance technology for the visually impaired.

II. Methodology

Our primary aim is to improve the spatial REC of Grounding DINO. We achieve this goal by (1) curating and generating datasets that emphasize spatial understanding and (2) fine-tuning a pre-trained Grounding DINO model on this dataset using elastic weight consolidation. For all fine-tuning and theoretical methods explored, we use the Grounding DINO model, Swin-B_cogcoor, as our baseline model. This model utilizes the baseline variant of the Swin Transformer [5] as the vision image feature backbone and has been pre-trained on the following datasets: COCO [6], O365 [7], GoldG [1], Cap4M [1], OpenImage [8], ODinW-35 [9], and RefCOCO [2]. We chose Swin-B_cogcoor as our baseline model for experimentation in order to enhance to spatial REC performance learned during the RefCOCO pre-training.

A. 2D Dataset Generation

Ultimately, since we are interested in improving spatial understanding in realistic scenarios, having a dataset that emphasizes these scenarios is imperative. To create our realistic 2D dataset, we identify a subset of the RefCOCO/+/g datasets entirely composed of images and associated captions that utilize spatial language. RefCOCO/+/g is a compilation of three different datasets containing referring expressions (e.g., "the man in the red shirt" or "the bowl on the right") [2]. We believe this will be a good dataset to build upon because (1) it is composed of real images (thus, it is "realistic") and (2) the authors of Grounding DINO trained and evaluated their model on this dataset.

To construct our spatial dataset, we use a keyword search (i.e., looking for words and phrases such as "rightmost" or "at the top"). Since we are extracting this dataset directly from RefCOCO/+/g, it is composed of "realistic" scenarios where spatial understanding is necessary for REC. This dataset, named RefCOCO-Spatial, is used as the ultimate evaluation regarding whether or not we have improved model performance in spatial understanding.

Furthermore, it is also imperative to evaluate the extent to which our training procedures cause the model to "overfit" to the specific task of spatial understanding. To test for this, we take the complement dataset of RefCOCO-Spatial (i.e., create a set of image/caption pairs that have no spatial component), called RefCOCO-NonSpatial.

Object detection performance is evaluated based on the mean average precision (mAP) metric commonly used by models, including Grounding DINO, for evaluating on the COCO 2017 validation dataset [10].

TABLE I: Comparison of our dataset (RefCOCO-3DS) with other popular 3D object detection datasets showing the number of images per dataset, which datasets have annotation data for depth information, and statistics on available object categories.

Dataset	**Images**	**Depth**	**Categories***
Objectron [11]	4M	yes	6 / 9
PROPS [12]	1k	yes	1 / 10
Pix3D [13]	10k	yes	4 / 9
ObjectNet3D [14]	90k	no	36 / 100
Ours	7k	yes	30 / 30

*object categories statistic format: the number of object categories the dataset shares with the COCO dataset / the number of object categories the dataset has in total

B. 3D Dataset Generation

In addition to preparing datasets for evaluating model performance on 2D spatial phrases ("leftmost", "bottommost"), we use a 3D dataset for evaluating the model on 3D spatial phrases that incorporate depth (e.g. "nearest", "furthest"). To leverage the pre-training of Swin-B_cogcoor, our REC OD 3D dataset consists of object categories shared by the COCO and RefCOCO datasets while containing information about object pose or distance relative to the camera. While there exist many 3D object detection datasets, to the best of our knowledge there do not exist any datasets that simultaneously share more than 6 object categories with the COCO dataset and contain depth-related information that can be used to accurately determine the 3D location of an object with respect to the camera (see I).

We use Blender (a popular 3D modeling application) to generate a custom synthetic object detection dataset, RefCOCO 3D Synthetic (RefCOCO-3DS). This dataset is composed of 30 object categories, all of which are shared by the COCO and RefCOCO datasets and all but one of which are shared by the O365 dataset, allowing us to leverage the pre-training of Swin-T_ogc and maintain consistency with the 2D spatial awareness evaluation.

This 3D dataset is automatically generated using Python scripts leveraging Blender's API. The resulting dataset is composed of object bounding boxes, segmentation contours, 3D pose locations, and associated class labels. Using a 3D modeling tool such as Blender enables us to utilize spatial labels that involve depth, such as "nearest" and "furthest". To simplify the scope of this study, we avoid all other spatial phrases that (a) require an understanding of position relative to a landmark (e.g. "second chair from the right") or (b) refer to multiple objects (e.g. "the two chairs on the right"). For more information about the generation of RefCOCO-3DS, refer to Section IV-B.

C. GDINOSAUR Architecture

We are building on top of the existing Grounding DINO architecture [1] (see Appendix for more details about its architecture). We utilize our 3D dataset to fine-tune the pre-trained Grounding DINO model. By evaluating the model on RefCOCO-Spatial, we are able to determine the extent to which this fine-tuning technique enables the model to enhance its spatial understanding.

In our initial tests, we found that fine-tuning the model on our synthetically created dataset significantly diminished its performance on the RefCOCO evaluation sets. This means that while the model gets better on our restricted dataset, it is losing the ability to perform well on the dataset used during pretraining. Fundamentally, this is a continual learning problem. Since the 3D dataset is synthetically generated, there is a distinct possibility that our fine-tuning procedure causes the model to overfit and "catastrophically forget" its original capabilities. To test for catastrophic forgetting, we further evaluate the model on RefCOCO-NonSpatial.

To address this continual learning challenge, we adopt two approaches known to help reduce the forgetting of the first task. In the first approach, we mix data from the original RefCOCO/+/g dataset together with data from our new synthetic 3D dataset during the training process. The idea here is that by constantly "reminding" the model of the first task (the pre-trained dataset of RefCOCO), we prevent the model from completely forgetting the first task, while still learning the second task (the synthetic dataset that we created).

The other major method we use in an attempt to mitigate catastrophic forgetting is Elastic Weight Consolidation (EWC) proposed by DeepMind [15]. This technique involves the addition of a regularizing term to the loss while learning a second task, which prevents the deviation of the model's parameters from the parameters learnt during the first task's training. Further, since not all parameters contribute equally to the learning of the model, it weights the importance of the parameters by using the diagonal of the Fisher information matrix. Their formulation, which we use in our implementation, is as follows:

$$\mathcal{L}_{EWC}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^\star_{A,i})^2 \qquad (1)$$

where $\mathcal{L}_B(\theta)$ represents the loss we use to train the model on the second task (in our case this is training on the synthetic data), F_i represents the i^{th} diagonal element of the Fisher Information Matrix, θ_i represents the i^{th} parameter and $\theta^\star_{A,i}$ is the i^{th} parameter of the model trained on task A which in our case were the weights of the pre-trained model. λ was a hyperparameter (that we set to 2).

The authors of the EWC paper ([15]) decide to use the diagonal of the Fisher information matrix as a prior that indicates the importance of that parameter to the model. One of the reasons they chose this was that these values can be estimated using the square of the gradients of the parameters. We also choose to implement the Fisher Information Matrix calculation in the same way, by squaring the gradients computed during a forward pass of RefCOCO training data on the pre-trained model.

III. Related Work

As mentioned in previous sections, the ability to jointly process text queries and associated images consists of a diverse set of hard problems. Various approaches have tackled different aspects of the problem. ViperGPT [16] is a framework that given a query about an image, is able to output a program that combines the visual and textual models and, when run will answer the query. Some approaches[17] train their models in an implicitly supervised manner, allowing the model to look at multiple pairs of images and textual prompts. This ability to correlate a word with its physical real-world meaning is referred to as grounding[18]. In their work, they show that grounding-based language models are able to learn new words rapidly, which in our opinion, makes them suitable candidates for open-set object detection problems.

Another problem that these models face is the ability to understand "referring expressions", i.e. they often struggle to localize specific objects from an image referred to in the text. This difficulty can be partly attributed to the large number of regions identified by the object model that cannot be connected to the output of the text model. Some authors [19] sought to solve this issue by designing a novel model that was able to boost these correlations and perform better at the task of matching the text to the desired bounding box. In our search for models that are able to understand referring expressions using grounding-based methods, we came across Grounding DINO [1], which fused grounded pre-training with the transformer based detector DINO [20].

Building up on, DETR (Detection Transformer) [21] framework, DINO [20] introduces a contrastive denoising training method, a mixed query selection strategy for initializing anchor boxes, and a novel "look forward twice" scheme for box prediction.

DINO employs a contrastive denoising training strategy that effectively handles both positive and negative samples, helping the model to avoid duplicate outputs and to select high-quality anchors for bounding box predictions. The mixed query selection approach initializes positional queries with top encoder features while keeping content queries learnable. This strategy improves the model's ability to pool comprehensive content features from the encoder. Furthermore, the "look forward twice" box prediction scheme leverages refined box information from later layers to optimize the parameters of early layers, resulting in improved prediction accuracy.

Grounding DINO [1] is a novel framework designed to address the challenges of grounding text in visual data, particularly for object detection tasks. The framework is a open-set detector and consists of three main phases: a feature enhancer, a language-guided query selection, and a cross-modality decoder. These components work together to fuse image and language features effectively, allowing for more accurate identification and localization of objects in images based on textual queries.

Grounding DINO's design builds upon the strengths of the DETR (DEtection TRansformer) [21] model and the DINO (DETR with Improved denoising aNchOr boxes) [20] detector, which are both based on Transformer architectures. The advantages of Grounding DINO over other models like GLIP [22] include:

- Its Transformer-based architecture is similar to that of language models, making it more natural for processing both image and language data.
- Transformer-based detectors have been shown to excel when leveraging large-scale datasets, which is crucial for improving the model's performance and generalization.
- As a DETR-like model, Grounding DINO can be optimized end-to-end without relying on handcrafted modules such as Non-Maximum Suppression (NMS) [23]. This simplification makes the grounding model design more streamlined and potentially more effective.

The Grounding DINO architecture is a dual-encoder-single-decoder system comprising an image backbone, a text backbone, a feature enhancer, a language-guided query selection module, and a cross-modality decoder. The process involves extracting features from both the image and text, fusing these features, initializing queries based on the fused features, and refining these queries to predict object boxes and extract corresponding phrases.

The ablation study within the Grounding DINO framework highlights the importance of encoder fusion as the most critical design aspect. While word-level text prompts have the smallest impact, they are still beneficial. The language-guided query selection and text cross-attention have a more significant influence on the performance on datasets such as LVIS [24] and COCO [6], respectively.

In contrast, the GLIP [22] model unifies object detection and phrase grounding by pre-training on a combination of human-annotated and web-crawled image-text pairs. This approach enables the model to learn object-level, language-aware, and semantically-rich visual representations. The key contributions of the GLIP model include:

- A unified framework for detection and grounding, allowing the model to learn from a broader set of visual concepts.
- Scaling up visual concepts with massive image-text data for pre-training.
- Transfer learning capabilities, enabling the model to be applied to various tasks with minimal additional annotations.

The GLIP model employs deep cross-modality fusion, which allows for early fusion of information from both image and text modalities, leading to enhanced language-aware visual representations. When tested on established benchmarks such as COCO and LVIS, the GLIP model demonstrates strong performance, surpassing many supervised baselines even in zero-shot settings.

Both Grounding DINO and GLIP models represent significant advancements in vision-language processing. Their ability to leverage large-scale image-text pairs for pre-training and their effective transfer to multiple tasks offer promising directions for future research in the field.

IV. Experiments and Results

A. 2D Dataset

As mentioned in Section II, we evaluate our model on a modified RefCOCO/+/g dataset and a synthetically generated custom data set. First, we extract the datasets for RefCOCO/+/g [2], which contains annotated COCO data. Then, we extract the informative data points from RefCOCO/+/g (ones that contain the referring words that we will be training on) and expand the annotations to use for training and validation.

B. Generating RefCOCO-3DS

This section outlines the pipeline for creating a comprehensive 3D dataset aimed at improving the spatial awareness capabilities of Grounding DINO. The dataset is built using 3D objects sourced from the following open-source distributions: sketchfab.com, free3d.com, turbosquid.com, and cgtrader.com. We specifically select object models with high-quality textures and belong to one of the COCO dataset object categories to curate a realistic and relevant dataset for finetuning Grounding DINO. We modify these models to standardize their origin points for consistency and automation efficiency built upon in the later stages of the dataset creation.

The main tool used for data preparation is Blender, the cornerstone tool in this pipeline, which is utilized for its capabilities in:

- **3D Modeling and Editing:** Precise manipulation of object origins and geometry is essential for data preparation.
- **Scripting with Python:** Blender's Python API is used to automate the manipulation of objects, cameras, and materials, facilitating the creation of a diverse dataset.
- **Rendering:** High-quality, photo-realistic images are rendered from the 3D scenes, contributing to the dataset with a range of scenarios and lighting conditions.

The following itemized points provide a brief overview of our dataset generation pipeline and are further explored later on:

1) **Selection and Preparation of 3D Objects:** Objects are sourced and modified to standardize origin points.
2) **Dataset Automation with Blender:** A Python script automates the dataset creation process within Blender, handling object, camera, and lighting manipulations.
3) **Scene Randomization:** Variability is introduced in indoor environment selection, object positions and orientations, camera positions and orientations, and lighting conditions to simulate diverse scenarios.

4) **Image Rendering:** The scenes are rendered to produce high-quality images for the dataset.

1) Selection and Preparation of 3D Objects: Out of the COCO dataset's 80 object categories, we narrow our objects of interest down to 69 non-living object categories (including "plants"). This decision is made to enhance the realism of our simulated images and simplify the procedures for randomizing object poses since synthetic living object categories are more difficult to make look realistic and higher degrees of articulation common to living object categories increase the number and difficulty of programmatically capturing all possible object poses respectively.

To simplify the selection of our environments which would serve as the background to our images, we chose a Blender model or a studio apartment, as seen in Figure 3, to place various objects into. Within this indoor environment, we can place and take images of objects in a living room, kitchen, and/or dining room environment. As a result, the number of objects we now consider is further filtered to indoor objects that could realistically appear in the living room, kitchen, or dining room environments of a studio apartment.

(a) Living Room (b) Kitchen (c) Dining Room

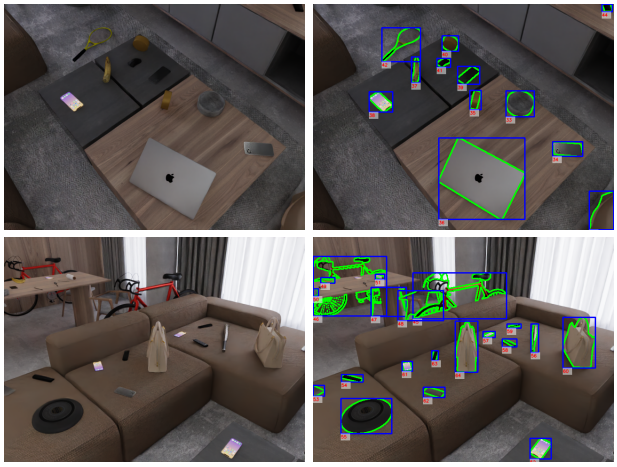
Fig. 3: Each of the sub-envionrments of the apartment model selected for generating RefCOCO-3DS

Next, we only consider objects that we determine indoor environments could have multiple of and would be easy to duplicate in our apartment model, eliminating object categories such as microwave and sink. To provide common indoor environment complexity without having to procedurally place large and/or foundational indoor objects, we choose to leave several models belonging to COCO object categories in the same place while choosing to not annotate the models. These include the following object categories: couch, table, sink, microwave, oven, and tv. Finally, our list of objects incorporated into the RefCOCO-3DS dataset is filtered down to the 30 object categories shown in Table II.

TABLE II: Object categories with annotations found in the RefCOCO-3DS dataset

backpack	banana	baseball bat	baseball glove	bicycle
book	bottle	bowl	cell phone	chair
clock	cup	fork	frisbee	hair drier
handbag	knife	laptop	mouse	potted plant
remote	skateboard	skiis	spoon	sportsball
suitcase	tennis racket	toaster	umbrella	vase

2) Dataset Automation with Blender: We use Blender's internal Python scripting to automate the scene generation and labeling processes as shown in Figure 4.



(a) raw images (b) annotated images

Fig. 4: Example images of the dataset with visuals of the bounding box and segmentation contour annotations.

To automate random scene generation, we've created three user-friendly useful tools in Blender: camera volumes, camera targets, and object planes. Camera volumes are rectangular prisms defined in the 3D environment and serve as possible spawning locations for our virtual camera. Multiple of these camera volumes are placed around the indoor scene of interest (living room, kitchen, or dining room) such that the majority of our images will look inwards toward the center of the scene. Camera targets are virtual locations toward which the virtual camera will aim while permitting the camera to still rotate or "roll" in the clockwise and counter-clockwise directions. We place multiple camera targets on or near the objects of interest for our dataset. Object planes are 2D-level planes serving as possible spawning locations for our manually imported 3D objects. These planes are placed on floors and table tops and require intuition-driven defining of which object can appear on each object plane, which we standardize using object plane dictionaries saved as JSON files.

Since a single object can have multiple stable or natural resting positions, (e.g. a soda can can be naturally resting in the upright, side, and upside down orientations), we manually define several of the object models such that they have multiple natural orientations. After a natural orientation has been defined, we automate additional orientation changes by defining rotational limits for each object thereby allowing for infinite possible orientations for each object. For objects with natural resting orientations that may require leaning up against a wall (e.g. a bicycle or upright skateboard), we specify several locations throughout the apartment for where they may appear. In total, we have 35 total objects converting 30 COCO object categories and 44 unique natural resting orientations.

3) Scene Randomization: While all the images in our dataset are generated in a model of a single apartment, we undertake many efforts to make each image as unique as possible. Each of the three sub-environments in the apartment equivalently contributes one-third of the total images, meaning the locations of the camera and camera targets were placed

TABLE III: Image render settings used for RefCOCO-3DS

Blender Setting	Value	Blender Setting	Value
samples	100	use fast gi	false
adaptive sampling	true	debug use spatial splits	true
adaptive threshold	0.5	debug use hair bvh	true
use denoising	true	use persistent data	true

in each sub-environment 33% of the time. Within each environment, 5 unique camera volumes are placed along with 2 camera targets. Considering that (1) the smallest camera volume has a volume $177\text{e}{+6} mm^3$ and (2) the camera location is chosen as a random point in a camera volume discretized to a resolution of $1mm^3$, we determine that the minimum number of possible camera views per sub-environment is $1.77\text{e}{+9}$. Throughout all sub-environments, we define 35 object planes with 14 unique object dictionaries to dictate which objects can appear on each object plane.

When automatically adding a candidate object to an object plane, a random point on the plane and random object orientation are proposed. To avoid objects overlapping with one another, we keep track of the 2D footprints created by all selected objects on the object plane. These footprints are represented as rectangles and are arbitrarily padded to ensure that objects do not appear too close together to avoid cluttered scenes. Before placing an object at a proposed object plane point and object orientation, we ensure the corresponding proposed footprint does not intersect with any existing footprints. Lastly, lighting is varied by controlling both the light coming from a pseudo-sun light source outside of the apartment and the light sources inside the apartment.

For every image, the following parameters are varied: camera placement, camera orientation, and lighting. For computational efficiency, we (1) allow a maximum number of 50 attempts to place an object on an object plane before discarding the candidate object and (2) randomize object placements once every 25 images.

4) Image Rendering: Thanks to Blender's cross-platform compatibility, we generate images and annotations on multiple laptops simultaneously and later combine multiple small datasets to form one large dataset. In particular, we are able to achieve an image generation rate of 1 image every 13 seconds or approximately 277 images per hour on a MacBook Pro Apple M1 Max with 32 GB RAM. Through trial and error, the Blender settings shown in Table III are found to provide the best trade-off between image quality and render speed.

C. Replicating Grounding DINO

Ideally, it would be possible to replicate Grounding DINO's results exactly by leveraging their original code. Unfortunately, the creators of Grounding DINO did not release their training code, so we instead chose to utilize an open-source implementation called Open-GroundingDINO (Open-GDINO) [25]. Open-GDINO is almost identical to the original Grounding DINO model except for minor differences in the training code. These minor differences result in a small fraction of the difference between the results achieved by Grounding DINO and Open-GDINO. More specifically, the creators of Open-GDINO were able to pre-train a Grounding DINO model on three datasets—O365, GoldG, and Cap4M—to achieve a 48.4% mAP (mean average precision) when evaluated on the COCO 2017 dataset. This is precisely the same number that the original Grounding DINO achieved under the same circumstances. Upon finetuning on the COCO 2017 dataset, Open-GDINO achieves 57.3%, which is marginally better than the original Grounding DINO's 57.2%. These differences can most likely be attributed to either stochasticity during optimization or minor differences in the training code itself. Regardless, Open-GDINO seems to be similar enough to Grounding DINO to be a suitable baseline to build upon.

TABLE IV: Comparison of training configurations and results (mAP) between Grounding DINO and Open-GDINO

	Grounding DINO	Open-Grounding DINO
O365, GoldG, Cap4M	48.4%	48.4%
O365, GoldG, Cap4M + COCO Fine-Tuning	57.2%	57.3%
COCO, O365, LIVS, V3Det, GRIT-200K, Flickr30k	N/A	55.1%

A more detailed comparison between Grounding DINO and Open-GDINO is available in Table IV. In addition to replicating two of Grounding DINO's results, the creators of Open-GDINO also trained a custom configuration (shown in the bottom row) on several different datasets, some of which were used by Grounding DINO, and others were not. After pretraining on this dataset, Open-GDINO achieved a 55.1% accuracy on the COCO 2017 dataset.

To prepare Open-GDINO for usage, our first step was to replicate the creators' evaluation results. This required several steps. First, we clone the Open-GDINO repository and download the pre-trained model weights. Second, we download the COCO 2017 evaluation dataset. Third, we adjust the authors' SLURM scripts to interface with Great Lakes and utilize the course account. Finally, we are able to run this script to launch pre-written Python evaluation code as a SLURM job.

The creators of Open-GDINO currently have two pre-trained weight configurations uploaded to GitHub. Although we have tried integrating both of them, only one has worked thus far (the other seems to be deprecated; it has the wrong tensor shapes). After evaluating the working model on the COCO 2017 evaluation dataset, we achieved a mAP of 55.1%, which precisely matches the Open-GDINO result achieved in row 3 of Table 1. Thus, we feel confident that we have achieved the goal of replicating one of Open-GDINO's evaluation results.

Having gotten the evaluation code working, we then move on to Open-GDINO's training code. Ideally, we would like

to be able to pre-train and fine-tune Open-GDINO on the datasets mentioned by the creators so that we can fully verify our experimental setup. As an initial experiment, we began by pre-training the model on the COCO 2017 training dataset. Launching a job for this task was relatively simple—we merely needed to create a SLURM script that interfaced with Open-GDINO's training code and launch a job. Once we had launched the job, however, we encounter some major computing problems. Given the computational constraints of the course accounts on Great Lakes—a maximum of 1 GPU for 8 hours—we are only able to train for half of one epoch before the experiment times out. The original Grounding DINO authors pre-train their model for 18 epochs on an even larger dataset (O365, GoldG, and Cap4M). (Note that, when fine-tuning from the DINO baseline, it only requires 7 epochs.) Thus, we have concluded that fully pre-training on any of the large datasets (or any combination thereof) is not feasible given our computational constraints.

Instead, we utilize two interventions to circumvent these computational constraints. First, we adopt the same strategy as the creators of Grounding DINO and Open-GDINO: freezing model weights. Rather than fine-tuning the entire model on COCO, we choose to freeze a portion of the model weights. Specifically, we choose to freeze weights in the model's image backbone and decoder layer, similar to the procedure used to train Grounding DINO. By freezing some model weights, we no longer have to compute as many gradients, and training thus proceeded more quickly.

Secondly, we decide to forego the training necessary to replicate Open-GDINO's fine-tuning results altogether. Instead, we have discovered that we could utilize the pre-trained version of Grounding DINO, which the authors host on GitHub, as a baseline. These model weights, referred to as `GroundingDINO-B` have been pre-trained on the following datasets: COCO, O365, GoldG, Cap4M, OpenImage, ODinW-35, and RefCOCO. By evaluating the performance of this pre-trained model and then comparing this performance to our fine-tuned model, we are able to analyze the effects of our proposed training procedures. Furthermore, without having to pre-train on RefCOCO, we are able to avoid a major computational undertaking.

D. Evaluation Procedure

To properly evaluate our proposed methodologies (mixture-based training and elastic weight consolidation, both of which are described in section II.C), we fine-tune `GroundingDINO-B` using the various configurations detailed in table V. The evaluation graphs for these experiments are all displayed in the appendix.

Due to the computational constraints described above, we are not able to train on the full RefCOCO training dataset. Instead, we choose a random subset of 5,000 images from the original RefCOCO dataset (which has roughly 20,000 images). Due to the fact that this randomly selected dataset has only 5,000 images, we refer to it in this paper as "1/4 RefCOCO."

TABLE V: Details of the various training configurations that we used.

Training Data	**Uses EWC?**	**Fig. #**
1/4 RefCOCO + Synthetic	No	6
1/4 RefCOCO Only	No	7
Synthetic Only	No	8
1/4 RefCOCO + Synthetic	Yes	9
Synthetic Only	Yes	10

Unfortunately, the authors of Grounding DINO did not publish any code for evaluating referring expression comprehension en masse (i.e., for a large dataset). They did, however, publish a Jupyter Notebook that evaluates the IoU performance for a model over singular images. To evaluate our models, we convert this Jupyter Notebook into a script that we could launch using SLURM to evaluate performance across a large dataset. Using this script, we are able to obtain mean average precision (mAP) estimates for the model over various splits of the data.

To properly evaluate performance, we choose to examine model performance over three data subsets. First, we feel that it would be prudent to evaluate performance over a validation split of our own synthetic dataset. Second and third, we would like to evaluate performance over a spatial and non-spatial split of RefCOCO's validation dataset. To generate this split, we loop through every referring expression in the RefCOCO validation dataset and searched for spatial words. Any referring expression using one of the following words was sorted into the spatial split of the dataset: 'right', 'rightmost', 'left', 'leftmost', 'furthest', 'farthest', 'far', 'near', 'nearest', 'top', 'topmost', 'upper', 'uppermost', 'bottom', 'close', 'closest', 'lower', 'foreground', 'background', and 'distant'. By doing this, we could isolate our analysis of model performance over spatial or non-spatial while still evaluating on real image data.

E. Results and Discussion

The complete results of our training are displayed in the appendix of this report. This section will mostly serve as a discussion of our findings.

Despite our hypothesis that synthetic data would improve spatial performance specifically, we found that our synthetic dataset improved model performance across both the spatial and non-spatial splits of the RefCOCO validation set. This result is slightly surprising, but given the wide variety of image scenes and object categories that we sampled in our synthetic dataset, it makes sense that we see a positive transfer to the domain of real images. Furthermore, it is extremely likely that referring expression comprehension requires some amount of spatial understanding even when the referring expressions do not explicitly emphasize spatial words. If true, it is possible that we enriched `GroundingDINO-B` over all types of referring expressions by improving its spatial understanding.

Figure 7 was slightly confusing, as the results were opposite to what we would expect. After fine-tuning specifically on

the 1/4 RefCOCO dataset, we see enhanced performance on the synthetic data and decreased performance on the RefCOCO validation set. Since we were training on one-fourth of RefCOCO rather than RefCOCO in its entirety, it is possible that we are simply overfitting to this dataset. The gains over the synthetic dataset are fairly minor and could potentially be explained by random chance. To validate this hypothesis, we would ideally run this same job with randomly selected subsets of RefCOCO that increase in size. If true, we would see the performance degradation decrease as the training set size increases. Due to time constraints, we would leave the validation of this hypothesis to future work.

On the whole, the most effective methodology seemed to be training on our synthetic dataset using elastic weight consolidation. In particular, EWC training on the mixture between RefCOCO and synthetic data resulted in almost a **15% increase** in mean average precision over both the spatial and non-spatial splits of the RefCOCO validation set. This result demonstrates that our synthetic dataset is capable of improving model performance on real image data. We feel confident that, if provided with more time and computational resources, our synthetic dataset architecture could serve as a method for substantially improving the performance of referring expression comprehension models at large.

V. Future Work

A. Improving the Synthetic Dataset

The synthetic dataset, RefCOCO-3DS, we create for this research serves as a valuable starting point for enhancing the spatial awareness capabilities of Grounding DINO. However, several avenues for future work could significantly improve the quantity, quality, and diversity of the dataset:

1) **Increasing the number of images**: With more computational resources and project hours, we could expand the dataset to include more images. This would provide the model with a more comprehensive set of examples to learn from, potentially leading to better generalization and robustness.
2) **Diversifying environments**: While our current dataset is based on a single studio apartment model, future iterations could incorporate a wider variety of indoor environments, such as different apartment layouts, houses, offices, and public spaces. This environmental diversity would expose the model to a broader range of spatial configurations and challenges.
3) **Expanding object diversity**: Our current dataset includes 30 object categories from the COCO dataset. In future work, we could aim to increase the number of object categories and include a greater variety of objects within each category. This would help the model learn to recognize and localize objects across a wider spectrum of shapes, sizes, and textures.
4) **Improving object realism**: While the current objects in our dataset are sourced from open-source 3D model repositories, future work could focus on creating or acquiring higher-quality, more photorealistic 3D models. This would help bridge the gap between the synthetic dataset and real-world images, potentially improving the model's performance on real-world data.
5) **Incorporating more complex spatial relationships**: Our current dataset focuses on simple spatial relationships such as "nearest" and "furthest." In future work, we could explore more complex spatial phrases that involve relative positions to landmarks or refer to multiple objects. This would challenge the model to develop a more nuanced understanding of spatial relationships.

It is important to note that we have created the RefCOCO-3DS dataset in a relatively short amount of time with limited resources. With more time, computational resources, and person-hours dedicated to dataset generation, we could significantly improve the quantity, quality, and diversity of the dataset. These enhancements would likely lead to even better performance gains for the Grounding DINO model in terms of spatial awareness and generalization to real-world scenarios.

B. Spatially-Aware Loss Functions for Grounding DINO

1) **Spatio-Temporal Graph Loss [26]:**
   - Construct a spatio-temporal graph $G = (V, E)$ of objects across frames.
   - Compute the Laplacian matrix $L = D - A$.
   - Implement the loss: $\mathcal{L}_{\text{graph}} = \sum_{i,j} L_{ij} \|x_i - x_j\|^2$.
2) **Spatial Distance Attention Regularization [27]:**
   - Compute spatial distance $d_{ij} = \|p_i - p_j\|_p$ between regions i and j.
   - Implement the penalty: $\mathcal{L}_{\text{attn}} = \sum_{i,j} f(d_{ij}) \alpha_{ij}$.
3) **Multi-Scale Spatial Alignment Losses:**
   - Implement alignment losses at each decoder layer l: $\mathcal{L}_{\text{align}}^{(l)} = \sum_i \text{IoU}(b_i^{(l)}, g_i)$.
   - Balance contributions: $\mathcal{L}_{\text{total}} = \sum_l \lambda^{(l)} \mathcal{L}_{\text{align}}^{(l)}$.

These spatially-aware loss functions can be incorporated into the training of DINO to improve its grounding capabilities.

VI. Conclusion

Despite the advancement in object detection models, a fully reliable model that processes referring expression has yet to exist. This project explores the approaches to improve the Grounding DINO model, including training on synthetic photorealistic 3D datasets, employing mixture datasets, and implementing Elastic Weight Consolidation. Two versions of synthetic datasets are curated, changes to the loss function are made, and the Ground DINO model is trained with different mixtures of the original RefCOCO datasets and the self-curated 3D datasets. The resulting model performs better in both spatial and non-spatial aspects of the RefCOCO dataset. Future research into Referring Expression Comprehension should focus on a more diverse and realistic 3D dataset and loss functions emphasizing spatial awareness.

VII. Author Contributions

All co-authors were involved in writing this report. All co-authors attended and contributed to all team meetings, and equally contributed to this project.

B.M. proposed improving Grounding DINO's 2D and 3D spatial REC performance, conceived the synthetic data generation method, and wrote the code for generating RefCOCO-3DS and for evaluating model REC performance. B.M., J.L., and A.S. created the pipeline for sourcing and preparing object models for automated dataset generation. J.L. and A.S. defined the scope of, sourced, and prepared the object models used in the final dataset. J.S. and A.V. conceived and implemented the mixed dataset training and elastic weight consolidation method respectively. J.S. and A.V. conducted model training and model evaluation on the Great Lakes super cluster. All team members generated data for the RefCOCO-3DS dataset, discussed the results and implications and commented on the manuscript at all stages.

REFERENCES

[1] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[2] L. Yu, P. Poirson, S. Yang, A. Berg, and T. Berg, "Modeling context in referring expressions," 07 2016.

[3] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, "Generation and comprehension of unambiguous object descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 11–20.

[4] J. A. Fodor and Z. W. Pylyshyn, "Connectionism and cognitive architecture: A critical analysis," *Cognition*, vol. 28, no. 1-2, pp. 3–71, 1988.

[5] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[7] S. Shao, Z. Li, T. Zhang, C. Peng, G. Yu, X. Zhang, J. Li, and J. Sun, "Objects365: A large-scale, high-quality dataset for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8430–8439.

[8] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *International journal of computer vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[9] C. Li, H. Liu, L. Li, P. Zhang, J. Aneja, J. Yang, P. Jin, H. Hu, Z. Liu, Y. J. Lee *et al.*, "Elevater: A benchmark and toolkit for evaluating language-augmented visual models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9287–9301, 2022.

[10] [Online]. Available: https://cocodataset.org/#detection-eval

[11] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann, "Objectron: A large scale dataset of object-centric videos in the wild with pose annotations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7822–7831.

[12] A. Opipari, "Pose estimation." [Online]. Available: https://deeprob.org/w24/datasets/props-pose/

[13] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2974–2983.

[14] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference Computer Vision (ECCV)*, 2016.

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, p. 3521–3526, Mar. 2017. [Online]. Available: http://dx.doi.org/10.1073/pnas.1611835114

[16] D. Surís, S. Menon, and C. Vondrick, "Vipergpt: Visual inference via python execution for reasoning," 2023.

[17] H. Shi, J. Mao, K. Gimpel, and K. Livescu, "Visually grounded neural syntax acquisition," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019. [Online]. Available: http://dx.doi.org/10.18653/v1/P19-1180

[18] Z. Ma, J. Pan, and J. Chai, "World-to-words: Grounded open vocabulary acquisition through fast mapping in vision-language models," 2023.

[19] J. Wang, J. Ke, H.-H. Shuai, Y.-H. Li, and W.-H. Cheng, "Referring expression comprehension via enhanced cross-modal graph attention networks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 19, no. 2, feb 2023. [Online]. Available: https://doi.org/10.1145/3548688

[20] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," 2022.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[22] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang *et al.*, "Grounded language-image pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 965–10 975.

[23] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4507–4515.

[24] A. Gupta, P. Dollar, and R. Girshick, "Lvis: A dataset for large vocabulary instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5356–5364.

[25] W. L. Zuwei Long, "Open grounding dino:the third party implementation of the paper grounding dino," https://github.com/longzw1997/Open-GroundingDino, 2023.

[26] M. Tchenegnon, S. Gibet, and T. L. Naour, "A new spatio-temporal loss function for 3d motion reconstruction and extended temporal metrics for motion evaluation," *arXiv preprint arXiv:2210.08562*, 2022.

[27] L. H. Mormille, C. Broni-Bediako, and M. Atsumi, "Regularizing self-attention on vision transformers with 2d spatial distance loss," *Artificial Life and Robotics*, vol. 27, no. 3, pp. 586–593, 2022.

[28] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," 2016.

[29] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," 2016.

[30] L. Jin, G. Luo, Y. Zhou, X. Sun, G. Jiang, A. Shu, and R. Ji, "Refclip: A universal teacher for weakly supervised referring expression comprehension," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 2681–2690.

[31] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," 2019.

[32] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[33] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 901–10 911.

APPENDIX

A. Grounding DINO Architecture

In Figure 5, we show the architecture used by Grounding DINO [1]. The model consists of a multimodal encoder (an image backbone, and a text backbone), a feature enhancer that uses text-to-image and image-to-text cross attention layers, a decoder that uses more attention layers. Observe in the figure that some of the blocks are colored in white; these represent the parts of the model that were frozen during their pretraining. In our finetuning process, we froze the same layers.

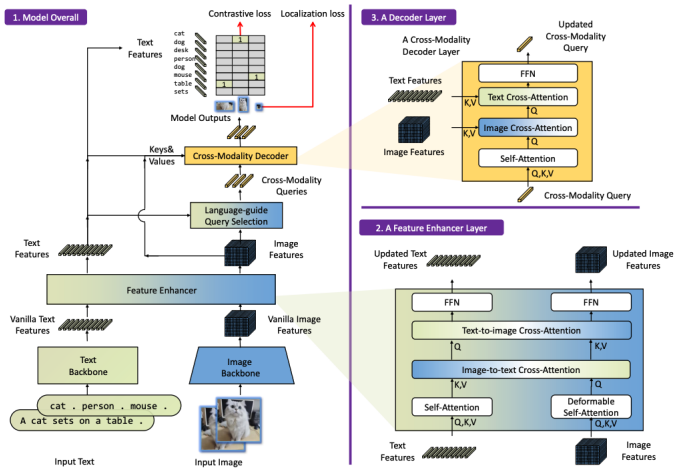


Fig. 5: GDINO Architecture

B. Training Curves

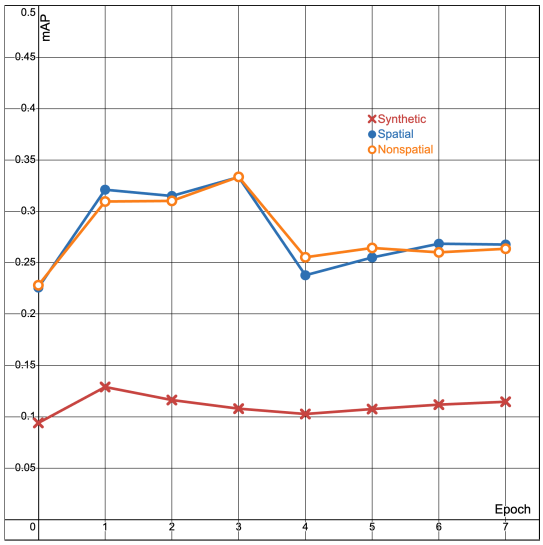


Fig. 6: Training on Mixture Dataset

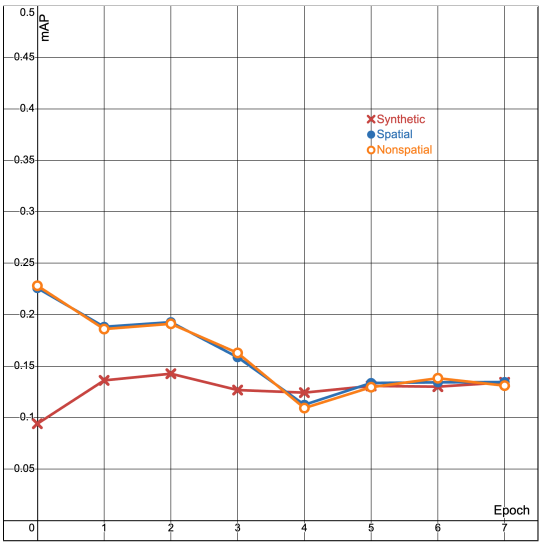


Fig. 7: Training on Quarter RefCOCO Dataset

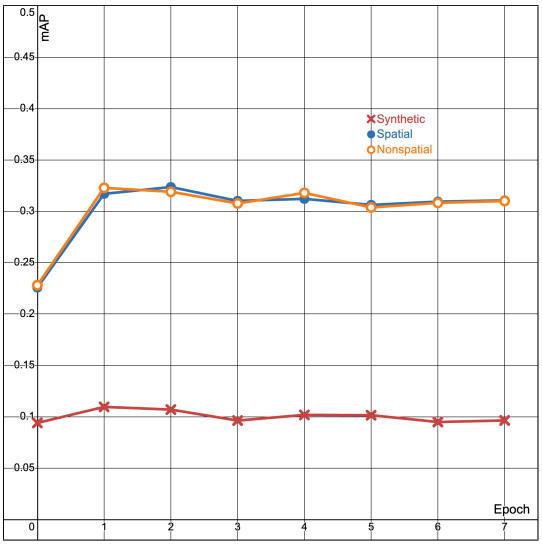


Fig. 8: Training on Synthetic Dataset

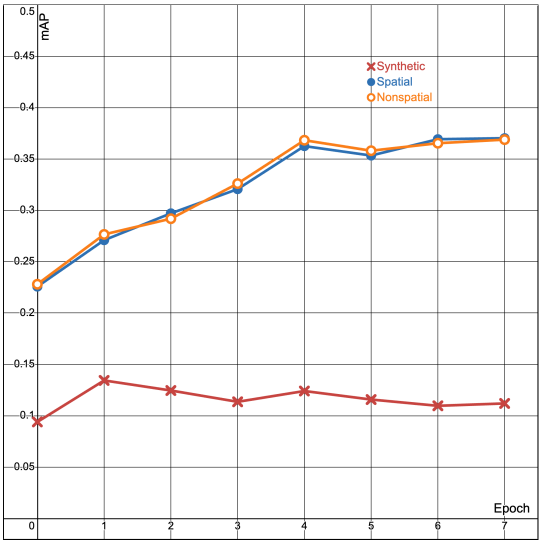


Fig. 9: Training on Mixture Dataset with EWC

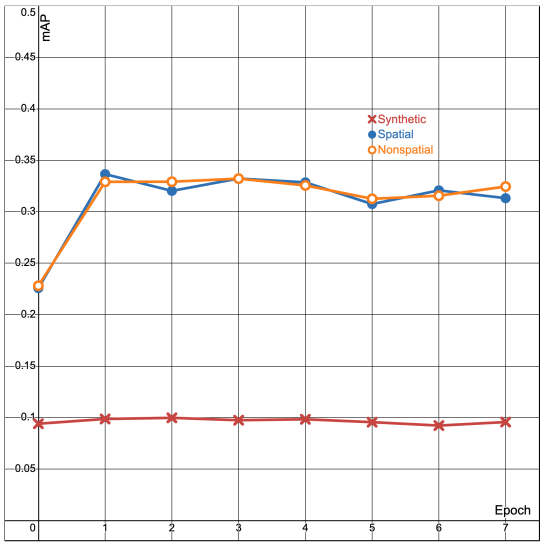


Fig. 10: Training on Synthetic Dataset with EWC

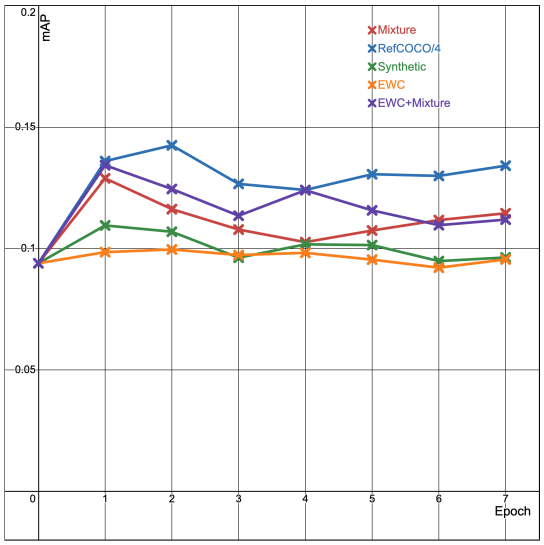


Fig. 11: Evaluation on Synthetic Dataset

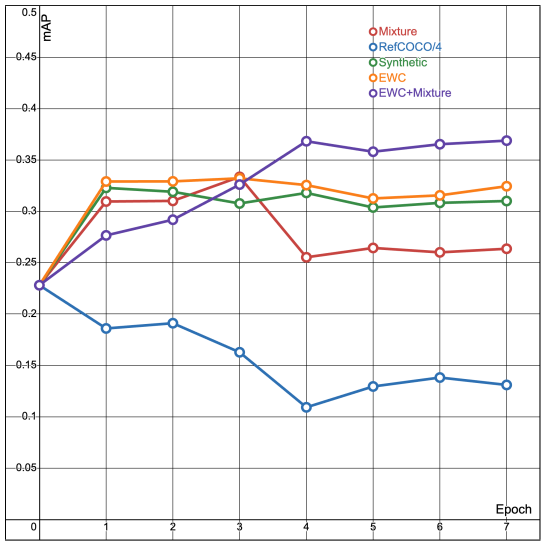


Fig. 13: Evaluation on RefCOCO Non-Spatial Split

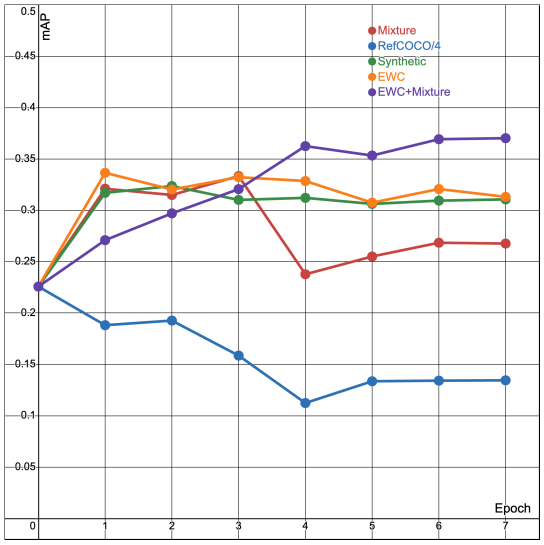


Fig. 12: Evaluation on RefCOCO Spatial Split