POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Adapting Vision-Language Models for Open-Vocabulary Object Detection through Prompt Learning

Supervisors
Prof. Barbara Caputo
Dr Fabio Cermelli
PhD Gabriele Rosi

Candidates
Claudio Macaluso
matricola: 317149

Academic year 2024-2025

Summary

In recent years, foundational vision-language models have opened new opportunities for addressing open-vocabulary object detection, with applications such as automatic image annotation. However, despite their generalization ability, these models often lack the specialization required to adapt efficiently to novel datasets or domains, especially in low-data regimes. This thesis investigates the use of prompt learning techniques, originally developed in the natural language processing field, to enhance the adaptability of vision-language models for object detection. By leveraging the intrinsic fusion of text and visual modalities in these architectures, we extend current baselines with prompt-based methods and evaluate their performance in few-shot setups. The results demonstrate that the proposed approaches consistently outperform the baseline method, showing the potential of prompt learning to specialize foundational models for the task of image annotation with minimal supervision.

Acknowledgements

Desidero innanzitutto ringraziare i miei relatori: Barbara Caputo, Fabio Cermelli e Gabriele Rosi, per la loro disponibilità, la loro competenza e per avermi guidato con attenzione e fiducia in questo percorso di ricerca. Accompagnandomi nella stesura di questa tesi, mi hanno dato l'opportunità di mettermi in gioco e di acquisire competenze che mi risulteranno sicuramente utili nel mio percorso futuro.

Un ringraziamento speciale va al team di FocoosAI, che mi ha accolto con professionalità e curiosità, offrendomi un ambiente stimolante in cui mettere alla prova le mie skills. In loro riconosco il perfetto connubio tra competenza e umanità, facendomi sentire sempre parte della squadra.

Alla mia famiglia: mamma, papà, Fabio, Vito e Aron, per il sostegno incondizionato, la pazienza e l'amore che non mi hanno mai fatto mancare. E' anche grazie a voi se ho avuto la possibilità e la capacità di portare a conclusione questo percorso. Il ricordo che ho di quando si viveva sotto lo stesso tetto è una delle memorie più confortevoli e calde che io possa richiamare, e potrò girare il mondo alla ricerca di me stesso, ma alla fine so per certo che una grossa parte di me è lì con voi che mi aspetta.

Ai miei amici di sempre, la mia seconda famiglia: Puglisi, Ciccio Piccolo, Giacomo, Mattia, Salvo, Putro e Mimmo, per avermi insegnato cosa vuol dire volersi bene. Alla loro capacità di portare leggerezza ed allo stesso tempo di farmi sentire ascoltato e capito.

Al mio coinquilino Enf, compagno di mille giornate, che mi ha insegnato a credere in me stesso anche nei momenti più bui.

Agli amici di Torino: Roberto, Riccardo, Filippo, Alex, Marianna, Giorgia, Mario e Gino, per aver reso questa città una casa, e per tutte le risate, le serate, e la leggerezza che solo voi avete saputo darmi.

A quelle persone che, anche se conosco da poco, mi hanno riempito di amore, gioia e calore in questo ultimo periodo. Alle tantissime conversazioni infinite che abbiamo avuto, perdendo totalmente il concetto del tempo e dello spazio.

A tutti voi devo tanto, perché senza di voi non sarei arrivato dove sono adesso.

Al passato, che continua a farmi da maestro, e al futuro, che mi chiama con la sua promessa.

A Torino, che mi ha cambiato, e a casa mia, che non mi ha mai lasciato. Grazie.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation: The Annotation Bottleneck

The field of computer vision has been one of the most fascinating areas in recent decades, attracting considerable interest from both researchers and companies. Giving machines the ability to perceive the world around them opens up a wide range of possibilities, paving the way for their deeper integration into human life.

The progress of modern computer vision models has been fueled by the availability of large-scale, richly annotated datasets. Resources such as PASCAL VOC [1] and MS COCO [2] have enabled the development of highly accurate supervised models, ranging from early frameworks like Faster R-CNN [3] to more recent approaches such as YOLO [4] and Transformer-based architectures [5, 6, 7]. However, this paradigm faces a persistent bottleneck: the creation of high-quality annotations is labor-intensive, costly, and slow. For the object detection task, annotating bounding boxes across several objects requires enormous human effort and expertise, especially in specialized domains such as medical imaging or industrial defect detection.

For many practical applications, the cost of annotation outweighs the benefits, leading to datasets that are small, weakly labeled, or partially annotated. This limits the deployment of computer vision models in real-world scenarios where the diversity of classes and contexts cannot be fully captured by large public benchmarks. Addressing this bottleneck requires automated or semi-automated annotation strategies that reduce dependence on extensive manual labeling while preserving annotation quality.

1.2 Problem Statement: Automatic Annotation in Low-Supervision Settings

Previous work has addressed the problem of automated or semi-automated annotation from several perspectives.

Early methods focused on semi-supervised detection, leveraging limited labels or image-level supervision to train models through pseudo-labeling and self-training pipelines to exploit unlabeled samples. Works like STAC [8], which uses a teacher model to extract high-confidence predictions for the student model for consistency regularizaition, or Unbiased Teacher [9], that that addresses the class imbalance of pseudo-labels using a Focal Loss [10] and improve incrementally the teacher model through Exponential Moving Average [11], significantly advanced the exploration of low-supervision paradigms in object detection. Although these approaches reduced the need for full annotations, they remained constrained to a closed vocabulary and required careful tuning to avoid error accumulation.

Few-shot and meta-learning methods later aimed to generalize to unseen classes using a handful of examples. Two-stage Fine-tuning Approach [12] showed that a simple tuning of the last layer can outperform more complex methods, while in Meta RCNN [13] a module capable of rescaling the input of the classificaiton head has been proposed in order to adapt the network to novel classes at test time. While these approaches don't require any model generated pseudo-label, they still rely on at least one labeled instance per class, limiting their applicability to fully automatic labeling.

More recently, the advent of Vision-Language foundation models such as CLIP [14] and ALIGN [15] has introduced open-vocabulary approaches, showing remarkable generalization capabilities: by aligning visual and textual features, model can be prompted through natural language rather than having fixed classification heads. Models such as YOLOE [16] and GroundingDINO [17] extended this concept to object detection. However, directly applying these models to new datasets often results in noisy predictions due to domain gaps, spurious correlations, and the lack of task-specific adaptation.

The central research question is therefore: how can we adapt foundation models for automatic dataset annotation in low-supervision scenarios, ensuring both efficiency and robustness?

1.3 Proposed Approach: Prompt Learning for Automatic Pseudo-Labeling

Building on the limitations of prior methods, recent research has explored prompt learning as a lightweight strategy to adapt large Vision-Language models to new tasks or domains with minimal supervision.

Born in the context of Natural Language Processing, prompt learning is a technique that replaces manually crafted prompts (e.g., "a photo of a class") with learnable prompt embeddings.

Handcrafted templates require domain expertise or extensive trial-and-error, and the same handcrafted phrase that worked in one task/dataset/domain could perform poorly in another. Prompt learning was introduced as a data-driven alternative aiming, through a training process, to optimize the prompt for the downstream application. This method provides a parameter-efficient means to adapt large-scale Vision-Language models to new tasks/-datasets/domains without fine-tuning their billions of parameters.

This approach is trivially adaptable to Vision-Language models due to their promptable nature. Given the text description of a class, the same can be specialized through a prompt refining process in order to improve model performance.

Works like CoOp [18] and LoCoOp [19] have proposed a simple yet effective way to specialize the CLIP [14] model using learnable prompts, enhancing its image classification capabilities.

The prompting approach can be naturally extended to models like GroundingDINO [17], which is exactly what this work aims to explore, focusing on the development of a prompt-enhanced detection module as a step toward a future automatic pseudo-labeling pipeline. Using these methods, we can specialize the model to a specific dataset distribution through prompt learning strategies, leveraging the foundational knowledge of the model to optimize prompts, while preserving the model's zero-shot inference capability using the same parameters.

1.4 Contributions

The main contribution of this thesis is the extension of prompt learning techniques to open-vocabulary foundational Vision-Language models. Specifically, two previously mentioned methods are adapted from global-level to region-level tasks, extending them to object detection:

- Context Optimization (CoOp) [18]: learns a set of continuous "soft prompts" that guide the model to better align with target categories using only a small labeled subset.
- Local regularized Context Optimization (LoCoOp) [19]: introduces a regularization loss to reduce the effect of spurious features (e.g., backgrounds, irrelevant textures) that often contaminate bounding boxes.

The methods are designed to be parameter-efficient, requiring optimization of only the prompt vectors while keeping the foundation model frozen. This allows for scalable adaptation without the computational burden of full fine-tuning.

Building on these adaptations, this thesis contributes the following:

- Benchmarking of open-vocabulary detectors (YOLOE and GroundingDINO) for their suitability in automatic annotation pipelines in the zero-shot setup.
- Extension of GroundingDINO with prompt learning methods (CoOp and LoCoOp) to enhance performance in low-supervision settings, which involves:
  - Implementation of a modular PromptLearner framework for integrating soft prompt tuning methods into GroundingDINO architectures.
  - Implementation of a modified loss to perform LoCoOp Local Regularization, adapted for object detection.
- Benchmarking of adapted prompt learning techniques for their suitability in automatic annotation pipelines under few-shot settings.
- Empirical validation on Roboflow-20VL FSOD dataset, demonstrating that prompt learning consistently improves pseudo-labeling quality.

Chapter 2

Related Work

2.1 The Evolution of Object Detection

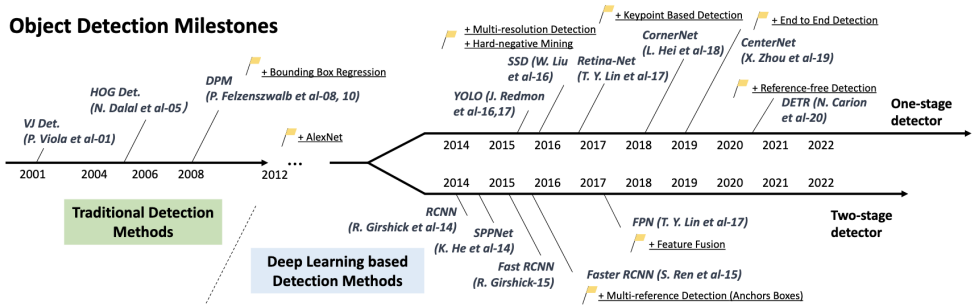

Figure 2.1: Object detection milestones from 2001 to 2022.

Object detection is a cornerstone of computer vision, enabling machines to interpret and interact with the visual world at a finer level of detail than simple classification. By localizing and identifying multiple objects within an image, detection provides the structural understanding required for numerous downstream applications, from autonomous driving and robotics to medical imaging and content moderation. Unlike image classification, which provides a global label, object detection offers spatially grounded predictions that bridge perception and action, forming the basis for higher-level reasoning tasks such as scene understanding and visual question answering.

2.1.1 Early Object Detection: Handcrafted Features

Before the rise of deep learning, object detection relied heavily on handcrafted features and statistical models. Early approaches focused on designing descriptors that captured discriminative visual cues, combined with classifiers such as Support Vector Machines (SVMs).

The Viola-Jones Detector [20] introduced the first real-time object detector, based on Haar-like features and a cascade of boosted classifiers. Although primarily successful for face detection, it demonstrated that detection could be practical at scale.

Histogram of Oriented Gradients (HOG) [21] became a widely used descriptor, leveraging edge orientation histograms to capture local shape. Coupled with SVMs, it achieved strong performance for pedestrian detection.

Deformable Part Models (DPMs) [22] extended this approach by modeling objects as collections of parts with geometric constraints, improving robustness to deformation and intra-class variability.

While these methods were foundational, they saturated in performance due to their limited representational power and reliance on hand-engineered features.

2.1.2 The Deep Learning Revolution: Two-Stage Detectors

The introduction of Convolutional Neural Networks (CNNs) marked a paradigm shift. Instead of relying on hand-engineered features, CNNs learned hierarchical visual representations directly from data. While the idea dates back to the 1990s [23], it only became feasible at scale with the availability of large datasets such as ImageNet [24] and the computational acceleration provided by modern GPUs, culminating in the breakthrough of AlexNet [25] in 2012.

Region Based Convolutional Neural Networks (R-CNNs) [26] pioneered CNN-based detection by applying selective search to generate region proposals, resizing each region, and classifying it with a CNN. While highly accurate, the approach was computationally expensive due to redundant feature extraction for overlapping regions.

Spatial Pyramid Pooling Networks (SPPNets) [27] addressed this inefficiency by computing a single feature map for the entire image and applying Spatial Pyramid Pooling to generate region features of arbitrary sizes, greatly improving efficiency.

Fast R-CNN [28] unified classification and bounding box regression into a

single network, enabling end-to-end training.

Faster R-CNN [3] introduced the Region Proposal Network (RPN), replacing selective search with a trainable, nearly cost-free proposal mechanism. This was the first near real-time detector.

Feature Pyramid Networks (FPNs) [29] improved multi-scale detection by combining high- and low-level features via lateral connections, enabling robust detection of objects at different scales.

Cascade R-CNN [30] further refined detection by applying multiple stages of progressively stricter IoU thresholds, reducing localization errors.

This family of methods established the two-stage detection paradigm, characterized by high accuracy but moderate speed.

2.1.3 One-Stage Detectors: Speed-Oriented Models

In parallel, researchers sought faster alternatives by eliminating the explicit proposal stage.

YOLO (You Only Look Once) [4] reframed detection as a single regression problem. The image was divided into a grid, with each cell directly predicting bounding boxes and class probabilities. This design enabled real-time inference but sacrificed some accuracy, particularly for small objects.

SSD (Single Shot MultiBox Detector) [31] improved performance by making predictions at multiple scales using feature maps from different convolutional layers.

RetinaNet [10] closed the accuracy gap with two-stage detectors by introducing focal loss, which down-weights easy negative examples and focuses training on hard positives. This addressed the severe class imbalance between foreground and background samples, allowing one-stage detectors to achieve competitive accuracy with significantly higher speed.

Anchor-Free Detectors emerged as an alternative to anchor-based approaches. CornerNet [32] predicted paired keypoints (top-left and bottom-right corners), while CenterNet [33] predicted object centers and sizes, simplifying detection and reducing hyperparameter tuning.

One-stage methods have since become dominant in scenarios requiring real-time inference or large-scale deployment.

2.1.4 The Transformer Era: DETR and Beyond

Despite these advances, CNN-based detectors still relied on complex components such as anchor generation, non-maximum suppression (NMS), and

region proposal networks (RPN). Thanks to the introduction of Transformers to vision, the need for hand-crafted components has been largely reduced through end-to-end architectures that model object relationships through attention.

DETR (Detection Transformer) [5] eliminated region proposals and anchor boxes altogether. Using a transformer encoder-decoder architecture, DETR modeled global context across the entire image and directly predicted a fixed set of objects via set-based bipartite matching using the Hungarian loss. This end-to-end design simplified training pipelines but required long convergence times and large datasets.

Subsequent improvements such as Deformable DETR [6] accelerated training by focusing attention on sparse, informative regions, while DN-DETR [7] improved stability using denoising queries.

Transformers thus opened a new research direction for object detection, shifting the focus from handcrafted priors to fully learnable architectures capable of modeling rich image-wide dependencies.

While DETR redefined how visual dependencies are modeled, the next breakthrough came from extending this reasoning to language itself, aligning visual and textual features to unlock open-vocabulary detection.

2.2 Foundation Models and Vision-Language Alignment

2.2.1 Representation of Information in Neural Networks

Neural networks work by transforming raw input data into internal representations, usually continuous vectors of a specific dimension, that encode the underlying information useful for a specific task. These representations are often latent: they're not directly interpretable, but they are optimized so that downstream tasks (classification, detection, retrieval, etc.) can use them effectively. Over time, researchers have put effort into designing or training networks so that these representations capture the aspects of input that are relevant for tasks while discarding irrelevant noise.

Based on which features of data are important, the information can be extracted through different methods, some of which were introduced in the previous chapter (e.g. CNNs, Transformer, etc.).

A particularly powerful way to formalize these learned representations is

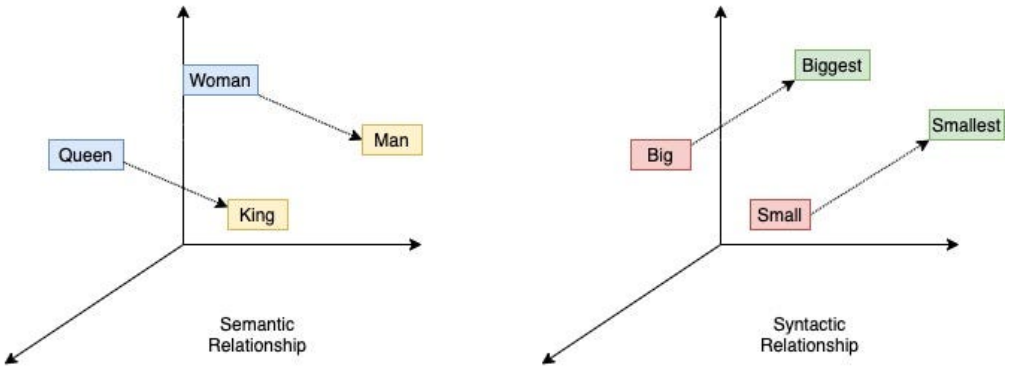

Figure 2.2: Visual representation of expected behaviour of the vector space of word embeddings.

through the concept of embeddings. Embeddings represent complex data, such as words, sentences, or images, as points in a continuous vector space, where geometric relationships (like distance or direction) capture semantic or structural similarity. Early work by Bengio et al. [34] introduced distributed word representations, paving the way for Word2Vec [35], which learned embeddings by predicting word contexts. Transformers such as BERT [36] extended this idea to contextual embeddings, dynamically adapting representations based on sentence meaning. In vision, as we will later discuss, models like CLIP [14] map both images and text into a shared embedding space, enabling cross-modal reasoning and zero-shot generalization.

Textual and Visual Representation

Textual representations encode semantic and syntactic properties of language in continuous vector spaces. These embeddings can be token-level (words, subwords) or sentence-level, and they capture both local and contextual information [34, 35, 36].

Visual representations encode images into feature vectors that preserve spatial and semantic content. CNNs traditionally extract hierarchical features, while attention-based transformers allow modeling of global dependencies across an image [37, 14].

By establishing robust representations for both modalities, models can then align text and image information, enabling tasks such as image captioning, text-to-image retrieval, and open-vocabulary object detection, which we will discuss in the next section.

2.2.2 From CLIP to Detection: Learning Transferable Visual Models from Natural Language Supervision

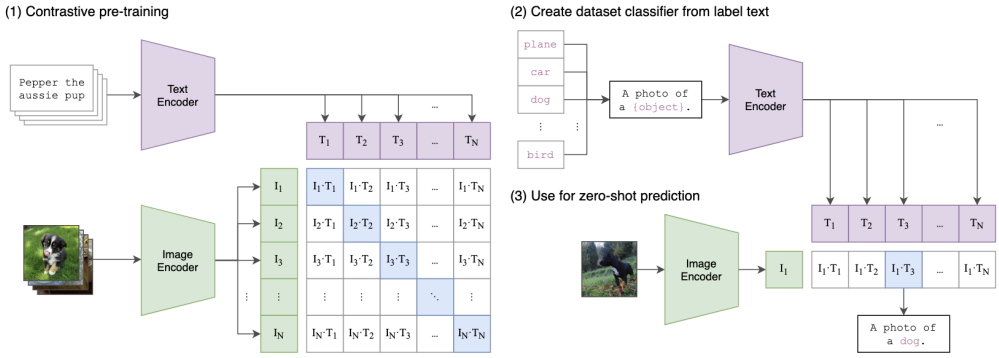

Figure 2.3: Overview of CLIP architecture.

A major leap in representation learning came with the introduction of CLIP (Contrastive Language-Image Pretraining) [14]. CLIP demonstrated that natural language could act as a rich and scalable source of supervision for visual representation learning, enabling models to generalize to new visual concepts without explicit task-specific fine-tuning.

CLIP: Model structure and innovation.

CLIP is based on a dual-encoder architecture composed of a text encoder and an image encoder, trained jointly to project both modalities into a shared embedding space. The text encoder is a Transformer similar to the one introduced in [37], while the image encoder can be either a Vision Transformer (ViT) [38] or a ResNet [39]. Unlike traditional supervised vision models that require labeled datasets with predefined categories, CLIP is trained on hundreds of millions of image-text pairs collected from the web. Each pair provides a weak but natural form of supervision, as the text roughly describes the corresponding image. The image-text pairs are used to connect visual concepts with language, an approach known as grounding.

CLIP: Alignment of textual and visual features.

The key innovation of CLIP lies in its training objective, a symmetric contrastive loss that aligns visual and textual embeddings. Given a batch of N

image-text pairs, the model computes the cosine similarity between all image and text embeddings, encouraging the similarity of matched pairs (v_i, t_i) to be higher than that of mismatched ones. Formally, this is achieved using a cross-entropy loss over the similarity matrix:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left(\log\frac{\exp(\text{sim}(v_i,t_i)/\tau)}{\sum_{j=1}^{N}\exp(\text{sim}(v_i,t_j)/\tau)} + \log\frac{\exp(\text{sim}(t_i,v_i)/\tau)}{\sum_{j=1}^{N}\exp(\text{sim}(t_i,v_j)/\tau)}\right)$$

where $\text{sim}(\cdot)$ denotes cosine similarity and τ is a learnable temperature parameter. This bidirectional contrastive objective ensures that both modalities learn a common semantic structure, making it possible to retrieve images given text, and vice versa.

From alignment to visual recognition and detection.

Once trained, CLIP can perform zero-shot classification by comparing the embedding of an image with embeddings of class descriptions written in natural language. This alignment between text and vision opened the path toward open-vocabulary models, which can recognize classes not seen during training, provided that textual descriptions are available.

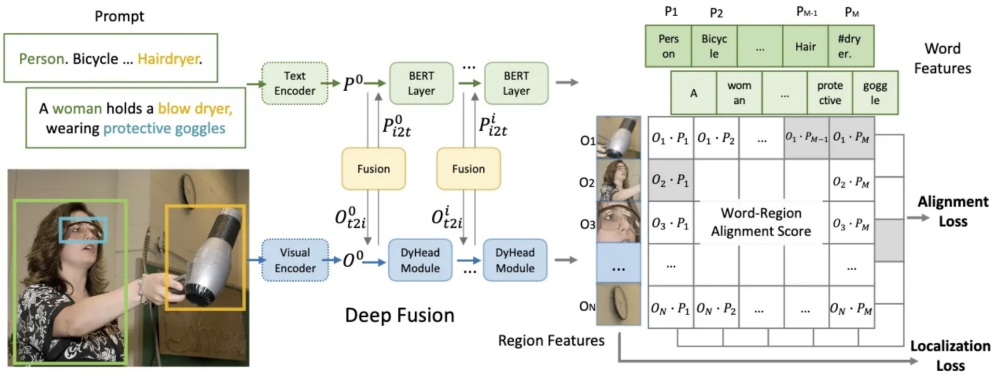

Figure 2.4: Overview of GLIP architecture.

The success of CLIP has inspired several extensions that adapt its architecture to downstream tasks beyond classification, including object detection and segmentation. Notably, GLIP (Grounded Language-Image Pretraining) [40] proposed a unified framework that integrates grounding and detection tasks into a single pretraining objective. By aligning region-level visual features with text spans in captions and introducing modality fusion layers,

GLIP effectively brings CLIP's global alignment principle to a spatial level, enabling the model to both localize and name objects in an open-vocabulary setting.

Following this direction, GroundingDINO [17] further refined the text-image alignment mechanism by combining the DINO [41] detection backbone with introduction of language for unseen object generalization.

These advancements represent a natural evolution of CLIP's philosophy, from global semantic matching to fine-grained, region-level understanding, ultimately bridging image-level representation learning with object detection.

Extending this principle to object detection required combining visual backbones with language-conditioned detection heads. The resulting architectures allow users to "prompt" the detector with arbitrary text queries, enabling prompt-based detection of unseen objects.

2.2.3 GroundingDINO: Language-Guided Open-Vocabulary Detection

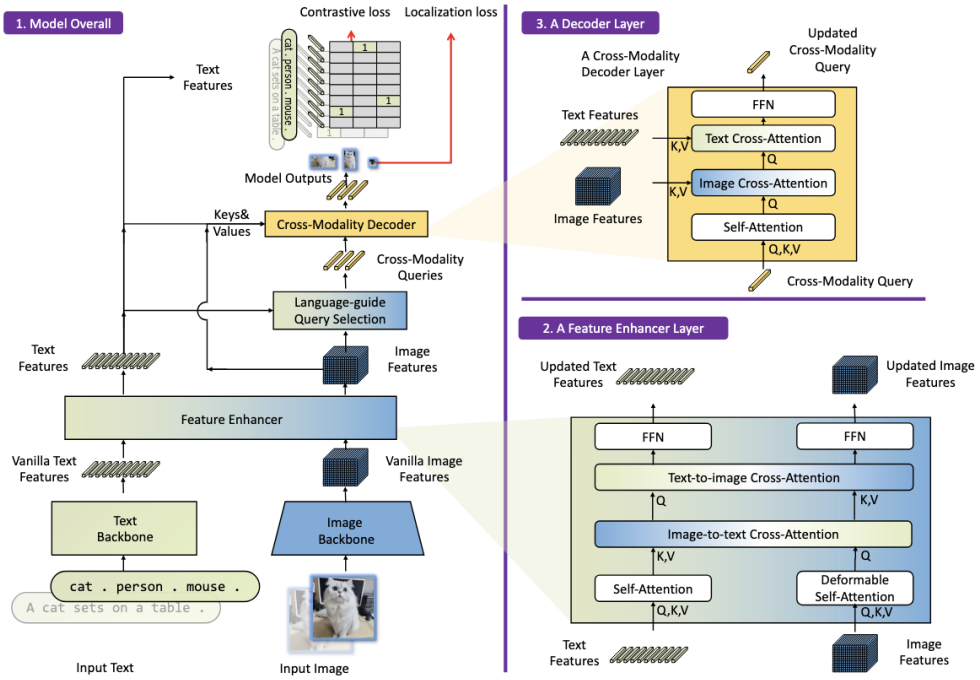

Figure 2.5: Overview of GroundingDINO architecture.

Building on the principles of CLIP and GLIP, GroundingDINO [17] represents one of the most significant advances in text-conditioned object detection. It extends the DINO object detector [41] by tightly coupling visual and textual modalities throughout the detection pipeline, enabling promptable and open-vocabulary detection. The model is capable of localizing and identifying arbitrary objects or phrases described by natural language prompts, even if those categories were never seen during training.

Model architecture

GroundingDINO inherits the transformer-based detection structure of DINO but introduces a multimodal fusion mechanism that allows textual information to directly guide the detection process. Its architecture can be divided into three principal modules:

1. **Feature Enhancer:** This module fuses multi-scale visual features from a Swin Transformer backbone with text embeddings produced by a Transformer text encoder, through different feature-enhancer layers. Each layer is composed by a deformable self-attention to enhance image features and the vanilla self-attention for text feature enhancers. Then, inspired by GLIP [40], an image-to-text and a text-to-image cross-attention module for feature fusion is added. This tight fusion ensures that features become aware of the other modality before entering the detection stage.

2. **Language-Guided Query Selection:** The goal of this module is to ensure that the decoder focuses on image regions that are semantically relevant to the input text.

   Let the encoded image features be $X_I \in \mathbb{R}^{N_I \times d}$ and the text features be $X_T \in \mathbb{R}^{N_T \times d}$, where d is the shared embedding dimension. To initialize the decoder, a fixed number of $N_q = 900$ queries must be selected from the image features. Instead of sampling them uniformly or randomly, GroundingDINO selects those that are most relevant to the textual description.

   This relevance is measured by computing a similarity matrix $X_I X_T^\top$ between image and text features. The maximum similarity score across all text tokens is taken for each image token, and the top N_q tokens are selected according to:

$$I_{N_q} = \mathrm{Top}_{N_q}\left(\max_{-1}(X_I X_T^\top)\right), \tag{2.1}$$

where Top_{N_q} extracts the indices of the most text-correlated image features. The features corresponding to these indices are then used to initialize the decoder queries.

Following DINO [41], each query is composed of two components: a positional part and a content part. The positional part encodes spatial information through dynamic anchor boxes derived from the encoder outputs, while the content part consists of learnable embeddings that are refined during training. By selecting queries in this text-aware manner, the model ensures that decoding begins from visually and semantically grounded regions, effectively bridging the gap between image and language modalities.

3. **Cross-Modality Decoder:** The decoder jointly processes the language-guided queries and multimodal features through a series of cross-attention and self-attention layers. Each query predicts both a bounding box and a textual similarity score, determining which text token or phrase best describes the localized object.

Training objective

GroundingDINO is trained end-to-end with a combination of detection and grounding objectives, designed to jointly optimize spatial localization and semantic alignment. Following previous DETR-like works [5, 42, 41], the model employs an ℓ_1 loss and a Generalized IoU (GIoU) loss [43] for bounding box regression, ensuring accurate localization of detected objects.

For classification, GroundingDINO adopts a contrastive loss inspired by GLIP [40], aligning predicted visual queries with the corresponding language tokens. Specifically, each decoder query is dot-multiplied with the text feature embeddings to produce logits over text tokens, after which a focal loss [10] is computed to handle class imbalance and emphasize harder examples.

A Hungarian bipartite matching strategy is used to associate each predicted box with its corresponding ground-truth object, based on both classification and regression costs. The final loss combines all these components as follows:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{cls}}\,\mathcal{L}_{\text{cls}} + \lambda_{\ell_1}\,\mathcal{L}_{\ell_1} + \lambda_{\text{giou}}\,\mathcal{L}_{\text{giou}},$$

where $\mathcal{L}_{\text{cls}}$ denotes the classification loss (computed via focal loss on text-aligned logits), $\mathcal{L}_{\ell_1}$ is the ℓ_1 regression loss on bounding box coordinates, and

$\mathcal{L}_{\text{giou}}$ represents the GIoU loss. The λ coefficients balance the contribution of each component.

The matching between predicted boxes $\hat{y}_i$ and ground-truth objects y_j is determined by minimizing a cost function that combines both localization and classification terms:

$$\mathcal{L}_{\text{match}}(i,j) = \lambda_{\text{cls}}\,\mathcal{C}_{\text{cls}}(\hat{y}_i, y_j) + \lambda_{\text{box}}\mathcal{C}_{\text{bbox}}(\hat{y}_i, y_j) + \lambda_{\text{giou}}\,\mathcal{C}_{\text{giou}}(\hat{y}_i, y_j).$$

Once the optimal bipartite assignment is computed using the Hungarian algorithm, the final loss is evaluated only on the matched pairs. As in DETR-like architectures, auxiliary losses are added after each decoder layer and from the encoder outputs to stabilize training and improve convergence.

This unified optimization framework ensures that the model not only localizes objects accurately but also learns to ground textual concepts within the visual domain, effectively bridging the gap between phrase grounding and open-vocabulary object detection.

Promptable and open-vocabulary capabilities

A key property of GroundingDINO is that it is inherently promptable: at inference time, the model accepts arbitrary text prompts that directly modulate the detection process. Because class names or descriptive phrases are encoded as text embeddings, the model generalizes naturally to unseen object categories, achieving zero-shot detection. This design represents a shift from closed-set recognition to open-ended, language-driven reasoning over visual content.

Performance and results

GroundingDINO exhibits strong performance across a range of zero-shot and referring-expression tasks. On COCO zero-shot transfer, it achieves **52.5 AP** for object detection, while on the ODinW benchmark (Object Detection in the Wild) it reports a mean AP of **26.1** across unseen domains. It also demonstrates excellent phrase grounding ability on datasets such as RefCOCO, RefCOCO+, and RefCOCOg. Later versions (GroundingDINO 1.5 and 1.6) further improved zero-shot COCO performance to **54.3 AP** and **55.4 AP**, respectively, by scaling model size and training on larger grounding datasets [44].

Compared to earlier CLIP-based adaptations for detection, GroundingDINO achieves tighter text-image integration by using linguistic features to train

the detection transformer. The combination of early fusion, language-guided queries, and cross-modal decoding make possible to reach spatial precision and semantic alignment. This allows a single model to seamlessly handle detection, grounding, and referring expression comprehension in a unified manner. In this sense, GroundingDINO marks a key step in the evolution from global vision-language alignment (as in CLIP) to fine-grained, region-level understanding required by open-vocabulary detection.

2.2.4 YOLOE: Real-Time Seeing Anything

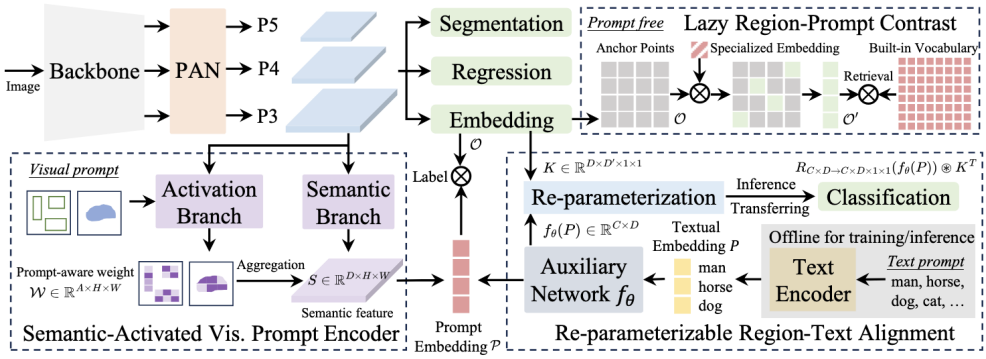

Figure 2.6: Overview of YOLOE architecture.

In contrast with GroundingDINO, YOLOE [16] represents a CNN-based approach to open-vocabulary detection, retaining YOLO's efficiency while incorporating promptable modules.

YOLOE extends the traditional YOLO detection framework to the open-vocabulary setting by introducing a unified representation that bridges visual and linguistic modalities. Unlike standard YOLO architectures, which rely on a fixed set of predefined classes, YOLOE learns a shared embedding space where both visual and textual features coexist, similarily to what we've seen with GroundingDINO. This design enables the model to recognize and localize novel object categories at inference time, guided solely by textual prompts.

Architecture Overview

YOLOE maintains the core one-stage detection paradigm of YOLO, ensuring real-time performance while introducing a text-conditioned similarity head in place of the conventional classification layer. Given visual features $F_v \in$

$\mathbb{R}^{H\times W\times d}$ extracted by the YOLO backbone and text embeddings $F_t \in \mathbb{R}^{C\times d}$ derived from a pretrained vision-language model such as CLIP, the model computes class scores by measuring the similarity between the two modalities:

$$S = \sigma\left(\frac{F_v F_t^\top}{\tau}\right), \tag{2.2}$$

where τ is a learnable temperature parameter and σ denotes the sigmoid or softmax activation. This formulation replaces the fixed classification layer with a flexible similarity computation, allowing YOLOE to generalize seamlessly to new categories described in natural language.

Training Objective

YOLOE is trained using a dual-branch objective that unifies traditional supervised detection and language-vision alignment. The supervised detection branch employs standard YOLO losses, including objectness prediction, bounding box regression, and confidence scoring. The language alignment branch, on the other hand, introduces a contrastive objective between visual region embeddings and their corresponding text embeddings, following the principles of CLIP-style supervision. This hybrid optimization allows YOLOE to learn both fine-grained localization and semantic alignment between image regions and linguistic concepts.

Open-Vocabulary Detection

At inference time, YOLOE performs open-vocabulary detection by simply replacing the category-specific classifier weights with text embeddings of arbitrary labels. The model thus predicts the similarity between visual regions and the given textual prompts, enabling zero-shot detection of unseen classes without any retraining or architectural modification. This approach makes YOLOE capable of both closed-vocabulary detection (on predefined categories) and open-vocabulary detection (on arbitrary textual queries) within a single unified framework.

Performance and Advantages

YOLOE achieves strong zero-shot generalization across datasets such as COCO and LVIS while maintaining the computational efficiency characteristic of YOLO-based models. It delivers performance comparable to transformer-based open-vocabulary detectors like GroundingDINO, but with substantially lower computational overhead. By integrating text-conditioned reasoning into a one-stage framework, YOLOE represents a significant step toward efficient and practical open-vocabulary object detection.

2.3 Prompt Learning for Vision-Language Models

As anticipated in Section 1.2, while foundational models like GroundingDINO exhibit remarkable zero-shot performance, proving their generalization ability, the learned semantic isn't enough to be effective among all domains. For these reason, several methods have been proposed to specialize these models on challenging datasets, aiming to preserve their generalization capability while effectively leveraging the rich knowledge acquired during pretraining while keeping training time acceptable.

One of these techniques is prompt learning, that originated in the field of Natural Language Processing (NLP) as a parameter-efficient alternative to full fine-tuning of large-scale language models. Instead of updating billions of parameters, prompt learning introduces soft prompts, learnable continuous vectors prepended or appended to token embeddings, that steer the model toward a downstream task [45, 46]. This approach enables rapid adaptation with minimal supervision and a drastically reduced number of trainable parameters, preserving the generalization ability of pretrained models while improving task specialization.

2.3.1 The Power of Scale

Lester et al. [46] demonstrated that prompt tuning remains highly effective even when tuning only a small number of parameters, provided that the underlying model is large enough. Their experiments across T5 [47] variants revealed a scaling law: the performance gap between prompt tuning and full fine-tuning diminishes as model size increases (Figure 2.7). This finding established prompt tuning as a scalable and robust strategy for large foundation models, capable of rivaling full fine-tuning while training a much lower

number of parameters.

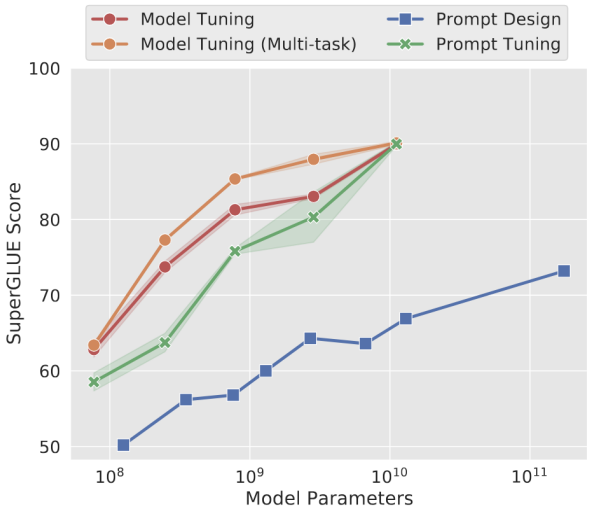

Figure 2.7: Visualization of prompt tuning scaling law.

2.3.2 Theoretical Motivation

From a theoretical perspective, prompt learning can be seen as learning in the input space rather than in the parameter space. The foundation model parameters θ remain fixed, while the prompt vectors P act as trainable inputs that condition the model's internal behaviour:

$$h = f_\theta(P, x), \tag{2.3}$$

where x denotes the original input (e.g., a textual description or an image embedding), and P encodes task-specific knowledge. By optimizing P, the model can adapt its internal activation patterns to new domains or tasks while maintaining the semantic structure learned during large-scale pretraining. This approach preserves the generalization of foundation models while introducing controllable, interpretable task conditioning.

Prompt learning is also related to the concept of in-context learning, where models are guided by examples or textual cues rather than explicit parameter updates. Unlike discrete text prompts, however, soft prompts are continuous and differentiable, allowing efficient optimization via gradient descent.

2.3.3 Prompt Learning in Vision-Language Models

In the context of vision-language models, prompt learning primarily targets the textual encoder. Since models like CLIP align image and text representations in a shared embedding space, modifying the text side via learned prompts effectively alters how the model interprets visual categories. This allows the model to adapt to new domains or tasks without altering the pretrained weights of the backbone.

Early works such as CoOp [18] introduced context optimization, where learnable prompt tokens replace manually designed text templates (e.g., "a photo of a [CLASS]"). This enables the model to discover optimal context representations for each class. Building on this idea, LoCoOp [19] proposed local regularized context optimization, adding a loss term in order to train prompts that adaptively capture possible intra-class nuisances. By optimizing prompts, LoCoOp achieves stronger robustness under the same CoOp conditions.

2.3.4 CoOp: Context Optimization

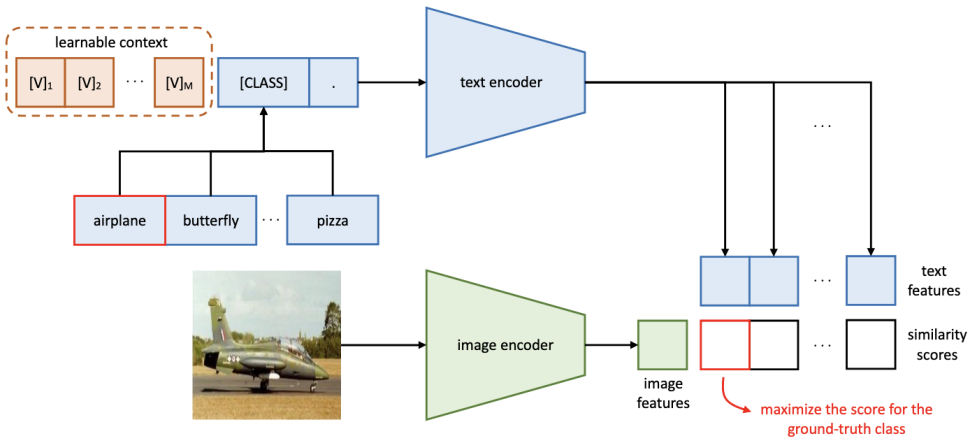

Figure 2.8: Overview of CoOp architecture.

Context Optimization (CoOp) [18] is one of the pioneering works that introduced prompt learning into the vision-language domain, extending techniques originally developed for Natural Language Processing (NLP) to models such as CLIP. Its core motivation lies in overcoming the inefficiency and suboptimality of manual prompt engineering, while enabling efficient adaptation of large pre-trained vision-language models to downstream datasets.

Motivation

Models like CLIP leverage hand-crafted textual templates such as "a photo of a [CLASS]" to encode class names into the shared vision-language embedding space. While this design allows zero-shot generalization to unseen categories, these manually designed prompts are not necessarily optimal for downstream datasets with domain shifts or task-specific nuances. The process of designing textual templates is labor-intensive, requires domain knowledge, and cannot guarantee performance consistency across datasets.

Core idea

CoOp replaces the fixed natural language context in textual templates with a set of learnable continuous vectors, also known as soft prompt tokens, which can be optimized end-to-end using a small number of labeled samples. These vectors serve as continuous substitutes for words in the textual prompt, allowing the model to automatically discover the most informative contextual cues that align textual and visual features.

Formally, let $f_V(\cdot)$ and $f_T(\cdot)$ denote the frozen CLIP image and text encoders, respectively. In CLIP, the textual representation of a class c is typically obtained by encoding a template such as:

$$t_c = \text{"a photo of a [CLASS]"},$$

and computing the corresponding text feature $z_c = f_T(t_c)$.

In CoOp, instead of relying on the manually fixed phrase "a photo of a" as prefix to guide classification, the template is parameterized by M learnable context vectors $P = \{v_1, v_2, \ldots, v_M\}$, each of dimension d (matching the token embedding dimension of the text encoder, i.e. the vector representation of the word). The method involve concatenating those vectors to token embeddings of the prompt, thus for class c we have:

$$e_c = [v_1, v_2, \ldots, v_M, \text{embed}(c)],$$

where embed(c) is the embedding of the class name token. These learnable tokens v_i are jointly optimized with respect to the downstream classification objective, while the text encoder f_T and image encoder f_V remain frozen.

Training objective

Given an image x and its corresponding label y, the CLIP-style similarity score between the visual feature $f_V(x)$ and the textual feature $f_T(t_c)$ for class

c is computed via cosine similarity:

$$\text{sim}(x, c) = \frac{f_V(x)^\top f_T(t_c)}{\|f_V(x)\| \, \|f_T(t_c)\|}.$$

The model's prediction probability is given by:

$$p(y|x) = \frac{\exp(\text{sim}(x, y)/\tau)}{\sum_{c=1}^{C} \exp(\text{sim}(x, c)/\tau)},$$

where τ is a learnable temperature parameter. The cross-entropy loss over the training dataset $\mathcal{D} = \{(x_i, y_i)\}$ is minimized to optimize the context vectors:

$$\mathcal{L}_{\text{CoOp}} = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i|x_i).$$

During training, gradients are propagated through the text encoder to the learnable prompt vectors v_i, enabling them to capture the contextual semantics that maximize alignment with visual features for the target dataset.

Parameter efficiency

A key advantage of CoOp is that only the context vectors P, typically a few dozen parameters, are trainable, while the multimodal backbone remains frozen. This drastically reduces computational cost and prevents catastrophic forgetting of the pre-trained knowledge. In practice, CoOp achieves significant performance gains over manual prompts using as few as 16 training samples per class, highlighting its strong data efficiency.

Context modalities

CoOp can be implemented in two variants:

- **Shared-context:** The same context vectors P are shared across all classes, enforcing a common embedding template that captures global contextual information.
- **Class-specific context:** Each class c has its own context set $P_c = \{v_{c,1}, \ldots, v_{c,M}\}$, allowing the model to learn finer, class-dependent textual cues. This variant provides greater flexibility but at the cost of additional parameters.

Empirically, class-specific CoOp tends to achieve higher accuracy on the seen (base) classes but may overfit, leading to weaker transfer to unseen categories. Shared-context CoOp, while less expressive, shows more robust generalization.

2.3.5 LoCoOp: Local Regularized Context Optimization

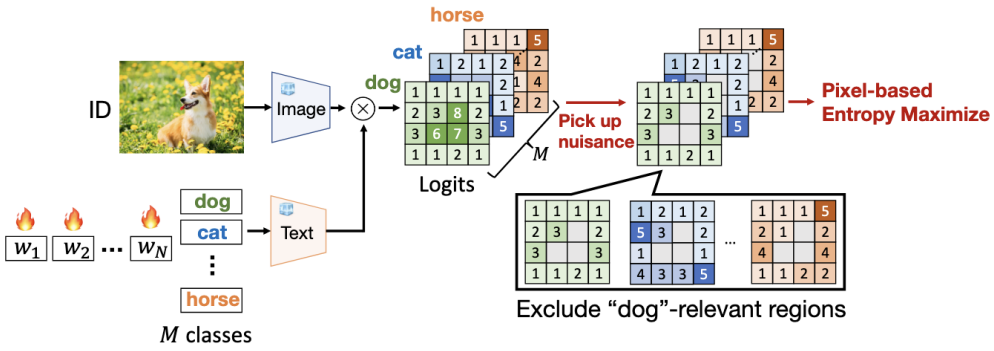

Figure 2.9: Overview of LoCoOp architecture [19]. CLIP local features from background regions are treated as pseudo-OOD samples during training, enforcing separation between in-distribution (ID) and OOD representations.

Local Regularized Context Optimization (LoCoOp) [19] was proposed to overcome some CoOp limitations. It introduces an Out-of-Distribution (OOD) regularization mechanism within the prompt learning framework, enabling CLIP to avoid overfitting on few-shot distributions.

Motivation

While CoOp [18] introduced learnable prompts that significantly improved few-shot adaptation efficiency, its formulation remains limited when applied to OOD data. CoOp optimizes prompt embeddings so that image features and class text embeddings of In-Distribution (ID) samples are closely aligned. However, this training objective also causes the model to inadvertently associate ID text embeddings with ID-irrelevant visual regions (e.g., backgrounds or co-occurring objects). As a result, the learned text features contain nuisance information, leading the model to assign artificially high confidence to OOD images that share similar background cues.

For this reason, LoCoOp method try to expoit ID-irrelevant regions to make the learned prompt nuisances-aware. To achieve this, instead of relying on external OOD data, LoCoOp leverages the local features of CLIP itself, particularly those corresponding to background or object-irrelevant regions, as pseudo-OOD features during training. By explicitly pushing these nuisance features away from all ID text embeddings, LoCoOp purifies the learned prompts, yielding cleaner and more discriminative representations for OOD detection.

Core idea

CLIP classification process relies on a global feature vector of the image, thereby discarding locality. LoCoOp idea is to leverage this local information in order to improve performance. For each local region, there should be a way to extract an indication of its ID-irrelevancy. The method relies on a simple yet effective assumption: a local region is likely an ID-irrelevant region if, based on its similarities with textual features describing the class, the ground truth class logit doesn't figure among the top-K values. In this way, using a strong foundational model, LoCoOp trusts the model and its zero-shot capability, assuming that it's capable at least to detect ID-irrelevancies, assigning lower scores to those patches that represent background information.

Training objective

LoCoOp builds upon CoOp's formulation, where an image encoder $f_V(\cdot)$ and a text encoder $f_T(\cdot)$ from CLIP are kept frozen, and a set of learnable context vectors $P = \{v_1, v_2, \ldots, v_M\}$ is optimized. Similarily to what we've seen with CoOp in Section 2.3.4 the classification probability for an ID image x^{in} and its label y is computed as:

$$p(y|x^{\text{in}}) = \frac{\exp(\text{sim}(x^{\text{in}}, y)/\tau)}{\sum_{c=1}^{C} \exp(\text{sim}(x^{\text{in}}, c)/\tau)},$$

, and the corresponding loss is the standard cross-entropy:

$$\mathcal{L}_{\text{CoOp}} = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i|x_i^{\text{in}}).$$

To enhance OOD robustness, LoCoOp introduces a regularization term. First, local region features $f_V(x_j^{\text{in}})$ are extracted from CLIP's feature map

and projected into the text embedding space. For each image x^{in}, regions that are irrelevant to the ID object, that is, whose ground-truth class does not appear among the top-K predicted text similarities, are collected into a set J:

$$J = \{j : \text{rank}(p_j(y|x_j^{\text{in}})) > K\}.$$

For each such region $j \in J$, LoCoOp maximizes the entropy of its predicted class distribution, encouraging it to be dissimilar from all ID text embeddings. This is achieved by maximizing the average of all the ID-irrelevant regions entropy in the image:

$$\mathcal{L}_{\text{OOD}} = -\frac{1}{N}\sum_{j}^{J} H(p_j),$$

where $H(\cdot)$ denotes the Shannon entropy. This operation effectively teaches the model that ID-irrelevant features (e.g., backgrounds) should not correlate with any known class representation.

The final training objective combines the standard CoOp loss with the OOD regularization term:

$$\mathcal{L}_{\text{LoCoOp}} = \mathcal{L}_{\text{CoOp}} + \lambda\,\mathcal{L}_{\text{OOD}},$$

where λ controls the strength of the OOD regularization.

This regularization scheme allows LoCoOp to better adapt to OOD conditions without requiring external datasets, by pushing background (pseudo-OOD) features away from class text embeddings. Consequently, LoCoOp reduces overconfidence on unseen data and improves separation between ID and OOD regions.

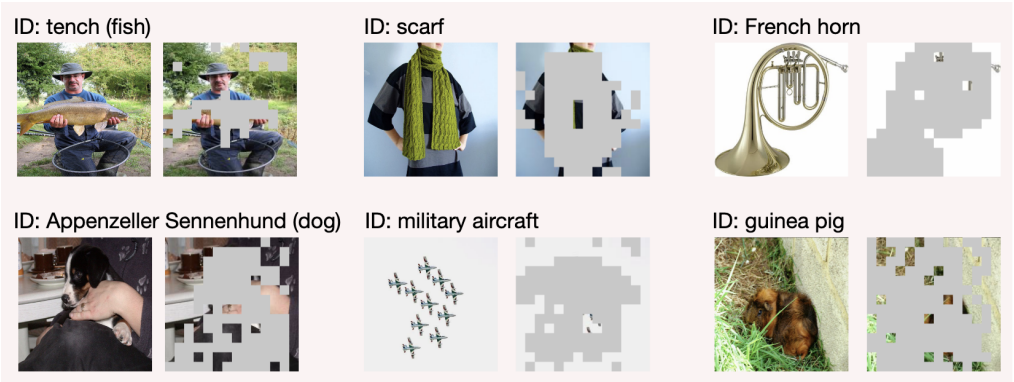

Figure 2.10: LoCoOp extracted ID-irrelevant regions.

CoOp vs LoCoOp

The results of the work, show that LoCoOp is better at imporoving the model performance with respect to CoOp, proving that regularization is beneficial for the prompt tuning process. Furthermore, its overhead is not a concern, since it takes only 1.4x training time compared to CoOp. The only downside is that LoCoOp introduces a new hyperparameter: the K parameter influence directly how ID-irrelevant regions are calculated, and it has to be carefully tuned to get the best results by the model.

2.3.6 Prompt Learning in Object Detection

While CoOp and LoCoOp were initially designed for image classification with CLIP, their principles naturally extend to other promptable models.

Foundational object detection models like GroundingDINO provide strong open-vocabulary detection backbones, but without adaptation, they may yield noisy pseudo-labels. By integrating CoOp and LoCoOp into GroundingDINO, this work aims to enhance the model's annotation quality under limited supervision, bridging the gap between foundational models and practical use cases.

The following chapter establishes the technical foundation for the proposed methodology, showing how advances in object detection, foundational models, and prompt learning enable better performance through efficient model tuning, thereby paving the way for automatic, low-cost dataset annotation.

Chapter 3

Methodology

3.1 Overview of the Proposed Methodology

What follows is the results of the extentions of Context Optimization (CoOp) [18] and Local Context Optimization (LoCoOp) [19] to the task of open-vocabulary object detection using the GroundingDINO [17] framework. These textual prompt learning strategies, originally developed for classification, are adapted to guide region-level detection. Figure 3.1 provides a high-level overview of the system architecture.

3.1.1 System Architecture Overview

As detailed in Section 2.2.3, GroundingDINO extends the DINO [41] detector by tightly coupling image and text modalities throughout the detection pipeline, enabling language-guided and open-vocabulary detection. Its transformer-based architecture fuses visual and textual features across multiple stages, allowing natural language prompts to influence decoding in the final layers of the model.

In this work, we extend GroundingDINO by introducing a `PromptLearner` module positioned between the text encoder and the multimodal fusion blocks. This component produces a set of learnable prompt embeddings $P \in \mathbb{R}^{n_{ctx} \times d}$ that act as contextual tokens conditioning the text encoder on the downstream dataset. Rather than relying solely on static textual representations such as tokenized class names, the `PromptLearner` generates additional vectors that are optimized to adapt the language branch result to the target detection domain.

For each class c, the resulting prompt-conditioned textual embeddings

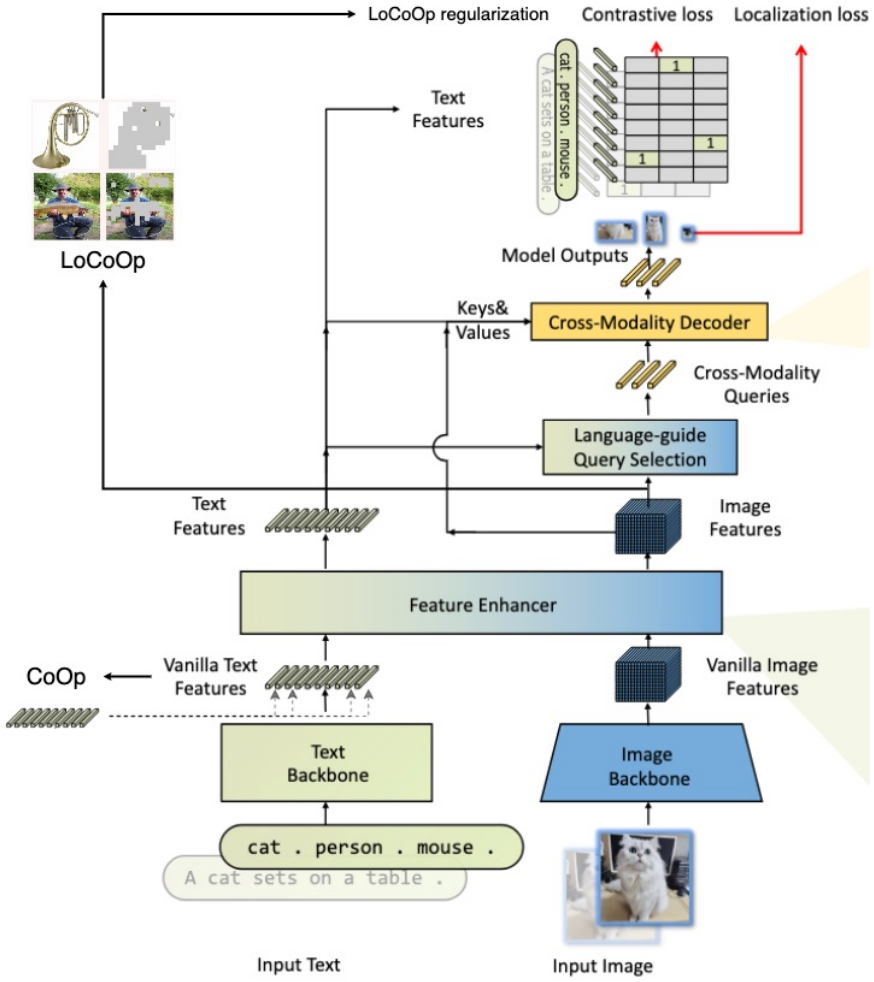

Figure 3.1: Overview of the proposed pipeline integrating CoOp and LoCoOp into GroundingDINO.

are processed by the other layers of the model, where they interact with the visual tokens extracted by the visual backbone through cross-attention layers. This integration allows the model to perform grounded detection guided by optimized textual context instead of static labels, improving adaptability under limited supervision or domain shifts.

The following sections describe how this architectural extension is specialized through the two previously introduced prompt learning strategies, CoOp and LoCoOp, and how they are adapted to the GroundingDINO framework for object detection.

3.2 Adaptation of CoOp to Object Detection in GroundingDINO

Theoretical Aspects

The original CoOp [18] method learns a set of continuous context tokens that condition the textual feature of the description of the image in order to improve classification precision. To adapt this to object detection, we treat the prompt embeddings as learnable modifiers of the class textual features extracted by GroundingDINO text backbone.

The learned prompts are prepended to the text encoder output of the class name, determining the first difference with the original CoOp, that learned the prompt at the token embedding level. Formally, for each class c, we concatenate n_{ctx} learnable context vectors $P = \{P_1, P_2, \ldots, P_{n_{ctx}}\}$ with the text encoder output, forming the final prompt embedding:

$$X_{T_c} = [\, P_1, P_2, \ldots, P_{n_{ctx}}, f_{\mathrm{T}}(c) \,]. \tag{3.1}$$

Here, $f_{\mathrm{T}}(\cdot)$ denotes the text encoder that maps the class description to its embedding space, and n_{ctx} is the number of context vectors to optimize. By indicating with n_c the number of tokens produced by the tokenizer for the class name c, $X_{T_c} \in \mathbb{R}^{(n_{ctx}+n_c)\times d}$ represents the final prompt-conditioned textual embedding used by the model to represent the class description.

The learnable context vectors can be either shared among all classes or unique for each class. In the first case, the model optimizes a parameter matrix $P \in \mathbb{R}^{n_{ctx}\times d}$, while in the second case $P \in \mathbb{R}^{(n_{ctx}\cdot n_{classes})\times d}$.

The context vectors are initialized from a normal distribution and optimized jointly with the detection objective, determining the another shift from original CoOp formulation. Indeed, instead of using a different loss with respect to the model in which it's used, the learning process is tailored naturally to the original model loss. In this way the prompts are optimized not only for classification, but even for bounding box regression, using the following loss:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{cls}}\, \mathcal{L}_{\text{cls}} + \lambda_{\ell_1}\, \mathcal{L}_{\ell_1} + \lambda_{\text{giou}}\, \mathcal{L}_{\text{giou}}$$

, where $\mathcal{L}_{\text{cls}}$ denotes the Focal Binary Cross Entropy, and its defined as:

$$\begin{aligned}
\mathrm{BCE}(z, y) &= -[y \log \sigma(z) + (1 - y) \log(1 - \sigma(z))] \\
p_t &= y\sigma(z) + (1 - y)(1 - \sigma(z)) \\
\alpha_t &= y\alpha + (1 - y)(1 - \alpha) \\
\ell_{i,q,c} &= \alpha_t \mathrm{BCE}(z_{i,q,c}, y_{i,q,c})(p_{t,i,q,c})^\gamma \\
\mathcal{L}_{\mathrm{cls}} &= \frac{1}{N_+} \sum_{i,q,c} \ell_{i,q,c}
\end{aligned}$$

; the term $\mathcal{L}_{\ell_1}$ is the ℓ_1 regression loss on bounding box coordinates, defined as:

$$\mathcal{L}_{\ell_1} = \frac{1}{N} \sum_{i,q} ||b_{i,q} - \hat{b}_{i,q}||_1$$

; while the last term $\mathcal{L}_{\mathrm{giou}}$ represents the GIoU loss, defined as:

$$\mathcal{L}_{\mathrm{GIoU}} = \frac{1}{N} \sum_{i,q} \Big(1 - \mathrm{GIoU}(b_{i,q}, \hat{b}_{i,q})\Big),$$

$$\mathrm{GIoU}(b_{i,q}, \hat{b}_{i,q}) = \frac{|b_{i,q} \cap \hat{b}_{i,q}|}{|b_{i,q} \cup \hat{b}_{i,q}|} - \frac{|C_{i,q}| - |b_{i,q} \cup \hat{b}_{i,q}|}{|C_{i,q}|},$$

Finally, the prompt-conditioned textual embeddings X_{T_c} are fused with the visual tokens extracted by the vision backbone through the following layers of the network, enabling grounded detection conditioned on textual prompts.

Motivation

The expected results of this adaptation is that prompts optimize to condition the textual branch in order to align better to the visual features of the specific dataset in exam. Since the training uses the same objective as the original model, it should learn not only to classify better the predicted bounding boxes but also to localize object more accurately. Moreover, appending prompts after the textual backbone should mimic the behaviour of a text embedding of a token, while removing a possible source of signal attentuation that is the textual backbone itself, concentrating more in the network fusion part.

Implementation Details

The `PromptLearner` is implemented as a learnable embedding layer initialized from a Normal distribution with $\mu = 0$ and $\sigma = std$ What the `PromptLearner` do is to take the encoded text from the text backbone of the

model, and, for each encoded token describing a specific class, it insert at its beginning the vectors of the context (the same for all classes if the context is unified, while a different context for each class if not). The pseudocode at Algorithm 1 outlines the class responsible for the modification of the texutal prompt.

Algorithm 1 PromptLearner

Input: $n_{\text{ctx}}, ctx_{\text{dim}}, csc, n_{\text{cls}}, std, des\_norm$
if csc is True **then**
    $ctx \sim \mathcal{N}(0, std^2)[n_{\text{ctx}} \times n_{\text{cls}}, ctx_{\text{dim}}]$
else
    $ctx \sim \mathcal{N}(0, std^2)[n_{\text{ctx}}, ctx_{\text{dim}}]$
end if
if des_norm defined **then**
    Normalize each ctx vector to norm = des_norm
end if

function FORWARD($embeddings, position\_ids, add\_ctx$)
    for each idx z where $position\_ids = 0$ (excluding [CLS], [SEP]) **do**
        Insert n_{ctx} before z the context vectors
    end for
    return modified embeddings
end function

As we outlined in Section 3.2, training follows the same objective as the base model, using the GroundingDINO loss to propagate gradients backward in order to optimize prompt values.

3.3 Adaptation of LoCoOp to Object Detection in GroundingDINO

Theoretical Aspects

LoCoOp [19] extends CoOp by introducing a regularization loss that discourages spurious correlations between object appearance and background. We follow the original formulation, adapting it to the bounding-box level.

Specifically, we add the LoCoOp entropy-based regularization term to the

loss, but this time it's computed over the image regions enclosed by each predicted bounding box. The ground truth label is extracted through Hungarian matching algorithm, that uses a cost function described in Section 2.2.3 to find a one-to-one matching between predicted and ground truth bounding boxes. The local regions cosine similarities between visual and textual features are then taken into consideration to regularize against possible nuisances inside the bounding box.

If the region-text similairy of the ground truth text description of the class do not rank within the top-K highest values, then the region is considered to be an ID-irrelevant one, and used for the regularization in the loss.

$$J = \{j : \text{rank}(p_j(y|x_j^{\text{in}})) > K\}$$

Given predicted probability distribution p_j for non-top$K$ local regions j, for each matched bounding box q we have:

$$\mathcal{L}_{\text{OOD}} = -\frac{1}{N_q}\sum_q \frac{1}{N_j}\sum_j H(p_j),$$

where $H(\cdot)$ denotes entropy. The term is then integrated in the original GroundingDINO loss as follow:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{GD}} + \lambda_{LoCoOp}\mathcal{L}_{\text{OOD}}$$

In this way, learned prompts encourages the model to remain uncertain about background areas, improving robustness to domain shift.

Motivation

LoCoOp prompts condition the model same way as CoOp do, sharing the same structure of concatenation and backpropagation. The main reason to prefer LoCoOp is that it tries to make the prompts uncertain about suspected spurious correlation. What we expect from this implementatin is a drop in wrong bounding boxes confidence. A direct influence of this method could be in the langauge-guided query initialization process: since the regularization tries to maximize entropy of non topK patches through similarity between text and visual, almost the same used for query selection, the process could learn to select better queries for initialization, discarding false positives. Qualitatively speaking, we expect that areas that are crowded by multiple bounding box at different confidence level will be in a way filtered, making wrong and duplicate boxes lower in confidence.

Implementation Details

LoCoOp adopts the same prompt-learning strategy as CoOp but introduces an additional loss module in the training loop. The `PromptLearner` module remains largely identical to the CoOp adaptation, with the main differences residing in the other components of the model.

In our Tranformer layer adaptation, by taking the fused multimodal features output by the Feature Enhancer, normalizing them, and performing a matrix multiplication, we obtain the cosine similarity between textual token features and visual patch features (Algorithm 2).

From this similarity matrix, the regions used for regularization are selected following the principles of the original LoCoOp, with two main adaptations

1. The similarities are now computed between image patches and text tokens; since GroundingDINO uses a sub-word tokenizer, a class can be described by more than one token; for this reason, token-level similarities belonging to the same class are projected to produce a patch-class similarity matrix through a one-hot encoding of class-token mapping (Algorithm 4).

2. The classification regularization is extended to detection by computing the regularization term within each predicted bounding box, so we have to aggregate across all boxes to obtain the final value (Algorithm 4).

For each prediction, the entropy calculation involves computing the average entropy over the selected ID-irrelevant patches of the bounding box. The ID-irrelevant region are selected by taking regions in which the similarities with ground truth class isn't among the topK values (Algorithm 3).

Regularization computation ends averaging over all the detections in the batch. (Algorithm 4).

Algorithm 2 CustomLoCoOpTransformer

Input: $srcs$ (multi-scale visual features), $masks$, $refpoint\_embed$, pos_embeds, tgt, $text\_dict$

▷ Step 1: Feature Enhancer

Encode multimodal inputs:

$memory \leftarrow$ visual embeddings

$memory\_text \leftarrow$ text embeddings

Initialize query embeddings (tgt) and reference boxes ($refpoint\_embed$).

▷ Step 2: Multimodal Decoder

Decode multimodal representations:

$(hs, references) \leftarrow \text{TransformerDecoder}(memory, tgt, refpoint\_embed)$

▷ Step 3: LoCoOp similarity computation

Normalize features:

$memory\_norm \leftarrow \text{normalize}(memory, L_2)$

$text\_norm \leftarrow \text{normalize}(memory\_text, L_2)$

Compute cosine similarity:

$sim\_flat \leftarrow memory\_norm \times text\_norm^T$

Store per level similarity results in $text\_dict$:

$\{sim\_flat\_lvl, sim\_flat\_lvl\_avg\}$

▷ Step 4: Return

return hs, $references$, hs_enc, ref_enc, $init\_box\_proposal$

Algorithm 3 EntropySelectTopK

Input: p (class similarities [n_features, n_classes]), top_k, $label$, $num\_of\_local\_feature$

Apply softmax over classes:

$p \leftarrow \text{Softmax}(p)$

▷ — Exclude regions containing the ground-truth label —

$pred\_topk \leftarrow \text{TopKIndices}(p, top\_k)$

$contains\_label \leftarrow (pred\_topk == label)$

$selected\_p \leftarrow p[\text{not } contains\_label]$

▷ — Compute entropy over selected regions —

$entropy \leftarrow -\text{mean}\Big(\sum selected\_p \cdot \log(selected\_p + 10^{-5})\Big)$

return $entropy$

Algorithm 4 SetCriterion with LoCoOp Regularization

Input: $outputs$, $targets$, $indices$, num_boxes
Parameters: $locoop\_top\_k$, $locoop\_weight$
Compute index mapping between matching prediction-target.
Normalize pos_maps ▷ one-hot token-class mapping
for each batch b in matching indexes **do**
    Get $boxes_b$, $labels_b$
    Select first similarity level $sim\_map \leftarrow sim\_levels[0][b]$
    for each $(box, label)$ in $(boxes_b, labels_b)$ **do**
        Convert box from (cx, cy, w, h) to pixel coordinates
        Extract local region:
        $local\_sim \leftarrow sim\_map[y1 : y2, x1 : x2, :]$
        Project to class space:
        $local\_class\_sim \leftarrow local\_sim \times pos\_maps_b^T$
        Compute entropy regularization:
        $loss \leftarrow -EntropySelectTopK(local\_class\_sim, locoop\_top\_k, label)$
        Append $loss$ to $locoop\_losses$
    end for
end for
▷ — Aggregate and return —
$loss\_locoop \leftarrow mean(locoop\_losses) \times locoop\_weight$
return $\{loss\_locoop\}$

Chapter 4

Experiments and Results

In this chapter, the focus will be moved to the evaluation of prompt-based conditioning, to explore whether it improves adaptation in few-shot detection scenarios consistently.

4.1 Experimental Setup

4.1.1 Benckmark

Datasets

Both CoOp and LoCoOp adaptations are trained on Roboflow-20VL, a challenging dataset built in order to measure the ability of foundation models to localize objects from a few visual examples and textual descriptions. Roboflow-20VL (RF20) is a set of different sub-datasets developed in collaboration between Roboflow and Carnegie Mellon University as part of the Foundational Few-Shot Object Detection Challenge [CVPR 2025].

Designed to assess the capability of foundation models to localize objects from a few visual examples and textual descriptions, provides a benchmark for few-shot multimodal learning. The dataset consists of 20 heterogeneous sub-datasets drawn from distinct domains, such as supermarket product localization, defect detection, and action contextualization, as well as diverse imaging modalities, including X-rays, thermal imagery, and aerial photography.

Each sub-dataset provides few annotation, encouraging the exploration of limited supervision techniques.

State-of-the-art foundation models such as Qwen 2.5VL and GroundingDINO achieve less than 1% accuracy on many RF20 datasets, highlighting the benchmark's difficulty. Consequently, RF20 provides an ideal testbed for analyzing the robustness, adaptability, and semantic alignment of promptable and few-shot-aware models such as the one proposed in this thesis.

Table 4.1: Overview of datasets composing the benchmark. Domain is choosen over dataset name since more informative. Each of the entries belong to one dataset only.

Domain	**#Images**	**#Classes**
Volleyball Actions	523	6
Aerial Airplans	42	1
UI Elements	183	10
Aquatic Life	152	7
Railway Components	253	4
Dental X-ray	175	4
FLIR Camera	564	4
Wheat Head	509	1
Lacrosse Components	113	4
Wood Defects	210	5
Snack Packages	132	8
Document Structure	853	19
Waste Material	302	6
Soda Bottles	246	3
Dreidels and Letter	158	6
Wildlife Trail	167	2
Water Meter Digits	220	10
White-Backed Animal	201	3
Wildfire Smoke	95	1
Hand X-ray	458	6
total	5556	110

Metrics

To compare a prediction with its ground truth in visual tasks, it's often used a metric called Intersection over Union (IoU) to give a measure of how two visual region are matching each other. IoU equation (Figure 4.2) is used to

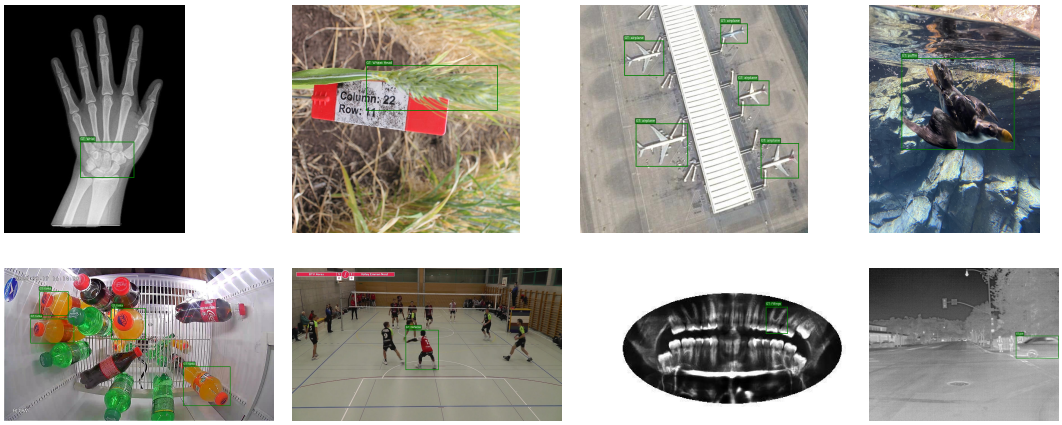

Figure 4.1: Examples of Roboflow 20VL datasets images.

calculate the ratio between the intersection and the union of two boxes or segmentation masks.

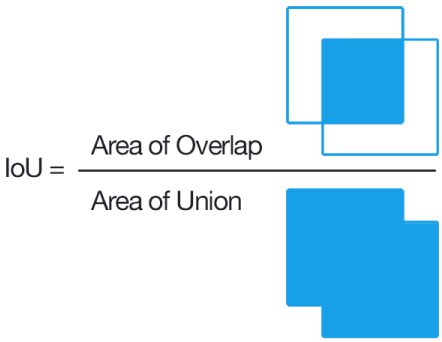

Figure 4.2: Visualization of IoU equation.

Through IoU, if two boxes are far away, not sharing any of their area, the ratio approaches zero; while, for perfectly matching boxes, it approaches 1. This metric is used in order to consider, given a specific threshold, if a predition is a true positive or not, given the ground truth to be compared with.

IoU is used inside metrics such as Average Precision (AP) in order to combine precision and recall into a single number. Precision and recall are formally described as:

$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN}$$

To get precision and recall values in vision tasks, predictions are sorted decreasingly based their confidence: at each level, just predictions having a greater or equal confidence score are considered. In this way we're saying that, above that confidence score, the model is capable to obtain that values of precision and recall, given an IoU threshold. For each new prediction considered, we one between TP and FP increases, respectively if that's a correct or wrong prediction, while the precision increases or decreases accordingly. FN decreases when a new TP is found, making the recall value always not decreasing. The ideal condition is that, for each new prediction, we have a TP, so that the value of precision is always 1 while recall increases. The general definition of AP is the area under the precision-recall curve (Figure 4.3):

$$AP = \int_0^1 p(r)dr$$

, greater value of this area means the model is performing better. In most framework, the value of AP is obtained discretely sampling precision at recall interval, using this value to numerically integrate the curve.

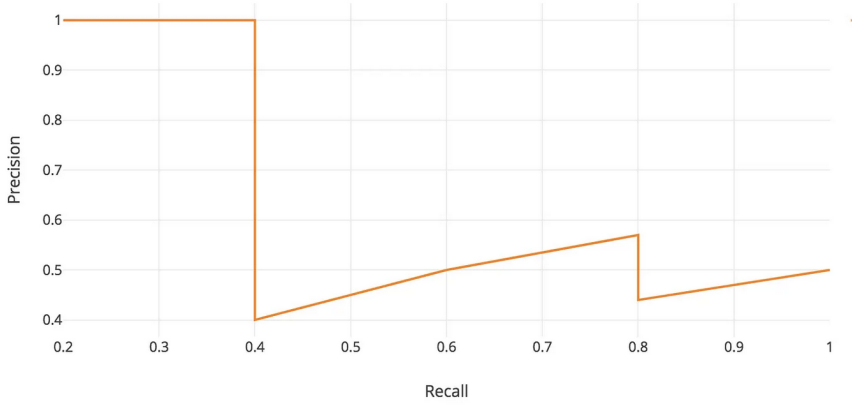

Figure 4.3: Precision recall curve example.

The COCO standard benchmark AP is indeed calculated sampling 101 points of the precision-recall curve. Moreover, COCO AP is an average of APs at different IoU threshold intervaled by a fixed range of 0.05, giving an

indication of both strong localization capabilities (e.g. $IoU = 0.9$) while still looking if the model is capable of detecting object at all (e.g. $IoU = 0.5$). Typical IoU values for the COCO IoU thresholds are $0.5 : 0.05 : 0.95$, 0.5 and 0.75.

Experiments

All the experiments are conducted in order to prove the effect of few-shots prompt-learning techniques on GroundingDINO model, comparing three main configurations:

- Baseline: the original GroundingDINO without any prompt adaptation.
- CoOp: GroundingDINO equipped with a prompt learner module as described in Section 3.2.
- LoCoOp: CoOp GroundingDINO with the additional local regularization term introduced in Section 3.3.

For the baseline, we report performance on a zero-shot setup, to check for any improvement from the base model, while for the other configuration as anticipated, experiments are conducted under few-shot regimes, varying the number of samples per class ($k \in \{4, 8\}$).

Sampling for few-shot setup

In order to perform few-shot learning, we have to implement a strategy to build a subset of the dataset composed ideally by n_{shots} samples (bounding boxes) for each class we have in our dataset. Dealing with a variable annotation number for each image, imposes to build a sampling process that either discard some annotation to match exactly the n_{shots} number, or accept some minimal variability to the number of samples taken per class. Among these two choises, the second seems more reasonable, since removing annotations from an image could negatively affect the model's ability to recognize the objects present in that image. So the sampling strategy could be summarized in the following points:

1. Each class must include at least the required number of annotations to satisfy the chosen number of shots for the experiment.
2. For classes that ends up having more than the specified number of shots, the number of included annotations is kept as low as possible while still meeting all constraints.

3. The final solution is found in a way that the number of classes exceeding the shots number is keep as low as possible.

Training Details

In accordance with the CoOp and LoCoOp training setups, we use the AdamW optimizer with a learning rate of 0.002, a linear warm-up over the first 10 epochs (from 0 to 0.002), and a cosine annealing schedule for the remaining epochs. The batch size is set to 4 and the number of epochs for all the experiments 100.

For the GroundingDINO loss, the weights used are 2.0 for the focal loss, 5.0 for the ℓ_1 loss, and 2.0 for the GIoU loss, while for the LoCoOp regularization, we used a weight of 0.25 following the original work.

For the pretrained checkpoint, all experiment are performed initializing GroundingDINO through official weights linked in the official repository.

A summary of training parameters is shown in Table 4.2.

Table 4.2: Training configuration summary.

Parameter	**Value**
Learning rate	0.002
Learnig rate scheduling	linear[: 10]+cosine[10 :]
Batch size	4
Epochs	100
Focal loss coef.	2.0
L1 loss coef.	5.0
GIoU loss coef.	2.0
LoCoOp reg. coef.	0.25

4.2 Quantitative Results

The initial experiment aimed to evaluate the performance of the baseline GroundingDINO models and determine which visual backbone between the lighter Swin-T or the heavier Swin-B is better to use in subsequent experiments. As results reported in Table 4.3 show, Swin-B achieves better performance in all IoU thresholds, so it will be used as a baseline for our methods evaluation.

Table 4.3: GroundingDINO baseline performances.

Visual Backbone	Performance		
	AP	**AP50**	**AP75**
Swin-T	13.62	18.63	15.49
Swin-B	**17.63**	**24.15**	**20.45**

After conducting multiple experiments with varying configurations for the CoOp and LoCoOp extentions, we have shown that prompr learning improve the accuracy of GroundingDINO consistently, as shown in Table 4.4. Even at higher IoU threshold, we achieve strong improvement over the baseline, showing that implemented techniques are contributing positively to the localization power precision of the model. Each setup reported in Table 4.4 is presented with its best-performing configuration, as identified through the ablation studies discussed in Section 4.4.

Table 4.4: GroundingDINO best-setup few-shot performances.

		Performance		
Method	**Shots**	**AP**	**AP50**	**AP75**
GroundingDINO	0	17.63	24.15	20.45
CoOp	4	33.93	45.18	37.73
CoOp	8	37.77	50.22	42.88
LoCoOp	4	35.47	47.02	39.89
LoCoOp	8	**39.75**	**53.24**	**44.68**

4.3 Qualitative Results

To evaluate qualitatively LoCoOp reuglarization, we report results of patches selection of best selected model through different training epochs in Figure 4.4. As we can see, through the learning process, more patches of the object that must be detected are excluded from the regularization process, suggesting that the model is learning better patch level text-visual aligning.

While in Figure 4.5, we can see that the baseline already have localization capabilities, but perform poorly in classification. LoCoOp regularization helps improving the model, correctly classifying objects that were previously

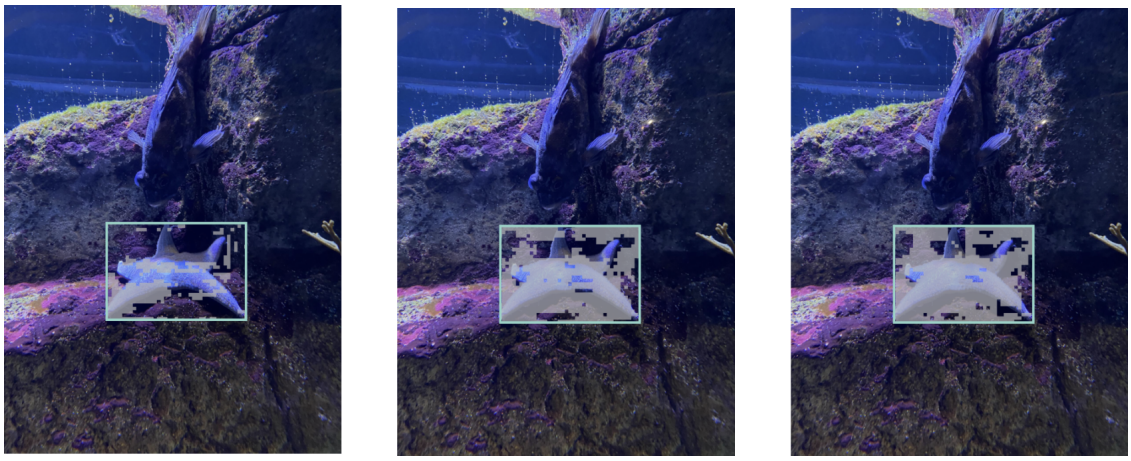

Figure 4.4: Example of locoop regularization on image patches over time. Greyed out patches are those that aren't taken to regularize the model.

misclassified.

4.4 Ablation Studies

We've setup different experiments in order to verify the contribution of each component of the model. The experiments are about the number of context tokens and the topK parameter in the Local Regularization.

Regarding the context length, the experiments (reported in Table 4.5) generally show that increasing the amount of context leads to improved performance. We have a large initial increase in performance just introducing context—about +8.98 between the chosen baseline and the lowest-performing setting—while variations in context length within the same context type and shot configuration yield an average improvement of approximately +0.55. Instead, regarding the context type, we can notice that it's the most conditioning setup overall, with its shift between the best configuration of Unified Context and Class-Specific Context of +7.18.

As expected, learning a specialized prompt for each specific class improve more than learning a unified context for the whole dataset. But, while class-specific context dispense better results, unified context is still interesting, since could possibly encode broader information about the dataset global distribution.

Regarding Local Regularization (experiments reported in Table 4.6), we

Figure 4.5: Comparison of base model with the best performing setup. On the left we can see the results of the baseline model, while on the right the best performing setup (LoCoOp with topK=1, shots=8, class-specific context = 8).

observe an average improvement of +2.28 compared to the CoOp-only implementation, when keeping all other parameters fixed. This shows the effectiveness of the regularization with respect to possible nuisances present in the data.

As for the topK ablation, the trend is less straightforward: in some configurations, strong regularization (with the maximum value, $topK = 1$) yields the best performance, while in others, the optimal setting is found close to the maximum but not at its extreme.

Table 4.5: CoOp context length ablation results.

Context			Performance		
Type	**Length**	**Shots**	**AP**	**AP50**	**AP75**
Uni	4	4	26.61	36.11	30.15
	8		**27.23**	**36.26**	**30.28**
	4	8	30.09	**40.22**	**33.74**
	8		**30.59**†	40.12	33.49
CS	4	4	32.95	43.67	37.03
	8		**33.93***	**45.18**	**37.73**
	4	8	37.64	50.16	42.46
	8		**37.77***†	**50.22**	**42.88**

* Best performance for few-shot setup configuration

† Best performance for context type configuration

Table 4.6: LoCoOp TopK value ablation results.

Context			Local reg.		Performance		
Type	**Length**	**Shots**	**Weight**	**TopK**	**AP**	**AP50**	**AP75**
Uni	8	4	0.25	0.4	**29.80**	38.94	34.19
				0.2	**29.80**	**39.32**	34.37
				1.0	29.64	38.80	**34.54**
CS	8	4	0.25	0.4	33.98	45.17	38.30
				0.2	33.84	45.51	38.29
				1.0	**35.47**	**47.02**	**39.89**
CS	8	8	0.25	0.4	39.16	51.99	44.09
				0.2	39.23	51.58	43.60
				1.0	**39.75**	**53.24**	**44.68**

4.5 Discussion

Prompt-based adaptation dominates zero-shot

Few-shot prompt tuning (CoOp) yields large improvements over the baseline (Table 4.4 and Table 4.5). With class-specific context and 8 shots, AP rises from 17.63 to 37.77 (+20.14 AP), and even with only 4 shots AP reaches 33.93 (+16.30). These gains confirm that learning textual prompts is a parameter

efficient way to adapt large vision-language models to unseed classes under scarce supervision.

Context semantics outweigh context length

Across matched settings, class-specific context outperforms uniform context by a wide margin: at 4 shots and length 8, class-specific outperform uniform context by +6.70 AP; at 8 shots and length 8, by +7.18 AP (Table 4.5). In contrast, increasing context length from 4 to 8 tokens brings only modest gains (on average ≈+0.56 AP across settings), with diminishing returns at higher data regimes (e.g., CS 8-shot: +0.13 AP). Learning highly specialized set of prompts, class-specific context learns to guide the model encoding the semantic useful to detect the class. This suggests that what the context encodes matters substantially more than how much context is provided.

Local Regularization (LoCoOp) provides further, reliable gains

Adding the locality regularizer on top of CoOp improves both AP and high-IoU metrics (Table 4.6, Table 4.4). For CS, 8-shot, length 8, LoCoOp with TopK=1.0 raises AP by +1.98, AP50 by +3.02, and AP75 +1.80. In a lower-data/less-informative setting (Uni, 4-shot, length 8), LoCoOp still brings +2.41 to +2.57 AP depending on TopK, with particularly strong improvements at AP75 (+3.91 to +4.26), suggesting better box tightness and spatial precision under data scarcity

By the higher AP50 value, we can state that the model learn to stabilize prompt vectors, matching visuals features across images more reliably. Moreover, the value of AP75, suggest that even localization has improved.

Sensitivity to the locality hyper-parameter

TopK interacts with the data regime: for CS/8-shot, the best AP is with TopK=1.0; for Uni/4-shot, several TopK values tie on AP but trade off AP50 vs. AP75. This indicates the regularizer can be tuned either for recall/confidence (higher AP50) or for tighter localization (higher AP75), depending on deployment needs.

4.6 Conclusion and Future Work

This work investigated prompt-based adaptation strategies for open-vocabulary object detection using the GroundingDINO framework. Through systematic ablations, we analyzed the influence of number of shots and the proposed adaptation of Context Optimization and Local Regularization modules.

Prompt learning methods like CoOp yield substantial gains even in extremely low-shot settings, and their effectiveness depends more on the semantic specificity of the context (class-specific vs. uniform) than on its length. About the Local Regularization (LoCoOp), it consistently enhances both detection accuracy and localization precision, providing a simple yet effective mechanism to stabilize learned prompts.

Possible future works could estabilish the difference between class-specific and uniform context under different constraint: rather than rough model performance, a valuable experiment could estabilish if uniform context capture global characteristics of the dataset as a whole, by comparing its performance with in-domain novel classes. Another possible interesting path could explore adaptive or learned strategies for selecting context length and regularization strength.

Overall, our results suggest that careful prompt design, supported by localized regularization, can unlock strong few-shot generalization in grounding-based object detection systems, and that, incrementally improving this techniques, can lead to fully automatic pipelines delivering high quality annotations.

Bibliography

[1] Mark Everingham et al. "The PASCAL Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136. DOI: 10.1007/s11263-014-0733-5.

[2] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

[3] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/abs/1506.01497.

[4] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640.

[5] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV]. URL: https://arxiv.org/abs/2005.12872.

[6] Xizhou Zhu et al. *Deformable DETR: Deformable Transformers for End-to-End Object Detection*. 2021. arXiv: 2010.04159 [cs.CV]. URL: https://arxiv.org/abs/2010.04159.

[7] Feng Li et al. *DN-DETR: Accelerate DETR Training by Introducing Query DeNoising*. 2022. arXiv: 2203.01305 [cs.CV]. URL: https://arxiv.org/abs/2203.01305.

[8] Kihyuk Sohn et al. *A Simple Semi-Supervised Learning Framework for Object Detection*. 2020. arXiv: 2005.04757 [cs.CV]. URL: https://arxiv.org/abs/2005.04757.

[9] Yen-Cheng Liu et al. *Unbiased Teacher for Semi-Supervised Object Detection*. 2021. arXiv: 2102.09480 [cs.CV]. URL: https://arxiv.org/abs/2102.09480.

[10] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV]. URL: https://arxiv.org/abs/1708.02002.

[11] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/68053af2923e00204c3ca7c6a3150cf7-Paper.pdf.

[12] Xin Wang et al. *Frustratingly Simple Few-Shot Object Detection*. 2020. arXiv: 2003.06957 [cs.CV]. URL: https://arxiv.org/abs/2003.06957.

[13] Xiaopeng Yan et al. "Meta r-cnn: Towards general solver for instance-level low-shot learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9577–9586.

[14] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[15] Chao Jia et al. *Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision*. 2021. arXiv: 2102.05918 [cs.CV]. URL: https://arxiv.org/abs/2102.05918.

[16] Ao Wang et al. *YOLOE: Real-Time Seeing Anything*. 2025. arXiv: 2503.07465 [cs.CV]. URL: https://arxiv.org/abs/2503.07465.

[17] Shilong Liu et al. *Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection*. 2024. arXiv: 2303.05499 [cs.CV]. URL: https://arxiv.org/abs/2303.05499.

[18] Kaiyang Zhou et al. "Learning to Prompt for Vision-Language Models". In: *International Journal of Computer Vision* 130.9 (July 2022), pp. 2337–2348. ISSN: 1573-1405. DOI: 10.1007/s11263-022-01653-1. URL: http://dx.doi.org/10.1007/s11263-022-01653-1.

[19] Atsuyuki Miyai et al. *LoCoOp: Few-Shot Out-of-Distribution Detection via Prompt Learning*. 2023. arXiv: 2306.01293 [cs.CV]. URL: https://arxiv.org/abs/2306.01293.

[20] Paul A. Viola and Michael J. Jones. "Rapid Object Detection using a Boosted Cascade of Simple Features." In: *CVPR (1)*. IEEE Computer Society, 2001, I:511–518. ISBN: 0-7695-1272-0. URL: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2001.html#ViolaJ01.

[21] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[22] Pedro F. Felzenszwalb et al. "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645. DOI: 10.1109/TPAMI.2009.167.

[23] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[24] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[26] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV]. URL: https://arxiv.org/abs/1311.2524.

[27] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. ISBN: 9783319105789. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[28] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV]. URL: https://arxiv.org/abs/1504.08083.

[29] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV]. URL: https://arxiv.org/abs/1612.03144.

[30] Zhaowei Cai and Nuno Vasconcelos. *Cascade R-CNN: Delving into High Quality Object Detection*. 2017. arXiv: 1712.00726 [cs.CV]. URL: https://arxiv.org/abs/1712.00726.

[31] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[32] Hei Law and Jia Deng. *CornerNet: Detecting Objects as Paired Keypoints*. 2019. arXiv: 1808.01244 [cs.CV]. URL: https://arxiv.org/abs/1808.01244.

[33] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. *Objects as Points*. 2019. arXiv: 1904.07850 [cs.CV]. URL: https://arxiv.org/abs/1904.07850.

[34] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. "A Neural Probabilistic Language Model". In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

[35] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[36] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[37] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[38] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[39] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[40] Liunian Harold Li et al. *Grounded Language-Image Pre-training*. 2022. arXiv: 2112.03857 [cs.CV]. URL: https://arxiv.org/abs/2112.03857.

[41] Hao Zhang et al. *DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection*. 2022. arXiv: 2203.03605 [cs.CV]. URL: https://arxiv.org/abs/2203.03605.

[42] Xizhou Zhu et al. *Deformable DETR: Deformable Transformers for End-to-End Object Detection*. 2021. arXiv: 2010.04159 [cs.CV]. URL: https://arxiv.org/abs/2010.04159.

[43] Hamid Rezatofighi et al. *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression*. 2019. arXiv: 1902.09630 [cs.CV]. URL: https://arxiv.org/abs/1902.09630.

[44] Tianhe Ren et al. *Grounding DINO 1.5: Advance the "Edge" of Open-Set Object Detection*. 2024. arXiv: 2405.10300 [cs.CV]. URL: https://arxiv.org/abs/2405.10300.

[45] Xiang Lisa Li and Percy Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. DOI: 10.18653/v1/2021.acl-long.353. URL: https://aclanthology.org/2021.acl-long.353/.

[46] Brian Lester, Rami Al-Rfou, and Noah Constant. *The Power of Scale for Parameter-Efficient Prompt Tuning*. 2021. arXiv: 2104.08691 [cs.CL]. URL: https://arxiv.org/abs/2104.08691.

[47] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG]. URL: https://arxiv.org/abs/1910.10683.