FEDERAL UNIVERSITY OF RIO GRANDE DO NORTE
TECHNOLOGY CENTER
GRADUATE PROGRAM IN ELECTRICAL AND COMPUTER ENGINEERING

Integrating Textual Queries with AI-Based Object Detection: A Compositional Prompt-Guided Approach

Silvan Ferreira da Silva Junior

Supervisor: Prof. Dr. Allan de Medeiros Martins

Co-supervisor: Prof. Dr. Ivanovitch Medeiros Dantas da Silva

Doctoral Thesis presented to Graduate Program in Electrical and Computer Engineering at UFRN (concentration area: Computer Engineering) as part of the requirements for obtaining the title of Doctor of Science.

PPgEEC Order Number: D384
Natal, RN, June 2025

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Silva Júnior, Silvan Ferreira da.
Integrating Textual Queries with AI-Based Object Detection: A Compositional Prompt-Guided Approach / Silvan Ferreira da Silva Júnior. - 2025.
73f.: il.

Tese (Doutorado) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Natal, 2025.
Orientação: Prof. Dr. Allan de Medeiros Martins.
Coorientação: Prof. Dr. Ivanovitch Medeiros Dantas da Silva.

1. Neuro-Symbolic AI - Tese. 2. Prompt-Guided Object Detection - Tese. 3. Crossmodal Reasoning - Tese. 4. Visual-Language Alignment - Tese. 5. Query-Driven Recognition - Tese. I. Martins, Allan de Medeiros. II. Silva, Ivanovitch Medeiros Dantas da. III. Título.

RN/UF/BCZM CDU 004

Elaborado por Raimundo Muniz de Oliveira - CRB-15/429

Acknowledgments

Agradeço à minha mãe, Conceição Miranda, à minha filha, Júlia Maria, e à minha tia Judith Miranda, por todo o amor, paciência e apoio incondicional ao longo desta jornada.

Aos meus orientadores e amigos Allan Medeiros e Ivanovitch Silva, pelo apoio constante desde o mestrado, sendo fundamentais não apenas na minha formação acadêmica, mas também no meu desenvolvimento profissional e pessoal.

Aos amigos que, de diferentes formas, contribuíram com este projeto, em especial Thiago Medeiros, Márcio Dahia, Brito Filho, Marcelo Queiroz e Daniel Costa, pelos diálogos, insights e apoio técnico e emocional durante o percurso.

À UFRN, pela estrutura acadêmica e institucional que possibilitou a realização deste trabalho, e ao CNPq, pelo apoio financeiro concedido durante o período da pesquisa.

Abstract

In the field of computer vision, object detection and recognition play a central role in many applications that support automated decision-making. Over recent years, new algorithms and methodologies have emerged to further enhance the automatic identification of target objects. In particular, the rise of deep learning and language models has opened many possibilities in this area, although challenges in contextual query analysis and human interactions persist. This thesis presents a novel neuro-symbolic object detection framework that aligns object proposals with textual prompts using a deep learning module while enabling logical reasoning through a symbolic module. By integrating deep learning with symbolic reasoning, object detection and scene understanding are considerably enhanced, enabling complex, query-driven interactions. Using a synthetic 3D image dataset, the results demonstrate that this framework effectively generalizes to complex queries, combining simple attribute-based descriptions without explicit training on compound prompts. We present the numerical results and comprehensive discussions, highlighting the potential of our approach for emerging smart applications.

Keywords: Neuro-Symbolic AI, Prompt-Guided Object Detection, Crossmodal Reasoning, Visual-Language Alignment, Query-Driven Recognition.

Resumo

No campo da visão computacional, a detecção e o reconhecimento de objetos desempenham um papel fundamental em diversas aplicações voltadas à tomada de decisão automática. Nos últimos anos, novos algoritmos e metodologias foram propostos para aprimorar a identificação automática de objetos-alvo. Em particular, o avanço do aprendizado profundo e dos modelos de linguagem abriu inúmeras possibilidades nessa área, embora persistam desafios na análise contextual de consultas e nas interações humanas. Esta tese apresenta um novo framework neuro-simbólico de detecção de objetos que alinha propostas de objetos a prompts textuais por meio de um módulo de aprendizado profundo, ao mesmo tempo em que possibilita o raciocínio lógico por meio de um módulo simbólico. Ao integrar aprendizado profundo com raciocínio simbólico, a detecção de objetos e a compreensão de cena são consideravelmente aprimoradas, viabilizando interações complexas orientadas por consultas. Utilizando um conjunto de dados sintéticos de imagens 3D, os resultados demonstram que o framework generaliza de forma eficaz para consultas complexas, combinando descrições baseadas em atributos simples sem treinamento explícito em prompts compostos. Apresentamos os resultados numéricos e discussões abrangentes, destacando o potencial de nossa abordagem para aplicações inteligentes emergentes.

Palavras-chave: IA Neuro-Simbólica, Detecção de Objetos Guiada por Prompt, Raciocínio Crossmodal, Alinhamento Visão-Linguagem, Reconhecimento Orientado por Consultas.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

AI	Artificial Intelligence
AOVR	Annotated Objects for Visual Reasoning
AST	Abstract Syntax Tree
BPE	Byte-Pair Encoding
CFG	Context-Free Grammar
CNN	Convolutional Neural Networks
DAG	Directed Acyclic Graph
DFA	Deterministic Finite Automaton
FC	Fully Connected
IoU	Intersection over Union
LLM	Large Language Model
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
MLP	Multi-Layer Perceptron
NFA	Non-Deterministic Finite Automaton
NMS	Non-Maximum Suppression
OVOD	Open-Vocabulary Object Detection
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RoI	Region of Interest
SSD	Single Shot MultiBox Detector
YOLO	You Only Look Once

Chapter 1

Introduction

This chapter presents the motivations and foundations that led to the development of this research. It begins by contextualizing the evolution of object detection, emphasizing the recent transition toward prompt-guided, open-vocabulary models and their current limitations in handling compositional and logical queries. Based on this gap, the central research question is formulated, guiding the investigation towards a neuro-symbolic framework that combines deep learning with symbolic reasoning. The main contributions of this work are then outlined, highlighting the proposed methodology and its potential impact on the field.

1.1 Background and Context

Object detection has undergone a remarkable evolution from early handcrafted approaches to modern AI-driven techniques. In the pre-deep-learning era, detectors relied on manually designed features and classifiers (Everingham et al. 2010, Zhao et al. 2019). For example, the seminal works on Haar cascades and deformable part models demonstrated object localization using edge and gradient features (Kaur & Singh 2024). These traditional methods, however, were limited in generality and struggled with varying object appearances (Preeti & Rana 2024, Ahmad & Rahimi 2022). The advent of deep learning brought a paradigm shift: Convolutional Neural Networks (CNNs) enabled end-to-end learning of object detectors from large datasets (LeCun et al. 1998, LeCun et al. 2015). Region-based models like Faster R-CNN (Ren et al. 2015) and single-shot models like YOLO (Redmon et al. 2016) drastically improved accuracy and speed by learning hierarchical visual features. Such CNN-based detectors became the de facto standard (Zhao et al. 2019, Manakitsa et al. 2024), but they were still constrained to a closed vocabulary of object classes determined by their training data. In practice, this meant a model could only detect the specific categories it had seen during training, limiting its usefulness in open-world scenarios where new object categories continually appear (Kaur & Singh 2024, Hossain et al. 2023).

In recent years, object detection has advanced towards a more flexible paradigm driven by vision-language models capable of understanding natural language prompts (Radford et al. 2021). Unlike traditional detectors constrained by fixed label sets (Ren et al. 2015, Redmon et al. 2016), these models enable Open-Vocabulary Object Detection (OVOD),

where the user can specify the target concept through free-form text (Minderer et al. 2022). This is achieved by learning joint representations of images and text, allowing the system to associate visual regions with arbitrary textual descriptions, even for concepts unseen during training (Radford et al. 2021). As a result, object detection has evolved from closed-set classification to a prompt-driven approach, where the model adapts to the user's intent expressed in natural language (Minderer et al. 2022, Gu et al. 2022). This shift represents a significant step towards more adaptable and generalizable visual recognition systems (Zhong et al. 2022, Yao et al. 2023).

Despite this progress, current open-vocabulary detectors remain limited in their ability to handle logical or compositional queries. Existing models excel at detecting single concepts or simple combinations, as demonstrated by ViLD (Gu et al. 2022), RegionCLIP (Zhong et al. 2022), and Grounding DINO (Wang et al. 2024), which allow users to detect multiple objects by providing independent keywords. However, they do not natively support reasoning with logical connectors such as AND, OR, and NOT within a single query. A user cannot directly ask a detector to find all vehicles that are not cars or to find red cubes and either blue spheres or green cylinders in one step, because the model does not parse or understand logical syntax; it only matches patterns to embeddings of literal phrases. Some compositional queries can be partially addressed by prompt engineering, as shown in CLIP (Radford et al. 2021) and Grounding DINO (Wang et al. 2024), where feeding the phrase red cube enables detection of red cubes, but this approach breaks down when facing more complex logic involving multiple attributes or disjunctions. This limitation stems from the fact that purely neural models fundamentally lack a mechanism for symbolic reasoning, treating the prompt as a holistic embedding without the capacity to disentangle multiple conditions or execute Boolean operations over detection results, a challenge broadly discussed in the neuro-symbolic AI literature (Garcez et al. 2019, Lamb et al. 2020, Mao et al. 2019). Furthermore, vision-language models frequently struggle with rare or out-of-distribution concept compositions that were not encountered during training, a limitation observed in models like Flamingo (Alayrac & others 2022) and Coarse-to-Fine Grounding (Jiang et al. 2024). As a result, there remains a significant gap in current methodologies, as no existing system robustly supports flexible, compositional queries that capture the richness and logical structure of natural human descriptions of visual scenes.

1.2 Research Motivation

Despite the considerable progress achieved by prompt-guided open-vocabulary object detection, current models remain fundamentally limited in their capacity for compositional and logical reasoning. These systems are predominantly designed to match individual prompts to visual regions but lack the ability to parse and execute queries involving logical operators such as AND, OR, and NOT. As a consequence, users cannot directly express complex queries like "find all vehicles that are not cars" or "detect red cubes and either blue spheres or green cylinders" in a single step. While prompt engineering may suffice for simple attribute conjunctions, it rapidly breaks down when faced with more intricate logical compositions.

This limitation is rooted in the fact that current models rely solely on dense, monolithic embeddings derived from textual inputs, without any capacity to explicitly decompose, combine, or negate semantic components. Furthermore, the absence of a symbolic reasoning mechanism renders these models fragile when dealing with rare, combinatorial, or out-of-distribution concepts not covered during pretraining. This disconnect between the expressive richness of natural language and the flat embedding-based matching in current OVOD systems presents a significant barrier to more sophisticated, human-aligned visual reasoning.

In this context, the central research question that guides this thesis is formulated as follows:

> **How can we extend prompt-guided, open-vocabulary object detection to support compositional and logical queries expressed in natural language, while preserving the generalization capabilities of vision-language models?**

Addressing this question is critical for advancing object detection beyond simple category matching toward systems capable of flexible, interpretable, and logic-driven visual understanding. It calls for the integration of symbolic reasoning mechanisms into the open-vocabulary detection pipeline, enabling models not only to localize objects based on textual descriptions but also to perform operations over sets of detections in accordance with logical constraints derived from user queries.

1.3 Main Contributions

In this PhD work, we address these limitations by proposing a neuro-symbolic architecture for prompt-guided object detection that integrates a neural vision model with a symbolic reasoning engine. The central idea is to combine the strengths of learned visual recognition and formal logic in a unified system. We design a two-module pipeline consisting of: (1) a neural open-vocabulary detector that can detect a wide range of objects given textual prompts, and (2) a symbolic interpreter that parses logical queries and orchestrates the detector to fulfill complex queries.

In our approach, a user's textual query (which may contain logical operators and multiple clauses) is first parsed into a logical form using a custom query language we developed. This logical form is then executed by the interpreter, which will call the neural detector on atomic sub-queries (e.g., detect "cube" or detect "sphere") and then combine the results according to the logic (e.g., intersect the sets of detections for "red" AND "cube"). The neural component provides the perceptual power to recognize objects in images, while the symbolic component provides the reasoning ability to compose queries, apply constraints, and derive high-level conclusions. Crucially, the two parts work in tandem: the neural detector produces structured outputs (like bounding boxes with class scores) that the symbolic module can treat as truth values or sets, enabling logical operations such as conjunction, disjunction, negation, and counting over detected objects. This fusion of statistical perception with discrete logic allows our system to answer queries

that are far more expressive than what prior detectors can handle. The core contributions of this thesis are summarized as follows:

- Vision-Language Fusion Model for Detection: We develop a novel open-vocabulary object detector that fuses visual and textual representations to localize objects from arbitrary text prompts. Our model extends existing vision-language frameworks with improved object-level feature alignment, yielding strong zero-shot detection performance on diverse concepts.
- Logical Query Language and Interpreter: We design a custom logical query language for compositional object queries, along with a symbolic interpreter capable of parsing and executing these queries. The language allows users to express boolean logic (AND, OR, NOT) and other operations (e.g. grouping, counting) over object categories and attributes in an intuitive text format. The interpreter integrates with the neural detector to compute query results, effectively acting as a neuro-symbolic reasoning engine for visual data.
- New Synthetic Dataset for Compositional Detection: We construct a synthetic benchmark dataset of complex scenes with ground-truth annotations for logical queries. This dataset contains images (both 2D rendered and 3D simulated scenes) where multiple objects with varying attributes are present, and is accompanied by automatically generated query sentences that involve combinations of object categories, attributes, and logical operations. It serves as a valuable resource to train the query parser and evaluate the system's ability to handle compositional queries that are difficult to source from existing real-image datasets.

By enabling flexible, compositional, query-based object detection, our approach paves the way for more intelligent visual understanding in both synthetic and real-world environments. Users are no longer restricted to asking "Is there an object of class X in the image?" but can pose rich questions like "Find any red cube that is not next to a blue cube" or "Are there either cats or dogs on the sofa, and if so, show me each." This capability is important for applications requiring fine-grained scene understanding, such as robotics (where an agent may need to find an object meeting multiple criteria), surveillance (querying complex scenarios from security footage), or image retrieval (searching for images by high-level descriptions). Furthermore, by testing our system on both challenging real images and highly controlled synthetic scenes, we demonstrate its robustness and versatility. In summary, this thesis contributes a new compositional paradigm to object detection – one that merges neural networks with symbolic logic – moving us closer to visual intelligence that can comprehend and answer complex queries about the world.

1.4 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 establishes the theoretical foundations that support this research, covering deep learning for visual perception, textual encoding, cross-modal alignment, and the fundamentals of symbolic reasoning based on formal language theory. Chapter 3 presents a comprehensive review of related

works, highlighting their strengths, limitations, and how this thesis positions itself within the current research landscape. Chapter 4 details the proposed neuro-symbolic framework, including the neural architecture, the symbolic reasoning module, and the dataset generation process. Chapter 5 presents the experimental results along with a quantitative and qualitative evaluation of the framework. Finally, Chapter 6 concludes the thesis, summarizing the main contributions, discussing current limitations, and proposing future research directions.

Chapter 2

Theoretical Background

This chapter establishes the theoretical foundation that supports the proposed neuro-symbolic framework. It begins in Section 2.1 with a detailed account of how convolutional architectures evolved into modern end-to-end object detectors, forming the visual backbone of this work. Section 2.2 explores how natural language is transformed into dense vector representations that are compatible with neural computation. Section 2.4 then formalizes the mechanisms that align these heterogeneous modalities within a shared embedding space, enabling joint reasoning across vision and language. Finally, Section 2.5 grounds the symbolic component in the formal machinery of compiler theory, including grammars, parsing, and semantic evaluation, which provides the interpretability and logical rigor central to this framework.

2.1 Deep Learning for Visual Perception

Deep learning has emerged as a fundamental paradigm in computer vision, significantly advancing the performance of object detection systems (LeCun et al. 2015, Gu et al. 2018). Unlike traditional approaches that relied on handcrafted features and separate classification pipelines, modern object detectors learn hierarchical visual representations directly from data through deep neural networks (Girshick et al. 2014, Ren et al. 2015, Redmon et al. 2016). This capability has enabled the development of end-to-end trainable architectures that jointly perform feature extraction, region proposal, classification, and localization. In this section, we focus on two key components underpinning most deep learning-based object detectors: Convolutional Neural Networks (CNNs), responsible for extracting spatial features from input images, and Region Proposal Networks (RPNs), which efficiently generate candidate object regions. These components form the backbone of widely adopted detection frameworks, including Faster R-CNN, Mask R-CNN, and their successors (Ren et al. 2015, He et al. 2017).

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks have become a cornerstone in the field of computer vision due to their ability to learn hierarchical representations of visual data directly from raw pixel inputs. Initially proposed by LeCun et al. (1998), CNNs are a class of deep

neural networks that leverage spatial hierarchies through the use of local receptive fields, weight sharing, and subsampling operations. These architectural properties make CNNs particularly effective in processing images, where spatially localized patterns play a central role (Krichen 2023).

A typical CNN architecture is composed of multiple layers, each transforming the input image or feature map into increasingly abstract representations. The fundamental building block of a CNN is the convolutional layer, which applies a set of learnable filters, also called kernels, across the spatial dimensions of the input. Mathematically, the output of a convolutional operation for a single filter can be expressed as shown in Equation 2.1.

$$y_{i,j}^{(k)} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{i+m,j+n} \cdot w_{m,n}^{(k)} + b^{(k)} \tag{2.1}$$

where x is the input feature map, $w^{(k)}$ denotes the k-th filter of size $M \times N$, $b^{(k)}$ is the corresponding bias term, and $y_{i,j}^{(k)}$ represents the activation at position (i, j) in the output feature map.

As shown in Figure 2.1, the convolution operation involves sliding a kernel over the input matrix, producing an activation map that highlights spatial features of interest.

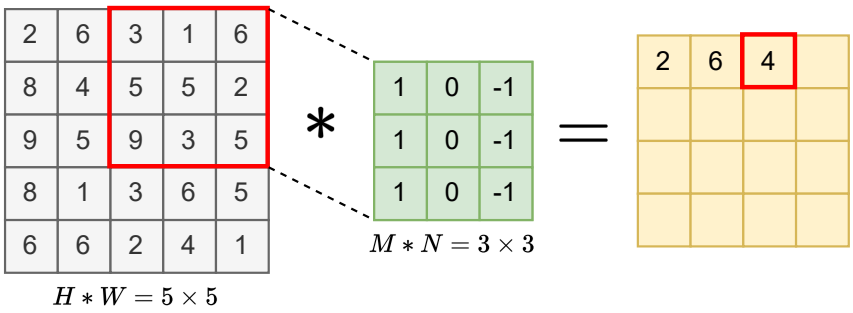

Figure 2.1: Illustration of the convolutional operation over a 2D image. A kernel slides over the input matrix, generating an activation map that highlights relevant spatial features.

CNNs are inherently structured to exploit the two-dimensional topology of images. Unlike fully connected layers where each neuron is connected to all units in the previous layer, convolutional layers restrict connections to local neighborhoods. This sparsity of connections not only reduces the number of parameters but also encourages the network to learn local patterns, such as edges, corners, and textures in the early layers, while deeper layers capture more global and semantically rich features (Li et al. 2021).

A typical CNN architecture is organized into two main components: the **feature extractor** and the **classification head**. The feature extractor is composed of a stack of convolutional layers, each typically followed by a nonlinear activation function, such as the Rectified Linear Unit (ReLU) (Nair & Hinton 2010), defined as $f(x) = \max(0, x)$, and a pooling operation, such as max-pooling or average-pooling. The convolutional layers are responsible for learning hierarchical spatial features from the input image, progressively encoding local patterns such as edges, textures, and shapes in the early layers, and

increasingly abstract representations in deeper layers.

Activation functions introduce non-linearity, enabling the network to learn complex mappings, while pooling layers reduce spatial resolution, provide translational invariance, and decrease computational cost. These components are typically arranged in a repeated sequence of convolution, activation, and pooling operations. The output of the feature extractor is a set of high-dimensional feature maps that are flattened into a one-dimensional feature vector. This vector is then passed to the classification head, composed of one or more Fully Connected (FC) layers, which act as a standard multilayer perceptron that maps the extracted features to the target output space. A final Softmax layer is commonly applied to convert the output logits into class probability distributions. This general structure of separating feature learning from decision making is a foundational design pattern in CNNs.

An example of this architecture is illustrated in Figure 2.2, where the feature extractor comprises convolution, activation, and pooling operations, and the classification head includes the flatten operation followed by fully connected layers and a Softmax function.

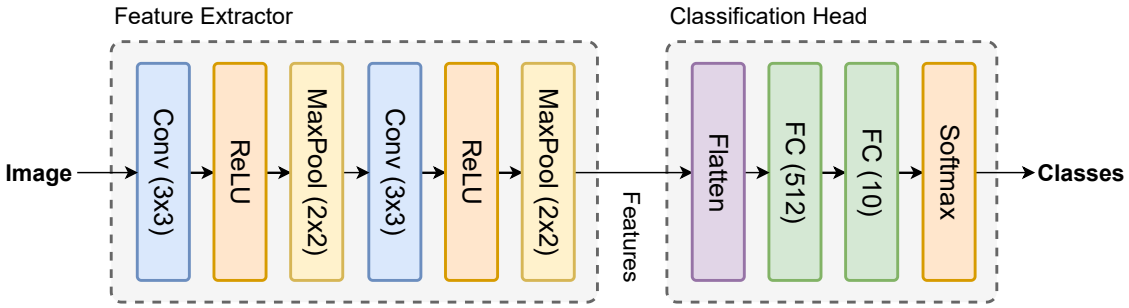

Figure 2.2: Illustrative architecture of a convolutional neural network showing alternating convolution, activation, and pooling layers followed by fully connected layers.

Over the past decade, several CNN architectures have been proposed to improve performance and efficiency across tasks and hardware configurations. Krizhevsky et al. (2012) introduced AlexNet, which popularized deep CNNs with ReLU activations and dropout regularization. VGGNet, proposed by Simonyan & Zisserman (2015), demonstrated the effectiveness of using small 3×3 filters stacked in depth. GoogLeNet (Inception) (Szegedy et al. 2015) introduced the concept of multi-path modules to increase representational capacity without dramatically increasing computation. ResNet, by He et al. (2016), incorporated residual connections to mitigate vanishing gradients in very deep networks. More recently, architectures such as EfficientNet (Tan & Le 2019) and MobileNet (Howard et al. 2017) have emphasized model efficiency for deployment on resource-constrained devices.

2.1.2 Object Detection

Object detection delivers a structured scene description by identifying every object instance and localizing each instance with a bounding box (Zhao et al. 2019). The model

returns a set $\mathcal{Y} = \{(\mathbf{b}_i, \mathbf{c}_i)\}_{i=1}^{N}$, where each localization vector is $\mathbf{b}_i = (x_i, y_i, w_i, h_i) \in \mathbb{R}^4$ and each categorical posterior is $\mathbf{c}_i \in \Delta^K$. This formulation preserves spatial structure and enables downstream tasks such as tracking and interaction analysis. Figure 2.3 illustrates the bounding-box geometry adopted in this thesis.

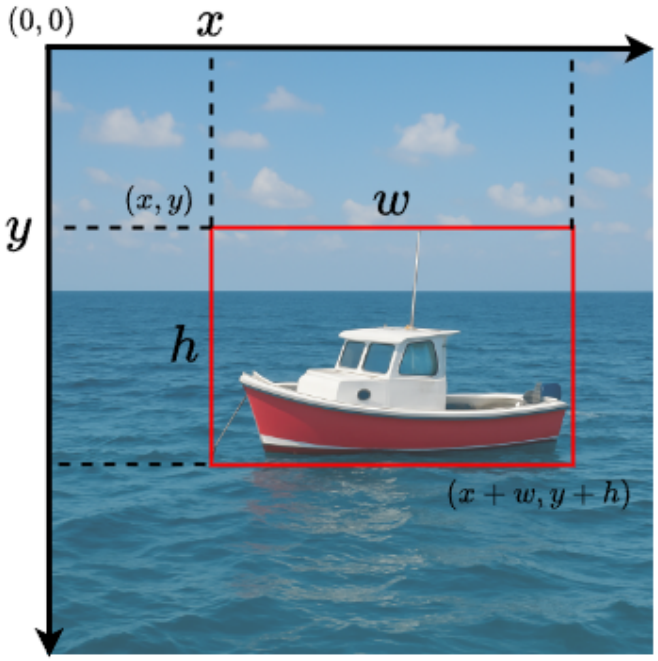

Figure 2.3: Bounding-box geometry with origin (x, y) and size (w, h).

Two architectural families dominate contemporary literature. Two-stage detectors first generate class-agnostic region proposals and then refine each proposal with a second head that performs classification and box regression. The R-CNN lineage, beginning with R-CNN (Girshick et al. 2014), followed by Fast R-CNN (Girshick 2015), Faster R-CNN (Ren et al. 2015), and culminating in Mask R-CNN (He et al. 2017), exemplifies this strategy. By decoupling localisation from recognition, these systems allocate more representational capacity to the second stage and achieve top accuracy.

Single-stage detectors such as YOLO (Redmon et al. 2016), SSD (Liu et al. 2016), RetinaNet (Lin et al. 2017), and FCOS (Tian et al. 2019) eliminate the proposal phase and predict class probabilities and offsets directly on dense feature maps. More recently, transformer-based designs like DETR (Carion et al. 2020) have further simplified the detection pipeline by replacing hand-designed components with global self-attention mechanisms. This unification yields real-time throughput and a compact footprint suitable for edge deployment, while innovations such as focal loss, anchor-free priors, and multi-scale fusion have narrowed the precision gap between one-stage and two-stage detectors .

Region Proposal Network

The Region Proposal Network (Ren et al. 2015) inserts a learnable proposal generator into the two-stage pipeline. A backbone produces a feature map $\mathbf{F} \in \mathbb{R}^{H \times W \times C}$. A small sliding convolution traverses $\mathbf{F}$ and at every spatial location (u, v) emits objectness logits $\{s_{u,v,a}\}_{a=1}^{A}$ and bounding-box offsets $\{(t_x, t_y, t_w, t_h)_{u,v,a}\}_{a=1}^{A}$.

Anchors provide translation-invariant reference boxes. Each anchor is a rectangle centred at (u, v) whose scale $s \in \mathcal{S}$ and aspect ratio $r \in \mathcal{R}$ are selected from predefined sets, yielding $A = |\mathcal{S}||\mathcal{R}|$ anchors per location. This dense tiling covers the image with minimal manual design and allows the network to regress precise boxes from coarse priors.

Figure 2.4 illustrates this mechanism using an image of a red apple as an object. On the left, the original image is shown. A CNN processes this image to extract spatial features, producing a feature map with reduced resolution. On the right, each location of this feature map corresponds to a grid cell in the input image, and multiple anchor boxes—shown in different colors to represent different scales and aspect ratios—are placed over each cell. These anchors serve as initial hypotheses for object location and shape, which are refined during training via regression to match ground-truth boxes.

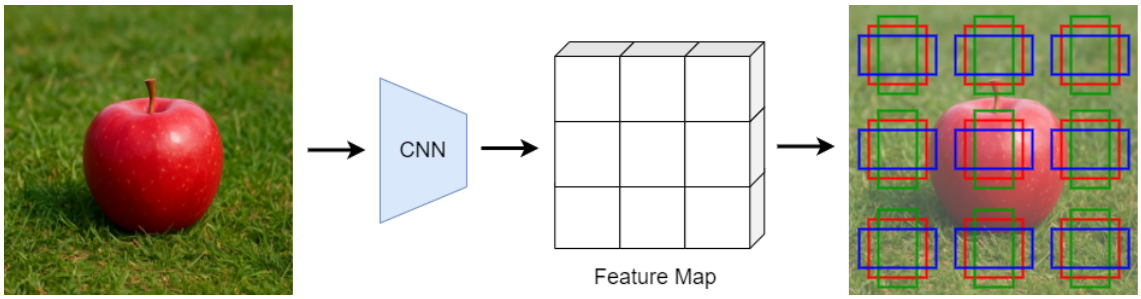

Figure 2.4: Anchor with different aspect ratios for an image

To enable the model to adjust rigid anchor priors into more precise object proposals, the bounding-box regression head predicts a set of offsets relative to each anchor. These offsets encode both translational shifts and scale adjustments that transform an anchor (x_a, y_a, w_a, h_a) into a refined bounding box (x, y, w, h). Translational offsets are normalized by the anchor dimensions, while scale changes are expressed logarithmically to model multiplicative differences. This formulation stabilizes training by keeping regression targets close to zero and facilitates learning across varying object sizes and aspect ratios. The transformation is defined in Equation 2.2.

$$t_x = \frac{x - x_a}{w_a}, \qquad t_y = \frac{y - y_a}{h_a}, \qquad t_w = \log \frac{w}{w_a}, \qquad t_h = \log \frac{h}{h_a}. \tag{2.2}$$

To assign training labels, each anchor is compared to ground truth boxes using Intersection over Union (IoU), a metric that quantifies the overlap between two regions as the ratio of their intersection to their union. Anchors with IoU above a positive threshold are labeled as positives, those below a negative threshold as negatives, and intermediate cases are ignored. This selective labeling exposes the model to both informative hard negatives and reliable positives while discarding ambiguous examples.

Operating directly over the convolutional feature map extracted by the backbone, the RPN applies a sliding window mechanism over this feature space. For each position in the sliding window, a 256-dimensional intermediate layer is computed, which serves as a shared representation for both classification and regression tasks. Attached to each window location are k predefined anchor boxes, covering various scales and aspect ratios. The classification layer outputs $2k$ scores, representing the probability of each anchor being an object versus background. Simultaneously, the regression layer predicts $4k$ coordinate adjustments to refine each anchor box, improving its alignment with potential objects. Figure 2.5 provides an overview of this process.

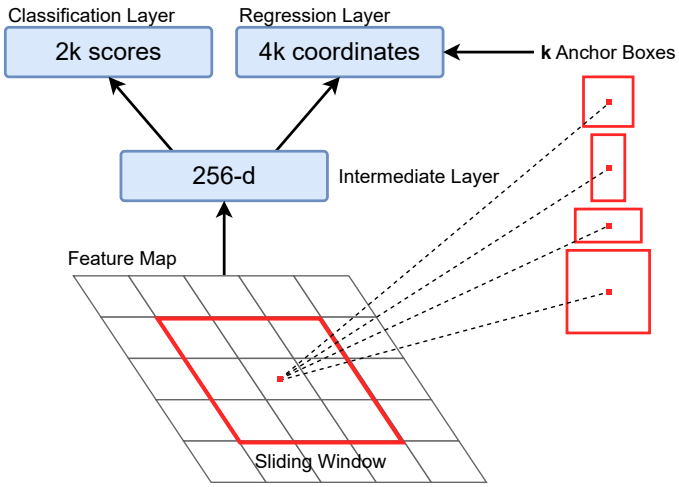

Figure 2.5: Region Proposal Network. A backbone creates a feature map, a convolutional head predicts objectness and offsets for each anchor.

After scoring all anchors, the RPN applies Non-Maximum Suppression (NMS), a technique that eliminates redundant proposals by retaining only the highest scoring region among those with high mutual IoU. The top N remaining proposals are then forwarded to the second stage detector.

Training Objective and Evaluation Metrics

Detector optimisation minimises the composite loss shown in Equation 2.3,

$$\mathcal{L} = \mathcal{L}_{\text{loc}} + \alpha\,\mathcal{L}_{\text{cls}} + \beta\,\mathcal{L}_{\text{conf}}, \tag{2.3}$$

where $\mathcal{L}_{\text{loc}}$ supervises bounding-box regression, $\mathcal{L}_{\text{cls}}$ supervises category prediction, and $\mathcal{L}_{\text{conf}}$ applies when an objectness branch is present. As shown in Equation 2.4, localisation quality is measured by the default Intersection-over-Union between a predicted box B and its matched ground truth B^*,

$$\text{IoU}(B,B^*) = \frac{|B \cap B^*|}{|B \cup B^*|}. \tag{2.4}$$

Equation 2.4 defines the overlap used both to assign positive and negative samples

during training and to score predictions during evaluation. Two-stage heads typically employ cross entropy for $\mathcal{L}_{\text{cls}}$, whereas single-stage detectors favour focal loss to attenuate easy negatives. Binary cross entropy guides $\mathcal{L}_{\text{conf}}$ using IoU-based targets.

Performance is reported using the mean Average Precision (mAP) metric. Average Precision integrates precision–recall curves for each class, and the mean across classes yields mAP. A prediction counts as correct if its IoU with ground truth exceeds a threshold, usually 0.50 for historical PASCAL VOC (Everingham et al. 2010) or multiple thresholds from 0.50 through 0.95 in steps of 0.05 for COCO (Lin et al. 2014). These metrics incentivise detectors to achieve both accurate localisation and correct classification under varying overlap requirements.

2.2 Representing Text

Statistical language models manipulate continuous vectors, yet human language is inherently discrete. This section follows the transformation pipeline from raw bytes to low-dimensional embeddings, dividing the discussion into four logically independent subsections: tokenization, one-hot encoding, dense embeddings, and objectives that endow those embeddings with meaning.

2.2.1 Tokenization

Tokenization transforms a stream of characters into discrete symbols that later map to vectors. Let $\mathcal{A}$ be a finite alphabet and let $\Sigma = \{\tau_1, \ldots, \tau_{|\Sigma|}\}$ denote the token vocabulary. As shown in Equation 2.20.

$$T : \mathcal{A}^* \longrightarrow \Sigma^*, \qquad s \mapsto (t_1, \ldots, t_n), \tag{2.5}$$

where s is the input string and $(t_1, \ldots, t_n)$ the resulting token sequence. To guarantee lossless reconstruction there must exist a decoding function $D : \Sigma^* \to \mathcal{A}^*$ satisfying $D \circ T = \text{id}$.

In practice a tokenizer must balance three tightly-coupled criteria. A large vocabulary reduces average sequence length but enlarges the embedding matrix; a small vocabulary has the opposite effect and increases computational cost in subsequent layers. Both extremes influence how often an unseen fragment is mapped to a special `UNK` symbol, which removes lexical information and harms generalisation. Designing T is therefore a question of locating a workable point on the surface spanned by vocabulary size, sequence length and out-of-vocabulary rate (Manning & Schutze 2003).

Granularity choices

Word-level tokenizers use whitespace and punctuation to approximate linguistic words. This approach yields short sequences but can require hundreds of thousands of embeddings and still leaves a non-zero `UNK` rate, particularly in morphologically rich languages (Dehdari & Duh 2016). At the opposite end, byte- or character-level tokenizers assign

every byte (or Unicode code point) a unique index; the vocabulary shrinks to a few hundred symbols and the UNK problem disappears, yet sequence length may increase by two orders of magnitude and each token carries minimal semantic content. Subword methods interpolate between these limits by discovering statistically frequent text fragments that can be recombined into new words, thereby offering a compromise on all three axes (Wu et al. 2016, Kudo & Richardson 2018).

Byte-Pair Encoding

The most widely used subword algorithm is Byte Pair Encoding (BPE) (Gage 1994, Sennrich et al. 2016). The procedure begins with the alphabet itself, $\Sigma_0 = \mathcal{A}$, and iteratively merges the single most frequent adjacent pair of symbols in the training corpus. If (a,b) is the pair selected at iteration k, the merged symbol $\langle ab \rangle$ is added to the vocabulary, the corpus representation is updated by replacing every occurrence of ab with $\langle ab \rangle$, and the process repeats. After K such merges the final vocabulary is given by Equation 2.6.

$$\Sigma_K = \Sigma_0 \cup \{\langle a_1 b_1 \rangle \dots \langle a_K b_K \rangle\} \tag{2.6}$$

The merge list is stored alongside the model so that encoding and decoding are deterministic and invertible. Frequent morphemes are typically created early, which shortens common sentences while still allowing any rare word to fall back on smaller fragments. Setting the target size $|\Sigma_K|$ in the range of thirty to fifty thousand tokens has been shown empirically to keep sequence lengths manageable without overloading memory in large scale deployments.

BPE represents a pragmatic midpoint on the granularity spectrum: sequences remain far shorter than byte-level encodings, the vocabulary remains orders of magnitude smaller than word-level schemes, and the UNK rate is effectively zero because every unseen string decomposes to characters. For these reasons BPE has become the de-facto standard tokenizer in contemporary neural-language pipelines.

2.2.2 One Hot Encoding

Given a fixed vocabulary Σ, each token is assigned a unique basis vector $e_i \in \mathbb{R}^{|\Sigma|}$ as defined in Equation 2.7.

$$\left(e_i\right)_j = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

These vectors are orthogonal, so all tokens are equidistant and the representation encodes no similarity structure. Moreover, the dimensionality scales linearly with the vocabulary size; a 50 000-token vocabulary results in vectors of dimension 50 000 for each word. One hot encodings therefore serve only as a lossless identity layer and must be compressed to enable the model to capture distributional structure.

2.2.3 Dense Embeddings

Neural models require a numerical representation of discrete linguistic symbols in order to perform gradient-based optimization. This representation is achieved by embedding tokens into a continuous vector space through a trainable projection (Incitti et al. 2023). Let Σ be a vocabulary of cardinality $|\Sigma|$. Each token $t_i \in \Sigma$ is initially represented by a one-hot vector $e_i \in \mathbb{R}^{|\Sigma|}$. An embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times |\Sigma|}$, where $d \ll |\Sigma|$, transforms this sparse representation into a dense vector $x_i \in \mathbb{R}^d$ through a simple matrix multiplication. This transformation is expressed in Equation 2.8.

$$x_i = \mathbf{E}e_i \tag{2.8}$$

The embedding x_i is a learned parameter that occupies a point in a continuous space whose geometry is shaped during training. The optimization process adjusts the embedding matrix $\mathbf{E}$ such that the positions of vectors in $\mathbb{R}^d$ reflect latent relationships present in the input distribution. This includes syntactic roles, semantic affinities, and contextual patterns. As a result, words that appear in similar contexts tend to be represented by vectors that are close in Euclidean or cosine distance.

Beyond mere proximity, the structure of the embedding space supports higher-order relationships. Clusters emerge for words belonging to similar categories, and directions in space may correspond to interpretable transformations, such as pluralization or gender shifts. Empirical evidence shows that linear operations on embedding vectors can approximate analogical reasoning. For instance, the vector difference between embeddings for "king" and "man" closely aligns with the difference between "queen" and "woman", suggesting that the space encodes relational structure in its directional geometry.

The dimensionality d plays a central role in shaping the capacity of the embedding space. A higher dimension increases representational power, but may lead to overfitting or redundancy. Conversely, a lower dimension enforces compression, potentially discarding subtle distinctions. The choice of d is therefore a design decision guided by the complexity of the linguistic domain and the scale of the dataset.

2.3 Transformers

The Transformer architecture introduced by Vaswani (2017) marked a fundamental shift in the modeling of sequential data. By replacing recurrence with self-attention and position-wise feed-forward networks, the architecture eliminated the sequential bottlenecks of prior models, enabling global context integration and significantly improved parallelism during training and inference. Each token in the sequence attends to all others simultaneously, which allows for direct modeling of long-range dependencies and syntactic or semantic patterns across entire documents. The complete structure of the Transformer encoder-decoder model is illustrated in Figure 2.6.

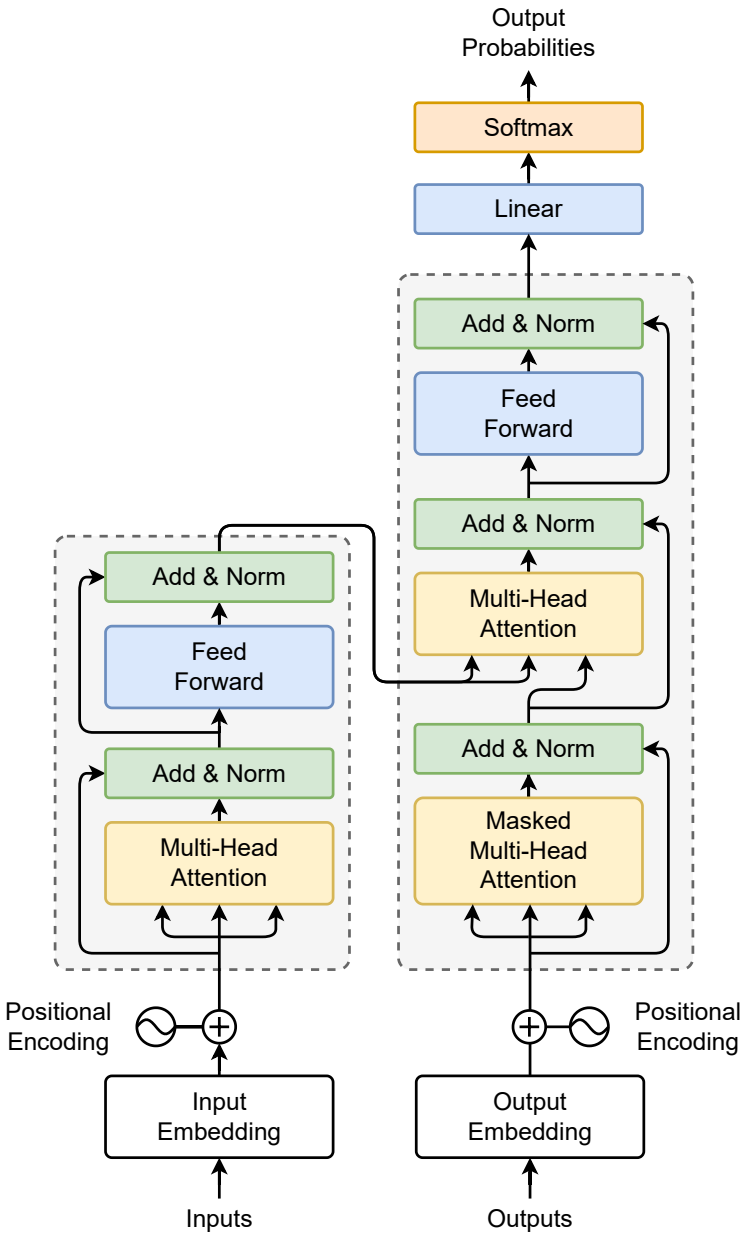

Figure 2.6: Transformer architecture comprising an encoder stack and a decoder stack.

2.3.1 Token Embeddings and Positional Encoding

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a matrix where each row $\mathbf{x}_i$ represents the embedding of the i-th token in a sequence of length n, and d is the embedding dimension. Since the self-attention mechanism is inherently permutation-invariant, the model requires an additional mechanism to encode positional information. This is introduced via a positional encoding matrix

$\mathbf{P} \in \mathbb{R}^{n \times d}$ that is added element-wise to $\mathbf{X}$, forming the input $\mathbf{E} = \mathbf{X} + \mathbf{P}$.

The original Transformer uses a fixed sinusoidal function to define the positional encoding. Each dimension of the positional vector follows a distinct frequency, with even and odd indices alternating between sine and cosine. The resulting structure is defined in Equation (2.9), where ω_k represents the frequency component associated with the k-th dimension.

$$P_{i,2k} = \sin(i\omega_k), \qquad P_{i,2k+1} = \cos(i\omega_k), \qquad \omega_k = 10000^{-2k/d} \tag{2.9}$$

This construction enables the model to infer relative distances between positions through linear transformations. The use of sinusoidal curves ensures that the encoding generalizes to sequence lengths beyond those observed during training.

2.3.2 Scaled Dot-Product Attention

Each layer of the Transformer projects the combined input matrix $\mathbf{E}$ into three distinct matrices: queries $\mathbf{Q} = \mathbf{E}\mathbf{W}^Q$, keys $\mathbf{K} = \mathbf{E}\mathbf{W}^K$, and values $\mathbf{V} = \mathbf{E}\mathbf{W}^V$, where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d_k}$ are learned weight matrices. The core operation that enables contextualization is the scaled dot-product attention, defined in Equation (2.10):

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.10}$$

where the dot products between queries and keys determine attention weights, while the division by $\sqrt{d_k}$ stabilizes gradients when d_k is large. The softmax operator normalizes the scores across all keys, ensuring that the attention weights form a probability distribution. This mechanism introduces quadratic complexity in sequence length, yet yields a fully connected token interaction graph, enabling dense long-range dependencies.

2.3.3 Multi-Head Attention and Feed-Forward Layers

Instead of performing attention computation once, the Transformer uses h independent heads to capture different types of interactions. Each head computes scaled dot-product attention in a lower-dimensional subspace and the results are concatenated and linearly transformed. This multi-head mechanism is defined in Equation (2.11).

$$\text{MHA}(\mathbf{E}) = \text{Concat}(\mathbf{Z}_1, \ldots, \mathbf{Z}_h)\mathbf{W}^O \tag{2.11}$$

where $\mathbf{Z}_j = \text{Attn}(\mathbf{E}\mathbf{W}_j^Q, \mathbf{E}\mathbf{W}_j^K, \mathbf{E}\mathbf{W}_j^V)$ and $\mathbf{W}^O$ is the output projection matrix. This formulation enhances the model's ability to capture multifaceted dependencies by distributing representational capacity across multiple heads.

Following the attention block, a position-wise feed-forward network is applied independently to each token embedding. This component, described in Equation (2.12), consists of two linear transformations with a nonlinearity in between.

$$\text{FFN}(\mathbf{z}) = \sigma(\mathbf{z}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{2.12}$$

The activation function σ is often a ReLU or a gated linear unit, depending on the architecture. Residual connections (He et al. 2016) and layer normalization (Ba et al. 2016) are applied before both the attention and feed-forward sublayers, ensuring signal stability and efficient gradient propagation through deep stacks.

2.3.4 Encoder-Only Transformers

Although the original Transformer was introduced with both encoder and decoder modules for autoregressive tasks such as translation, most modern language understanding systems utilize only the encoder. Encoder-only architectures are trained to produce contextual embeddings that capture semantic and syntactic information from input sequences. These models are used extensively for classification, retrieval, and representation learning.

Pretraining objectives vary and include masked language modeling, contrastive learning, and sentence-level prediction. The resulting models encode high-dimensional representations that preserve meaning, context, and discourse-level information. Architectures such as BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019), and SimCSE (Gao et al. 2021) exemplify this paradigm and serve as foundations for downstream tasks explored in later chapters.

2.4 Cross-Modal Alignment

Cross-modal alignment endows heterogeneous sensory streams with a unified representational basis that enables downstream reasoning without bespoke task-specific heads (Baltrušaitis et al. 2018). Instead of treating vision and language as disjoint modalities, contemporary frameworks learn a common geometric manifold on which visual and textual concepts occupy proximate directions. This section explains the architectural mechanisms that realize such alignment, the metric-learning objectives that sculpt the manifold, the role of similarity kernels and temperature in probabilistic calibration, and the downstream behaviours, such as retrieval, localization, and zero-shot transfer, that emerge from a well-structured latent space (Alayrac & others 2022).

2.4.1 Shared Latent Spaces

Cross-modal reasoning relies on a latent manifold where heterogeneous sensory inputs admit comparable geometric interpretations. Let $\mathbf{v}_i \in \mathbb{R}^{d_v}$ denote the visual feature extracted by a convolutional or Transformer backbone and $\mathbf{t}_i \in \mathbb{R}^{d_t}$ the linguistic embedding produced by a text encoder. Modality-specific projection heads $g_v : \mathbb{R}^{d_v} \to \mathbb{R}^d$ and $g_t : \mathbb{R}^{d_t} \to \mathbb{R}^d$ map these signals onto a shared d-dimensional space, as shown in Equation 2.13.

$$\mathbf{z}_i^v = g_v(\mathbf{v}_i), \qquad \mathbf{z}_i^t = g_t(\mathbf{t}_i), \tag{2.13}$$

where $\mathbf{z}_i^v$ and $\mathbf{z}_i^t$ are ℓ_2-normalised so that inner products coincide with cosine similarity. Orthogonal transformations, batch normalisation, and learned scaling parameters refine the alignment (Radford et al. 2021). Figure 2.7 depicts this architecture and illustrates how independent visual and textual representations converge in a unified geometric space suitable for cross-modal reasoning.

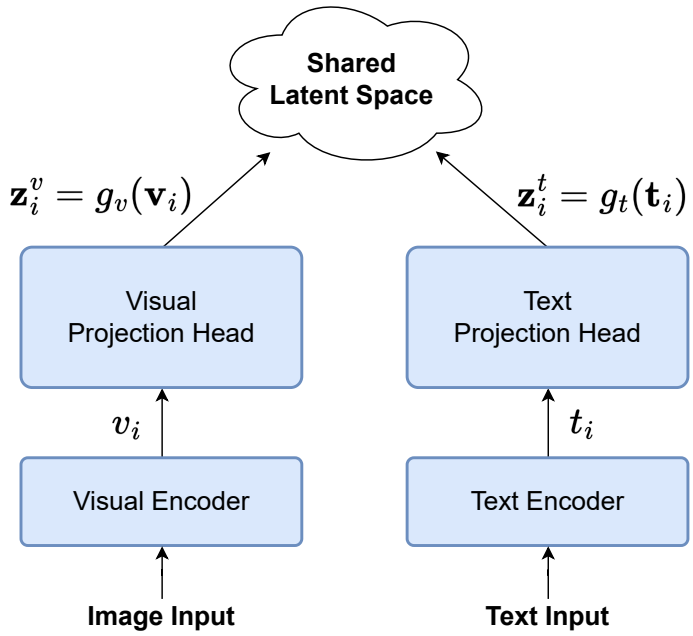

Figure 2.7: Architecture for cross-modal embedding alignment. Independent visual and textual encoders feed projection modules that produce embeddings, enabling direct geometric comparison between modalities.

2.4.2 Contrastive Learning Objectives

Metric-learning objectives attract paired samples and repel mismatched ones, thereby imposing a discriminative geometry on the shared manifold. Four losses: InfoNCE, contrastive (pairwise), triplet, and cosine embedding, that are prominent in cross-modal literature; each loss is examined in this Section.

Contrastive Loss

As shown in Equation 2.14, the classic pairwise contrastive loss tightens positive image–text pairs while pushing mismatched pairs beyond a Euclidean margin m; by explicitly encoding binary similarity labels y_i, it carves a metric space where semantic neighbourhoods are demarcated by the margin, yielding robust separation in coarse-grained datasets but struggling in fine-grained settings where a single negative radius is insufficient (Hadsell et al. 2006).

$$\mathcal{L}_{\text{Ctr}} = \frac{1}{N}\sum_{i=1}^{N} y_i \,\|\mathbf{z}_i^v - \mathbf{z}_i^t\|_2^2 + (1-y_i)\,\max\bigl(0,\, m - \|\mathbf{z}_i^v - \mathbf{z}_i^t\|_2\bigr)^2, \tag{2.14}$$

Triplet Loss

Equation 2.15 generalises pairwise supervision by ranking an anchor closer to its positive than to a negative by at least a margin α; this hinge formulation sharpens local ordering of embeddings, and when coupled with hard-negative mining can produce highly discriminative manifolds—albeit at the cost of additional sampling complexity and sensitivity to mining heuristics (Schroff et al. 2015).

$$\mathcal{L}_{\text{Tri}} = \frac{1}{N}\sum_{i=1}^{N} \max\bigl(0,\, \langle \mathbf{z}_i^a, \mathbf{z}_i^p\rangle - \langle \mathbf{z}_i^a, \mathbf{z}_i^n\rangle + \alpha\bigr), \tag{2.15}$$

Cosine Embedding Loss

Focusing on angular rather than Euclidean relationships, the cosine embedding loss of Equation 2.16 penalises deviations in cosine similarity, naturally complementing the unit-norm projections produced earlier in Equation 2.13; by being scale-invariant, it excels in high-dimensional representation learning where vector magnitudes vary widely, and the threshold β provides a tunable angular margin separating unrelated pairs while preserving correlated ones (Wang et al. 2014).

$$\mathcal{L}_{\text{Cos}} = \frac{1}{N}\sum_{i=1}^{N} y_i\bigl(1 - \langle \mathbf{z}_i^v, \mathbf{z}_i^t\rangle\bigr) + (1-y_i)\,\max\bigl(0,\, \langle \mathbf{z}_i^v, \mathbf{z}_i^t\rangle - \beta\bigr), \tag{2.16}$$

InfoNCE and CLIP Objective

The InfoNCE objective shown in Equation 2.17 is a contrastive loss that encourages paired samples to have similar embeddings while pushing apart unpaired ones. It maximises mutual information between modalities by comparing each positive pair against all other elements in the minibatch. This bidirectional formulation handles both visual to text and text to visual retrieval symmetrically. A learnable temperature parameter τ controls the sharpness of the similarity distribution, and large batch sizes introduce diverse negatives that improve generalisation. This objective underlies models such as CLIP, which jointly trains an image encoder and a text encoder to align vision and language representations in a shared embedding space (Oord et al. 2018, Radford et al. 2021).

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{2N}\sum_{i=1}^{N}\Biggl[\log\frac{\exp(\langle \mathbf{z}_i^v, \mathbf{z}_i^t\rangle/\tau)}{\sum_{j=1}^{N}\exp(\langle \mathbf{z}_i^v, \mathbf{z}_j^t\rangle/\tau)} + \log\frac{\exp(\langle \mathbf{z}_i^t, \mathbf{z}_i^v\rangle/\tau)}{\sum_{j=1}^{N}\exp(\langle \mathbf{z}_i^t, \mathbf{z}_j^v\rangle/\tau)}\Biggr] \tag{2.17}$$

2.4.3 Cosine Similarity and Zero-Shot Applications

Similarity in the joint embedding space is quantified by the cosine kernel of Equation 4.9,

$$\mathrm{sim}(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|_2 \, \|\mathbf{b}\|_2}, \tag{2.18}$$

which reduces to an inner product when unit-normalised projections from Equation 2.13 are used. Scores of $\mathrm{sim} = 1$ denote maximal semantic alignment, values near 0 imply orthogonality, and negative scores encode explicit dissimilarity. Because contrastive pre-training shapes the embedding manifold so that semantically related samples cluster angularly, a text-to-image retrieval query embedding $\mathbf{z}_q^t$ can simply sort a gallery $\{\mathbf{z}_j^v\}_{j=1}^M$ by descending similarity as prescribed in Equation 4.9; the highest-ranking items emerge as the most semantically consistent without any additional task-specific parameters (Alayrac & others 2022). The same geometric property endows the model with zero-shot transfer: lightweight linear probes or nearest-neighbour classifiers inherit meaningful decision boundaries from the pre-trained space, enabling image classification, captioning, or visual question answering to achieve significant accuracy even when their label sets differ entirely from the data used during training. Thus, cosine similarity functions both as the metric governing cross-modal correspondence and as the mechanism that enables generalization to unseen tasks (Radford et al. 2021).

2.5 Interpreters and Compilers

In order to enable logical queries over detected objects, our framework leverages parsing and semantic analysis techniques derived from compiler theory. This ensures that natural language prompts can be transformed into structured, executable queries against the neural representation of the scene. This section introduces the theoretical foundations of interpreters, compilers, and parsing systems, which serve as the backbone for the symbolic reasoning component of our framework.

The transformation of source code into executable instructions or actions is fundamentally governed by the principles of formal language theory. Both interpreters and compilers rely on these principles to process and understand program text. This section elaborates on the key theoretical concepts and practical stages involved in this process, from lexical analysis to semantic interpretation and execution strategies (Aho et al. 2007).

A typical interpreter or compiler pipeline is organized as a sequence of stages that progressively transform raw source code into structured, validated, and executable forms. The first stage, **lexical analysis**, uses a lexer to scan the input and convert it into a stream of tokens—atomic units like keywords, identifiers, and symbols. Next, the **parser** applies a context-free grammar to this token stream to construct a syntactic structure, usually represented as a parse tree or abstract syntax tree (AST). Following syntax validation, **semantic analysis** verifies context-sensitive properties such as variable scope, type correctness, and function resolution. Once the code passes these checks, a compiler proceeds with **intermediate code generation**, optimization, and finally emits machine code or bytecode.

In contrast, interpreters directly execute the validated AST or intermediate representation step-by-step, without producing a standalone binary. This pipeline ensures that high-level source code is both syntactically and semantically coherent before execution or translation. The complete flow of the pipeline of an interpreter is illustrated in Figure 2.8, where each transformation step receives a structured input and produces a refined output for the next stage.

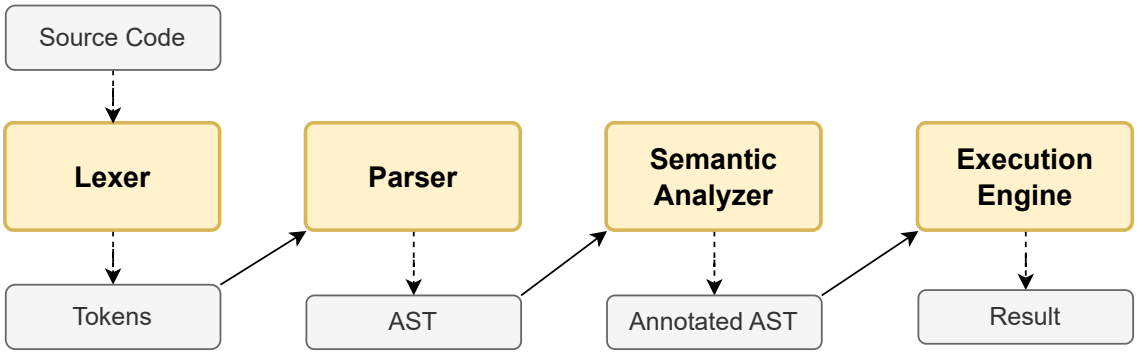

Figure 2.8: Interpreter pipeline. Source code is transformed through lexical, syntactic, and semantic analysis before being executed by the interpreter.

2.5.1 Formal Languages and Grammars

The theoretical foundations of interpreters and compilers originate in the study of *formal languages*. Given a finite alphabet Σ, a formal language L is any subset of the Kleene closure Σ^*, the set of all finite-length strings, including the empty string, that can be formed from symbols in Σ. The syntactic well-formedness of the strings in L is prescribed by a *formal grammar* G. For most programming languages a *context-free grammar* (CFG) suffices; it is defined, as shown in Equation 2.19, by the quadruple:

$$G = \langle N, \Sigma, P, S \rangle, \tag{2.19}$$

where N is the finite set of non-terminal symbols, Σ the finite set of terminal symbols called tokens, P the finite set of production rules mapping N to $(N \cup \Sigma)^*$, and $S \in N$ the start symbol. Repeated application of rules in P, beginning with S, induces the derivation relation $\Rightarrow^*$ that generates exactly the strings of the language $L(G)$; the parser later exploits this relation to verify syntactic correctness (Hopcroft et al. 2019).

Although CFGs are effective at capturing the hierarchical structure of programs, they cannot handle context-sensitive rules—such as making sure a variable is declared before it is used, or that two values in an operation have compatible types. To address this, attribute grammars enrich the parse tree by attaching extra information, called attributes, to each node v_i. These attributes, denoted a_i, can be either synthesised from the values of child nodes or inherited from parent or sibling nodes. By combining these attributes with the production rules from Equation 2.19, attribute grammars allow compilers to check not only the structure but also the semantics of the code during compilation (Knuth 1968).

2.5.2 Lexical Analysis and Tokenization

The initial phase of processing source code is lexical analysis, also known as scanning or tokenization. This stage reads the raw input stream, which is a sequence of characters $s = c_1 c_2 \ldots c_n$, and transforms it into a sequence of tokens $t_1, t_2, \ldots, t_k$. Each token represents a syntactically meaningful lexical unit, such as a keyword like `if` or `while`, an identifier like variable names, an operator like + or =, or a literal value like numbers or strings (Aho et al. 2007). The complete transformation process is illustrated in Figure 2.9.

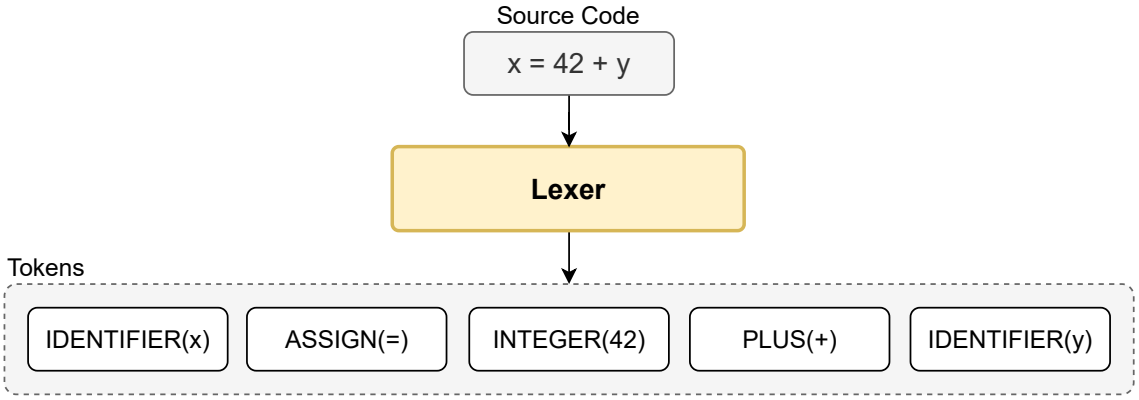

Figure 2.9: Lexer process. Source code is transformed into a stream of tokens.

The patterns that define the valid forms of tokens are typically specified using a finite set of regular expressions, $\mathcal{R} = \{r_1, \ldots, r_m\}$, where each r_i describes a particular category of token. Based on the theory of automata, these regular expressions can be systematically converted into a finite automaton that recognizes the specified patterns. A common approach involves constructing a non-deterministic finite automaton (NFA) using Thompson's construction, followed by the subset construction algorithm to convert the NFA into an equivalent deterministic finite automaton (DFA), denoted here as δ (Aho et al. 2007).

The lexical analyzer, effectively implementing the state transitions of the DFA δ, performs the mapping shown in Equation 2.20:

$$\mathcal{T} : \Sigma_{\text{char}}^* \longrightarrow \mathcal{C}^*, \qquad \mathcal{T}(s) = \langle c(t_1), \ldots, c(t_k) \rangle, \tag{2.20}$$

where Σ_{char} is the alphabet of characters in the source language, $\mathcal{T}$ represents the tokenization function, $\mathcal{C}^*$ is the resulting sequence of token classes (or types), and $c(t_i)$ denotes the specific class (e.g., IDENTIFIER, KEYWORD, OPERATOR) associated with the recognized token t_i. This process partitions the character stream efficiently, typically operating in time linear to the length of the input s. During this phase, lexically irrelevant elements, such as whitespace (spaces, tabs, newlines) and comments, are usually identified and discarded, as they do not contribute to the syntactic structure recognized by the parser.

The correctness and feasibility of lexical analysis rely heavily on the mathematical properties of regular languages. Specifically, the closure properties of regular languages under operations like union, concatenation, and Kleene star allow complex token patterns

to be described compositionally. Furthermore, the fundamental equivalence between regular expressions and finite automata guarantees that any set of patterns describable by regular expressions can be recognized efficiently by a deterministic finite automaton.

2.5.3 Syntax Analysis and Parsing

Following lexical analysis, the sequence of tokens $\langle c(t_1), \ldots, c(t_k) \rangle$ is passed to the syntax analyzer, or parser. The primary goal of parsing is to determine whether this token sequence conforms to the syntactic structure defined by the language's grammar G. If the sequence is grammatically valid, the parser typically constructs a data structure that explicitly represents the hierarchical relationships between the tokens, reflecting the derivation process according to G. This structure is commonly known as a parse tree or, more frequently in practice, an AST, denoted here as τ (Aho et al. 2007). The overall parsing process is illustrated in Figure 2.10.

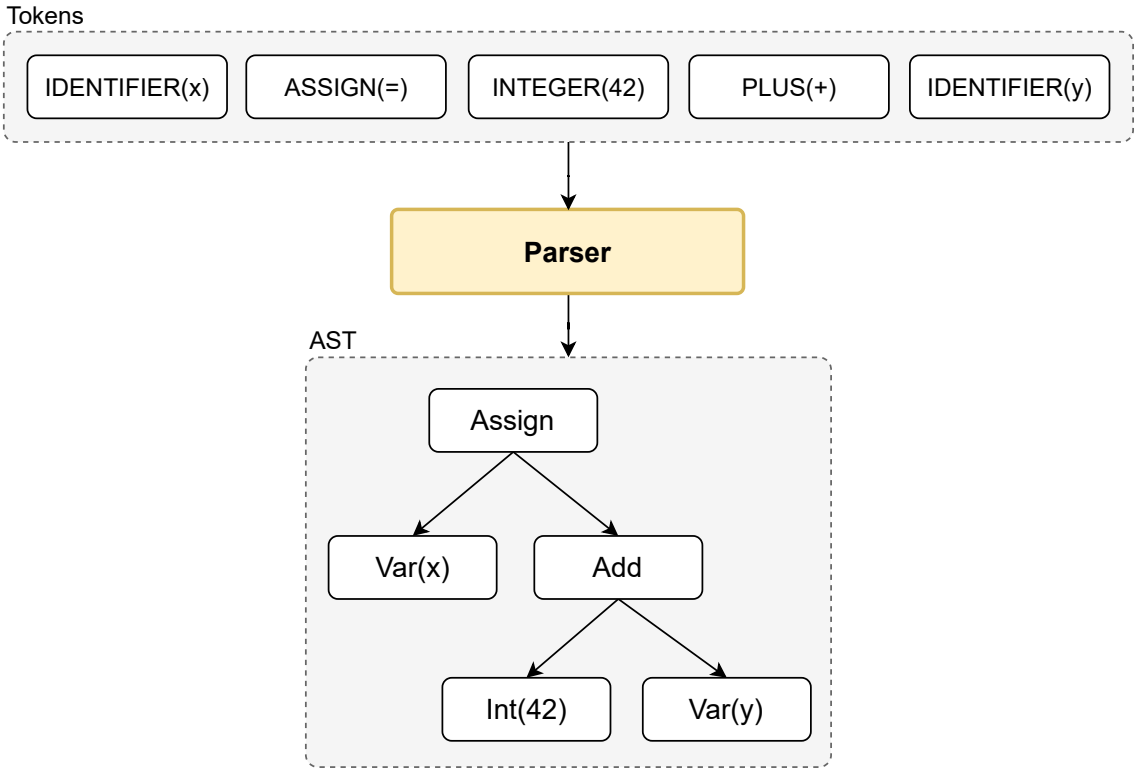

Figure 2.10: Parser process. Tokens are transformed into an AST that captures the program's structure.

Numerous parsing algorithms have been developed, broadly categorized into top-down and bottom-up strategies (Grune & Jacobs 2012).

- **Top-down parsing** methods, such as recursive descent or predictive parsers like LL(k), start from the grammar's start symbol S and attempt to derive the input token sequence. These methods use the next k input tokens (the lookahead) to predict which production rule to apply at each step.

- **Bottom-up parsing** methods, such as shift-reduce parsers like LR(k) and its variants (SLR, LALR), work by attempting to reconstruct the derivation in reverse. They read tokens (shift) onto a stack and, whenever the top portion of the stack matches the right-hand side of a production rule, they replace that sequence with the corresponding non-terminal (reduce). This process continues until the entire input is consumed and the stack contains only the start symbol S.

LR parsers, widely used due to their ability to handle a large class of grammars, operate using a state machine (the LR automaton). The states in this automaton typically represent sets of LR items (productions marked with a position indicating parsing progress). Given a current state (representing the viable prefix γ processed so far, implicitly stored on the stack) and the lookahead token(s) ℓ, the LR automaton consults a parsing table (derived from the canonical collection of LR items, C). Based on the current state and lookahead, the table dictates whether to perform a 'shift' action (consuming the lookahead ℓ and transitioning to a new state) or a 'reduce' action (applying a specific production rule from P).

The overall parsing process can be abstracted as a function that maps the token sequence to a parse tree, as shown in Equation 2.21:

$$\mathcal{P} : \mathcal{C}^* \longrightarrow \mathcal{T}ree, \qquad \mathcal{P}\big(\langle c(t_1), \ldots, c(t_k)\rangle\big) = \tau, \tag{2.21}$$

where $\mathcal{T}ree$ represents the domain of possible parse trees (or ASTs). For a grammar G to be practically useful for programming languages, it should ideally be unambiguous, meaning that any valid input string has exactly one unique parse tree. Deterministic parsing algorithms, such as LL(k) and LR(k) for their respective grammar classes, guarantee the construction of this unique tree for unambiguous grammars. This structural uniqueness is essential for ensuring consistent semantic interpretation in subsequent phases (Aho et al. 2007).

Once the parser constructs the parse tree τ, the process extends beyond syntactic validation into semantic analysis. This stage enriches τ with context-sensitive information such as type constraints, symbol bindings, and scope resolution, such as elements that CFGs alone cannot capture. Formally, this involves computing attributes for each node $v \in V(\tau)$ via an evaluation function $\mathcal{E} : V(\tau) \to D$, where D represents semantic domains like types or values. These attributes are computed over a Directed Acyclic Graph (DAG) of dependencies to ensure correctness and termination. This annotated tree then serves as the foundation for the subsequent execution or logical evaluation within the symbolic engine.

2.5.4 Interpreters, Compilers, and Hybrid Execution Models

An interpreter directly executes the operations specified by the source program. It typically works by processing the source code, often via an intermediate representation like an AST, and performing the actions dictated by the program constructs in sequence. Conceptually, an interpreter implements a function $\theta : \Sigma^* \times \mathbb{S} \to \mathbb{S}$, which takes the source program and an initial machine state and produces a final state by simulating the pro-

gram's execution. The interpretation process often involves traversing the AST and invoking routines corresponding to the operations at each node (Aho et al. 2007).

A compiler, on the other hand, translates the source program p, written in a source language L_s, into an equivalent program π, written in a target language L_t. The target language is often machine code for a specific processor architecture or an intermediate bytecode for a virtual machine. This translation process can be represented shown in Equation 2.22.

$$\mathcal{C}_{s\to t} : L_s \longrightarrow L_t, \qquad \mathcal{C}_{s\to t}(p) = \pi. \tag{2.22}$$

The crucial aspect is semantic equivalence: the translated program π, when executed on the target platform, must produce the same results, or exhibit the same observable behavior, as the direct execution of the source program p. After compilation, the resulting target code π is executed separately, potentially on a different machine than the one used for compilation (Muchnick 1997).

In summary, the symbolic module of our framework is deeply grounded in the formal principles of compiler design, leveraging lexical analysis, parsing, and semantic evaluation to transform natural language queries into structured and executable logical plans. This pipeline is crucial for bridging high-level user intents with the low-level perceptual outputs generated by the neural module. By adopting an interpreter-like architecture based on ASTs, the system can execute compositional, transparent, and logically grounded queries over the visual scene. This design choice not only enables robust interaction with complex, multi-attribute prompts but also ensures that the reasoning process remains interpretable and extensible, fully aligned with the modular neuro-symbolic architecture proposed in this thesis.

Chapter 3

Related Works

This chapter reviews the main research directions that form the foundation of this work, focusing on advances in object detection, multimodal reasoning, and neuro-symbolic AI. While deep learning-based object detectors have achieved remarkable success in perceptual tasks, they remain fundamentally limited in handling logical inference, relational reasoning, and compositional generalization. Concurrently, the evolution of visual-language models has expanded the capabilities of AI systems in integrating textual and visual information, such as Open-Vocabulary Object Detection, yet often without offering interpretability or symbolic reasoning. In response to these limitations, the neuro-symbolic AI paradigm has emerged as a promising approach that combines neural perception with symbolic logic, enabling models to execute structured, logical queries over visual scenes. This chapter situates the proposed framework within this broader context, highlighting how it builds upon and extends prior work to address the challenges of compositionality, interpretability, and flexible reasoning in vision-language tasks.

3.1 Object Detection and Deep Learning Foundations

Object detection has undergone a transformative evolution, largely propelled by advancements in deep learning. Early CNNs based pipelines, such as Faster R-CNN (Ren et al. 2015), YOLO (Redmon et al. 2016), and SSD (Liu et al. 2016), established fundamental architectures for achieving accurate and efficient localization of objects within images. These models demonstrated the power of learned hierarchical features for visual recognition tasks. Mask R-CNN (He et al. 2017) further extended the capabilities of object detection by simultaneously performing instance segmentation, thereby providing pixel-level masks for detected objects. More recent innovations have focused on specific challenges, such as handling rotated objects. Oriented R-CNN (Xie et al. 2024) addressed this by introducing rotation-aware region proposals, significantly improving the detection of objects with arbitrary orientations. Additionally, MADet (Xie et al. 2023) proposed a mutual-assistance regression mechanism to jointly optimize the classification and localization branches of a one-stage detector, leading to enhanced performance. While these deep learning models have achieved remarkable success in perceptual accuracy, they typically lack explicit mechanisms for relational reasoning or symbolic logic. This limitation hinders their effectiveness in tasks that require a deeper understanding of context or the

compositional nature of visual scenes.

3.2 Multimodal Reasoning and Visual-Language Models

Visual Question Answering has emerged as a prominent benchmark for evaluating the ability of models to reason about visual content based on natural language queries. Initial approaches in VQA often involved fusing features extracted from CNNs for images and Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM), for encoding questions. However, subsequent research has highlighted the importance of aligning information across modalities. Attention mechanisms, as employed in models like LXMERT (Tan & Bansal 2019) and ViLBERT (Lu et al. 2019), have proven crucial for selectively focusing on relevant parts of the image and the question, leading to improved reasoning capabilities. VQACL (Zhang, Zhang & Xu 2023) introduced the concept of continual learning to VQA, enabling models to adapt to evolving multimodal data streams and improve cognitive reasoning over time. Xiao et al. (Xiao & Huang 2023) explored the use of explicit attention mechanisms to enhance cross-modal alignment between visual and textual features, further boosting VQA performance. Comprehensive surveys, such as the one by Li et al. (Li, Yang, Geng, Zhou, Ding & Nian 2024), have documented the evolution of VQA methodologies, tracing the progression from basic encoding strategies to sophisticated generative and attention-based approaches capable of intricate query understanding.

The advent of Transformer-based architectures has significantly impacted the field of VQA and related visual-language tasks. Their ability to model long-range dependencies and capture complex contextual relationships has made them a cornerstone of modern VQA systems. Layout-aware Transformers (LaTr) (Biten et al. 2022) demonstrated the importance of incorporating spatial context, particularly for tasks involving scene text understanding within images. Compact Transformer models, such as LiT-4-RSVQA (Sundar & Goel 2023), have been developed to address the need for efficient VQA solutions in resource-constrained environments. The mGTE series (Zhang et al. 2024) has further advanced the capabilities of text encoders by enabling them to handle multilingual inputs and long-context text representations, which is crucial for complex cross-modal reasoning scenarios. These works collectively underscore the vital role of robust and versatile text encoders in facilitating effective communication and reasoning between visual and linguistic modalities, although they often lack explicit mechanisms for symbolic reasoning or providing interpretable explanations.

3.3 Open-Vocabulary Object Detection

Open-vocabulary object detection has rapidly evolved through the integration of vision-language pretraining and prompt-guided inference. A seminal work in the field, ViLD (Gu et al. 2022), introduced vision-language knowledge distillation from CLIP into object detectors, laying the groundwork for aligning visual and semantic spaces. OWL-ViT (Minderer et al. 2022) extended this paradigm with a scalable transformer-based architecture and

enabled zero-shot generalization to novel categories. RegionCLIP (Zhong et al. 2022) advanced region-level alignment between image patches and language, improving localization for unseen categories. DetCLIPv2 (Yao et al. 2023) further scaled OVOD training by enhancing word-region correspondence via contrastive learning. Meanwhile, CoDet (Ma et al. 2023) introduced co-occurrence-aware alignment strategies to better ground textual prompts in cluttered scenes. More recent approaches explored enhanced query mechanisms: OV-DINO (Wang et al. 2024) unified open-vocabulary training with selective fusion of language and vision features, and T-Rex2 (Jiang et al. 2024) leveraged visual-text prompt synergy to support generic object detection. AggDet (Zheng & Liu 2024) proposed a training-free framework that boosts performance via confidence aggregation from frozen backbones. At the intersection of segmentation and OVOD, CLIFF (Li, Liu, Ma & Yuan 2024) employed latent diffusion models for continual learning. Finally, CCKT-Det (Zhang et al. 2025) introduced cyclic contrastive knowledge transfer for robust generalization in unseen categories. Collectively, these works demonstrate the field's transition from static label classifiers to dynamic, prompt-driven systems capable of detecting arbitrary objects via textual guidance.

3.4 Neuro-Symbolic AI and Visual Reasoning

To overcome the limitations of purely neural models in tasks requiring abstract reasoning and logical inference, neuro-symbolic AI has emerged as a promising hybrid paradigm. This approach aims to integrate the strengths of neural networks for perception and pattern recognition with the power of symbolic systems for logical reasoning and knowledge representation. Early efforts in this direction explored connecting neural classifiers with rule-based logic systems (Garcez et al. 2019), laying the foundation for more sophisticated integrations. NS-CL (Mao et al. 2019) provided a compelling example of how neural object detectors, when coupled with a symbolic program execution engine, could effectively answer complex compositional visual queries. Models like Neural Theorem Provers (Rocktaschel & Riedel 2017) and Neural Logic Machines (Dong et al. 2019) have taken this integration further by embedding logical inference directly into differentiable neural modules, enabling end-to-end training and enhancing compositional generalization and interpretability.

Recent surveys have provided a broader perspective on the landscape of neuro-symbolic AI in the context of visual reasoning. Khan et al. (Khan & others 2023) offer a comprehensive survey of neuro-symbolic visual reasoning techniques, with a particular focus on the role of scene graphs and common-sense knowledge in achieving richer scene understanding. Colelough (Colelough & Regli 2024) systematically maps the field of neuro-symbolic AI, identifying key research directions, challenges, and opportunities. Mashayekhi et al. (Mashayekhi & Group 2024) highlight the benefits of incorporating symbolic constraints into neural models for visual tasks, advocating for hybrid modular designs that leverage the complementary strengths of both approaches. These works collectively demonstrate the significant potential of integrating symbolic reasoning with neural perception to tackle visual reasoning tasks that demand structured understanding and logical inference beyond the capabilities of purely data-driven models.

3.5 Compositionality and Structured Generalization

A key advantage of neuro-symbolic systems lies in their ability to handle compositional reasoning, where the understanding of complex concepts or queries arises from the combination of simpler components. For instance, the CLEVRER dataset (Yi et al. 2020) was specifically designed to evaluate models' ability to perform physical and causal reasoning in a compositional manner, often involving the integration of neural dynamics modeling with symbolic inference. KRISP (Marino et al. 2021) demonstrated how the integration of external symbolic knowledge can significantly improve the performance of VQA systems on real-world datasets, highlighting the benefits of establishing feedback loops between perception and reasoning. Logical Neural Networks (Serafini & Garcez 2016) further exemplify this by tightly integrating symbolic rules into the learning process, leading to enhanced consistency and generalization, particularly in scenarios requiring logical deductions.

While purely neural models have shown impressive performance on many benchmark datasets, they often struggle with extrapolation to unseen combinations of attributes or when faced with compositional queries that deviate from the training distribution (Li, Yang, Geng, Zhou, Ding & Nian 2024). In contrast, neuro-symbolic approaches offer inherent interpretability, improved data efficiency, and robust logical reasoning capabilities, making them well-suited for tasks requiring structured generalization. However, challenges remain in areas such as scalability to complex real-world scenarios, the complexity of integrating disparate neural and symbolic modules, and the intricacies of training such hybrid systems effectively (Colelough & Regli 2024).

3.6 Positioning of This Work

The framework proposed in this thesis directly builds upon the foundational concepts and advancements discussed in the related works. Our approach integrates a neural object detector with a dedicated symbolic module that is designed to execute logical queries over the perceptual outputs generated by the detector. A key distinction of our system compared to traditional VQA or object detection systems trained on compound queries is its reliance on a custom query language and compositional logic. This design allows our system to interpret multi-attribute prompts without requiring explicit training on every possible combination of attributes. This inherent compositionality, facilitated by the symbolic reasoning module, offers significant advantages in terms of interpretability and generalization to novel, unseen queries—a capability that often poses a challenge for end-to-end neural systems (Mao et al. 2019). By decoupling perception and reasoning through a well-defined interface and a custom query language, our work aims to address key limitations in existing models, enabling flexible, transparent, and efficient reasoning over complex visual scenes based on structured, logical queries. This approach allows for a more direct and interpretable form of interaction with visual information, moving beyond pattern recognition towards a more comprehensive understanding of visual content through logical inference.

Finally, a comparison of the strengths and limitations of various neuro-symbolic approaches is summarized in Table 3.1.

Approach	Pros	Cons
Early Hybrid Models	Straightforward rules; clear symbolic logic	Limited scalability; rigid rule design
Neural Theorem Provers	End-to-end training; capable of complex inferences	High computational cost; requires large annotated datasets
NS-CL-like Methods	Good compositionality; improved interpretability	Domain-specific constraints; fragile logical forms
Transformer-based VQA	Strong perceptual capabilities; good language alignment	Weak in deep symbolic reasoning; heavily reliant on data
Our Framework	Flexible integration of neural and symbolic reasoning; robust to uncertainty; supports a custom query language	Requires alignment training; demands careful dataset design

Table 3.1: Pros and cons of existing neuro-symbolic approaches, including our proposed framework.

This chapter has outlined how the proposed framework is situated at the intersection of three major research trajectories: deep learning for visual perception, cross-modal reasoning with visual-language models, and the growing field of neuro-symbolic AI. While object detection models have achieved remarkable perceptual accuracy, and visual-language models have expanded multimodal reasoning capabilities, both remain fundamentally constrained in their ability to perform explicit logical inference and handle compositional generalization. Neuro-symbolic approaches offer a promising solution to these limitations by bridging pattern recognition and symbolic reasoning, yet existing solutions often suffer from scalability challenges, domain specificity, or limited expressiveness in user interaction. These limitations highlight the continued need for hybrid approaches that integrate the strengths of both neural and symbolic paradigms. As the field evolves, recent advances in foundation models, large-scale multimodal pretraining, and differentiable reasoning suggest promising directions for overcoming these challenges and pushing the boundaries of visual understanding through structured, explainable, and versatile AI systems.

Chapter 4

Methodology

Our proposed neuro-symbolic framework integrates a deep neural network for object detection guided by textual prompts with a symbolic reasoning layer that enables logical query-driven interaction with detected objects. This section details our framework, highlighting the operation of the Neural and Symbolic modules, as well as the synthetic dataset used for training and evaluation. Figure 4.1 presents a general schema of our proposed approach.

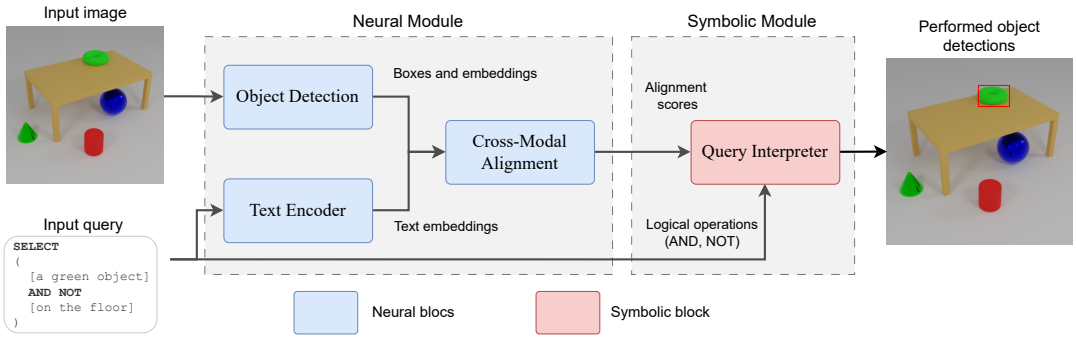

Figure 4.1: Overview of the proposed framework.

The Neural Module is composed of several blocks, which are responsible for processing images and text to generate alignment scores between object and textual prompts. The processing blocks within the Symbolic Module interpret these scores through a pipeline of tokenization, parsing into a structured query (AST), semantic analysis into an executable query plan, and evaluation to produce the final result. This integration enables context-aware reasoning and crossmodal object detection.

Next subsections describe the characteristics of each processing block that composes our framework.

4.1 The Neural Module

The proposed neural module is a fully AI-based component that integrates deep learning techniques to process visual and textual information. The neural module is responsi-

ble for detecting objects in images and aligning them with user-defined textual prompts. It consists of four processing blocks: a Region Proposal Network, an Object Proposal Encoder, a Text Encoder, and a Proposal-Text Alignment Module.

The core architecture of the neural module follows a structured processing pipeline. First, a RPN generates candidate regions likely to contain objects. These regions are processed via RoI pooling to ensure uniform feature dimensions, after which the features are passed through a Multi-Layer Perceptron (MLP) to extract visual embeddings. In parallel, a Transformer-based encoder processes natural language prompts to produce corresponding text embeddings. These two embeddings are subsequently aligned through a shared latent space using a learnable projection, enabling cross-modal comparison between visual and textual modalities.

4.1.1 Region Proposal Network

The RPN block generates a set of object proposals from an input image. Given an image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, the RPN processes it through a convolutional feature extractor, producing feature maps of size $H' \times W' \times C$. Anchors of various scales and aspect ratios are placed over the feature maps to generate K region proposals $\{R_1, R_2, \ldots, R_K\}$. Figure 4.2 further depicts this process.

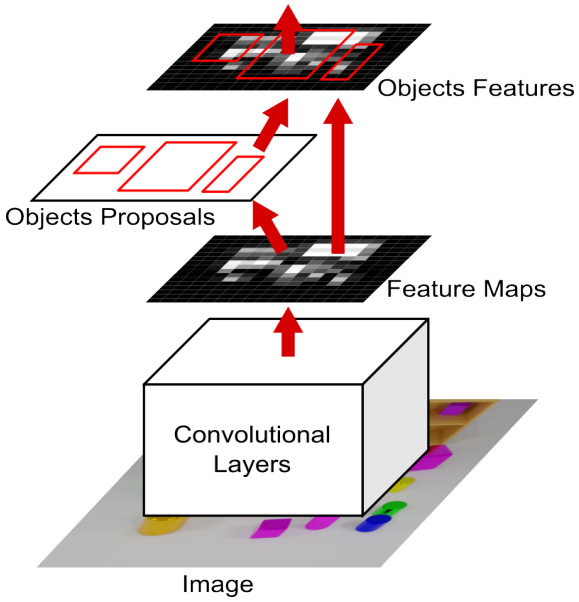

Figure 4.2: An overview of the Faster R-CNN pipeline, illustrating how the RPN block generates object proposals, which are then refined and classified to produce final detections.

For each anchor R_i, the RPN predicts an objectness score c_i and a set of bounding box regression parameters $\mathbf{t}_i = (t_{x_i}, t_{y_i}, t_{w_i}, t_{h_i})$. These parameters encode the offset and scale transformations required to adjust the anchor box to better fit a potential object. As shown in Equation 4.1, the center offsets (t_{x_i}, t_{y_i}) are normalized by the width and height of the

anchor, while the scale factors (t_{w_i}, t_{h_i}) are represented in logarithmic space to capture multiplicative changes.

$$\begin{aligned} t_{x_i} &= \frac{x_i - x_{a_i}}{w_{a_i}}, \qquad t_{y_i} = \frac{y_i - y_{a_i}}{h_{a_i}}, \\ t_{w_i} &= \ln\left(\frac{w_i}{w_{a_i}}\right), \quad t_{h_i} = \ln\left(\frac{h_i}{h_{a_i}}\right), \end{aligned} \tag{4.1}$$

The variables $(x_{a_i}, y_{a_i}, w_{a_i}, h_{a_i})$ denote the anchor's center coordinates, width, and height.

The RPN is trained using a multi-task loss that simultaneously optimizes objectness classification and bounding box regression. As shown in Equation 4.2, the loss function $\mathcal{L}_{\text{RPN}}$ combines a classification term and a regression term, each normalized by the number of anchors contributing to their respective losses.

$$\mathcal{L}_{\text{RPN}} = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(c_i, c_i^*) + \lambda \frac{1}{N_{\text{reg}}} \sum_i c_i^* L_{\text{reg}}(\mathbf{t}_i, \mathbf{t}_i^*), \tag{4.2}$$

where c_i^* denotes the ground-truth objectness label (1 for positive anchors, 0 for negative), and $\mathbf{t}_i^*$ are the ground-truth bounding box regression targets. The function L_{cls} corresponds to the binary cross-entropy loss, while L_{reg} is the smooth L_1 loss. The hyperparameter λ controls the trade-off between the two objectives.

After training, the RPN outputs a set of refined object proposals, which are then subjected to non-maximum suppression to eliminate redundant proposals, resulting in a set $\mathcal{R} = \{R_1, R_2, \ldots, R_N\}$ of N proposals.

4.1.2 Object Proposal Encoder

The Object Proposal Encoder block maps each proposal R_i to a fixed-size embedding $\mathbf{v}_i \in \mathbb{R}^{d_v}$, where $d_v = 256$. It is designed to capture both spatial and semantic information for accurate alignment with textual descriptions. For each R_i, RoI pooling (Girshick 2015) extracts a $7 \times 7 \times 256$ feature map $\mathbf{F}_i$ from the RPN outputs, ensuring uniform representation regardless of the proposal's original size. Next, $\mathbf{F}_i$ is passed through three 3×3 convolutional layers (with 256, 512, and 256 filters, each followed by ReLU) to learn higher-level patterns. The transformed features are flattened and fed into two fully connected layers (1024 units with ReLU, then 256 units) to produce the final embedding vector $\mathbf{v}_i$. We apply L2-normalization. As shown in Equation 4.3.

$$\mathbf{v}_i = \frac{\mathbf{W}_e \,\text{vec}(\mathbf{F}_i) + \mathbf{b}_e}{\|\mathbf{W}_e \,\text{vec}(\mathbf{F}_i) + \mathbf{b}_e\|_2}, \tag{4.3}$$

where $\mathbf{W}_e$ and $\mathbf{b}_e$ are the embedding-layer parameters, and $\text{vec}(\cdot)$ denotes vectorization. This normalization ensures the embeddings lie on a hypersphere, facilitating alignment via cosine similarity.

4.1.3 Text Encoder

The Text Encoder block maps each textual prompt T_j to a fixed-size embedding $\mathbf{u}_j \in \mathbb{R}^{d_u}$. A Transformer-based architecture (Vaswani 2017) captures the semantic meaning of the prompt. First, T_j is tokenized into words or subword units and mapped to embedding vectors; positional encodings are then added to preserve token order. Multiple self-attention layers follow, capturing contextual relationships between tokens. The output is aggregated (e.g., via mean or max pooling) and passed through fully connected layers to produce the embedding $\mathbf{u}_j$, which is then L2-normalized, as shown in Equation 4.4.

$$\mathbf{u}_j = \frac{\mathbf{W}_t \,\text{Aggregate}(\mathbf{H}_j) + \mathbf{b}_t}{\left\| \mathbf{W}_t \,\text{Aggregate}(\mathbf{H}_j) + \mathbf{b}_t \right\|_2}, \tag{4.4}$$

where $\mathbf{H}_j$ represents the outputs from the self-attention layers. This approach allows the encoder to handle complex prompts by capturing the relationships between words, enabling meaningful comparisons with the object proposal embeddings.

4.1.4 Proposal-Text Alignment Module

As shown in Figure 4.3, this processing block computes alignment scores between object proposals and textual prompts by projecting their embeddings into a common latent space and measuring their similarity.

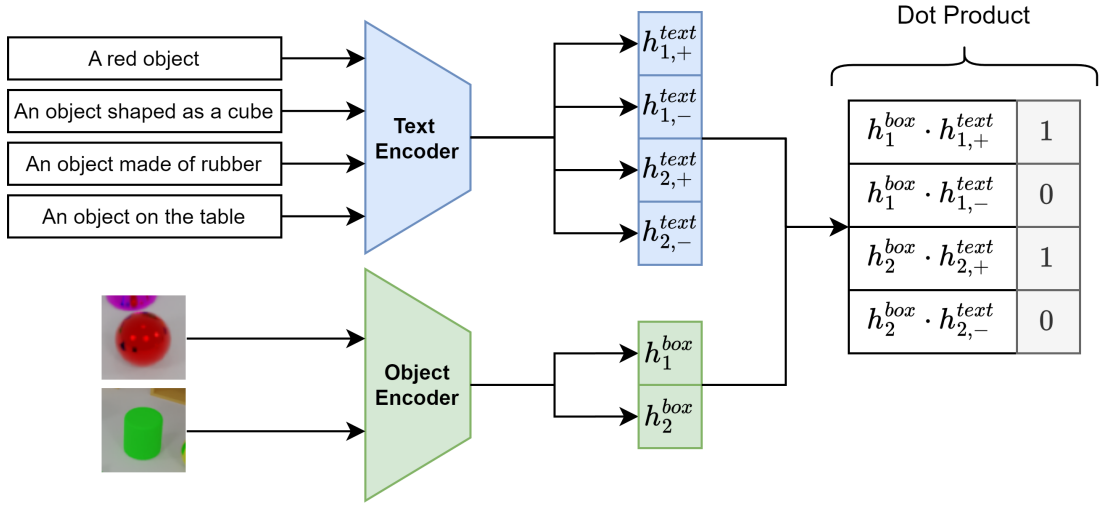

Figure 4.3: The Proposal-Text Alignment block, demonstrating how object proposal embeddings and text embeddings are projected into a common latent space and aligned via cosine similarity

.

Given the object proposal embeddings $\{\mathbf{v}_i\}_{i=1}^{N} \subset \mathbb{R}^{d_v}$ obtained from the Object Proposal Encoder and the text embeddings $\{\mathbf{u}_j\}_{j=1}^{M} \subset \mathbb{R}^{d_u}$ generated by the Text Encoder,

the alignment process begins by projecting both sets of embeddings into a common latent space of dimension d_h. This is accomplished via learnable linear transformations, as defined in Equations 4.5 and 4.6.

$$\mathbf{h}_i^{\text{box}} = \mathbf{W}_b \mathbf{v}_i + \mathbf{b}_b, \quad \mathbf{W}_b \in \mathbb{R}^{d_h \times d_v}, \quad \mathbf{b}_b \in \mathbb{R}^{d_h} \tag{4.5}$$

$$\mathbf{h}_j^{\text{text}} = \mathbf{W}_t \mathbf{u}_j + \mathbf{b}_t, \quad \mathbf{W}_t \in \mathbb{R}^{d_h \times d_u}, \quad \mathbf{b}_t \in \mathbb{R}^{d_h} \tag{4.6}$$

Subsequently, the projected embeddings are normalized to unit vectors, as described in Equations 4.7 and 4.8.

$$\hat{\mathbf{h}}_i^{\text{box}} = \frac{\mathbf{h}_i^{\text{box}}}{\|\mathbf{h}_i^{\text{box}}\|_2} \tag{4.7}$$

$$\hat{\mathbf{h}}_j^{\text{text}} = \frac{\mathbf{h}_j^{\text{text}}}{\|\mathbf{h}_j^{\text{text}}\|_2} \tag{4.8}$$

The similarity score s_{ij} between object proposal i and text prompt j is computed as the cosine similarity between the corresponding normalized embeddings, as shown in Equation 4.9.

$$s_{ij} = \hat{\mathbf{h}}_i^{\text{box}} \cdot \hat{\mathbf{h}}_j^{\text{text}} = \langle \hat{\mathbf{h}}_i^{\text{box}}, \hat{\mathbf{h}}_j^{\text{text}} \rangle \tag{4.9}$$

To map the similarity score from the interval $[-1,1]$ to $[0,1]$, a linear transformation is applied as expressed in Equation 4.10.

$$s_{ij} = \frac{s_{ij}+1}{2} \tag{4.10}$$

Finally, the model is trained using a binary cross-entropy loss, defined in Equation 4.11, which compares the predicted alignment scores with ground-truth labels.

$$\mathcal{L}_{\text{align}} = -\frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \Big[y_{ij} \log(s_{ij}) + (1-y_{ij}) \log(1-s_{ij}) \Big] \tag{4.11}$$

In this formulation, the binary label y_{ij} is set to 1 if the object proposal R_i corresponds to the semantic content of the text prompt T_j, and 0 otherwise, as determined by the dataset annotations.

4.2 The Symbolic Module

Using a custom query language, the Symbolic Module enables logical reasoning and complex interactions with the detected objects. It interprets the alignment scores from the Neural Module and applies logical operations to execute queries. The module comprises four key stages:

- **Lexer:** The lexer processes the raw query string and produces a stream of tokens, each representing the smallest meaningful unit (e.g., `PROMPT`,`OPERATOR`, `NUMBER`). Whitespace and comments are discarded, allowing subsequent parsing steps to work with well-defined tokens;
- **Parser:** The parser reads the token stream and constructs an abstract syntax tree (AST) based on predefined grammar rules. For example, `[a red object] and [an object on the floor]` is transformed into a logical conjunction node with two prompt node children, while expressions like `COUNT [a red object] > 5` generate a comparison node linking a `COUNT` operation to a `NUMBER`;
- **Semantic Analyzer:** This stage checks the consistency and data types throughout the AST. It ensures that logical operators apply to Boolean-like or fuzzy/probabilistic scores, arithmetic operators appear only with numeric data, and each prompt node is assigned an uncertainty interpretation (Boolean, fuzzy, or probabilistic);
- **Evaluator:** Finally, the evaluator executes the AST by fetching alignment scores from the Neural Module and applying the user-defined operations.

4.2.1 Proposed query language

In order to support the operation of our proposed framework, a new query language was defined to support conceptual object detection. The proposed query language supports three primary data types. The `NUMBER` type represents integers or real numbers, `BOOLEAN` accepts only `true` or `false`, and `PROMPT` consists of natural language text enclosed in brackets.

The supported operators include arithmetic, logical, and comparison operators. Specifically, `^` is used for exponentiation (precedence level 1); `*`, `/`, and `%` denote multiplication, division, and modulus operations (precedence level 2); `+` and `-` perform addition and subtraction (precedence level 3); comparison operators (`==`, `!=`, `<`, `<=`, `>`, `>=`) have a precedence level of 4; logical NOT (`not`) operates at level 5; logical AND (`and`) at level 6; logical XOR (`xor`) at level 7; and logical OR (`or`) at level 8.

4.2.2 Interpreting Scores

The interpretation of alignment scores influences how logical operations are computed. Three approaches are considered: Probabilistic, Fuzzy, and Boolean. In the Probabilistic interpretation, scores are treated as probabilities and operations use probability theory (e.g., the product rule for `AND` and the inclusion-exclusion principle for `OR`). In the Fuzzy interpretation, scores represent degrees of truth with operations defined via fuzzy logic (using the minimum for `AND` and maximum for `OR`). In the Boolean interpretation, scores are binarized using a threshold (e.g., $\theta = 0.5$) before applying standard Boolean algebra.

4.2.3 Clause Structures

The query language provides clauses for interacting with detected objects, which can be combined with logical operators to form complex queries.

Select

Identifies entities matching a prompt, returning object indices above a score threshold.
How to use:

```
select [entity selection]
sort by [attribute] asc/desc limit N
```

Example: To select the three biggest green objects:

```
select [an object with color green]
sort by [a big object] desc limit 3
```

Count

Counts the number of entities matching a prompt, returning an integer.
How to use:

```
count [entity selection]
```

Example: Number of objects on the floor:

```
count [an object on the floor]
```

All and Any

Checks whether all or any entities satisfy a given condition, returning a Boolean.
How to use:

```
all/any [entity selection]
```

Example: Check if all objects are made of metal:

```
all [an object made of metal]
```

Example: Check if at least one object is inside the box:

```
any [an object inside the box]
```

4.3 Integration and Uncertainty Handling

The interaction between the Neural and Symbolic Modules is achieved through the alignment scores provided by the Neural Module. The Symbolic Module leverages these scores as inputs for its logical operations, thereby grounding symbolic reasoning in perceptual outputs. This integration enables complex, context-aware analysis of detected objects by combining neural perception with symbolic logic.

To handle uncertainties inherent in crossmodal recognition, the framework supports three interpretations of the alignment scores: probabilistic, fuzzy, and Boolean. The choice of interpretation affects how evidence from multiple conditions is aggregated when executing complex queries.

- Probabilistic Interpretation: Alignment scores s_{ij} are treated as probabilities, representing the likelihood that object proposal R_i corresponds to textual prompt T_j. Logical operations are performed using probabilistic logic;
- Fuzzy Interpretation: Alignment scores are interpreted as degrees of truth. Logical operations are defined according to fuzzy set theory—for example, the AND operation is implemented as the minimum of the scores, while the OR operation is implemented as the maximum;
- Boolean Interpretation: Scores are first binarized using a threshold θ (typically $\theta = 0.5$), after which standard Boolean algebra applies.

This flexibility in score interpretation enhances system adaptability and enables experimentation with different reasoning paradigms. The integration ensures that uncertainties at the perceptual level are propagated and incorporated during the reasoning process, yielding robust and interpretable results even in the presence of noise or ambiguity.

Based on the defined query language and modules, users can search for objects with different characteristics. Figure 4.4 presents a query example. By fusing perception with symbolic inference, the system can isolate specific entities—such as selecting “an object on the floor” but “not a metal object’ and provide transparent answers, even when multiple attributes must be satisfied or ruled out.

4.4 Dataset

To train and evaluate our framework, we construct the Annotated Objects for Visual Reasoning Dataset (AOVR-Dataset), a synthetic 3D dataset designed to facilitate research in visual reasoning and object detection. The dataset comprises 3D objects arranged in various container settings, with each object annotated using precise bounding boxes and natural language descriptions. Examples of the generated 3D scenes with annotated objects from the AOVR-Dataset are shown in Figure 4.5.

4.4.1 Dataset Description

The AOVR-Dataset was created to support tasks in visual reasoning, object detection, and natural language understanding. It encompasses a wide variety of objects with

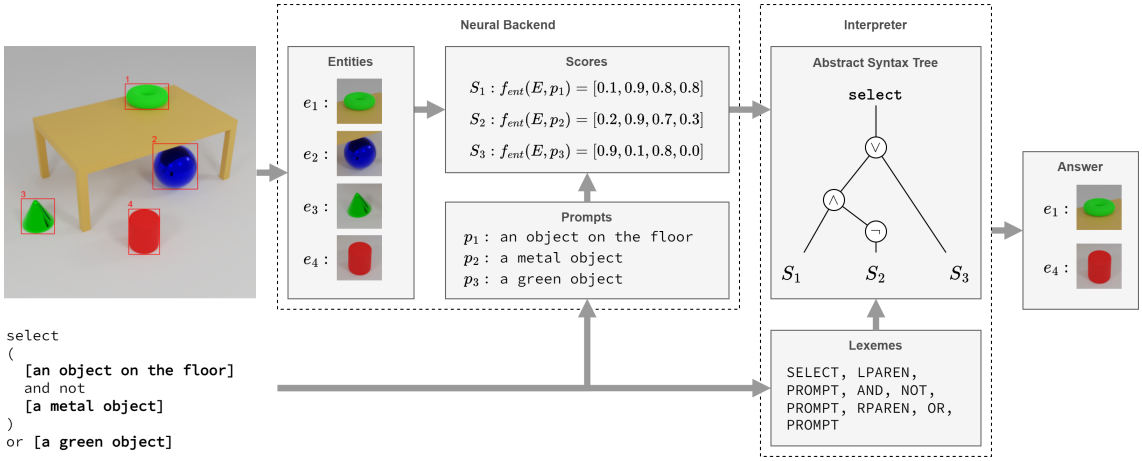

Figure 4.4: The Neural Module (left) takes input images and textual prompts, generating object proposals and producing alignment scores between visual features and text. The Symbolic Module (right) uses these scores to execute logical queries, enabling context-aware, explainable reasoning over detected objects.

multiple attributes. The objects come in shapes such as cylinder, cube, torus, cone, and sphere; display colors including blue, magenta, black, red, orange, purple, green, yellow, and cyan; are made from materials like metal and rubber; and are classified as small or big. Additionally, objects are placed in various containers including floor, shelf bottom, shelf top, table top, table under, crate, box, shelf, and table. Each object is annotated with precise bounding boxes for localization and natural language descriptions—captions are generated using a Large Language Model (LLM) to provide context-rich details.

4.4.2 Dataset Generation

The dataset generation process begins with the construction of a generic 3D scene in Blender (version 3.4), where both objects and containers are carefully modeled. Rather than relying solely on geometric primitives, a list of custom object shapes was created, with manual adjustments applied to smooth edges, refine corners, and ensure geometric consistency. Container types were similarly designed to support structured spatial organization and varied interactions with the placed objects. A neutral environment was configured with the addition of area lights, which provide diffuse and uniform illumination across the scene. Rendering was performed using the Cycles engine, chosen for its physically-based lighting and material realism. Figure 4.6 shows the modeling interface and the scene configuration within Blender.

The algorithm for dataset generation is outlined in Algorithm 1. It begins by clearing the current scene and initializing the material library using the predefined material set M. For each of the N scenes, two distinct containers c_1 and c_2 are randomly selected from C and instantiated to define the placement grounds $G = \{g_1, g_2\}$. Within each container, k objects are created by randomly sampling a shape $s \in S$, and assigning random attributes: rotation angle $z_{angle} \sim \mathcal{U}(-\theta, \theta)$, discrete size (small or big), and material $m \in M$. Each

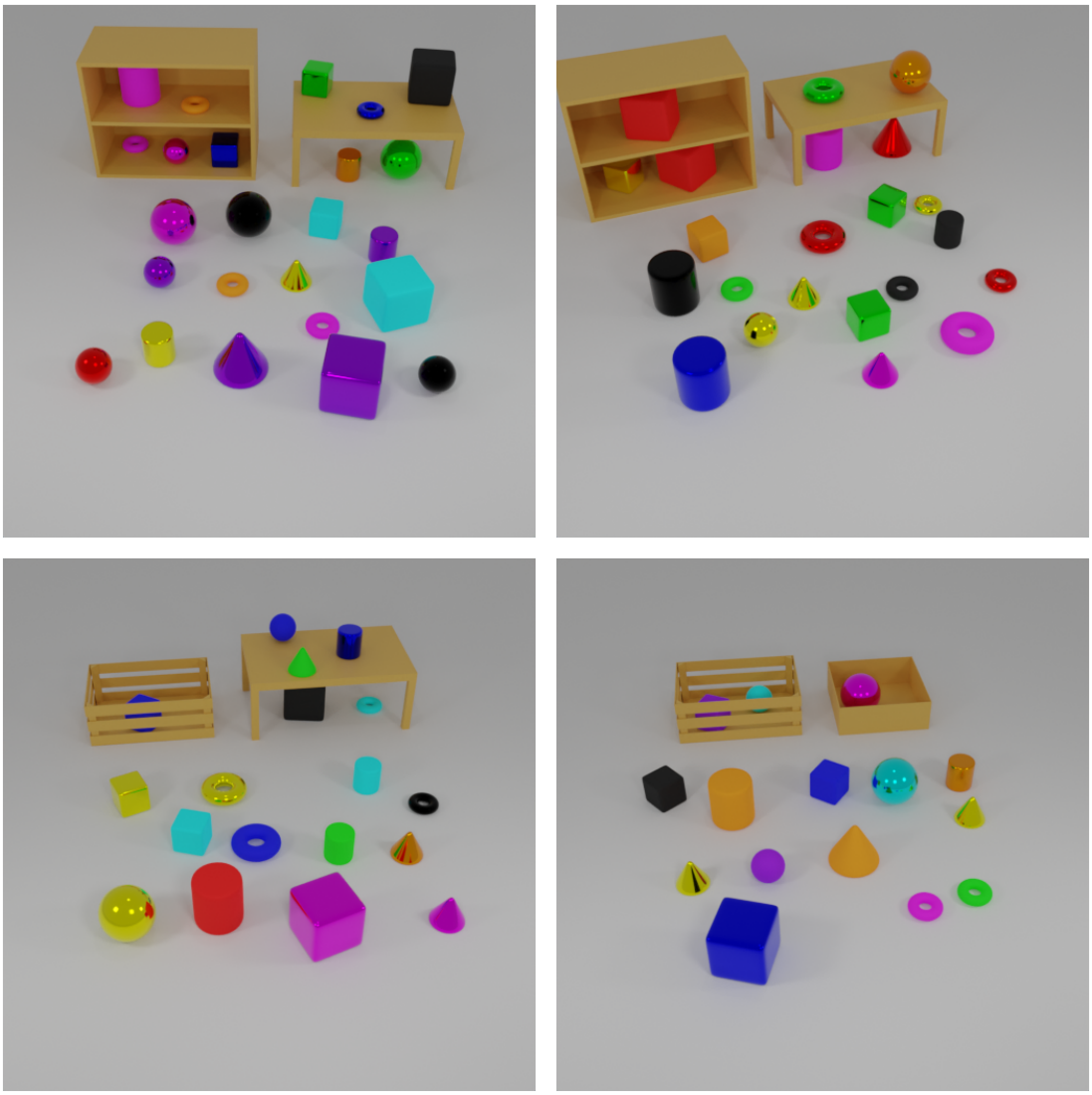

Figure 4.5: Examples from the AOVR-Dataset. Each image depicts 3D objects exhibiting diverse shapes, colors, materials, and sizes in different container settings. Bounding box annotations and natural language descriptions accompany each object.

object is placed on the ground without intersection and within the spatial bounds of the container. The metadata—comprising shape, size, material, container, 3D position, and 2D bounding box—is stored in a per-scene list L_{scene}. Based on this metadata, a language model generates natural language captions describing the scene. The camera is constrained to follow a predefined path while always pointing to the center of the scene; within this constraint, its specific position is randomized to introduce visual diversity across scenes. Finally, the configured scene is rendered using the parameters specified in R, and both the resulting image and its corresponding JSON file (containing metadata and captions) are saved.

For each image, a corresponding JSON file with the same name is generated, provid-

Algorithm 1: Compact procedure for AOVR-Dataset generation.

Input: S: object shapes, C: container types, M: materials, N: scenes, k: objects/ground, R: render config

Output: $N \times (\texttt{image}, \texttt{JSON})$ pairs

Clear scene; Init material library from M;

for *scene_id* $\leftarrow 1$ **to** N **do**

  $L_{scene} \leftarrow \emptyset$;

  `/* Initialize metadata list for scene */`

  Select distinct $c_1, c_2 \in C$ randomly; Instantiate & position to define grounds $G = \{g_1, g_2\}$;

  Sample camera position $\mathbf{p}_{cam} \sim \mathcal{P}_{path}$; Constrain look-at target $\mathbf{t}_{cam} \leftarrow$ center;

  foreach *ground* $g \in G$ **do**

    for $i \leftarrow 1$ **to** k **do**

      Select random $s \in S$; Instantiate s;

      Randomize props: $z_{angle} \sim \mathcal{U}(-\theta, \theta)$, size $\in \{\text{small, big}\}$, material $m \in M$;

      Place object on g (non-intersecting, within footprint);

      Append shape, size, color, material, container, 3D pos and 2D box to L_{scene};

    end

  end

  Generate captions via LLM using L_{scene};

  Render scene with R; Save `image` and `JSON` (L_{scene} + captions);

end

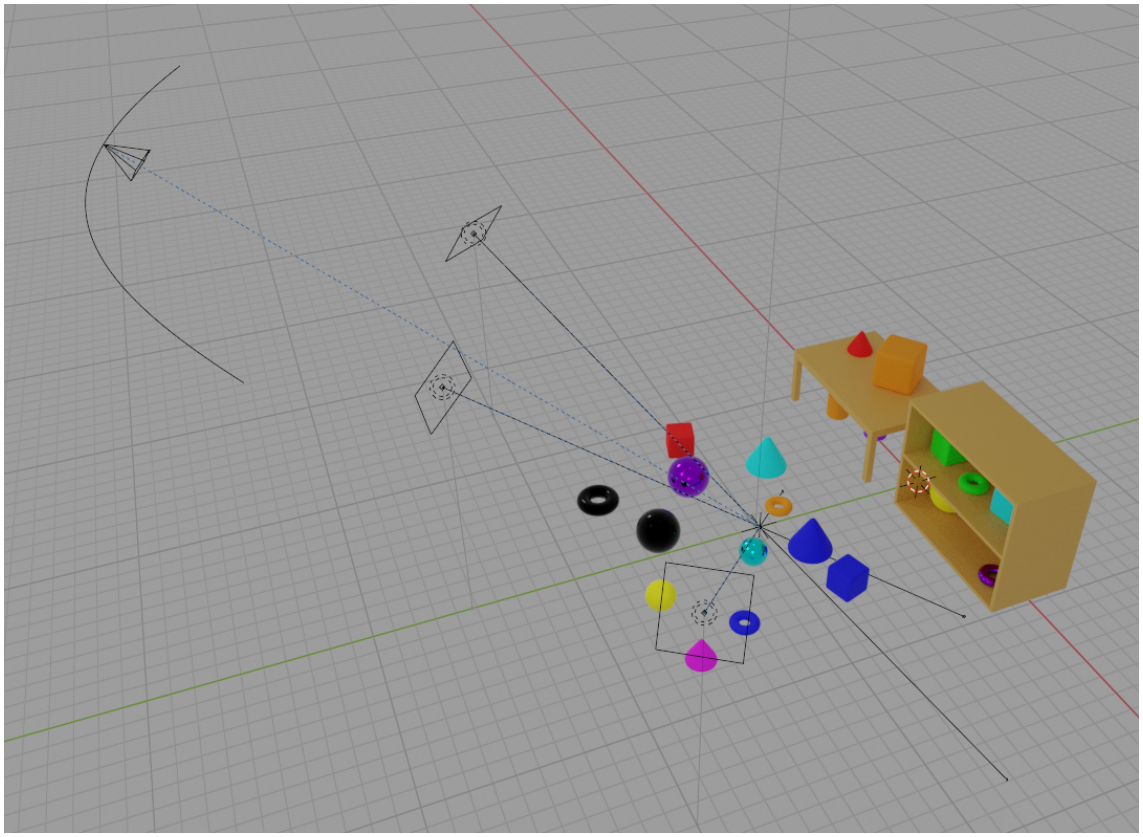

Figure 4.6: Modeling environment in Blender used for dataset generation.

ing a structured annotation of all objects in the scene. Each object entry includes visual attributes, such as material, color, size, and shape, along with its precise spatial location in 3D space, the container it belongs to, and a 2D bounding box for accurate image-level localization. This comprehensive annotation not only facilitates downstream tasks like object detection and scene understanding but also ensures that each instance is richly described in both geometric and semantic terms. An example of such a JSON file is shown in Listing 4.1.

Listing 4.1: Example of JSON annotation for a single image.

```
[
  {
        "material": "rubber",
        "color": "orange",
        "size": "small",
        "shape": "torus",
        "location": [
            1.7428562641143799,
            -10.725440979003906,
            0.0
        ],
```

```
        "container": "floor",
        "bounding_box": [267, 262, 299, 287]
    },
  ...
]
```

The process of generating captions for object attributes involves systematically querying a Large Language Model (LLM) to produce diverse linguistic descriptions grounded in specific visual properties. For each attribute–value pair (e.g., `color: orange`), the LLM is prompted to generate a set of unique, lowercase sentences that capture the semantics of the property without introducing redundancy or irrelevant information. Using the GPT-4o model from OpenAI, each attribute is expanded into a collection of 100 distinct descriptive phrases. Table 4.1 presents illustrative examples of these descriptions across different attributes.

Attribute	**Value**	**Descriptions**
Shape	Cylinder	a cylindrical object; an item shaped like a cylinder; something that has a cylindrical form
Color	Red	a red object; an item that is colored red; something that has a red hue
Material	Metal	a metal object; an item made of metal; something that is constructed from metal
Size	Big	a big object; an item that is large in size; something that occupies a lot of space
Container	Shelf Top	a container placed on the top of a shelf; an item located on the shelf's upper surface; a shelf top container

Table 4.1: Examples of natural language descriptions for object attributes.

The prompt used to guide the LLM in generating these descriptions was carefully designed to enforce clarity, conciseness, and consistency in the output format. It specifies the task, the expected output style, and provides a concrete example to ensure the model adheres to the intended structure. Listing 4.2 shows the exact prompt template employed during the dataset creation process.

Listing 4.2: Prompt for generating synthetic sentences based on object properties.

```
### Instruction ###
You are an expert in generating synthetic data. Your task is to
generate N unique sentences, all in lowercase, describing an object
 based on the provided property in the format "property: value".
Avoid repeating sentences and do not include any additional
information.

### Output Format ###
Your response should be a list of comma-separated values, e.g.,
sentence1, sentence2.

### Example ###
# Input #
property: color
value: red
N: 3

# Output #
a red object, an item that is colored red, something that has a red
 hue

Begin!
```

In conclusion, the AOVR-Dataset provides a controlled yet richly annotated environment designed to support research in visual reasoning and cross-modal understanding. By combining procedurally generated 3D scenes, precise geometric annotations, and diverse natural language descriptions, the dataset enables systematic evaluation of models' capabilities in compositional reasoning, object recognition, and language grounding. Its synthetic nature ensures flexibility in scaling complexity, while the structured metadata facilitates transparent benchmarking for tasks that require both perceptual accuracy and semantic understanding.

Chapter 5

Results

In this section, we evaluate the performance of our proposed neuro-symbolic framework on the AOVR-Dataset. Our experiments aim to demonstrate the framework's effectiveness in object detection, alignment with textual prompts, and handling complex queries via the symbolic module. We present the experimental setup, including dataset details and implementation specifics, followed by results and analysis.

5.1 Experimental Setup

In this section we outline the empirical framework used to validate the proposed neuro–symbolic architecture. We begin by describing the AOVR-Dataset, whose deliberately varied objects, attributes and scene layouts enable a stringent test of generalization. We then detail the neural and symbolic components, implementation environment and training regimen.

5.1.1 Preparing the Dataset

The AOVR-Dataset comprises 10,000 examples, partitioned into a training set of 8000 and a validation set of 2000. Each example consists of an image with annotated objects and associated textual prompts. The dataset includes various objects, attributes, and scenes to ensure effective generalization. The validation set contains novel combinations of attributes to assess the model's ability to handle new scenarios.

5.1.2 Implementation Details

Our neural network is implemented in PyTorch (version 2.5.1) (Paszke et al. 2019). The RPN uses a lightweight MobileNetV3 backbone (Howard et al. 2019), pretrained on ImageNet (Deng et al. 2009), chosen for its efficiency and strong performance in object detection. The Object Proposal Encoder is a two-layer MLP with 1024 neurons per layer. For text, we use the large version of the Transformer-based mGTE encoder (Zhang, Li, Zhang, Li, Zhang, Li, Zhang, Li, Zhang & Li 2023), trained on large-scale multilingual data with 1024-dimensional embeddings, enabling rich semantic understanding. Both visual and textual embeddings are projected into a shared 64-dimensional space by the

Proposal-Text Alignment Module, which computes alignment scores via element-wise multiplication and summation. All components are trained end-to-end for joint optimization. Figure 5.1 presents the implemented neural module architecture.

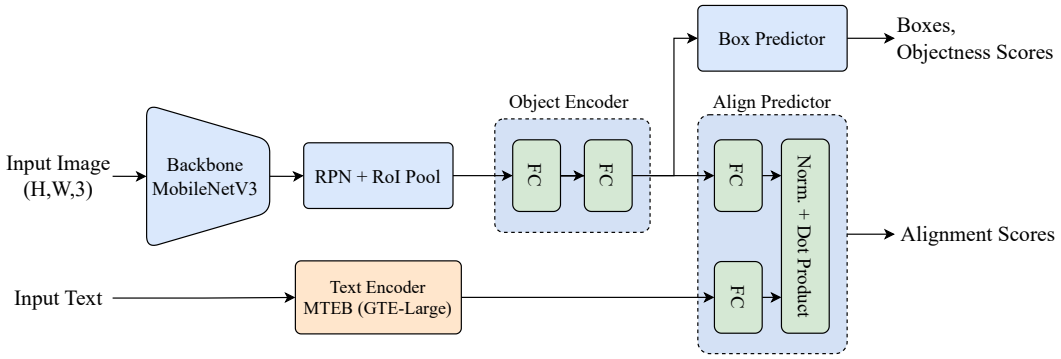

Figure 5.1: Overview of the neural module.

5.1.3 Training Procedure

Model training is conducted using the AdamW optimizer (Loshchilov & Hutter 2018) with an initial learning rate of 1×10^{-3}. We employ a learning rate scheduler that dynamically reduces the learning rate based on the validation loss, using a "reduce on plateau" strategy. Specifically, the learning rate is reduced when the validation loss ceases to improve, ensuring optimal convergence during training. The model is trained for 20 epochs with a batch size of 16. Both training and inference are performed on an NVIDIA GeForce GTX 3060 GPU equipped with 12 GB of VRAM. The model is trained by minimizing the combined loss function shown in Equation 5.1.

$$\mathcal{L} = \mathcal{L}_{\text{RPN}} + \mathcal{L}_{\text{align}}, \tag{5.1}$$

where $\mathcal{L}_{\text{RPN}}$ encourages accurate object proposal generation, and $\mathcal{L}_{\text{align}}$ ensures correct association between visual features and textual descriptions. Figure 5.2 shows the training and validation loss curves over the 20 epochs. Both training and validation losses decrease and converge, indicating that the model learns effectively without significant overfitting.

Evaluation Metrics

We evaluate our framework using the following metrics:

- Mean Average Precision (mAP): Assesses object detection performance, calculated at Intersection over Union (IoU) thresholds of 0.5 and 0.75.
- Average Intersection over Union (IoU): Measures the average overlap between predicted bounding boxes and ground truth.

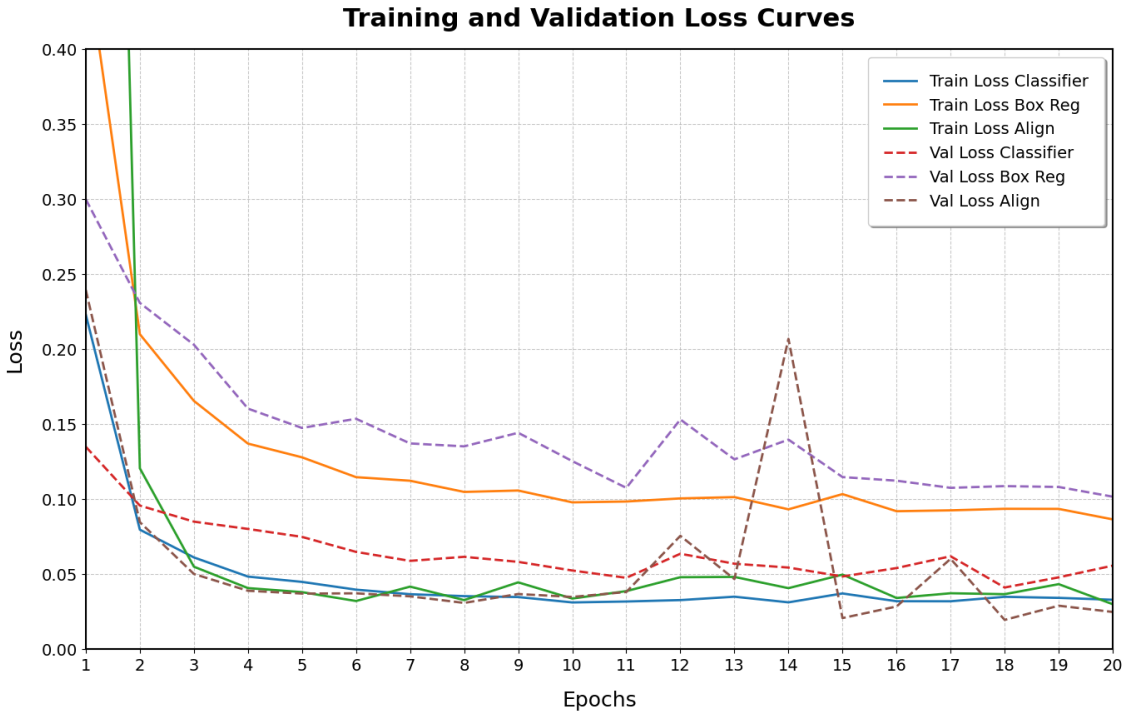

Figure 5.2: Training and validation loss curves during model training.

- Alignment Accuracy: Measures the accuracy of aligning object proposals with textual prompts, expressed as the percentage of correct alignments.
- Query Execution Accuracy: Evaluates the correctness of answers produced by the symbolic module when executing queries, compared against ground-truth answers.

5.2 Quantitative Analysis

This section consolidates the numerical evidence supporting the proposed framework. We begin with localization and classification scores that benchmark the neural module, proceed to cross-modal alignment metrics, and conclude with an evaluation of symbolic query execution under multiple uncertainty interpretations.

5.2.1 Object Detection Performance

To assess the baseline object detection capabilities of the neural module without considering alignment with textual prompts, the performed evaluation aims to determine the model's effectiveness in accurately localizing objects within images.

We evaluate the object detection performance of the neural module on the validation set of the AOVR-Dataset. The evaluation procedure involves the following steps:

1. **Model Inference:** The model processes each image in the validation set, generating predicted bounding boxes and associated objectness scores for detected objects.

2. **Performance Metrics:** We compute the mean Average Precision (mAP) at Intersection over Union (IoU) thresholds of 0.5 and 0.75. The mAP metric provides a comprehensive evaluation by considering the precision and recall across different confidence levels. Additionally, we calculate the average IoU across all detected objects to assess the precision of the bounding box localization.
3. **Evaluation Protocol:** The mAP is calculated following the standard protocol used in object detection benchmarks, where precision and recall are evaluated at varying confidence thresholds, and the average is taken over all thresholds.

The model achieves an mAP of 99.48% at IoU = 0.5 and 98.01% at IoU = 0.75, with an average IoU of 93.39%.

The results indicate that the neural module achieves high detection accuracy on the AOVR-Dataset. The mAP of 99.48% at an IoU threshold of 0.5 demonstrates the model's effectiveness in correctly identifying and localizing objects within the images.

At a stricter IoU threshold of 0.75, the mAP remains high at 98.01%, suggesting that the predicted bounding boxes closely match the ground truth annotations even under more stringent overlap criteria. The average IoU of 93.39% further confirms the precision of the bounding box predictions.

This high localization accuracy is critical for ensuring that the detected objects correspond accurately to the entities present in the scenes, which is essential for subsequent tasks such as alignment with textual prompts and symbolic reasoning.

5.2.2 Alignment Performance

To evaluate the model's ability to correctly align object proposals with their corresponding textual prompts, effectively associating visual features with semantic descriptions, this assessment focuses on the model's capability to distinguish between correct (positive) and incorrect (negative) textual descriptions for each detected object using prompts that refer to a single attribute.

We assess the alignment performance of our model using a controlled experiment designed to test its ability to differentiate between positive and negative captions for each object proposal. The experiment is conducted as follows:

1. **Data Preparation:** For each object in the validation set, we generate a set of positive captions that accurately describe a single attribute of the object (e.g., `[a red object]`, `[a cube-shaped object]`). Additionally, we generate negative captions that describe an attribute not present in the object (e.g., `[a green object]` for a red object). Positive and negative captions are balanced to ensure the equal representation of correct and incorrect prompts. This ensures that the model is tested on its ability to recognize correct descriptions and reject incorrect ones based on single attributes.
2. **Feature Extraction:** The neural module processes each detected object proposal to extract visual embeddings representing the object's features, and the text encoder converts both positive and negative captions into textual embeddings.

3. **Alignment Score Computation:** We compute alignment scores with both their positive and negative captions using the Proposal-Text Alignment Module for each object proposal. A confidence threshold set at 0.5 is applied to the alignment scores to generate binary predictions.
4. **Accuracy Calculation:** We compare the model's predictions with the ground truth labels. The alignment accuracy is calculated as the percentage of correct predictions from the total number of caption–object pairs evaluated.

The model achieves an alignment accuracy of 99.35%, indicating that it correctly aligns object proposals with their corresponding positive captions and rejects negative captions with high precision.

The high alignment accuracy indicates that the model effectively associates the visual features with corresponding textual descriptions based on single attributes. This performance suggests the model successfully captures the semantic relationships between objects and language.

5.2.3 Accuracy Analysis with Uncertainty Handling Methods

To evaluate the model's ability to handle complex queries involving multiple object attributes and assess the impact of different uncertainty handling methods in the symbolic module on query execution accuracy, we aim to highlight the advantages of using a symbolic method to combine multiple simple prompts over training the model with complex sentences.

We design an experiment where the model is trained exclusively on simple prompts, each describing a single attribute of an object (e.g., `[a red object]`). During the evaluation, we test the model's capacity to handle complex queries constructed by combining multiple simple prompts using logical `and` operations within the symbolic module. This approach leverages symbolic reasoning to interpret and execute complex queries without requiring the neural model to process complex sentences directly.

The experiment is conducted as follows:

1. **Attribute Selection:** For each of 1000 images from the test set, randomly select attributes to create queries of varying lengths (n), where n ranges from 1 to 5.
2. **Query Generation:** Generate complex queries by combining n randomly chosen attributes using logical `and` operations. For example, a query with $n = 3$ could be the following:
   ```
   select [a red object]
   and [a metal object]
   and [an object on the floor]
   ```
3. **Ground Truth Extraction:** Identify the ground-truth bounding boxes of objects that satisfy all selected attributes.
4. **Model Inference and Reasoning:** Use the neural module to generate object proposals and compute alignment scores for each simple attribute prompt. Employ the symbolic module to execute the complex query under three different uncertainty handling interpretations:

- *Probabilistic Interpretation (PI):* Treat alignment scores as probabilities and combine them using probabilistic logic.
- *Fuzzy Interpretation (FI):* Interpret scores as degrees of truth in fuzzy logic and combine them accordingly.
- *Boolean Interpretation (BI):* Binarize alignment scores using a threshold (e.g., 0.5) and apply standard Boolean logic.

5. **Accuracy Computation:** Calculate the query execution accuracy by comparing the model's predicted bounding boxes with the ground truth. A prediction is correct if the Intersection over Union (IoU) exceeds 0.5.

The query execution accuracy for each number of attributes and uncertainty handling method is presented in Table 5.1.

Num. of Attributes (n)	**Probabilistic (%)**	**Fuzzy (%)**	**Boolean (%)**
1	91.06	90.49	91.30
2	82.62	83.83	82.87
3	74.88	74.69	75.27
4	68.44	67.65	69.05
5	62.52	64.16	62.85

Table 5.1: Query execution accuracy.

The results demonstrate that the model effectively handles complex queries by combining simple prompts using the symbolic module. Even though the model is trained only on simple prompts, it achieves reasonable accuracy on complex queries involving multiple attributes.

We observe a decreasing trend in accuracy as the number of attributes in the query increases across all uncertainty handling methods. This decline aligns with theoretical expectations based on the probabilistic combination of independent attribute recognitions.

Assuming that the recognition of each attribute is an independent event with an individual attribute recognition accuracy of approximately 91%, the expected accuracy for queries involving n attributes can be approximated by the Equation 5.2.

$$\text{Expected Accuracy} = (0.91)^n \times 100\%. \tag{5.2}$$

Comparing the calculated theoretical values with the observed accuracies in Table 5.1, we find that they closely align, confirming that the decline in accuracy is mainly due to the compounding effect of combining multiple attributes.

Regarding the impact of different uncertainty handling methods, we observe that the differences in accuracy among the probabilistic, fuzzy, and Boolean interpretations are relatively small, generally within 1% to 2% of each other. This suggests that the system is robust to the choice of the uncertainty handling method. Additionally, for queries with a higher number of attributes ($n = 5$), the fuzzy interpretation achieves slightly higher accuracy, possibly because fuzzy logic better handles partial truths in complex combinations.

These observations highlight the effectiveness of using a symbolic module to combine simple prompts, allowing the system to handle complex queries without requiring the neural model to process complex sentences directly. This approach offers the following advantages:

- **Scalability:** Training the model exclusively on simple prompts reduces the complexity of the training data and the model. It eliminates the need to cover all possible combinations of attributes during training, which would be impractical due to the combinatorial explosion of possible complex sentences.
- **Modularity:** The symbolic module provides a flexible framework to combine simple concepts into more complex ones. This modularity allows for easier debugging, extension, and maintenance of the system.
- **Interpretability:** By using a symbolic method to combine attributes, the reasoning process becomes more transparent and interpretable. Each step of the logical combination is explicit, facilitating understanding and trust in the system's decisions.
- **Generalization:** The approach allows the model to generalize to unseen combinations of attributes. Since the neural module learns representations for individual attributes, the symbolic module can recombine these to handle novel queries without additional training.

5.3 Qualitative Analysis

This section demonstrates the interpretability of our neuro-symbolic framework through logical queries executed on selected images from the AOVR-Dataset. Each test evaluates the system's ability to handle complex reasoning tasks by combining alignment scores from the neural module with logical operations from the symbolic module.

5.3.1 Test 1: Basic Attribute and Logical Queries

To evaluate the system's ability to interpret basic attribute-based queries and logical compositions, we designed a set of tests involving color, shape, and spatial location, combined using operators such as `and`, `or`, and `not`. These queries were executed on the object arrangement depicted in Figure 5.3. The results are summarized in Table 5.2.

The first five examples produce correct outputs, demonstrating the framework's capability to handle simple property-based filtering and basic logical composition. An interesting observation arises from the query `select [a green object or a donut]`, which returns the expected result despite the likely absence of explicit logical disjunction modeling in the encoder. This behavior can be attributed to the fact that the embedding of the entire phrase effectively captures both `green object` and `donut` as distinct visual–linguistic patterns. Since these attributes are disjoint in the dataset, the model retrieves objects matching either one, even though the logical `or` is not explicitly parsed.

Conversely, the query `select [a red colored cylinder]` does not return the correct result, instead producing a superset of relevant and irrelevant objects. This discrepancy is likely due to the text encoder's limited capacity to enforce logical conjunction. In

Description	Query	Output	Ground Truth
Selects red, spherical, metallic objects	`select [a red object] and [an object shaped like a ball] and [a metallic piece]`	`Indices([6])`	`Indices([6])`
Selects purple objects not in the shelf	`select [a purple object] and not [an object in the shelf]`	`Indices([17])`	`Indices([17])`
Selects the largest red cylinder	`select [a cylinder] and [a red object] sort_by [a big object] limit 1 desc`	`Indices([0])`	`Indices([0])`
Selects black objects not on the floor	`select [a black object] and not [an object on the floor]`	`Indices([18])`	`Indices([18])`
Checks if there is at least one purple cone-shaped object	`any [a purple object] and [an object shaped like a cone]`	`Boolean(True)`	`Boolean(True)`
Selects green objects or objects shaped like donuts	`select [a green object or a donut]`	`Indices([15, 3, 13, 14])`	`Indices([15, 3, 13, 14])`
Selects an object on the table	`select [an object on the table]`	`Indices([20])`	`Indices([20, 22])`
Selects a red cylinder	`select [a red colored cylinder]`	`Indices([12, 7, 19, 6, 22, 0, 16])`	`Indices([0])`

Table 5.2: Examples of attribute-based and logical queries, including logical combinations, along with their outputs and ground truth.

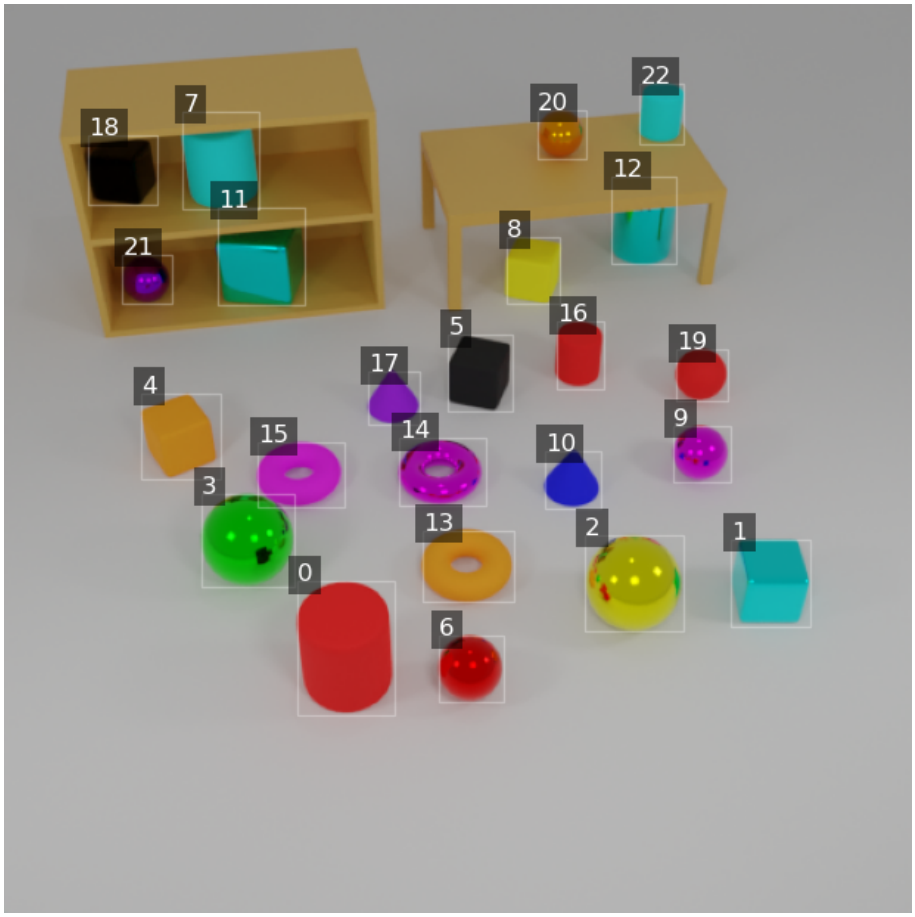

Figure 5.3: Objects used in tests with basic attributes and logical queries.

practice, the encoder does not interpret conjunctions such as `and` or composite phrases as strict intersections of properties. Rather, it embeds the entire phrase into a joint vector representation, where multiple attributes are blended rather than logically composed. As a result, the model tends to retrieve objects that exhibit partial matches to the combined description.

These results suggest that the system relies on the emergent alignment between textual and visual embeddings, rather than explicit logical parsing. In cases where precise attribute combination is required, the absence of symbolic composition limits the model's ability to produce exact matches.

5.3.2 Test 2: Semantic and Relational Queries

To assess the system's capacity for semantic understanding and relational reasoning, we designed a set of more advanced queries. These were intentionally phrased using expressions not encountered during training but that convey equivalent meanings, in order to evaluate generalization and inference of abstract properties. The tests were conducted on the scene shown in Figure 5.4, and the corresponding results are presented in Table 5.3.

Description	Query	Output	Ground Truth
Selects objects resembling a globe	`select [similar to a globe]`	`Indices([2, 11])`	`Indices([2, 11])`
Selects all scarlet-colored objects	`select [a scarlet thing]`	`Indices([6])`	`Indices([6])`
Selects objects colored like trees (green)	`select [an object with the color of trees]`	`Indices([4, 8, 5])`	`Indices([4, 8, 5])`
Selects donut-shaped objects	`select [an object shaped as a donut]`	`Indices([12])`	`Indices([12])`
Selects cubic, dark-colored objects	`select [a cubic form] and [a dark object]`	`Indices([3])`	`Indices([3])`
Selects pointy, small, iron objects excluding green	`select [a pointy thing] and [made of iron] and [small proportions] and not [green colored]`	`Indices([1])`	`Indices([1])`
Selects objects inside a crate	`select [inside a crate]`	`Indices([15])`	`Indices([15])`
Selects tube-shaped objects not on the shelf	`select [shaped like a tube] and not [in the shelf]`	`Indices([6])`	`Indices([6])`

Table 5.3: Examples of visual or metaphorical queries interpreted via similarity, shape, and exclusion logic.

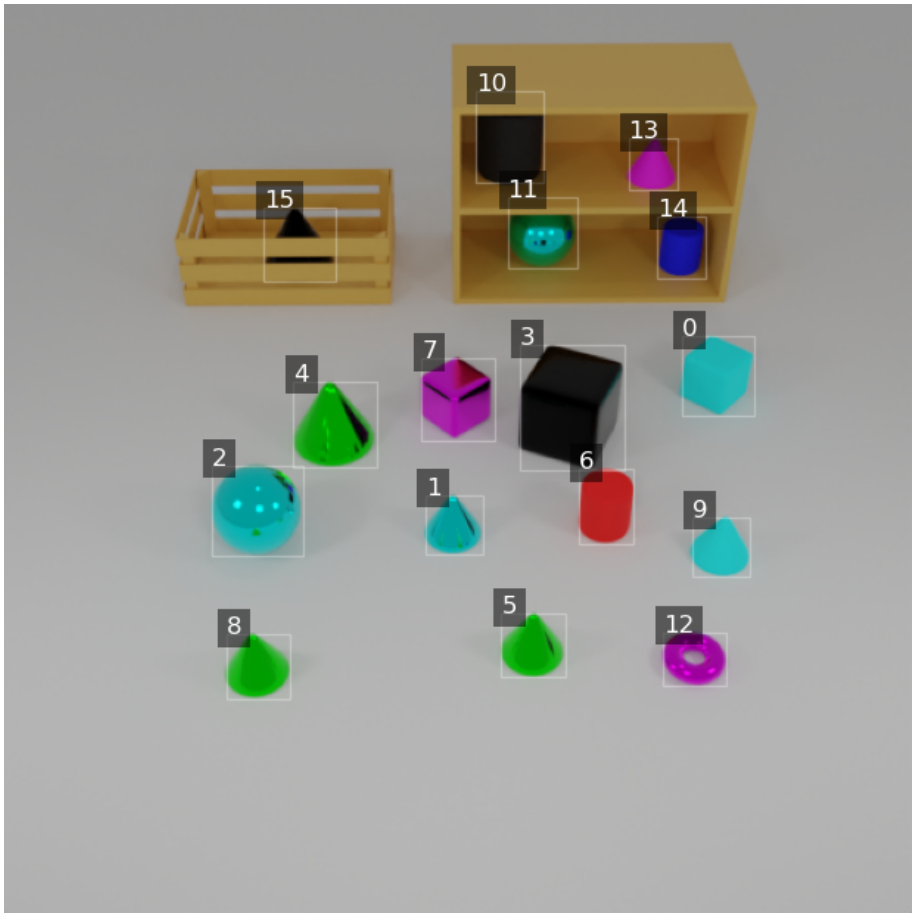

Figure 5.4: Objects used in tests with basic semantic and relational queries.

The results reveals that the system reliably grounds paraphrastic and metaphorical expressions whose lexical forms are absent from the training set, thereby evidencing a semantic generalization. Queries formulated through indirect references to color, material, or geometry are resolved with high fidelity, indicating that the joint vision-language manifold learned during pre-training retains sufficient density to bridge linguistic variability and visual attributes. This performance is contingent on the representational adequacy of the frozen text encoder: its contextualized embeddings, acquired from large-scale corpora, embed semantic relations such that phrases sharing abstract meaning occupy neighboring regions of the latent space. When these embeddings are compared with visual features, alignment is achieved through distributional proximity rather than explicit symbolic decomposition, enabling correct retrieval even in the absence of exact lexical overlap.

5.3.3 Test 3: Quantitative and Comparative Queries

To evaluate the system's ability in quantitative and comparative reasoning, we designed queries involving object counting, ratio estimation, and threshold-based comparisons. These tests were applied to the configuration illustrated in Figure 5.5. Table 5.4

lists the set of queries used in this evaluation.

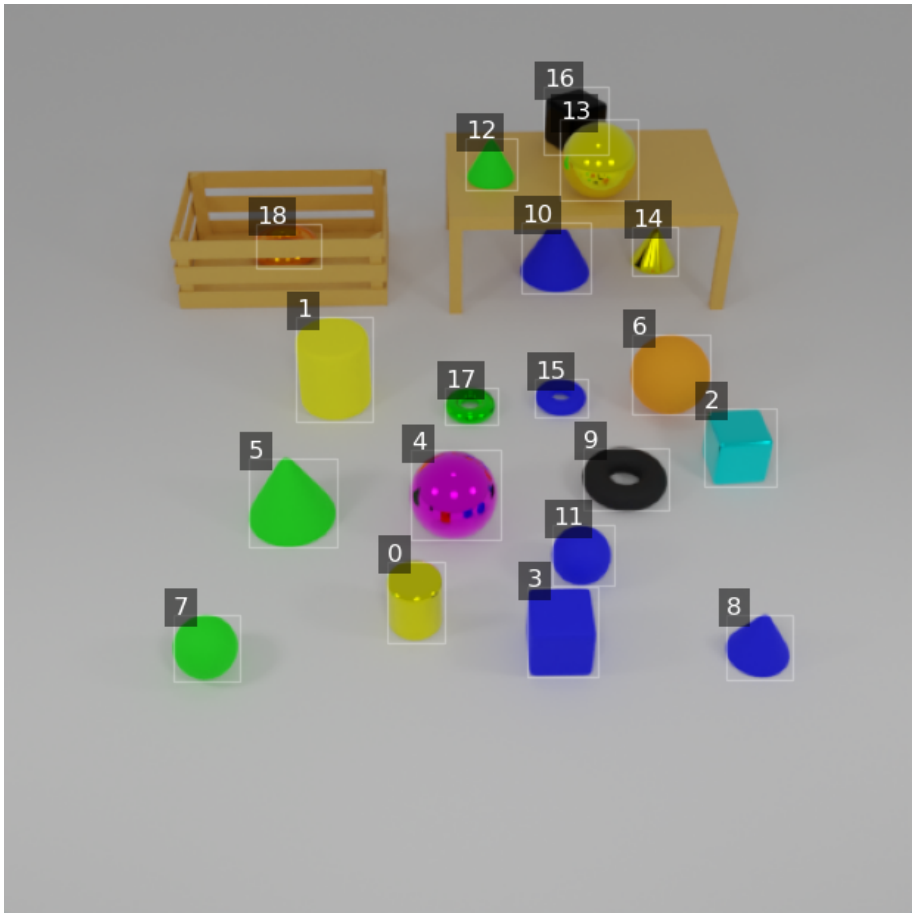

Figure 5.5: Objects used in tests with quantitative and comparative queries.

The first six queries show consistent results, indicating that the system can reliably count objects based on attributes, combine conditions, and compare numerical values. The query `count [a metal object]` correctly returns the number of metallic items, and the comparison `count [a metal object] > 5` is also evaluated correctly. The total object count and the ratio computation `count [a sphere] / count [*]` both match the ground truth, showing that the system handles basic arithmetic over counts. Also, the query `count ([a blue object] and [on the floor])` confirms that compound filters are working numerically, and the model is able to compare quantities across distinct categories as seen in the query comparing blue objects on the floor to green ones.

However, the last two rows in Table 5.4 highlight the limitations. The count of pink objects is off by one, and the model incorrectly identifies six black spheres, while none exist in the ground truth. These errors suggest the model is susceptible to blending or misinterpreting multi-attribute combinations. This shows a similar behavior observed in Test 1: the model tends to match semantically close but incorrect instances when multiple conditions are tightly bound in a single phrase.

Description	Query	Output	Ground Truth
Counts objects made of metal	`count [a metal object]`	`Number(8)`	`Number(8)`
Checks if metal objects exceed 5	`count [a metal object] > 5`	`Boolean(True)`	`Boolean(True)`
Counts all objects	`count [*]`	`Number(19)`	`Number(19)`
Calculates the ratio of spheres to total objects	`count [a sphere] / count [*]`	`Number(0.263)`	`Number(0.263)`
Counts blue objects on the floor	`count ([a blue object] and [on the floor])`	`Number(5)`	`Number(5)`
Checks if blue objects on the floor exceed green objects	`count ([a blue object] and [on the floor]) > count [a green thing]`	`Boolean(True)`	`Boolean(True)`
Counts pink objects	`count [a pink object]`	`Number(2)`	`Number(1)`
Counts black spheres	`count [a black sphere]`	`Number(6)`	`Number(0)`

Table 5.4: Examples of quantitative and comparative queries, covering counts, ratios, and logical evaluations over object sets.

Chapter 6

Conclusion and Future Work

This work proposed a neuro-symbolic framework designed to overcome the limitations of existing prompt-guided object detection models in handling compositional and logical queries. Conventional open-vocabulary detectors are primarily constructed to associate single textual prompts with visual regions, relying exclusively on embedding-based similarity without mechanisms for handling structured reasoning or logical composition. In contrast, the architecture presented in this research introduces a clean separation between perceptual grounding and symbolic reasoning. The neural module is responsible for localizing objects and aligning them with textual prompts via a deep learning pipeline that integrates a Region Proposal Network, an object proposal encoder, a transformer-based text encoder, and an alignment module that projects both modalities into a shared embedding space. This pipeline effectively captures cross-modal relationships, providing a reliable mapping between visual object proposals and natural language descriptions through cosine similarity over normalized embeddings.

Building upon these neural outputs, the symbolic component operates as a formal interpreter, explicitly designed to process and execute logical queries over the detected objects. The interpreter consists of a pipeline that begins with lexical analysis, where the input query string is tokenized into atomic units representing prompts, logical operators, and structural elements. This token stream is then passed to a parser, which constructs an AST reflecting the hierarchical structure of the logical formula. The AST is subsequently validated and enriched through semantic analysis, ensuring consistency in operator usage, type compatibility, and uncertainty handling. Finally, the evaluator traverses the AST, executing set-theoretic operations such as conjunction, disjunction, and negation over the detection results produced by the neural module. Each leaf node corresponds to an atomic prompt whose alignment scores determine the set of detected objects satisfying that condition, while internal nodes represent logical compositions that manipulate these sets according to the specified query. This architecture not only enables the resolution of arbitrarily complex queries expressed in natural language but does so in a fully transparent manner, providing interpretable intermediate steps that are inaccessible in purely embedding-based models.

An immediate and significant consequence of this design choice is the enhancement of transparency and interpretability in the detection process. Unlike purely neural or embedding-based approaches, where the semantic content of a query is collapsed into a single high-dimensional vector, the symbolic layer preserves the intermediate results

associated with each sub-prompt. This transparency allows users to trace precisely how individual components of a query contribute to the final detection outcome. Each logical clause in the query produces a corresponding set of detections, and the symbolic reasoning process clearly documents how these sets are intersected, unified, or subtracted to arrive at the final result. This explicit reasoning pathway is particularly valuable in contexts where explainability and accountability are critical, such as decision-making systems, safety-critical applications, and interactive AI tools.

In order to empirically validate the proposed framework and to rigorously evaluate its capacity for compositional generalization, the AOVR-Dataset was developed as part of this research. This dataset was designed to provide a controlled experimental environment where both perceptual recognition and logical reasoning could be systematically tested. Each image in the dataset is a synthetically generated 3D scene composed of multiple objects, where the properties of shape, color, and spatial configuration are parameterized to produce diverse yet structured variations. Crucially, the dataset includes not only visual annotations but also a comprehensive set of logical queries paired with ground truth answers. The training partition contains only atomic prompts, corresponding to single-attribute descriptions, whereas the validation set introduces a wide spectrum of compositional queries, including combinations of up to five attributes. This partitioning ensures that success on compositional tasks cannot be attributed to memorization but instead reflects the system's ability to correctly execute symbolic compositions over atomic perceptual detections.

The evaluation of the proposed architecture focused on validating three core aspects: the accuracy of perceptual grounding, the reliability of text-to-object alignment, and the correctness of logical reasoning over detection outputs. The neural module demonstrated strong perceptual performance, achieving a mean average precision of 99.48% at an IoU threshold of 0.5 and 98.01% at 0.75, with an average IoU of 93.39%. These results confirm that the detector consistently produces bounding boxes that closely match ground truth annotations, providing a robust foundation for the subsequent symbolic reasoning.

In parallel, the alignment between textual prompts and visual objects achieved an accuracy of 99.35% on single-attribute queries. This indicates that the embedding space effectively captures the semantic correspondence between language and vision, ensuring that object proposals are correctly associated with their respective descriptions. As the symbolic reasoning relies directly on these alignments, this high accuracy is critical for the integrity of the overall system.

The symbolic module was evaluated using a diverse set of compositional queries combining multiple attribute-based prompts with logical operators such as conjunction (AND), disjunction (OR), and negation (NOT). Reasoning was tested under three uncertainty-handling strategies: probabilistic, fuzzy, and boolean interpretations. For queries involving one or two attributes, accuracy consistently exceeded 90% across all strategies. However, performance gradually declined as query complexity increased, reaching approximately 63% for compositions involving five attributes. This trend aligns with the expected combinatorial effect, where the likelihood of satisfying all constraints diminishes with each additional condition. Among the reasoning strategies, the fuzzy interpretation consistently performed slightly better on complex queries, likely due to its capacity to

tolerate partial matches and smooth over the brittleness inherent in strict threshold-based boolean logic.

In addition to quantitative analyses, a set of qualitative experiments was conducted to evaluate how the system handles more naturalistic and semantically varied queries. These experiments involved testing the framework on paraphrased expressions, metaphorical descriptions, spatial relations, and quantitative comparisons such as object counts and ratios. The system demonstrated a notable degree of flexibility in handling paraphrases and relational conditions, correctly mapping them to their intended semantic interpretations in most cases. Nonetheless, certain limitations became apparent. Specifically, when queries contained tightly coupled multi-attribute descriptions expressed as single phrases, the embedding space occasionally produced ambiguous representations that hindered accurate alignment. Similarly, in queries involving explicit counts or ratios, the system sometimes generated overcounts or introduced false positives, suggesting that purely set-based composition is insufficient for capturing the precise semantics of numerical reasoning.

These observations highlight a fundamental tension inherent in the interplay between continuous embedding spaces and discrete logical operations. While the symbolic module excels at compositionality once accurate atomic detections are available, the reliance on embedding-based semantic similarity for prompt alignment introduces potential sources of ambiguity, especially when linguistic expressions involve subtle nuances or dense attribute combinations.

In summary, the findings presented in this work provide strong empirical support for the hypothesis that a modular neuro-symbolic architecture can enable prompt-guided object detection with logical compositionality, without sacrificing the generalization benefits afforded by large-scale vision-language pretraining. By cleanly decoupling visual recognition from symbolic reasoning, the proposed framework offers a practical and interpretable approach to complex scene understanding. This research contributes to the broader endeavor in artificial intelligence to integrate the strengths of deep perceptual models with the rigor and structure of formal symbolic systems, advancing the field towards more robust, flexible, and cognitively inspired AI frameworks.

6.1 Difficulties and Limitations

The development of this research faced challenges related to the absence of existing datasets and tools tailored to prompt-based detection with symbolic composition. This led to the design and construction of both a custom dataset and a dedicated symbolic execution engine. Moreover, while the framework achieved excellent performance in controlled synthetic environments, its evaluation was limited to static scenes with predefined parameters.

6.2 Broader Applications and Implications

The capabilities unlocked by this neuro-symbolic approach extend far beyond academic benchmarks:

- **Robotics and Embodied AI**: Robots require the ability to reason about their environment, often needing to satisfy complex conditions, such as "pick up the red cube that is not next to any green objects." The proposed framework offers a natural way to interface between visual perception and task-level planning.
- **Surveillance and Security**: Security systems could leverage logical queries to monitor for complex situations, e.g., "detect all people who are not wearing helmets in construction zones" or "find vehicles parked outside designated areas."
- **Interactive AI and Visual Assistants**: Applications such as AI-driven photo organization, content moderation, or virtual assistants can benefit from the ability to answer nuanced questions like "show me all images with cats sitting on chairs but no dogs."
- **Scientific Analysis**: In domains like biology or astronomy, researchers frequently require systems that can query complex patterns in visual data, e.g., "find all cells that are circular and do not overlap with others" or "identify galaxies with certain structural attributes."
- **Cognitive AI Research**: This framework contributes to the broader goal of building AI systems that better mirror human cognitive abilities, combining perception, language, and reasoning in a unified pipeline.

6.3 Future Work

This research opens several avenues for future development and exploration, including the following directions:

1. **Knowledge Integration**: Integrate external knowledge sources, such as ontologies, knowledge graphs, or large language models, into the symbolic module. This enhancement would provide the system with richer semantic understanding and enable reasoning grounded in common-sense knowledge.
2. **Scalability and Optimization**: Improve the computational efficiency of the symbolic reasoning engine to handle larger scenes with hundreds of objects and multiple simultaneous prompts. This includes algorithmic optimizations, parallelization strategies, and possibly approximate reasoning techniques to maintain low latency in real-time applications.
3. **Cross-Modal Generalization**: Generalize the proposed neuro-symbolic framework beyond the visual domain, applying the same architecture to other modalities such as audio, text, LiDAR, or multimodal sensor fusion. The principle of prompt-based grounding combined with symbolic composition could be extended to any structured perceptual task where compositionality is valuable.
4. **Real-World Deployment**: Transition from synthetic benchmarks to real-world datasets and applications. This involves testing the framework under realistic conditions, including noise, occlusions, and domain shifts, as well as deploying it in robotics, autonomous systems, and interactive AI applications.
5. **Dataset Expansion**: Further develop the AOVR-Dataset by incorporating more diverse object categories, richer attribute variations, occlusions, lighting conditions,

and dynamic scenes. Expanding the dataset would not only support the evaluation of future versions of the framework but also benefit the broader research community working on neuro-symbolic AI.

6.4 Publications and Datasets

The research conducted in this thesis has led to the following publications and publicly available datasets:

- Ferreira, S., Martins, A., Costa, D. G., & Silva, I. (2025). **Integrating Textual Queries with AI-Based Object Detection: A Compositional Prompt-Guided Approach**. Sensors, 25(7), 2258. DOI: 10.3390/s25072258
- Ferreira, S., Martins, A., Costa, D. G., & Silva, I. (2025). Annotated Objects for Visual Reasoning Dataset. Mendeley Data, V1. DOI: 10.17632/bn5cbjts6j.1

Bibliography

Ahmad, Hafiz Mughees & Afshin Rahimi (2022), 'Deep learning methods for object detection in smart manufacturing: A survey', *Journal of Manufacturing Systems* **64**, 181–196.

Aho, Alfred V, Monica S Lam, Ravi Sethi & Jeffrey D Ullman (2007), *Compilers: Principles, Techniques, and Tools*, 2ª edição, Pearson Education.

Alayrac, Jean-Baptiste et al. (2022), Flamingo: a visual language model for few-shot learning, *em* 'arXiv preprint arXiv:2204.14198'.

Ba, Jimmy Lei, Jamie Ryan Kiros & Geoffrey E Hinton (2016), Layer normalization, *em* 'arXiv preprint arXiv:1607.06450'.

Baltrušaitis, Tadas, Chaitanya Ahuja & Louis-Philippe Morency (2018), 'Multimodal machine learning: A survey and taxonomy', *IEEE transactions on pattern analysis and machine intelligence* **41**(2), 423–443.

Biten, Ali Furkan, Emre Arabaci, Atikah Mandal et al. (2022), 'Latr: Layout-aware transformer for scene-text vqa', *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pp. 10735–10745.

Carion, Nicolas, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov & Sergey Zagoruyko (2020), End-to-end object detection with transformers, *em* 'European Conference on Computer Vision (ECCV)', pp. 213–229.

Colelough, Brandon & William Regli (2024), 'Neuro-symbolic ai in 2024: A systematic review', *arXiv preprint arXiv:2401.01040* .

Dehdari, Jon & Kevin Duh (2016), 'How to design an nlp pipeline for a morphologically rich language', *arXiv preprint arXiv:1610.02517* .

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li & Li Fei-Fei (2009), 'Imagenet: A large-scale hierarchical image database', pp. 248–255.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee & Kristina Toutanova (2019), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *Proceedings of NAACL* .

Dong, Hongyu, Jiayuan Mao, Chuang Gan et al. (2019), 'Neural logic machines', *International Conference on Learning Representations* .

Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn & Andrew Zisserman (2010), 'The pascal visual object classes (voc) challenge', *International journal of computer vision* **88**(2), 303–338.

Gage, Philip (1994), 'A new algorithm for data compression', *C Users J.* **12**(2), 23–38.

Gao, Tianyu, Xingcheng Yao & Danqi Chen (2021), Simcse: Simple contrastive learning of sentence embeddings, *em* 'Proceedings of EMNLP'.

Garcez, Artur d'Avila, Marco Gori, Luis C Lamb, Luciano Serafini, Michael Spranger & Pietro Tran (2019), 'Neuro-symbolic ai: The third wave', *arXiv preprint arXiv:1905.06088* .

Girshick, Ross (2015), Fast r-cnn, *em* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 1440–1448.

Girshick, Ross, Jeff Donahue, Trevor Darrell & Jitendra Malik (2014), Rich feature hierarchies for accurate object detection and semantic segmentation, *em* 'Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)', pp. 580–587.

Grune, Dick & Ceriel JH Jacobs (2012), *Parsing Techniques: A Practical Guide*, Springer Science & Business Media.

Gu, Jiuxiang, Zhenhua Wang, Jason Kuen, Lin Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xinchao Wang & Gang Wang (2018), 'Recent advances in convolutional neural networks', *Pattern recognition* **77**, 354–377.

Gu, Xiuye, Tsung-Yi Lin, Weicheng Kuo & Yin Cui (2022), Open-vocabulary object detection via vision and language knowledge distillation, *em* 'International Conference on Learning Representations (ICLR)'.

Hadsell, Raia, Sumit Chopra & Yann LeCun (2006), Dimensionality reduction by learning an invariant mapping, *em* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 2, pp. 1735–1742.

He, Kaiming, Georgia Gkioxari, Piotr Dollár & Ross Girshick (2017), Mask r-cnn, *em* 'Proceedings of the IEEE international conference on computer vision (ICCV)', pp. 2961–2969.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren & Jian Sun (2016), Deep residual learning for image recognition, *em* 'Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)', pp. 770–778.

Hopcroft, John E, Rajeev Motwani & Jeffrey D Ullman (2019), *Introduction to Automata Theory, Languages, and Computation*, Pearson.

Hossain, Md. Yearat, Ifran Rahman Nijhum, Md. Tazin Morshed Shad, Abu Adnan Sadi, Md. Mahmudul Kabir Peyal & Rashedur M. Rahman (2023), 'An end-to-end pollution analysis and detection system using artificial intelligence and object detection algorithms', *Decision Analytics Journal* **8**, 100283.

Howard, Andrew G, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto & Hartwig Adam (2017), Mobilenets: Efficient convolutional neural networks for mobile vision applications, *em* 'arXiv preprint arXiv:1704.04861'.

Howard, Andrew, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le & Hartwig Adam (2019), 'Searching for mobilenetv3', *arXiv preprint arXiv:1905.02244* .

Incitti, Francesca, Federico Urli & Lauro Snidaro (2023), 'Beyond word embeddings: A survey', *Information Fusion* **89**, 418–436.

Jiang, Qing, Feng Li, Zhaoyang Zeng et al. (2024), 'T-rex2: Towards generic object detection via text-visual prompt synergy', *arXiv preprint arXiv:2403.14610* .

Kaur, Jaskirat & Williamjeet Singh (2024), 'A systematic review of object detection from images using deep learning', *Multimedia Tools and Applications* **83**(4), 12253–12338.

Khan, Muhammad et al. (2023), 'A survey of neurosymbolic visual reasoning with scene graphs and common sense knowledge', *Neuro-Symbolic AI Journal* .

Knuth, Donald E (1968), 'Semantics of context-free languages', *Mathematical systems theory* **2**(2), 127–145.

Krichen, Moez (2023), 'Convolutional neural networks: A survey', *Computers* **12**(8), 151.

Krizhevsky, Alex, Ilya Sutskever & Geoffrey E Hinton (2012), Imagenet classification with deep convolutional neural networks, *em* 'Advances in neural information processing systems (NeurIPS)', Vol. 25, pp. 1097–1105.

Kudo, Taku & John Richardson (2018), Subword regularization: Improving neural network translation models with multiple subword candidates, *em* 'ACL', pp. 66–75.

Lamb, Luis C, Artur d'Avila Garcez et al. (2020), 'Graph neural networks meet neural-symbolic computing: A survey and perspective', *Frontiers in Artificial Intelligence* **3**.

LeCun, Yann, Léon Bottou, Yoshua Bengio & Patrick Haffner (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

LeCun, Yann, Yoshua Bengio & Geoffrey Hinton (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

Li, Panfeng, Qikai Yang, Xieming Geng, Wenjing Zhou, Zhicheng Ding & Yi Nian (2024), 'Exploring diverse methods in visual question answering', *arXiv preprint arXiv:2404.13565* .

Li, Wuyang, Xinyu Liu, Jiayi Ma & Yixuan Yuan (2024), Cliff: Continual latent diffusion for open-vocabulary object detection, *em* 'ECCV'.

Li, Yang, Wei Li, Qi Wang & Fei-Yue Chen (2021), 'A survey of convolutional neural networks: analysis, applications, and prospects', *IEEE Transactions on Artificial Intelligence* **2**(4), 336–352.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár & C Lawrence Zitnick (2014), Microsoft coco: Common objects in context, *em* 'ECCV', Springer, pp. 740–755.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He & Piotr Dollár (2017), Focal loss for dense object detection, *em* 'Proceedings of the IEEE International Conference on Computer Vision (ICCV)', pp. 2980–2988.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu & Alexander C Berg (2016), Ssd: Single shot multibox detector, *em* 'European Conference on Computer Vision', Springer, pp. 21–37.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer & Veselin Stoyanov (2019), Roberta: A robustly optimized bert pretraining approach, *em* 'arXiv preprint arXiv:1907.11692'.

Loshchilov, Ilya & Frank Hutter (2018), 'Decoupled weight decay regularization', *arXiv preprint arXiv:1711.05101* .

Lu, Jiasen, Dhruv Batra, Devi Parikh & Stefan Lee (2019), 'Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks', *Advances in Neural Information Processing Systems* **32**.

Ma, Chuofan, Yi Jiang, Xin Wen et al. (2023), Codet: Co-occurrence guided region-word alignment for open-vocabulary object detection, *em* 'NeurIPS'.

Manakitsa, Nikoleta, George S. Maraslidis, Lazaros Moysis & George F. Fragulis (2024), 'A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision', *Technologies* **12**(2).

Manning, Christopher D & Hinrich Schutze (2003), *Foundations of Statistical Natural Language Processing*, MIT press.

Mao, Jiayuan, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum & Jiajun Wu (2019), Neuro-symbolic concept learner, *em* 'International Conference on Learning Representations'.

Marino, Kenneth, Mohammad Rastegari, Ali Farhadi & Hannaneh Hajishirzi (2021), KRISP: Integrating implicit and symbolic knowledge for open-domain knowledge-based VQA, *em* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 14111–14121.

Mashayekhi, Zahra & The Neuro-Symbolic AI Group (2024), 'Neuro-symbolic AI in 2024: A systematic review', *arXiv preprint arXiv:2501.05435* .

Minderer, Matthias, Alexey Gritsenko, Austin Stone et al. (2022), Simple open-vocabulary object detection with vision transformers, *em* 'European Conference on Computer Vision (ECCV)'.

Muchnick, Steven S (1997), *Advanced Compiler Design and Implementation*, Morgan Kaufmann.

Nair, Vinod & Geoffrey E Hinton (2010), Rectified linear units improve restricted boltzmann machines, *em* 'Proceedings of the 27th international conference on machine learning (ICML)', pp. 807–814.

Oord, Aaron van den, Yazhe Li & Oriol Vinyals (2018), 'Representation learning with contrastive predictive coding', *arXiv preprint arXiv:1807.03748* .

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai & Soumith Chintala (2019), 'Pytorch: An imperative style, high-performance deep learning library', *Advances in Neural Information Processing Systems* **32**.

Preeti & Chhavi Rana (2024), 'Artificial intelligence based object detection and traffic prediction by autonomous vehicles – a review', *Expert Systems with Applications* **255**, 124664.

Radford, Alec, Jong Wook Kim, Christopher Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pam Mishkin, Jack Clark, Gretchen Krueger & Ilya Sutskever (2021), Learning transferable visual models from natural language supervision, *em* 'Proceedings of ICML'.

Redmon, Joseph, Santosh Divvala, Ross Girshick & Ali Farhadi (2016), You only look once: Unified, real-time object detection, *em* 'Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)', pp. 779–788.

Ren, Shaoqing, Kaiming He, Ross Girshick & Jian Sun (2015), Faster r-cnn: Towards real-time object detection with region proposal networks, *em* 'Advances in neural information processing systems (NeurIPS)', Vol. 28.

Rocktaschel, Tim & Sebastian Riedel (2017), 'End-to-end differentiable proving', *Advances in Neural Information Processing Systems* pp. 3788–3800.

Schroff, Florian, Dmitry Kalenichenko & James Philbin (2015), Facenet: A unified embedding for face recognition and clustering, *em* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 815–823.

Sennrich, Rico, Barry Haddow & Alexandra Birch (2016), Neural machine translation of rare words with subword units, *em* 'ACL', pp. 1715–1725.

Serafini, Luciano & Artur S Garcez (2016), 'Logic tensor networks: Deep learning and logical reasoning from data and knowledge', *Proceedings of the European Conference on Machine Learning* pp. 539–554.

Simonyan, Karen & Andrew Zisserman (2015), Very deep convolutional networks for large-scale image recognition, *em* 'International Conference on Learning Representations (ICLR)'.

Sundar, Ravi & Priya Goel (2023), Lit-4-rsvqa: Lightweight transformer for resource-constrained vqa, *em* 'Proceedings of the AAAI Conference on Artificial Intelligence', pp. 1234–1240.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke & Andrew Rabinovich (2015), Going deeper with convolutions, *em* 'Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)', pp. 1–9.

Tan, Hao & Mohit Bansal (2019), 'Lxmert: Learning cross-modality encoder representations from transformers', *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* pp. 5100–5111.

Tan, Mingxing & Quoc Le (2019), Efficientnet: Rethinking model scaling for convolutional neural networks, *em* 'International Conference on Machine Learning (ICML)', PMLR, pp. 6105–6114.

Tian, Zhi, Chunhua Shen, Hao Chen & Tong He (2019), Fcos: Fully convolutional one-stage object detection, *em* 'Proceedings of the IEEE International Conference on Computer Vision (ICCV)', pp. 9627–9636.

Vaswani, A (2017), 'Attention is all you need', *Advances in Neural Information Processing Systems* .

Wang, Hao, Pengzhen Ren, Zequn Jie et al. (2024), 'Ov-dino: Unified open-vocabulary detection with language-aware selective fusion', *arXiv preprint arXiv:2407.07844* .

Wang, Jian, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen & Ying Wu (2014), Learning fine-grained image similarity with deep

ranking, *em* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 1386–1393.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le et al. (2016), Google's neural machine translation system: Bridging the gap between human and machine translation, *em* 'arXiv preprint arXiv:1609.08144'.

Xiao, Lei & Yue Huang (2023), 'Attention-driven multimodal alignment for visual question answering', *Pattern Recognition Letters* **177**, 88–97.

Xie, Xingxing, Chunbo Lang, Shicheng Miao, Gong Cheng, Ke Li & Junwei Han (2023), 'Mutual-assistance learning for object detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**(12), 15171–15184.

Xie, Xingxing, Gong Cheng, Jiabao Wang, Ke Li, Xiwen Yao & Junwei Han (2024), 'Oriented r-cnn and beyond', *International Journal of Computer Vision* **132**(7), 2420–2442.

Yao, Lewei, Jianhua Han, Xiaodan Liang et al. (2023), Detclipv2: Scalable open-vocabulary object detection pre-training via word-region alignment, *em* 'CVPR'.

Yi, Kunyang, Jiajun Wu, Chuang Gan & Joshua B Tenenbaum (2020), 'Clevrer: Collision events for video reasoning', *arXiv preprint arXiv:2001.01320* .

Zhang, Chuhan, Chaoyang Zhu, Pingcheng Dong et al. (2025), Cckt-det: Cyclic contrastive knowledge transfer for open-vocabulary object detection, *em* 'International Conference on Learning Representations (ICLR)'.

Zhang, Hanwang, Kun Li, Lei Zhang, Yafeng Li, Lei Zhang, Yafeng Li, Lei Zhang, Yafeng Li, Lei Zhang & Yafeng Li (2023), 'mgte: Multilingual generative text encoder for cross-lingual text-image retrieval', *arXiv preprint arXiv:2301.12345* .

Zhang, Wei, Ming Li, Lei Wang et al. (2024), 'mgte: Generalized long-context text representation and reranking models for multilingual text retrieval', *arXiv preprint arXiv:2407.19669* .

Zhang, Xi, Feifei Zhang & Changsheng Xu (2023), Vqacl: A novel visual question answering continual learning setting, *em* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 27218–27227.

Zhao, Zhiqiang, Peng Zheng, Shou-tao Xu & Xindong Wu (2019), 'Object detection with deep learning: A review', *IEEE transactions on neural networks and learning systems* **30**(11), 3212–3232.

Zheng, Yanhao & Kai Liu (2024), 'Training-free boost for open-vocabulary object detection with confidence aggregation', *arXiv preprint arXiv:2404.08603* .

Zhong, Yiwu, Jianwei Yang, Pengchuan Zhang et al. (2022), Regionclip: Region-based language-image pretraining, *em* 'CVPR'.