University of Naples "Parthenope"
Department of Science and Technology
M. Sc. in Applied Computer Science (Machine Learning and Big Data)

Master Degree Thesis

A CLIP-based approach to Text-to-Image Retrieval in Marine Monitoring

Advisor
Prof. Antonino Staiano

Co-Advisor
Dr. Alessio Pierluigi Placitelli

Examiner
Prof. Francesco Camastra

Candidate
Renato Esposito

STUDENT ID
0120000278

Academic Year 2024-2025

ABSTRACT

The conservation of marine biodiversity is one of the priorities defined by the United Nations 2030 Agenda, specifically under Sustainable Development Goal 14 (Life below water). In this context, sea turtles represent a keystone species whose monitoring is crucial but often limited by the high costs and inefficiencies of traditional approaches. Recent advances in Artificial Intelligence and particularly in Vision–Language Models, offer new opportunities for supporting marine research through the automated analysis and retrieval of visual data.

This thesis presents the design and implementation of a specialized text-to-image retrieval system for sea turtles, built on top of the CLIP model. To address the lack of suitable training data, a domain-specific dataset was created by generating image captions with automatic models and refining them through manual review. The model was then fine-tuned with Low-Rank Adaptation, an efficient parameter adaptation strategy, allowing specialization to the marine domain while preserving general knowledge from large-scale pretraining.

From a methodological perspective, the work proposes a combined optimization strategy: alongside the CLIP standard contrastive loss, the Unified Contrastive Learning loss was integrated to support multi-positive learning. This design enables the model to better capture subtle distinctions across similar visual scenarios, such as poses, interactions, or environmental conditions, without collapsing semantically related samples into a single representation.

Experimental results demonstrate that the proposed system improves retrieval accuracy in the sea turtle domain, supporting both category-level and fine-grained instance-level queries. The contributions of this work lie in the creation of a dataset for marine monitoring, the adaptation of a general-purpose Vision–Language Model to a highly specialized ecological context, and the proposal of a loss design that balances global discrimination and fine-grained differentiation. These outcomes not only advance the application of AI in marine conservation but also

align with the objectives of Sustainable Development Goal 14, fostering more effective and scalable biodiversity monitoring solutions.

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

1. INTRODUCTION

This chapter introduces the context and motivations underlying this thesis. It presents the relevance of Artificial Intelligence in environmental monitoring, with a specific focus on its role in protecting marine biodiversity. The case of sea turtles as the target domain is discussed, followed by the objectives of the work and an outline of the thesis structure.

1.1 Context and Motivations

In 2015, the **United Nations Organization** (ONU) adopted the *2030 Agenda for Sustainable Development* [49], a global action plan aimed at ensuring a more equitable, prosperous, and environmentally responsible future for current and future generations. The Agenda consists of 17 **Sustainable Development Goals** (SDGs) and *169 specific targets*, which cover crucial areas such as poverty reduction, education, health, gender equality, decent work, climate change, and ecosystem protection.
The 17 goals cover a broad spectrum of global priorities, including:

1. Eliminate poverty.
2. Eliminate hunger.
3. Good health and well-being.
4. Quality education.
5. Gender equality.
6. Clean water and sanitation.
7. Affordable and clean energy.

8. Decent work and economic growth.
9. Enterprise, innovation, and infrastructure.
10. Reduced inequalities.
11. Sustainable cities and communities.
12. Responsible consumption and production.
13. Climate action.
14. **Life below water**.
15. Life on land.
16. Peace, justice, and strong institutions.
17. Partnerships for the goals.

In this work, we focus specifically on SDG 14, which aims to "*conserve and sustainably use the oceans, seas and marine resources for sustainable development.*" As a key pillar of the Agenda for Environmental Sustainability, this goal addresses the critical role of marine ecosystems in supporting life on Earth. Oceans, covering over 70% of the planet's surface, serve essential functions: they regulate the global climate, act as carbon sinks, and provide vital resources, including food, transportation, and livelihoods, for billions of people. Nevertheless, these ecosystems face escalating threats, such as pollution, biodiversity loss, and the intensifying impacts of climate change, jeopardizing their ability to sustain future generations.
Among the most relevant targets related to this goal are:

- 14.1: by 2025, prevent and significantly reduce marine pollution of all kinds, in particular from land-based activities, including marine debris and nutrient pollution.
- 14.2: sustainably manage and protect marine and coastal ecosystems, preventing significant adverse impacts and strengthening their resilience.

Within this scenario, monitoring marine biodiversity and protecting vulnerable species are key actions. Sea turtles, for example, are considered keystone species of ocean ecosystems, but are currently severely threatened by floating plastic, abandoned fishing nets, and invasive human activities. Automatically tracking their presence or distinguishing them from potentially harmful elements such as marine debris or other animals is a critical and complex task.
In this context, **Artificial Intelligence (AI)** and in particular **Machine Learning (ML)** and **Deep Learning (DL)** techniques, offer new opportunities for analyzing large amounts of environmental and visual data, making marine fauna monitoring, species recognition and the classification of potentially dangerous elements for the underwater environment, more efficient and scalable.

1.1.1 The Role of Artificial Intelligence

In recent years, **AI** has played an increasingly central role in addressing complex environmental challenges, thanks to its ability to process large volumes of data, uncover latent patterns, and support data-driven decision-making. In particular, techniques from **ML** and **DL** have found wide application in domains related to the monitoring and protection of both marine and terrestrial ecosystems.
One of the major advantages of AI is its capacity to automate the analysis of heterogeneous data sources, including remote sensing, underwater photography, drone footage, smart buoys, and weather stations. This has enabled the development of intelligent systems for biodiversity monitoring, detection of pollution, protection of endangered species, and prevention of environmental disasters.
In the marine domain, **convolutional neural networks** (CNNs) have proven particularly effective in visual recognition tasks such as the automatic identification of marine species from images and videos, seabed mapping, and real-time tracking of aquatic animals [43, 70].
Moreover, several studies have leveraged ML algorithms to detect floating debris (e.g., macroplastics), hydrocarbons, and other contaminants using multispectral data from Sentinel-2 satellites and Synthetic Aperture Radar (SAR). For instance, the *MariNeXt* model outperformed baseline segmentation systems by over 12% in multimodal tasks [34, 51].
Recent developments have also explored the integration of AI into embedded

systems for on-board processing. These solutions, often deployed on drones or autonomous marine vehicles, use compressed DL architectures to detect oil spills and estimate their thickness with high accuracy and low energy consumption [48].
In parallel, the emergence of **Vision-Language Models** (VLMs), such as CLIP [56] and BLIP-2 [37], is opening new frontiers in the semantic indexing and retrieval of visual environmental data, enabling the exploration of large image archives through natural language queries.
However, despite progress, challenges remain, including **insufficient labeled data**, **natural variability in environmental conditions**, and the **risk of model bias caused by unbalanced or poorly representative datasets**. For this reason, the development of curated datasets and robust methodologies remains a priority in the scientific community.
As a result, the adoption of AI technologies in the context of marine monitoring is not only technologically feasible but also strategically aligned with the objectives of the United Nations 2030 Agenda, particularly Goal 14, which focuses on the conservation of marine ecosystems and the promotion of sustainable ocean management.

1.1.2 The Case of the Sea Turtles

Among the many marine organisms affected by human-induced threats, **sea turtles** represent a species of high ecological value and a conservation priority. All seven existing species are included in the Appendices of the Convention on International Trade in Endangered Species of Wild Fauna and Flora [8] and most are classified as **endangered** or **critically endangered** by the International Union for Conservation of Nature [29].
These animals face multiple challenges, including habitat degradation [77], fisheries bycatch [81], plastic ingestion [66], and climate change [20]. Their migratory behavior, spanning thousands of kilometers in international waters, further complicates conservation efforts and requires coordinated global monitoring strategies.
Traditional methods of monitoring sea turtles, such as physical tagging, manual photographic identification, and field surveys, are expensive, labor-intensive, and limited in spatial and temporal coverage [60]. Recently, image-based techniques (e.g., underwater camera traps, drones, autonomous vehicles) have demonstrated

potential for the automated identification of individuals and behaviors, although their application to sea turtles remains experimental [63], in particular:

- **Drones**: Used for surface census of nests and individuals [75].
- **Underwater camera traps**: Used in pilot studies for facial pattern recognition [14].
- **Neural networks**: Tested for photo-ID analysis as a replacement for manual methods [5].

However, large-scale adoption of these technologies is currently limited by two key factors:

1. *Lack of standardization in image acquisition and annotation methods.*
2. *Scarcity of public datasets for model training.*

Adding to this is the *growing availability of unstructured visual data*, which presents new challenges in terms of organization, indexing, and retrieval. The ability to efficiently search and retrieve relevant images from large datasets, possibly using natural language queries, could support researchers, NGOs, and conservation agencies in cataloging, analyzing, and interpreting visual data related to sea turtles.

1.1.3 Objective of the Work

This thesis work, which has been developed in collaboration with marine biologists from the Turtle Point "Anton Dohrn" research center in Portici, aims to **design and implement a specialized text-to-image retrieval system** able to:

- Semantically **align visual and textual representations** in a shared embedding space, optimized for the sea turtle domain.
- **Allow natural language queries** (e.g., "turtle with a net on its shell") to retrieve relevant images from unstructured datasets.

Existing models such as CLIP [56] and BLIP-2 [37] are designed to be general-purpose, achieving robust zero-shot performance across large domains. However, this same generality limits their effectiveness in specialized domains, where objects or scenarios exhibit unique features that are poorly represented in standard training datasets (e.g., COCO [39], Flickr30k [87]). In our use case of tracking sea turtles in images at different angles, such as from above, this translates into two concrete problems:

- **Bias toward generic features**: representations learned from pre-trained models favor common visual attributes (e.g., the mere presence of the animal) over details critical to biologists, such as specific poses ("solitary turtle swimming at the surface") or interactions with the environment ("a turtle swimming next to another turtle").
- **Poor adaptability to low-resolution contexts**: in images from above, where turtles occupy few pixels and are subject to noise (reflections, waves), generalist models often fail to distinguish subtle variations, essential for behavioural studies.

To overcome the limitations of generalist models in the domain of images of sea turtles, we developed a tailored approach based on three main components:

- **A specialized dataset**, constructed manually, containing image-text pairs that emphasize morphological and behavioral attributes relevant for ecological analysis.
- **Efficient fine-tuning of the CLIP model**, based on LoRA (Low-Rank Adaptation), allows the model to be adapted to the new domain by modifying only a small portion of its parameters, while preserving the knowledge acquired during large-scale pretraining.
- **A combined optimization strategy**, based on a combination of two loss functions: CLIP's contrastive loss [78], to preserve intra-category variability (e.g., different poses, interactions, or environmental contexts) and Unified Contrastive Learning loss [85], to enforce a clear separation between the target class and all other categories.

The system contributes to marine conservation (**SDG 14**) (explained in Section 1.1) through:

- **Research support**: accelerating the analysis of visual datasets for studies on distribution, threats, or behavior.
- **Practical applications** such as identifying individuals or critical situations (e.g., entanglement in nets).

1.2 Thesis Structure

The thesis is organized into six main chapters, each addressing a specific aspect of this work:

- **Chapter 1 – Introduction:** provides the overall context and motivations behind the work, presenting the role of AI in environmental monitoring and its application to the case study of sea turtles. The objectives of the thesis and its structure are also outlined.
- **Chapter 2 – Technical Background and Related Work:** reviews the theoretical foundations relevant to this work, including the evolution from ML to DL, optimization processes, and loss functions. It introduces Transformer architectures, contrastive learning, vision–language models, and retrieval systems, as well as the evaluation metrics employed in this domain.
- **Chapter 3 – Proposed Method:** details the adopted methodology, starting with the CLIP architecture and its components, followed by a discussion on the limitations of the standard CLIP loss and the rationale for alternative strategies such as Unified Contrastive Learning. The chapter concludes by describing the combined loss function and the integration of LoRA for efficient fine-tuning.
- **Chapter 4 – Experiments:** describes the experimental setup, including dataset construction, preprocessing, and caption generation. It also explains the implementation of the dynamic batch sampler and presents the evaluation protocols designed for both category-level and instance-level retrieval.

- **Chapter 5 – Results and Discussion:** reports the experimental results and analyzes them in detail, discussing retrieval performance at both category and instance levels and summarizing the overall findings.
- **Chapter 6 – Conclusions and Future Developments:** presents the conclusions drawn from the study, highlighting the main contributions and limitations of the proposed approach. It also discusses possible future developments and practical implications of this thesis work.

2. TECHNICAL BACKGROUND AND RELATED WORK

This chapter provides the theoretical foundations and a review of related work relevant to this study. It begins with an overview of ML and DL principles, including optimization strategies and loss functions. The chapter then introduces Transformer architectures for vision and text, discusses contrastive learning paradigms, and reviews vision–language models and cross-modal retrieval systems. Finally, it presents commonly adopted evaluation metrics for retrieval tasks.

2.1 Introduction: From Machine Learning to Deep Learning

ML is a branch of **AI** focused on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data. ML has its conceptual roots in Alan Turing's pioneering studies on computational intelligence and the first mathematical models of artificial neurons developed by McCulloch and Pitts (1943) [46].

These works laid the theoretical foundations for what would become one of the most influential fields of modern computer science. However, the transition from theory to applied practice has only been fully realized in the last decade, thanks to the convergence of three critical factors: the availability of large datasets, the exponential evolution of computational capabilities, and the development of increasingly sophisticated algorithms. Today, ML is the driving force behind numerous technological innovations, finding strategic applications in diverse domains such as image recognition and classification, natural language processing (NLP), advanced robotics, and recommender systems.

ML methodologies can be classified into three main categories:

- **Supervised learning**: Algorithms learn from labeled data, that is, input-output pairs, to build models that can predict outcomes on new data. Ex-

amples include support vector machines (SVMs) [22] and Random Forest [6].

- **Unsupervised learning**: used to identify patterns in unlabeled data, using techniques such as clustering (e.g., k-means) and dimensionality reduction (e.g., PCA [84]).
- **Reinforcement Learning**: Based on a reward system, the model learns through interactions with a dynamic environment, as in the case of autonomous robot navigation [74]. Examples of these techniques are Q-Learning [83], Deep Q-Networks (DQN) [26], and Policy Gradient Methods [73].

The evolution of ML reached a decisive turning point with the development of **artificial neural networks** (ANNs) [86], computational architectures inspired by the functioning of biological neurons. These models, initially conceived as shallow networks, have progressively gained complexity and depth thanks to two key factors: the unprecedented availability of large datasets and the advent of advanced computational resources, particularly graphics processing units (GPUs). **ANNs** draw inspiration from the structure and function of biological nervous systems, conceptually materializing the pioneering insights of McCulloch and Pitts [46] into concrete computational models. The introduction of the backpropagation algorithm [62] represented a fundamental methodological breakthrough, enabling the effective training of these architectures.
The structure of ANNs, as shown in figure 2.1, is divided into interconnected components: layers organize artificial neurons in a hierarchy that proceeds from the input layer through one or more hidden layers to the output layer. Each neuron processes incoming information using activation functions, such as sigmoid or ReLU, while weights serve as adaptive parameters that are optimized during training to minimize predictive error. Mathematically, the output of a single neuron can be expressed as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{2.1}$$

where:

- y represents the neuron output;

- $f(\cdot)$ is the activation function (e.g., sigmoid, ReLU, tanh);
- x_i are the neuron inputs ($i = 1, 2, \ldots, n$);
- w_i are the weights associated with each input;
- b is the bias term;
- n is the total number of inputs.

This fundamental computational unit forms the building block of more complex neural architectures.

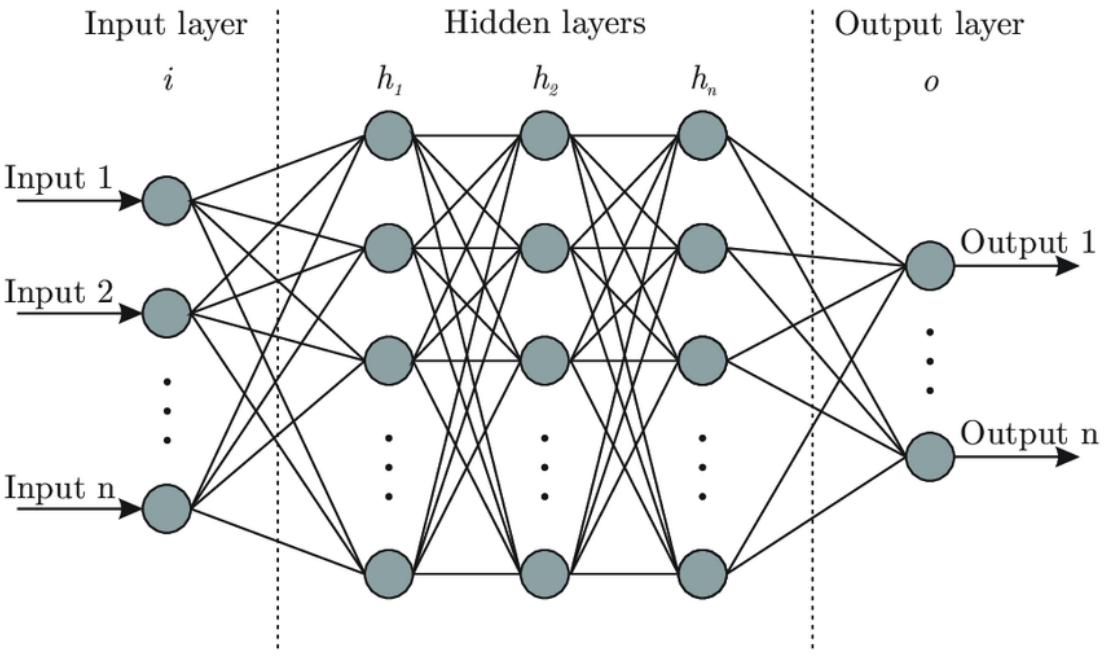

Fig. 2.1: ANN architecture with representation of the fundamental layers. The structure shows the input layer (left) with input nodes ($Input_1, Input_2, ..., Input_n$), the hidden layers (center) with computational neurons ($h_1, h_2, ..., h_n$) that process intermediate features through nonlinear activation functions, and the output layer (right) with prediction results ($Output_1, ..., Output_n$). The connections between nodes, represented by synaptic weights $w_{i,j}$, are learned during training using optimization algorithms such as backpropagation. The mathematical transformation in each neuron can be expressed with 2.1.

Traditional neural networks, characterized by relatively simple architectures with few hidden layers (shallow networks), have found effective application in elementary pattern classification and regression problems. However, despite their theoretical foundation, traditional ANNs have significant limitations in learning complex patterns, namely those represented by high-dimensional data. The **Universal Approximation Theorem** [9] states that a single hidden layer network with a sufficient number of neurons can approximate any continuous function with arbitrary accuracy. However, this theoretical guarantee presents a problem: the number of neurons required can grow exponentially with the complexity of the problem, making the approach computationally intractable for real-world applications. This gap between theoretical possibility and practical feasibility has led to the development of DL architectures.

DL thus emerges as a natural evolution of ANNs, overcoming their structural constraints through deeply layered architectures. This transition from shallow to deep networks not only represents a quantitative increase in layers, but also introduces a new qualitative paradigm: hierarchical representation learning. Each layer of the network extracts progressively more abstract and complex features, moving from primitive elements, such as edges and contours in images, to high-level semantic concepts.

The success of DL is manifested through the development of specialized architectures, each designed to exploit specific data characteristics and computational models. **CNNs** [53] (figure 2.2) exploit spatial locality and translation invariance to achieve extraordinary performance in computer vision tasks, while **Recurrent neural networks** (RNNs) [69] (figure 2.3) and their advanced variants, such as long-short-term memory (LSTM) networks [24], effectively capture temporal dependencies in sequential data, including natural language and time series. The field underwent a significant shift with the introduction of **Transformers** [80], described in Section 2.2, whose attention mechanisms eliminated the need for recurrence while enabling unprecedented scalability and performance, giving rise to transformative models such as BERT [12] and the GPT family [55].

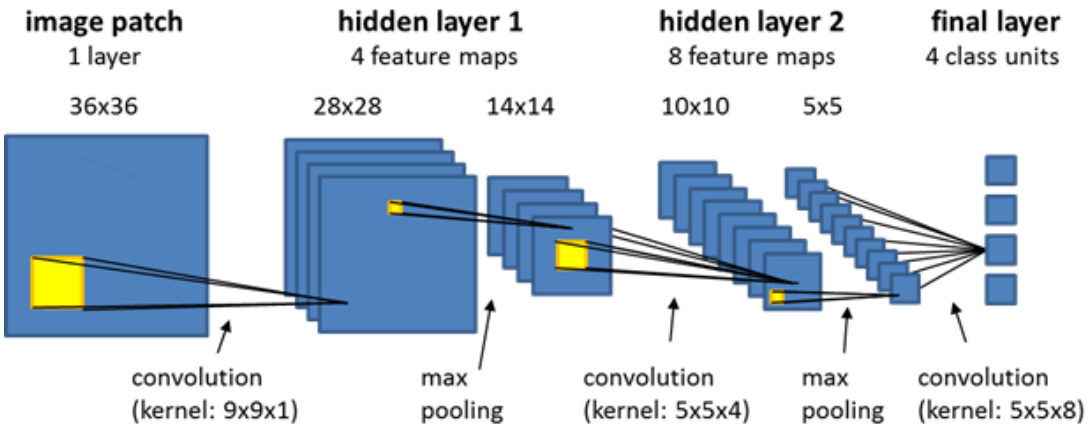

Fig. 2.2: Example of a CNN architecture. The network processes a 36×36 input image (image patch) through a series of hierarchical layers: layer 1 (Hidden Layer 1) applies a convolution with 4 filters (9×9×1 kernel), producing 4 28×28 feature maps, followed by a max-pooling operation that reduces the feature size to 14×14; layer 2 (Hidden Layer 2) uses 8 filters (5×5×4 kernel), generating 8 10×10 feature maps, further reduced to 5×5 via max-pooling; finally, the final layer applies a convolution with a 5×5×8 kernel and a 4-unit fully-connected layer, corresponding to the output classes. The architecture combines multi-scale feature extraction and classification, typical of CNNs for visual recognition tasks.

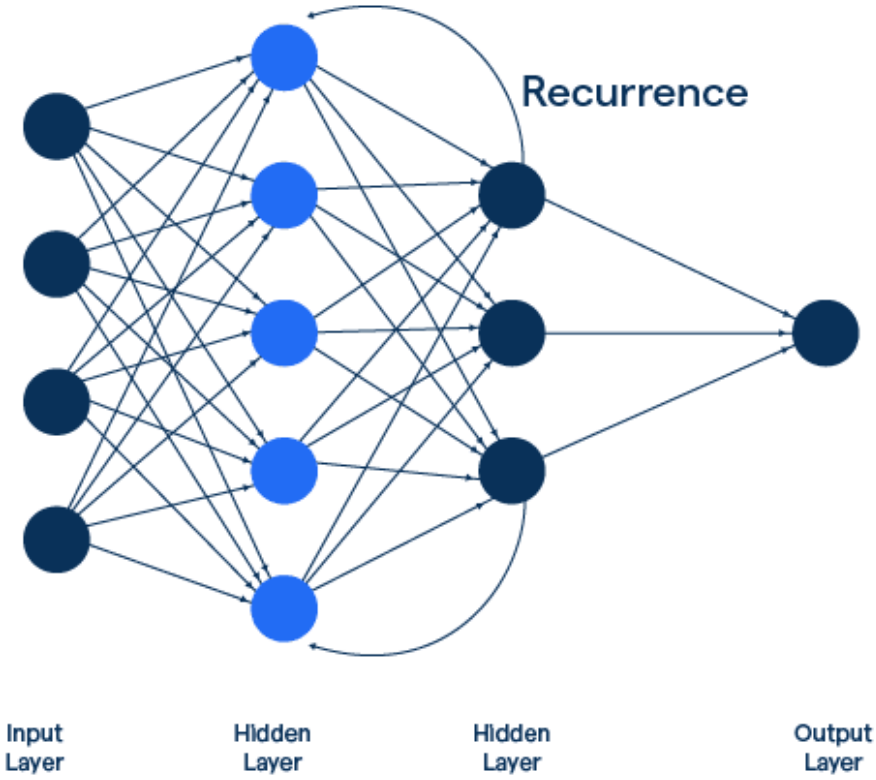

Fig. 2.3: The architecture shows the main components of a standard RNN: the input layer receives sequential data, followed by one or more recurrent hidden layers where information is processed temporally while maintaining a hidden state that captures the temporal dependencies between the elements of the sequence. Recurrence is represented by the cyclic connections that allow the network to propagate information across successive time steps. The output layer generates predictions based on both the current input and the accumulated state of the network. This structure makes RNNs particularly suitable for processing sequential data such as text, time series, or signals.

The transition from traditional shallow networks to deep architectures was made possible by the confluence of three crucial technological advances. First, the availability of large-scale datasets, exemplified by ImageNet [11], provided the volume of data necessary to train complex models without severe overfitting problems. Second, the evolution of computational hardware, particularly the adoption of GPUs for parallel matrix operations [57] and later specialized tensor processing units (TPUs) [32], has made training deep networks computationally feasible. Third, algorithmic innovations such as dropout regularization [71], batch normal-

ization [28], and advanced optimizers such as Adam [35] have effectively addressed historical challenges, including vanishing gradients and training instability.
Contemporary DL architectures demonstrate superior performance in complex scenarios requiring automatic feature extraction and learning hierarchical representations, effectively replacing traditional shallow networks in domains involving high-dimensional data. However, this computational power introduces significant challenges that deserve careful consideration. The limited interpretability inherent in deep models creates "black box" systems whose decision-making processes remain opaque [61], raising concerns in safety-critical applications. Furthermore, the significant computational requirements for training and inference raise questions of environmental sustainability [72] and equitable access to advanced AI technologies. The figure 2.4 clearly highlights the structural difference between the two approaches, showing how DNNs enable more hierarchical and complex feature learning.

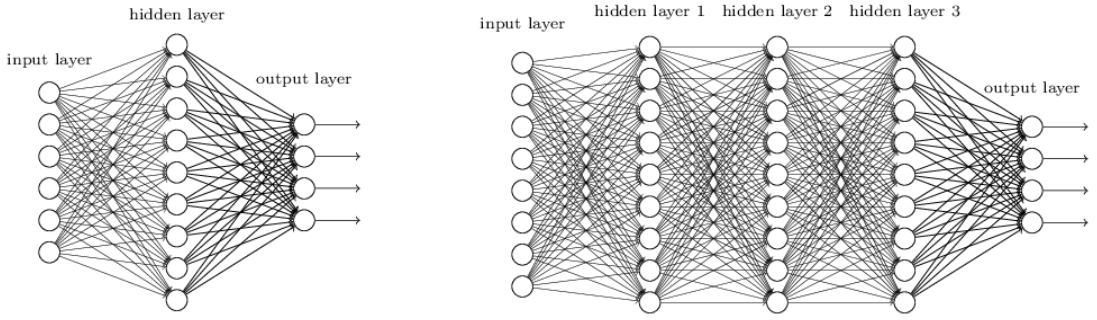

Fig. 2.4: Comparison of neural architectures. **Left**: Traditional ANN with a single hidden layer. **Right**: Deep Neural Network (DNN) with multiple hidden layers that enable hierarchical feature learning.

2.1.1 The Training Process: Optimization and Loss Functions

The training process of an ML model can be formalized within the framework of **Empirical Risk Minimization** (ERM) [79]. The core idea is to find the model parameters θ that minimize the loss not on the entire data distribution (which is intractable), but on a finite training dataset. This finite approximation is called the **empirical risk**. The goal of ERM is therefore stated as:

$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x_i;\theta), y_i)$ where $f(x_i;\theta)$ is the model's prediction for the input x_i, y_i is the ground truth label and $\mathcal{L}$ is the loss function. However, it's crucial to remember that the true objective of ML is to minimize the **expected risk**, or generalization error, that is, the loss on unseen data: ERM is the practical strategy we use to approximate this ideal goal. The discrepancy between minimizing the empirical risk and the expected risk is a fundamental source of overfitting, which is why techniques like regularization, validation, and cross-validation are essential components of the modern ML pipeline.

The Role of the Loss Function

The choice of the loss function $\mathcal{L}$ is as critical as the choice of the model architecture, as it directly encodes the learning objective. Different tasks require fundamentally different loss functions:

- **Classification Task**: Cross-Entropy loss [45] is the "de facto standard" for multi-class classification. It measures the dissimilarity between the true label distribution y and the predicted distribution p. For a single training example, it is defined as:

$$\mathcal{L}_{CE}^{(i)} = -\sum_{c=1}^{M} y_{i,c} \log(p_{i,c}) \tag{2.2}$$

  where M is the number of classes, $y_{i,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for the i-th example and $p_{i,c}$ is the predicted probability (from the softmax function) for the i-th example and class c.

  For a batch of B examples, the loss is averaged over all examples:

$$\mathcal{L}_{CE} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{c=1}^{M} y_{i,c} \log(p_{i,c}) \tag{2.3}$$

  Minimizing this loss is equivalent to maximizing the log-likelihood of the correct labels, effectively pushing the model to assign high probability to the correct class.

- **Regression Task**: Mean Squared Error Loss (MSE) is commonly used: $\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$.
  where:
  - N: Number of examples.
  - y_i: The ground truth for the example, i.
  - $\hat{y}_i$: The model's prediction for the example, i.
- **Ranking and Retrieval Task**: More complex functions such as Triplet Loss [64] or, central to this thesis, Contrastive Losses, which will be explored in more detail in 2.3, are essential. These functions do not attempt to classify a sample into a predefined category, but rather to learn a latent space in which similar samples are close and dissimilar samples are distant.

For a comprehensive comparative analysis of loss functions in ML, we refer the reader to the survey by Qi Wang et al. [82].

Gradient-Based Optimization and Advanced Optimizers

Once the learning objective is defined by the loss function, the next step is to find the parameters θ that minimize it. Most algorithms for this are based on the principle of **Gradient Descent**. The central idea is to iteratively update the θ parameters by moving in the direction opposite to the gradient of the cost function $\nabla_\theta \mathcal{L}(\theta)$, since the gradient points towards the direction of maximum increase of the function. The single-step update is formalized as: $\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta \mathcal{L}(\theta_t)$ where:

- θ_t represents the parameters at time step t.
- η is the learning rate, a critical hyperparameter that determines the size of the update step.
- $\nabla_\theta \mathcal{L}(\theta_t)$ is the gradient of the cost function with respect to the parameters.

A naive version of this algorithm, known as **Batch Gradient Descent**, computes the gradient using the entire training dataset. This approach, while theoretically

sound, is computationally prohibitive for large datasets and is not suitable for online training.

To overcome the limitations of Batch Gradient Descent, the **Stochastic Gradient Descent** (SGD) algorithm has become the "de facto standard". Instead of calculating the gradient on the entire dataset, SGD estimates the true gradient using a small subset of data, called a mini-batch B, randomly drawn from the training dataset D_{train} at each iteration. The gradient estimate is therefore:

$$\nabla_\theta \mathcal{L}(\theta_t) \approx \frac{1}{|B|} \sum_{i \in B} \nabla_\theta \mathcal{L}_i(\theta_t) \tag{2.4}$$

where $\mathcal{L}_i$ is the loss for the i-th example of the mini-batch.

This estimate introduces some statistical noise, which has been shown to be beneficial in that it helps the model escape shallow local minima of the loss function, potentially leading to better generalizable solutions.

However, SGD suffers from some issues:

- **Slow convergence**: the direction of the stochastic gradient may be suboptimal.
- **Sensitivity to the choice of learning rate**: a learning rate that is too high causes oscillations and instability, while a learning rate that is too low slows convergence excessively.
- **Problems with cost functions** with different curvatures (e.g., ravines or saddle points).

To address these challenges, several advanced, so-called **adaptive**, optimizers have been developed, which extend the basic concept of SGD by introducing mechanisms to dynamically adapt the learning rate for each parameter. These incorporate the concept of **momentum** and **learning rate**, some of the most famous:

- **SGD with Momentum** [54]: introduces a **"velocity" term** v_t that accumulates a moving average of past gradients, adding inertia to the direction of descent. This helps smooth oscillations and accelerate convergence in the relevant directions. The term v_t is calculated as:

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta \mathcal{L}(\theta_t) \tag{2.5}$$

where γ is the momentum coefficient.
The parameters are then updated:

$$\theta_{t+1} = \theta_t - v_t \tag{2.6}$$

- **RMSprop** [23]: adjusts the learning rate for each parameter by dividing the current gradient by the square root of an exponential moving average of the squares of past gradients ($E[g^2]_t$), with $g_t = \nabla_\theta \mathcal{L}_t(\theta_{t-1})$. This normalizes the update, reducing the amplitude for parameters with large gradients and increasing it for those with small gradients.

- **Adam** (Adaptive Moment Estimation) [35]: combines the advantages of RMSprop and moment. It is perhaps the most popular optimizer in modern practice due to its robustness and consistent performance. Adam calculates two moments:

  – *First moment* (average of gradients, similar to momentum):

  $$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.7}$$

  – *Second moment* (average of squared gradients, similar to RMSprop):

  $$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.8}$$

  These moments are then bias-corrected (since they are initialized to zero) and used for a parameter update for which each dimension has its own dynamic learning rate:

  $$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.9}$$

  where $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected versions of m_t and v_t and ϵ is a regularizing term to avoid division by zero.

The choice of optimizer (SGD, SGD with momentum, Adam, etc.) and the configuration of its hyperparameters (learning rate, β_1, β_2) are critical decisions that profoundly influence the stability, convergence speed, and final performance of the model.

The interaction between optimizer and loss function is crucial: an efficient optimizer is necessary to effectively minimize a well-defined loss function, especially in high-dimensional spaces and with complex models such as DNNs. Understanding this binomial is preparatory to the analysis of specific architectures, such as Transformers, and learning paradigms such as the contrastive one on which the CLIP model is based.

2.2 Transformer Architectures for Vision and Text

Transformer [80], which revolutionized DL thanks to its attention mechanism and initially developed for language but later successfully extended to computer vision (Vision Transformer), is an architecture based exclusively on **self-attention mechanisms**, overcoming the limitations of RNN and CNN, as opposed to which, it can process all tokens in parallel (RNNs require sequential steps) and captures direct dependencies between distant tokens thanks to self-attention.
The input is a sequence of vectors $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{n \times d}$ where each vector represents a token (in the case of text) or an image patch (in the case of ViT). The main element of this architecture is the **self-attention mechanism** defined as:

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

where:

- $Q = XW^Q, K = XW^K, V = XW^V$ are the **query**, **key** and **value** matrices obtained through linear projections (learned parameters).
- d_k is the size of the key vectors.
- $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k}$ are learned weights.

To enhance the model's capacity to focus on different contextual relationships, the Transformer employs **multi-head self-attention**. This mechanism executes the self-attention operation in parallel across h distinct heads, each with independent learned projections:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.11}$$

where:

- $\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V)$
- $W^O \in \mathbb{R}^{hd \times d_k}$ is a learned weight matrix.

In its original architecture, as shown in Figure 2.2, the Transformer is composed of two main components:

- **Encoder**: Processes the entire input sequence in parallel, producing contextualized representations.
- **Decoder**: Generates the output autoregressively, leveraging information learned from the encoder through a cross-attention mechanism.

Each layer of the encoder is composed of two main sublayers:

- **Multi-Head Self-Attention (MHSA)**: Enables each token in the sequence to attend to all other tokens, dynamically computing weighted combinations based on relevance.
- **Feed-Forward Network**: A position-wise fully connected feed-forward network applied independently to each token.

Each of these sublayers is wrapped with a **residual connection** followed by **layer normalization**, applied as follows:

$$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

Each layer of the *decoder* is composed of three main sublayers:

- **Masked Multi-Head Self-Attention (Masked MHSA)**: Similar to the encoder's self-attention, but with a causal mask that prevents attending to future positions in the output sequence. This is essential during autoregressive generation to ensure that the prediction for a token only depends on previous tokens.

- **Cross-Attention (Encoder-Decoder Attention)**: This sublayer allows the decoder to attend to the output of the encoder, effectively integrating information from the input sequence. It uses the encoder outputs as *keys* and *values*, while the decoder's hidden states serve as *queries*.
- **Feed-Forward Network**: Identical in structure to the one used in the encoder, it is applied independently to each position.

As in the encoder, each sublayer is followed by a **residual connection** and a **layer normalization**, according to the general pattern:

$$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

This design ensures stability and deep signal propagation during training.
The output of a decoder layer can be expressed as the following sequence of operations:

$$\begin{aligned} h_1 &= \text{LayerNorm}\left(x + \text{MaskedMHSA}(x)\right) \\ h_2 &= \text{LayerNorm}\left(h_1 + \text{CrossAttention}(h_1, z)\right) \\ \text{Output} &= \text{LayerNorm}\left(h_2 + \text{FFN}(h_2)\right) \end{aligned}$$

One of the most powerful features of the Transformer architecture is its intrinsic modularity. Unlike sequential models such as RNNs, which struggle to scale in depth due to vanishing gradients, Transformers allow stacking of multiple encoder and decoder layers, enabling the model to progressively refine its internal representations.
In practice, a Transformer typically consists of a stack of N encoder layers and M decoder layers. Each encoder layer receives the output of the previous one and applies a series of sublayers with residual connections and layer normalization. Formally, a single encoder layer can be described as:

$$Z^{(\ell)} = \text{LayerNorm}\left(X^{(\ell)} + \text{MultiHead}(X^{(\ell)})\right) \tag{2.12}$$

$$X^{(\ell+1)} = \text{LayerNorm}\left(Z^{(\ell)} + \text{FFN}(Z^{(\ell)})\right) \tag{2.13}$$

where:

- $X^{(\ell)}$ is the input to the ℓ-th layer.
- `MultiHead` is the MHSA mechanism.
- `FFN` is a position-wise feed-forward network, typically a two-layer MLP with ReLU or GELU.
- `LayerNorm` stabilizes training and ensures smooth gradient flow.

The layers do not operate in isolation; instead, they form a deep hierarchy: early layers capture local relationships (e.g., between adjacent words or image patches), intermediate layers aggregate mid-range patterns and deeper layers model global dependencies across the entire sequence.
Originally $N = 6$ and $M = 6$, but this modularity has allowed architectures like GPT or CLIP itself to scale to hundreds of layers while maintaining training stability.
While the original Transformer includes both an encoder and a decoder (as depicted in Figure 2.2), many modern architectures such as CLIP employ only the encoder component, leveraging its ability to produce rich contextual representations for both visual and textual inputs.

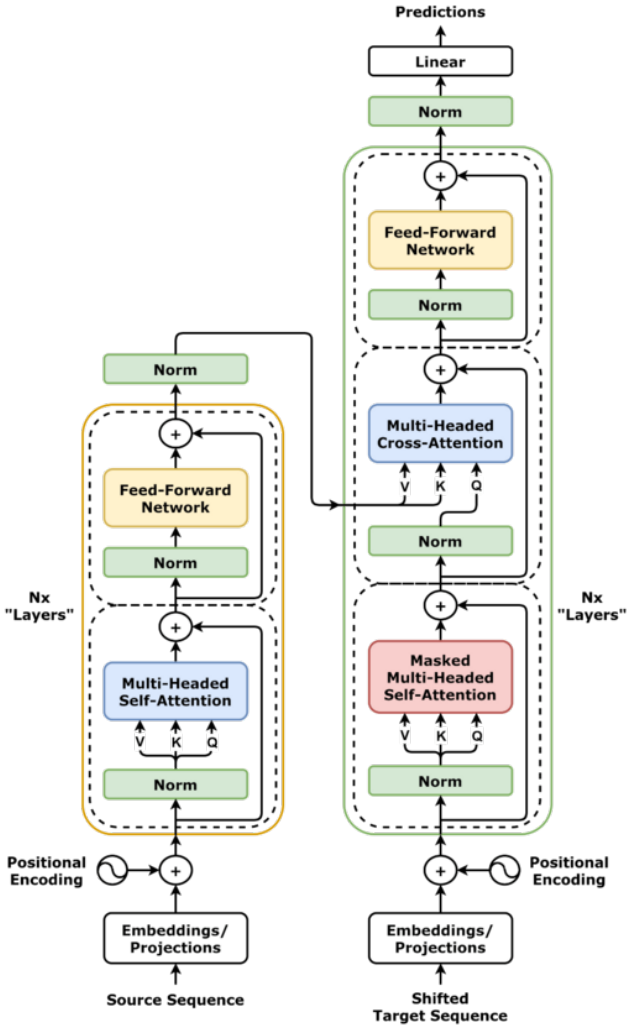

Fig. 2.5: Overview of the Transformer architecture.*Source: Wikipedia*

2.2.1 Vision Transformers

In recent years, the **Vision Transformer** (ViT) [13] has demonstrated that an architecture originally conceived for natural language can achieve competitive and sometimes superior performance to CNNs in visual classification and recognition tasks. The underlying idea is to treat an image as a sequence of elements (patches) analogous to the sequence of tokens in a text, thus enabling the direct application of a *Transformer Encoder*.

Given an input image $x \in \mathbb{R}^{H \times W \times 3}$, it is divided into N non-overlapping *patches* of size $P \times P$. Each patch $x_{i,j}$ is then "linearized" into a vector of size $(P \times P \times 3)$

and projected into a vector space of size d using a linear projection:

$$z_p = \text{Flatten}(x_{i,j})W^P \tag{2.14}$$

where $W^P \in \mathbb{R}^{(P\times P\times 3)\times d}$ is the projection matrix of the patches.

A learnable `[CLS]` token embedding $x_{[CLS]}$ is prepended to the sequence, and a positional embedding $E_{\text{pos}} \in \mathbb{R}^{(N+1)\times d}$ is added to preserve spatial information.

$$Z^{(0)} = [x_{\texttt{[CLS]}}; z_1^{(0)}, z_2^{(0)}, \ldots, z_N^{(0)}] + E_{pos}$$

The resulting sequence is then fed into a standard Transformer encoder, and the final representation of the `[CLS]` token is used as the global descriptor of the image.

With sufficiently large-scale pre-training and suitable regularization, ViT can match or surpass CNNs on many vision benchmarks, benefiting from scalability and the ability to model long-range dependencies from the earliest layers. Figure 2.2.1 shows an overview of the architecture of a ViT.

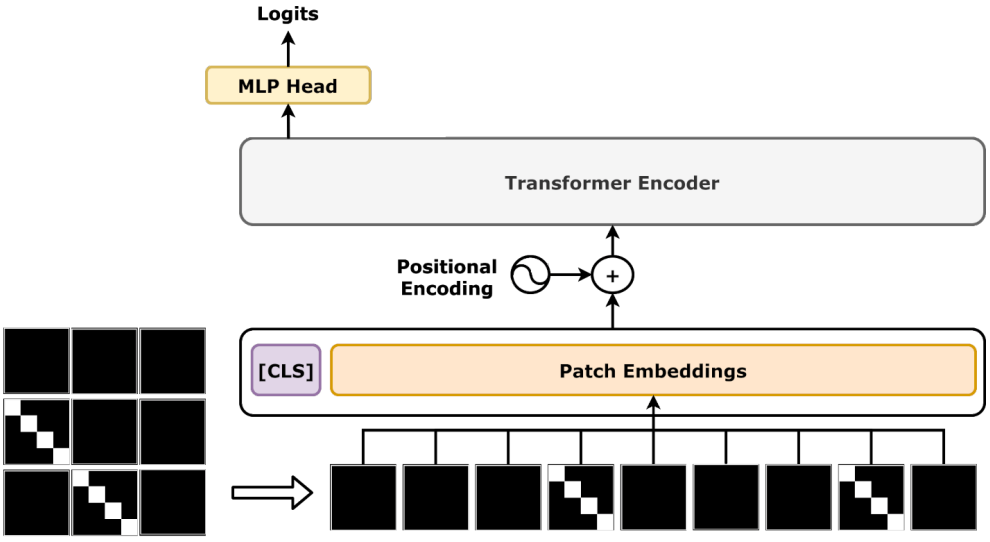

Fig. 2.6: In the ViT architecture, an input image is first divided into fixed-size patches. These patches are then linearly projected into embeddings, which are fed into a standard Transformer encoder. *Source: Adapted from Wikipedia.*

2.3 Contrastive Learning

Contrastive learning is one of the most significant methodologies in ML, particularly in the field of **representational learning**. It is based on the idea of

learning effective representations of data through a process of comparing (contrasting) positive and negative examples, to maximize the similarity between representations of "similar" examples and minimize that between "dissimilar" examples. In practical terms, each example is associated with at least one positive pair (another semantically similar example) and a certain number of negative pairs. The model is trained to maximize the similarity between the embeddings of the positive pairs and minimize it between the negative pairs. This approach has proven effective in a wide range of tasks, including classification, retrieval, clustering, and multimodal learning.

2.3.1 Supervised vs Self-Supervised Contrastive Learning

Contrastive Learning can be mainly classified into two categories: Supervised and Self-Supervised.

- **Supervised contrastive learning**: In this context, the model leverages explicit class labels during training. Positive pairs consist of examples belonging to the same class, while negative pairs consist of examples belonging to different classes. This approach allows the supervised signal to be used to improve the quality of learned representations, facilitating the semantic aggregation of similar examples. A relevant example is the supervised contrastive loss [33], which extends the self-supervised methodology by including such class information.

- **Self-supervised contrastive learning**: In the absence of labels, this approach automatically generates positive examples through data augmentation techniques applied to the same original data (e.g., rotations, cropping, color variations on images). In this way, the model implicitly learns a representation robust to these transformations. Negative pairs, on the other hand, are typically obtained from other data within the batch. This paradigm has established itself with methods such as SimCLR [7] and MoCo [21], which have demonstrated excellent performance even in the absence of explicit supervision.

2.3.2 Contrastive Loss Functions

Contrastive learning is therefore based on the concept of learning by discriminating between pairs of data. More specifically, given a reference data point, called **anchor**, the model learns to distinguish between:

- A positive example, closely related or semantically similar to the anchor.
- And various negative examples, which are distinct and semantically different from the anchor.

The theoretical roots of contrastive learning date back to concepts of metric learning, where the objective is to learn a representation space in which semantic similarity is reflected in the distances between samples. In the literature, the term contrastive loss is often used broadly to refer to any loss function that increases the similarity between positive pairs and decreases it between negative pairs; however, each specific formulation has its own distinct name and properties. One of the first modern formulations is the Contrastive Loss "Margin-based" [19].
Defined on pairs of samples (x_i, x_j), minimizes the distance between similar (*positive*) elements and maximizes that between dissimilar (*negative*) elements, with a margin m:

$$\mathcal{L}_{\text{contrastive}}(x_i, x_j, y_{ij}) = \begin{cases} \frac{1}{2}\|f(x_i) - f(x_j)\|^2, & \text{if } y_{ij} = 1, \\ \frac{1}{2}\left[\max\left(0, m - \|f(x_i) - f(x_j)\|\right)\right]^2, & \text{if } y_{ij} = 0, \end{cases} \tag{2.15}$$

where:

- $f(x_i)$ is the embedding function of the sample x_i.
- $y_{ij} = 1$ indicates a positive pair, $y_{ij} = 0$ a negative.
- $m > 0$ is a hyperparameter that defines the minimum margin for negatives.

This loss has been fundamental to metric learning, but requires a careful balance between positive and negative pairs in training.
Another very well-known loss function is the **Triplet Loss** [64], which was introduced for facial recognition and is based on the idea that the distance between

an anchor and a positive example is less than the distance between the anchor and a negative example by at least a margin:

$$\mathcal{L}_{\text{triplet}} = \max\left(0,\ \|f(x_a) - f(x_p)\|^2 - \|f(x_a) - f(x_n)\|^2 + \alpha\right), \tag{2.16}$$

where:

- x_a is the anchor;
- x_p is the positive;
- x_n is the negative;
- α is a margin;
- $f(\cdot)$ is the embedding function.

Subsequently, one of the best well known loss functions is the InfoNCE (Information Noise Contrastive Estimation) [78], which is based on Noise Contrastive Estimation [18]. InfoNCE is a contrastive loss function used for representation learning and mutual information estimation. While NCE was developed for density estimation, InfoNCE aims to learn useful representations by maximizing mutual information between correlated variables. It is important in self-supervised learning frameworks such as contrastive predictive coding and image-based contrastive methods. Unlike NCE, which requires explicit noise sampling, InfoNCE uses in-batch negatives and can be formally defined as:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{\exp\left(s(x, y)/\tau\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(s(x, y')/\tau\right)}\right] \tag{2.17}$$

where the expectation is calculated on the distribution of positive (x, y) and negative (x, y') pairs. In practice, this expectation is estimated by averaging the samples of a batch, yielding the form:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(s(x_i, y_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(s(x_i, y_j)/\tau\right)}, \tag{2.18}$$

where:

- $s(x, y)$ is a similarity function;

- τ is a temperature hyperparameter;
- (x_i, y_i) is the positive torque for the anchorx_i;
- $\{y_j\}_{j \neq i}$ are the in-batch negatives.

InfoNCE is widely used in self-supervised methods such as SimCLR [7] and MoCo [21], where the data structure allows for defining positive pairs via transformations, called augmentations, or temporal/spatial co-occurrences. However, its formulation is asymmetric, as it only evaluates the probability that the anchor x_i retrieves the positive y_i and not vice versa.
Another interesting loss function in the context of contrastive learning is the **Unified Contrastive Learning** (UniCL) loss [85], which extends the InfoNCE formulation to naturally handle multiple positive pairs per sample. This approach will be described in detail in Section 3.2.4.
In conclusion, adopting a contrastive approach proves crucial for the text-to-image retrieval task, as it allows for the learning of highly discriminative and semantically coherent multimodal representations. Through the joint optimization of corresponding and non-corresponding pairs, the model not only memorizes direct associations between text and image, but also develops a shared representation space in which related concepts are close and unrelated ones are distant. This ability to organize information in a structured manner is the basis of effective representation learning, capable of generalizing to unseen data and maintaining robustness even in complex scenarios or with high semantic variability. In the proposed application context, this results in greater accuracy and reliability in retrieving relevant images, making contrastive learning a fundamental element of the learning pipeline.
A visual representation of the concept of contrastive learning is provided in Figure 2.7.

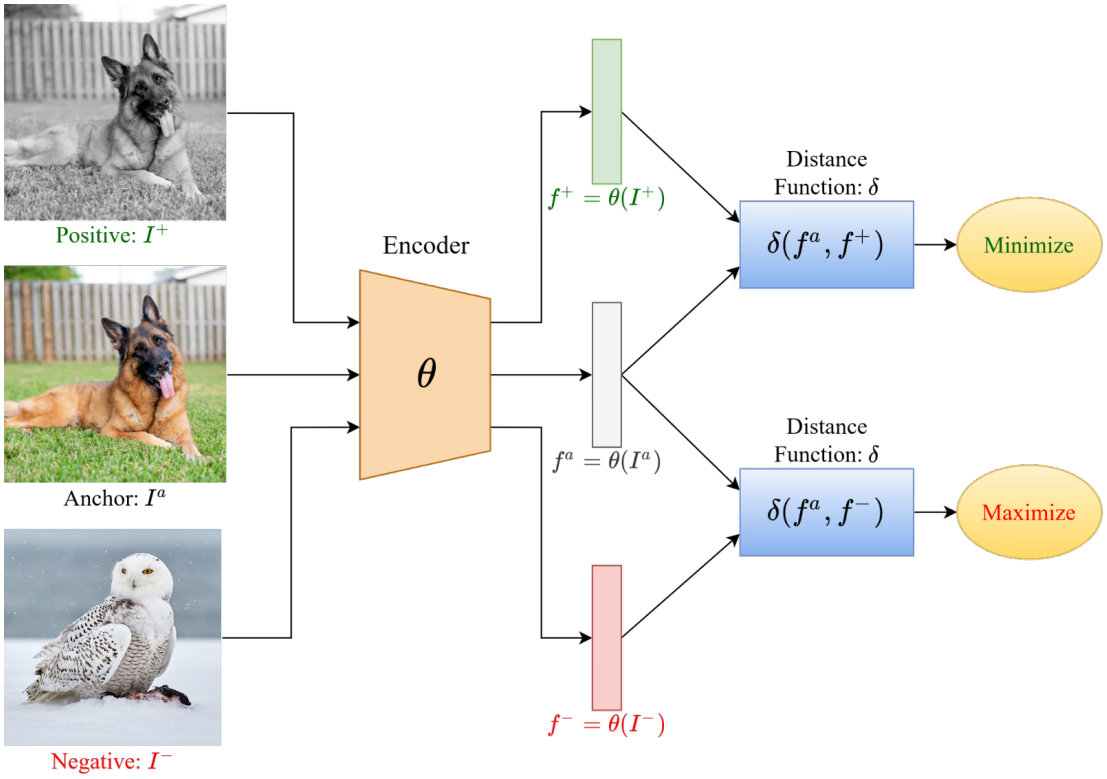

Fig. 2.7: In contrastive learning, the goal is to learn effective representations by maximizing the similarity between positive instances and minimizing it between negative ones. In the illustrated scheme, I^+ represents a positive instance (e.g., an augmentation of the same image), while I^- is a negative instance (a different sample). The θ function transforms an image I into its latent representation f. The similarity between representations is measured by a δ. The goal is therefore to minimize $\delta(f^a, f^+)$, where $f^a = \theta(I^a)$ and $f^+ = \delta(I^+)$, to bring positive instances closer together and to maximize $\delta(f^a, f^-)$, with $f^- = \theta(I^-)$, to remove negative instances.

2.4 Vision-Language Models and Cross-Modal Retrieval

The growing availability of multimodal data, such as images accompanied by textual descriptions, has spurred strong interest in developing systems capable of simultaneously processing and understanding information from different domains. In this context, **VLM** represents a class of ML models designed to acquire a joint understanding of visual and linguistic content, enabling them to tackle a wide range of tasks, including image captioning, visual question answering (VQA), vi-

sual grounding, and, in particular, cross-modal retrieval.
The core of a VLM is the concept of a **shared representation space** (shared embedding space), in which data from different modalities, typically images and text, are projected into a shared vector space. In this space, semantically similar elements are close to each other, while conceptually distant elements are separated. This property allows for the direct comparison of representations of heterogeneous modalities using similarity metrics, such as cosine similarity, enabling the automatic association between a text and the image describing its content, or vice versa. Cross-modal retrieval falls within this paradigm as the task of retrieving relevant elements in one modality from a query formulated in another. Two common configurations are:

- **Text-to-image retrieval**, in which a textual description is used to identify the most relevant images with respect to the expressed content.
- **Image-to-text retrieval**, in which an image constitutes the query to find the most consistent captions or text documents.

To enable this matching, VLMs learn a semantic alignment between visual and linguistic representations. This alignment is often achieved using contrastive learning strategies, which, as explained in detail in the 2.3 section, aim to bring corresponding image-text pairs closer together in the embedding space and separate unrelated ones. In this way, the model develops a coherent and generalizable multimodal representation, capable of supporting both supervised tasks and zero-shot scenarios.
The result is a versatile conceptual framework that combines vision and language in a single semantic space and can be applied to heterogeneous contexts, from content search in large multimedia archives to data classification and filtering, to monitoring and surveillance in scientific and environmental applications. It is important to note that, for simplicity, images and text are considered as examples, but there is nothing to prevent systems from being built based on other signals, such as voice.
This conceptual framework and its main multimodal capabilities are illustrated in Figure 2.8.

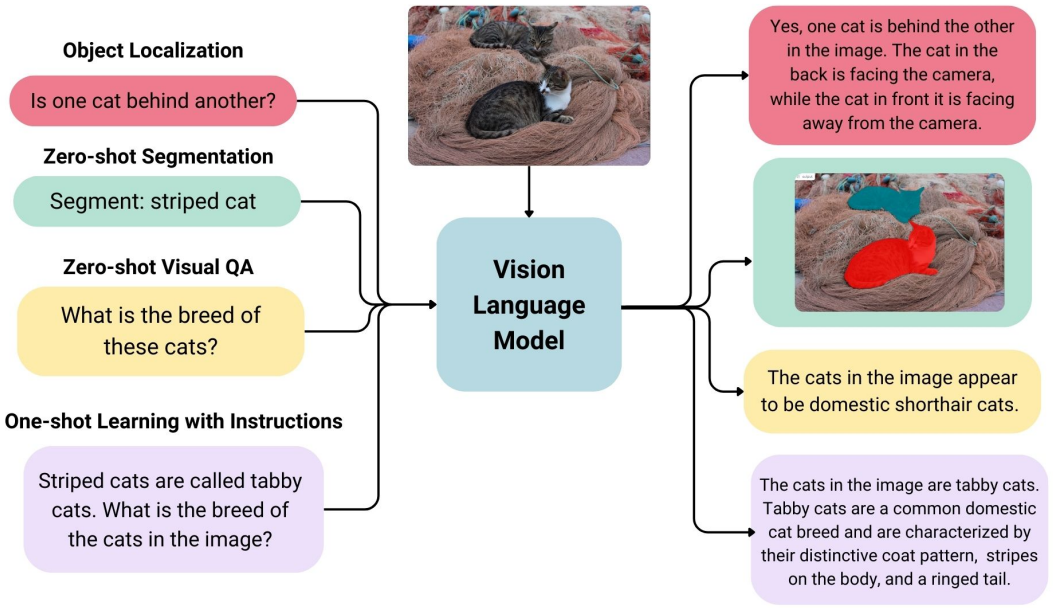

Fig. 2.8: The Vision Language Model demonstrates several multimodal capabilities: (1) **Object Localization**: identify the relative position of objects (e.g., "One cat is behind the other"). (2) **Zero-shot Segmentation**: isolates objects without specific training (e.g., "striped cat" segmentation). (3) **Zero-shot Visual QA**: answers questions about the picture without previous examples (e.g., "What breed of cat is this?"). (4) **One-shot Learning**: learns from contextual instructions (e.g., "Tabby cats are tabbies" → recognizes the breed). (5) **Descriptive Generation**: provides detailed descriptions (locations, tabby breed, coat pattern).

2.5 Evolution of Vision-Language Models

The development of VLMs has been characterized by a progressive shift in both architectural choices and training paradigms, driven by the need to achieve more effective alignment between visual and textual modalities. Early approaches, such as Deep Visual-Semantic Embedding [59] and VSE++ [15], represented the first attempts to map images and text into a joint embedding space. These systems typically use CNNs to extract visual features and RNNs for text encoding, optimizing their parameters through ranking losses or triplet losses. While effective on small- and medium-scale datasets, these models were limited in their ability to generalize beyond the training distribution and struggled to capture detailed semantic relationships between modalities. A major breakthrough was introduced

with CLIP (Contrastive Language–Image Pretraining) by OpenAI [56], which marked a paradigm shift in VLM design. CLIP employs separate Transformer-based encoders for vision and text, trained on an unprecedented scale of 400 million image-text pairs collected from the web. Its large-scale contrastive learning objective allowed the model to acquire highly transferable multimodal representations, demonstrating remarkable zero-shot performance in both classification and retrieval tasks without further optimization. The scalability of this approach highlighted the potential of leveraging noisy web-scale datasets to produce robust and versatile multimodal embeddings. Following CLIP, several models have attempted to expand the capabilities of VLMs by introducing modular architectures and more sophisticated integration strategies. BLIP-2 [37], for example, decouples visual and language pretraining by connecting a frozen image encoder with a large frozen language model via a lightweight Querying Transformer. This design enables more efficient image-to-text and text-to-image alignment while maintaining the zero-shot capabilities of large-scale pretrained components. Similarly, ClipCap [47] extends CLIP's visual encoder with a mapping network that produces a prefix vector for a language model, thus enabling caption generation by reusing the pretrained multimodal space. These examples illustrate the flexibility of contrastively trained embeddings as a basis for a wide range of multimodal tasks. In addition to improving retrieval and captioning performance, recent research has increasingly focused on domain adaptation techniques for VLMs, aiming to specialize pre-trained models for specific application scenarios without incurring the costs of extensive retraining. These strategies, which will be described in detail in the next section, include efficient fine-tuning methods that preserve most of the pre-trained knowledge while adapting to domain-specific data. Despite these advances, the application of VLM to marine wildlife monitoring remains largely unexplored. In this work, we address this gap by developing a text-to-image retrieval system specifically designed for sea turtles, leveraging contrastive learning and efficient fine-tuning strategies, which will be described in detail in subsequent chapters.

2.6 Pretraining and Transfer Learning in VLMs

The success of modern VLM is largely due to their pretraining on massive multimodal datasets, often comprising hundreds of millions of image–text pairs. This large-scale training phase enables the model to acquire general-purpose multimodal representations that capture broad visual concepts and linguistic patterns, supporting a variety of downstream tasks, including zero-shot inference. However, in many practical scenarios, the target domain differs significantly from the data distribution used during pretraining. For example, datasets for marine wildlife monitoring are considerably smaller and more specialized than the large-scale collections used to train general-purpose models like CLIP. To address this gap, researchers use **transfer learning**, a paradigm in which a pre-trained model is adapted to a new domain or task by leveraging the knowledge gained during pre-training. Transfer learning can be achieved through several strategies:

- **Feature extraction**, in which the pre-trained model is kept frozen and used exclusively as a feature generator, with a separate lightweight model trained on its outputs for the new task.
- **Fine-tuning**, in which some or all of the pre-trained parameters are updated to better fit the new domain.

When adopting fine-tuning, there are several approaches:

- **Complete fine-tuning**, in which all model parameters are updated, and this can produce a strong fit, but requires significant computational resources and can cause overfitting when the new dataset is small.
- **Partial fine-tuning**, in which only selected components, such as the text encoder or projection layers, are updated, preserving most of the pre-trained weights.
- **Parametrically efficient fine-tuning** (PEFT), which adds a small set of trainable parameters to the model while keeping the original weights largely frozen, significantly reducing computational and memory footprint.

An example of partial fine-tuning is **LiT** (Locked Image Tuning) [88], which adapts the model by optimizing only the text encoder while keeping the vision encoder fixed, reducing both training costs and the risk of overfitting. Among PEFT methods, **LoRA** [25], which will be explained in detail in Section 3.3, introduces small rank decomposition matrices that can be trained at specific levels (e.g., attention projections), enabling efficient domain adaptation with minimal additional parameters. By clearly separating the pre-training, transfer learning, and fine-tuning phases, this framework allows VLMs to be effectively deployed in specialized domains, such as text-to-image retrieval for sea turtles, while maintaining the efficiency and generalization capabilities of pre-trained models at a large scale.

2.7 Retrieval Systems: Architectures and Applications

Retrieval systems aim to efficiently identify and return the most relevant items from a large database in response to a user query. In the context of VLMs, retrieval is often multimodal, meaning that the query and target items belong to different modalities, such as retrieving images with a textual description (text-to-image retrieval) or retrieving captions for a given image (image-to-text retrieval) [2]. At the heart of retrieval is the concept of a similarity function that operates in a shared embedding space, in which semantically related items are mapped close to each other. This design enables direct comparison between queries and candidates using distance metrics such as cosine similarity or Euclidean distance [44].

2.7.1 Architectural Paradigms for Cross-Modal Retrieval

Several architectural choices have been explored for implementing retrieval systems, each with specific tradeoffs between accuracy and efficiency:

- **Dual encoder architecture**: In this paradigm, as shown in the figure 2.9, the query and the candidate items are processed independently through modality-specific encoders (e.g., a vision encoder for images and a language encoder for text). The outputs are projected into a shared embedding space and compared via a similarity function [56, 88]. Dual encoders are highly

scalable because embeddings for all items in the database can be precomputed and indexed, allowing retrieval through fast nearest-neighbor search. However, their independent encoding may limit fine-grained interactions between modalities.

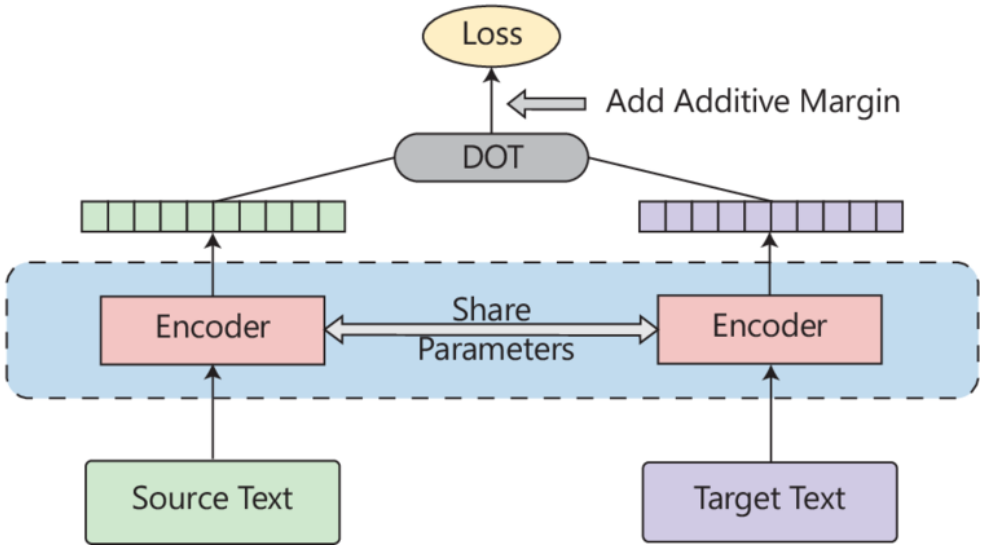

Fig. 2.9: Dual-encoder architecture with shared parameters for cross-modal alignment. Two independent encoders process the Source Text and Target Text, respectively, generating optimized vector representations through a dot product-based loss function with additive margin. Sharing parameters between the encoders promotes learning of consistent features across the two domains.

- **Cross encoder architecture**: Unlike dual encoders, cross encoders, as shown in the figure 2.10, jointly process the query and candidate item through a single model, enabling richer interaction between modalities [36, 42]. While this often leads to higher retrieval accuracy, the computational cost is significantly higher, since the model must process each query–candidate pair at runtime. This makes cross-encoders less suitable for large-scale retrieval tasks.

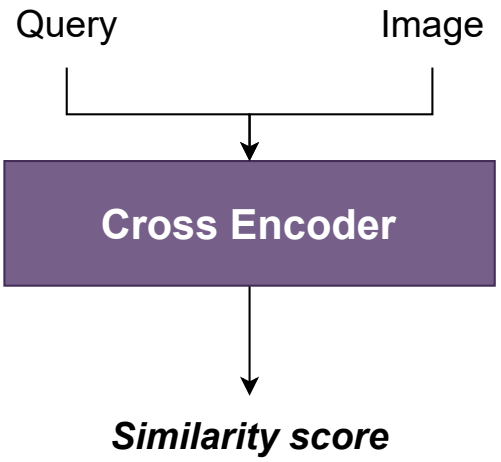

Fig. 2.10: **Cross-Encoder** architecture: computes a *similarity score* by jointly processing the query and the image, enabling direct interactions between modalities.

- **Late fusion and hybrid approaches**: These strategies combine the efficiency of dual encoders with the precision of cross encoders. A common approach is to use a dual encoder for initial retrieval (candidate selection) and then re-rank the top results using a cross encoder [27, 50]. This two-stage pipeline balances retrieval speed with improved accuracy.

Once embeddings are obtained, an efficient indexing structure is required for large-scale retrieval. Traditional methods include tree-based data structures such as KD-trees and Ball-trees [17], which are effective for lower-dimensional spaces. For high-dimensional embeddings typical of VLMs, approximate nearest neighbor methods such as FAISS [31] or Annoy are commonly employed. These methods enable sub-linear time search by sacrificing exactness for speed, which is crucial for real-time applications.

2.7.2 Applications of Retrieval Systems

Retrieval systems are applied in a wide range of domains, including:

- **Content-Based Image Retrieval (CBIR)**: Used in large multimedia databases, digital asset management, and e-commerce platforms to search for visually similar products [10].
- **Information Retrieval in Multimodal Search Engines**: Systems like Google Images or Bing Visual Search integrate cross-modal retrieval to allow both text-to-image and image-to-text queries.
- **Surveillance and Environmental Monitoring**: Retrieval of relevant footage or imagery from large-scale sensor networks, including applications in wildlife monitoring and conservation [4, 52].

By combining robust visual–textual representations, efficient indexing strategies, and scalable architectures, modern retrieval systems are able to operate across diverse domains and data modalities. In the context of this work, such systems provide the foundation for text-to-image retrieval tailored to marine turtles, enabling efficient navigation and querying of domain-specific image repositories.

2.8 Text-to-Image Retrieval: Task Definition

The task of **text-to-image retrieval** consists, as shown in the figure 2.11, in retrieving the most relevant images from a dataset based on a textual query that describes their content. Formally, let $\mathcal{I} = \{i_1, i_2, ..., i_N\}$ be a set of images and $\mathcal{T} = \{t_1, t_2, ..., t_N\}$ the corresponding set of text captions, where each pair (i_k, t_k) describes semantically related content. Given a query text $t_q \in \mathcal{T}$, the objective is to retrieve the image $i^* \in \mathcal{I}$ (or a ranked list of images) that best matches the semantic content of t_q.

Modern approaches to this problem leverage **dual-encoder architectures**, explained in Section 2.7.1, where a vision encoder f_θ maps images into an embedding space $\mathbb{R}^d$ and a text encoder g_ϕ does the same for text. The aim is to learn these functions so that the embeddings of matching image-text pairs are close in the shared space, typically using a **contrastive loss function**, which is **one-way InfoNCE**:

$$\mathcal{L}_{\text{contrastive}} = -\log \frac{\exp(\text{sim}(f_\theta(i_k), g_\phi(t_k))/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(f_\theta(i_k), g_\phi(t_j))/\tau)}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function (e.g., cosine similarity) and τ is a temperature parameter.

In this work, the text-to-image retrieval task is adapted to the specific context of marine turtle monitoring, aiming to build a system capable of retrieving turtle images from a textual description, thus facilitating data exploration and annotation by experts and researchers.

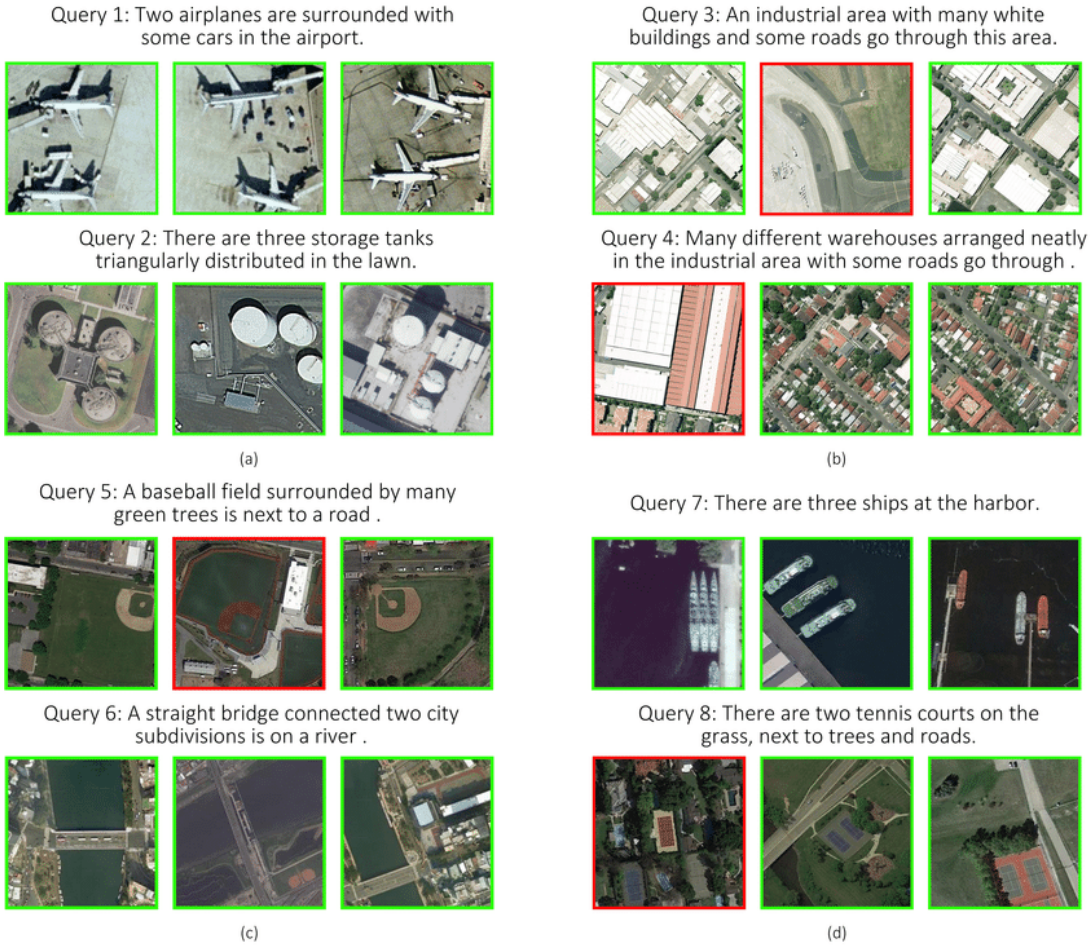

Fig. 2.11: Text-to-Image retrieval example on dataset *UCMerced-LandUse-Captions*. Text queries (a-h) retrieve matching images based on semantic content; retrieved images are marked as correct if with a green border, otherwise, it is red.

2.9 Evaluation Metrics for Retrieval

Performance evaluation in retrieval systems is essential for measuring how effectively the model retrieves relevant elements in response to a query. Among the most commonly used metrics for text and multimodal retrieval tasks are **Recall@K**, **Precision@K**, **Mean Average Precision**, and **Normalized Discounted Cumulative Gain**.

2.9.1 Recall@K

Recall@K measures the system's ability to include the correct result within the first K elements returned. Formally, given a set of queries Q and for each query $q \in Q$ the position $\text{rank}(q)$ of the first relevant result, we define:

$$\text{Recall@}K = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I}[\text{rank}(q) \leq K]$$

where $\mathbb{I}[\cdot]$ is the indicator function that is 1 if the condition is true and 0 otherwise [58].
Recall@1 measures the accuracy of the first result, while Recall@5 and Recall@10 evaluate the quality of the returned list over a broader horizon.

2.9.2 Precision@K

Precision at K returns the fraction of relevant results among the first K items retrieved. It is defined as:

$$\text{Precision@}K = \frac{1}{K} \sum_{i=1}^{K} \mathbb{I}[i \in \text{relevant}]$$

This metric is particularly useful when you want to limit the number of results shown to the user, ensuring high quality in the top positions [68].

2.9.3 Mean Average Precision

When a query may have multiple relevant results, the Mean Average Precision (mAP) represents the average of the *Average Precisions* (AP) calculated across

all queries. For a single query, the AP is:

$$\text{AP} = \frac{1}{R} \sum_{k=1}^{N} P(k) \cdot \mathbb{I}[k \in \text{relevant}]$$

where R is the total number of relevant results, N is the total number of results considered and $P(k)$ is the precision calculated up to position k [44]. The mAP is the arithmetic mean of the APs across all queries.

2.9.4 Normalized Discounted Cumulative Gain

Normalized Discounted Cumulative Gain (nDCG) is a metric that takes into account the position and graded relevance of results, very useful in ranking systems where items have different levels of importance. Discounted Cumulative Gain (DCG) is defined as:

$$\text{DCG}_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

where rel_i is the relevance of the result at position i. The nDCG normalizes the DCG with respect to the maximum ideal value (IDCG):

$$\text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}$$

This metric is useful for evaluating the overall ranking by considering the position and quality of the results [30].

2.9.5 Context-Specific Considerations

In this work, where each text query corresponds to a single ground-truth image, Recall@K remains the primary metric for evaluating the performance of the retrieval system. However, if the system were extended to scenarios with multiple query-relevant images or with graded relevance levels, mAP and nDCG would become more appropriate metrics.

3. PROPOSED METHOD

This chapter describes the methodology developed in this work. It first examines the CLIP architecture and its components, including vision and text encoders and the original contrastive loss. It then discusses the limitations of the standard CLIP approach in high-similarity domains and introduces an alternative training strategy based on multi-positive and combined contrastive losses. Finally, it explains the integration of LoRA for efficient fine-tuning.

3.1 Contrastive Language–Image Pretraining

This work is based on the use of **CLIP** [56], developed by OpenAI in 2021, which is among the most widely used VLMs. Its architecture is designed to learn joint visual and textual representations in a shared semantic space, thus developing the ability to associate images and linguistic descriptions even in zero-shot contexts, i.e., without specific training on specific tasks or categories. Unlike traditional supervised models, which are trained on manually annotated datasets, CLIP uses a "*web-supervised*" learning paradigm: the model is trained on hundreds of millions of image-text pairs collected from the web, where the texts act as implicit descriptions of the images.

The key mechanism that allows clip to work is **contrastive learning**, explained in detail in Section 2.3, the idea of which is to optimize alignment in the latent space by bringing the representations (embeddings) of related pairs (positive pairs) closer together and distancing those of unrelated pairs (negative pairs), more formally, the model maximizes the similarity between the embeddings of positive pairs and minimizes that between negative pairs.

3.1.1 Clip Architecture

CLIP, as shown in figure 3.1.3, is based on an architecture composed of two main components: an **image encoder** and a **text encoder**, designed to project multimodal data (visual and linguistic) into a shared semantic space. This allows for retrieval (text→image or image→text) by evaluating the vector similarity between learned representations, both encoders adopt the **Transformer** [80] architecture, also explained in detail in the Section 2.2

3.1.2 Vision Encoder – Vision Transformer

For images, CLIP employs a ViT, with the commonly used configuration ViT-B/32, where "B" indicates the Base variant of the model and "32" corresponds to the patch size (32×32 pixels) used to split the input image. Other configurations exist, such as ViT-L/14, where "L" refers to the Large variant and the smaller patch size of 14 allows for a finer spatial resolution [13].
As detailed in Section 2.2.1, an input image $x \in \mathbb{R}^{H\times W\times 3}$ is divided into $N = \frac{HW}{P^2}$ patches of dimension $P \times P$. Each patch is flattened and projected into a d-dimensional space through a learnable matrix W^P:

$$x_p = \text{Flatten}(x_{i,j})W^P \tag{3.1}$$

where $x_{i,j}$ is the patch at index (i, j) and $W^P \in \mathbb{R}^{(P\times P\times 3)\times d}$ is the patch projection matrix.
Positional embeddings $E_{pos} \in \mathbb{R}^{(N+1)\times d}$ are then added to the sequence, which also includes a learnable `[CLS]` token prepended to the patch embeddings.
The output embedding of this token, after processing by a classic Transformer Encoder, is used as the global representation of the image.

3.1.3 Text Encoder – Transformer for Text

For the text part, CLIP uses a standard Transformer Encoder 2.2, where each word, or subword token, is mapped into a vector embedding via a matrix W_{embed} with the addition of sinusoidal positional encoding:

$$e_i = W_{\text{embed}}(t_i) + PE(i), \tag{3.2}$$

where:

- $W_{embed} \in \mathbb{R}^{|V| \times d}$ is the embedding matrix;
- $PE(i) \in \mathbb{R}$ is the sinusoidal positional encoding at position i;
- t_i is the token.

Both encoders produce dense representations, which are projected into a shared embedding space and ℓ_2-normalized. The final representations are obtained as follows:

$$z_I = \frac{W_I f_\theta(x)}{\|W_I f_\theta(x)\|_2}, \quad z_T = \frac{W_T g_\phi(t)}{\|W_T g_\phi(t)\|_2}, \tag{3.3}$$

where:

- $f_\theta(x)$ is the output of the image encoder;
- $g_\phi(t)$ is the output of the text encoder;
- $W_I, W_T \in \mathbb{R}^{d \times d_{enc}}$ are learnable projection matrices (projection heads);
- $z_I, z_T \in \mathbb{R}^d$ are the final normalized vectors;
- $\| \cdot \|_2$ denotes the Euclidean norm.

The training objective is then to maximize the cosine similarity between z_I and z_T for aligned image–text pairs, while minimizing it for mismatched pairs, as detailed in the next section.

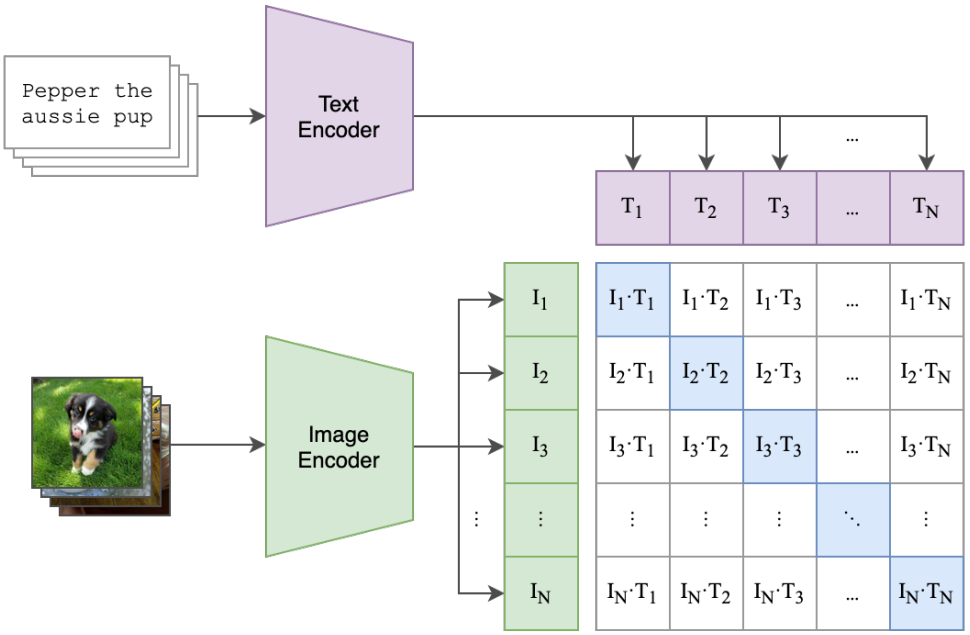

Fig. 3.1: The diagram describes CLIP's architecture. On the left, there are the two encoders, respectively, a Vit-B-32 and a Transformer-based text encoder that processes both images and captions. These two encoders map the outputs in the same embedding space. On the right, there is a logits matrix, computed via cosine similarity between all image-text pairs. The symmetric contrastive loss optimizes the alignment by maximizing similarity for correct pairs on the diagonal (positive pairs) and minimizing it for mismatched pairs off-diagonal (negative pairs).*Source: Adapted from Clip's paper* [56].

3.1.4 CLIP Loss: Symmetric Cross-Entropy

As introduced in Section 2.3.2, the InfoNCE loss is asymmetric, as it compares an anchor x against a set of candidates y. However, in multimodal contrastive learning, such as in CLIP, it is crucial to enforce a *symmetric alignment* between the two modalities (image and text).

To this end, CLIP adopts the **Symmetric Cross-Entropy Loss**, commonly referred to as *CLIP loss* [56]. This loss extends InfoNCE by applying it in both directions: image-to-text and text-to-image, and averaging the two contributions:

$$L_{\text{CLIP}} = \frac{1}{2}\left(L_{\text{img}\to\text{text}} + L_{\text{text}\to\text{img}}\right) \tag{3.4}$$

where:

$$L_{\text{text}\to\text{img}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(s(T_i, I_i)/\tau)}{\sum_{j=1}^{N}\exp(s(T_i, I_j)/\tau)} \tag{3.5}$$

$$L_{\text{img}\to\text{text}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(s(I_i,T_i)/\tau)}{\sum_{j=1}^{N}\exp(s(I_i,T_j)/\tau)} \tag{3.6}$$

Here, $s(\cdot,\cdot)$ denotes the similarity function (cosine similarity), τ is the temperature parameter and N is the batch size. Both terms are essentially InfoNCE applied in opposite directions. The objective is to learn a **shared embedding space** where each image-caption pair (I_i, T_i) is mapped to nearby points, while non-matching pairs are pushed apart. This bidirectional formulation ensures that the model can retrieve images from text queries and text from image queries with equal effectiveness.

3.2 Training Strategy: Beyond the Standard CLIP Loss

3.2.1 The Diagonal Assumption

The CLIP loss is structured around a similarity matrix $S \in \mathbb{R}^{N\times N}$, where each entry $S_{i,j} = sim(I_i, T_j)$ quantifies the similarity between the i-th image and the j-th text in the batch. The loss enforces that:

- $S_{i,j}$ for $i = j$, that is the diagonal, should be **maximized** because these are the true positive pairs.
- $S_{i,j}$ for $i \neq j$ should be **minimized** because these are all the other image-text pairs that are considered negatives.

This mechanism implicitly enforces a strict binary separation: each anchor (image or text) has exactly one positive and $N-1$ negatives, and doing that, it penalizes any semantic overlap between non-diagonal pairs, even if two captions describe similar images or two images represent similar scenes described by nearly identical captions.

3.2.2 Limitations of the CLIP Loss

As described in 3.1, CLIP was designed for generalist scenarios to learn a shared image-text space by optimizing a symmetric contrastive loss. During training, this loss favors similarity between correct pairs (diagonal of the matrix) and penalizes all other combinations in the batch, which are treated as negative.

However, this optimization scheme relies on a strong assumption: each image (or text) has only one positive match in the batch, while all other combinations are considered incorrect. While this assumption is correct for a generalist model oriented toward zero-shot classification, it is too rigid in our context, because it introduces critical limitations when applied to narrow-domain scenarios, such as the one considered in this project, where images and captions exhibit **high intra-class similarity** both semantic and visual.

3.2.3 Limitations in High-Similarity Domains

In our context, focused on marine wildlife, specifically sea turtles, such assumptions are not only unrealistic but also actively detrimental to the learning process for two main reasons:

- **Visual Similarity**: Many images in the dataset depict sea turtles from above, in similar poses, under comparable lighting conditions, and against nearly indistinguishable seascapes.
- **Semantic Similarity**: Captions, whether generated automatically or manually curated, often include recurring structures and lexical fields (e.g., "a sea turtle swimming in clear water", "a turtle gliding through the ocean", "an underwater view of a turtle"). These descriptions may differ slightly in syntax but convey nearly identical semantics.

In this scenario, considering such pairs as completely unrelated is counterproductive: semantically similar images and captions should be attracted into the embedding space, not removed as if they were completely unrelated.
To address these challenges, it is essential to adopt training strategies that **relax the one-positive assumption** and explicitly account for the presence of multiple positives in each batch.

3.2.4 Multi-Positive Loss: Unified Contrastive Learning

To overcome the limitations of the standard CLIP loss in narrow-domain scenarios, the *UniCL* approach [85] is adopted, which generalizes contrastive learning by allowing multiple semantically consistent positives per anchor. This strategy

is designed to better reflect the complexity of real-world data, where different samples may describe similar semantic concepts without being exact matches. In standard CLIP training, each image-text pair is treated as the only correct (positive) match in the batch, and all other combinations are considered negatives. This assumption, although effective in large-scale, open-domain training, becomes overly restrictive when applied to specialized domains like ours, where data often present a high degree of semantic and visual redundancy. In our dataset, for instance, many images depict sea turtles photographed from above, often swimming in visually uniform marine environments and accompanied by similar textual descriptions, and consequently, treating each of these as entirely unrelated would be counterproductive. Instead, it is desirable to attract semantically similar samples into the same region of the joint embedding space, enabling the formation of robust clusters around high-level semantic concepts (e.g., "sea turtle swimming near the surface").

The goal of UniCL is precisely this: to extend the contrastive loss to a multi-positive case by explicitly allowing more than one matching sample per anchor, and this allows the model to better learn intra-class variability while maintaining inter-class discriminability. The goal is therefore to minimize the symmetric bidirectional contrastive loss defined as:

$$\min_{\theta,\phi} \mathcal{L}_{BiC} = \mathcal{L}_{i2t} + \mathcal{L}_{t2i} \tag{3.7}$$

where θ and ϕ are the parameters of the visual and text encoders and $\mathbf{v}$ and $\mathbf{u}$ denote the resulting image and text features, respectively.

The image-to-text contrastive loss to align matched images in a batch with a given text:

$$L_{i2t} = -\sum_{i\in\beta} \frac{1}{|P(i)|} \sum_{k\in P(i)} log \frac{\exp(\tau \mathbf{u}_i^T \mathbf{v}_k)}{\sum_{j\in\beta} \exp(\tau \mathbf{u}_i^T \mathbf{v}_j)} \tag{3.8}$$

where β is the batch that contains pairs image-text, $k \in P(i) = \{k|k \in \beta, y_k = y_i\}$, y_i is the concept (label) associated with the i image and y_k is the concept (label) associated with the k text.

Symmetrically, the text-to-image contrastive loss aligns matched texts to a given image:

$$L_{t2i} = -\sum_{j\in\beta} \frac{1}{|P(j)|} \sum_{k\in P(j)} log \frac{\exp(\tau \mathbf{u}_k^T \mathbf{v}_j)}{\sum_{i\in\beta} \exp(\tau \mathbf{u}_i^T \mathbf{v}_j)} \tag{3.9}$$

where β is the batch that contains pairs image-text, $k \in P(j) = \{k|k \in \beta, y_k = y_j\}$, y_j is the concept (label) associated with the j text and y_k is the concept (label) associated with the k image.

This formulation naturally extends CLIP's unique approach by leveraging semantic labels as weak supervision. This leads to better local alignment and more consistent clustering across the multimodal embedding space, which is particularly useful when categories (e.g., different turtle species or behaviors) are visually and semantically similar.

The figure 3.2 shows the operation of the UniCL from a visual point of view.

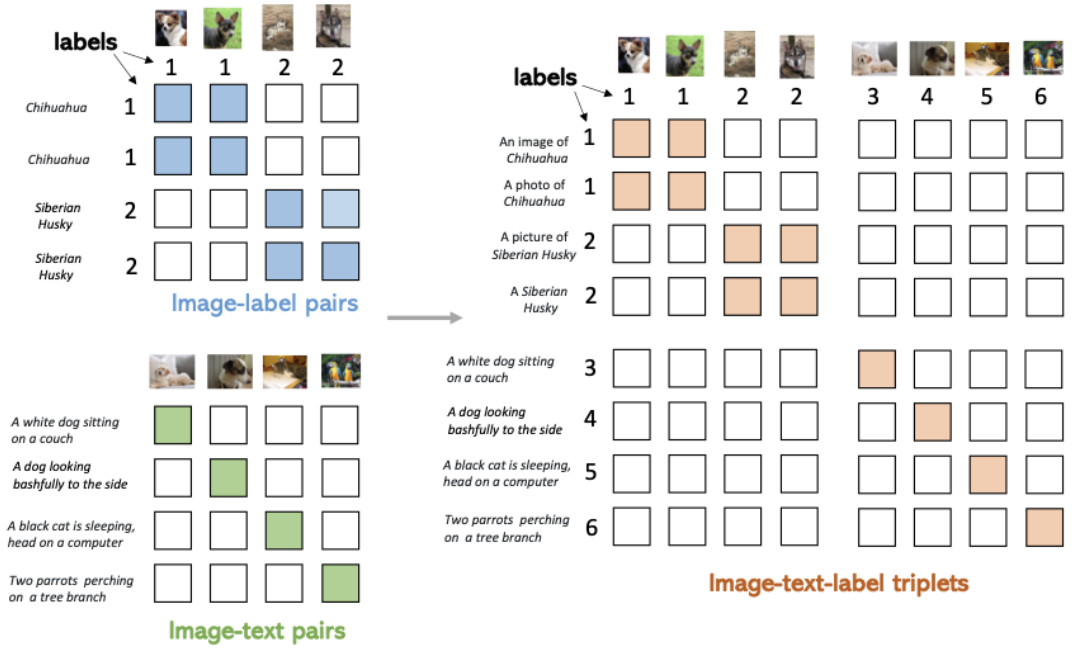

Fig. 3.2: An illustration showing the integration of image-label and image-text data in the image-text-label space. For image-text data, we associate a textual concept with each label, aligning images and textual concepts based on the annotated labels (blue boxes). For image-text data, each pair has a unique label index, resulting in alignment exclusively along diagonal entries (green boxes). On the right, the two data types are simply combined into image-text-label triplets, where red boxes indicate positive pairs and empty boxes represent negative pairs.

Source: Adapted from Microsoft's UniCL paper [85].

3.2.5 Our Combined Loss Strategy

Based on the previous analysis, a training strategy that integrates the strengths of the original CLIP loss and UniCL was designed. The goal of this combination is twofold: on the one hand, we want to maintain a clear separation between the "turtle" category and all the other classes present in the dataset (dolphins, debris, empty sea, etc.) using UniCL; on the other hand, thanks to CLIP loss, we want to prevent all turtle images from collapsing into the same point in space, thus losing intra-category variability. This variability is essential to distinguishing aspects such as different poses, interactions with other animals, different environmental contexts, or particular behavioral states.

This strategy allowed us to obtain an embedding space in which the images and descriptions of sea turtles are well separated from the distractors but internally organized in a semantically meaningful way.

The ***final loss function*** to be minimized is defined as the average of the two components:

$$Loss = \frac{L_{BiC} + L_{CLIP}}{2} \tag{3.10}$$

3.3 Low-Rank Adaptation for Efficient Fine-Tuning

In this project, the **LoRA** technique [25] is adopted as a strategy for efficient CLIP model tuning in a narrow-domain scenario. LoRA is particularly suitable in contexts where full tuning of large pre-trained models is computationally expensive or requires a prohibitive amount of memory. Models like CLIP are typically pre-trained on huge datasets and consist of hundreds of millions of parameters. Optimizing the entire model on a relatively small, domain-specific dataset, as in this work, not only risks overfitting but also entails high computational costs. Furthermore, full optimization requires storing a copy of all model weights for each domain, making implementation and experimentation less efficient. To address this issue, we turned to PEFT; LoRA stands out among PEFT techniques by introducing a minimal number of trainable parameters while maintaining the expressive power of the original model.

The **central idea** of LoRA is to decompose the parameter updates into a low-rank matrix product. Rather than updating the full weight matrix $W \in \mathbb{R}^{d \times k}$ of

a given layer (typically a linear projection in the attention mechanism), LoRA introduces a low-rank decomposition:

$$W' = W + \Delta W = W + BA \tag{3.11}$$

where:

- W is the original (frozen) weight matrix.
- $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{k \times r}$ are trainable matrices with rank $r \ll \min(d, k)$.
- $\Delta W = BA$ is the low-rank update that captures task-specific adaptation.

By learning only A and B, we significantly reduce the number of trainable parameters, from dk to $r(d + k)$), allowing for efficient optimization and reduced memory footprint. In our implementation, LoRA was applied to both the text encoder and visual encoder of the CLIP model. Specifically, we targeted the MHSA layers and their associated projection layers:

- Query, Key, Value projections: $q_{proj}, k_{proj}, v_{proj}$.
- Output projection: out_{proj}.

These components play a central role in the representation learning process of both encoders. By fine-tuning only these subspaces, we allowed the model to adapt to domain-specific variations (e.g., visual and linguistic features of turtles) without altering the entire pre-trained backbone.
The adoption of LoRA yielded several benefits:

- Reduced training time and GPU memory usage, allowing us to train on consumer-grade hardware.
- Improved generalization by avoiding overfitting due to excessive parameter updates.
- Modularity and reusability, as different LoRA weights can be swapped or merged with the base model without retraining from scratch.

4. EXPERIMENTS

This chapter outlines the experimental setup used to validate the proposed approach. It describes the dataset construction process, including image collection, preprocessing, and caption generation. The implementation of the dynamic batch sampler is then presented, along with the evaluation protocol adopted for category-level and instance-level retrieval experiments.

4.1 Dataset Construction

4.1.1 Collection and Selection of Images

The dataset used for training and evaluating the model was constructed by combining data from various sources, with the goal of maximizing semantic and visual variety, which is essential for robust learning. The core of the dataset consists of high-resolution images extracted from a proprietary archive, consisting of frames sampled from three different video sources at an aerial angle. These sequences capture real marine environments containing turtles, dolphins, floating debris, and portions of the open sea without any human subjects. However, because these images are extremely similar in terms of angle, environmental conditions, and visual content, using the entire dataset would have resulted in a high risk of overfitting. For this reason, a small portion of distinctive images, characterized by greater variety, was manually selected. Specifically, approximately 10% of the images from the original dataset were selected, of which approximately 60% belonged to the "turtle" class, while the remaining 40% was evenly distributed among the other categories (dolphins, debris, and open sea). Figure 4.1 shows an example of the class "images", while Figure 4.2 of the "debris", "sea", and "dolphin" class images.

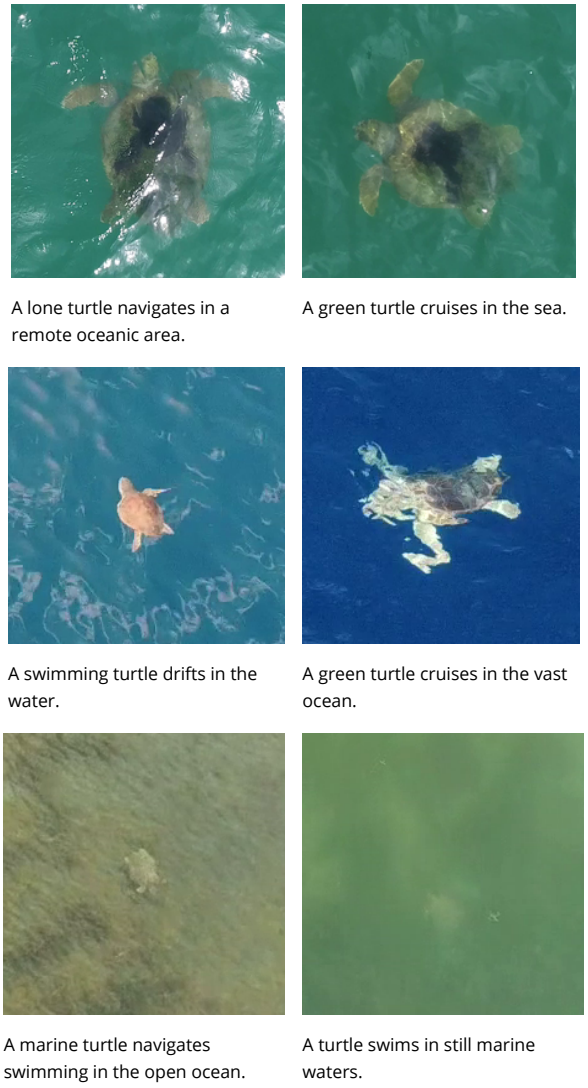

Fig. 4.1: This image shows some examples of images from our dataset. As can be seen, the specimens tend to display similar poses, with morphological and behavioral repetitiveness characterizing the majority of the sample. Furthermore, some aspects pose critical challenges for retrieval, including reflections on the water, noise artifacts, and seabeds that hinder turtle recognition. The images shown are selected from those with the greatest surface diversity, but the dataset includes numerous nearly identical cases.

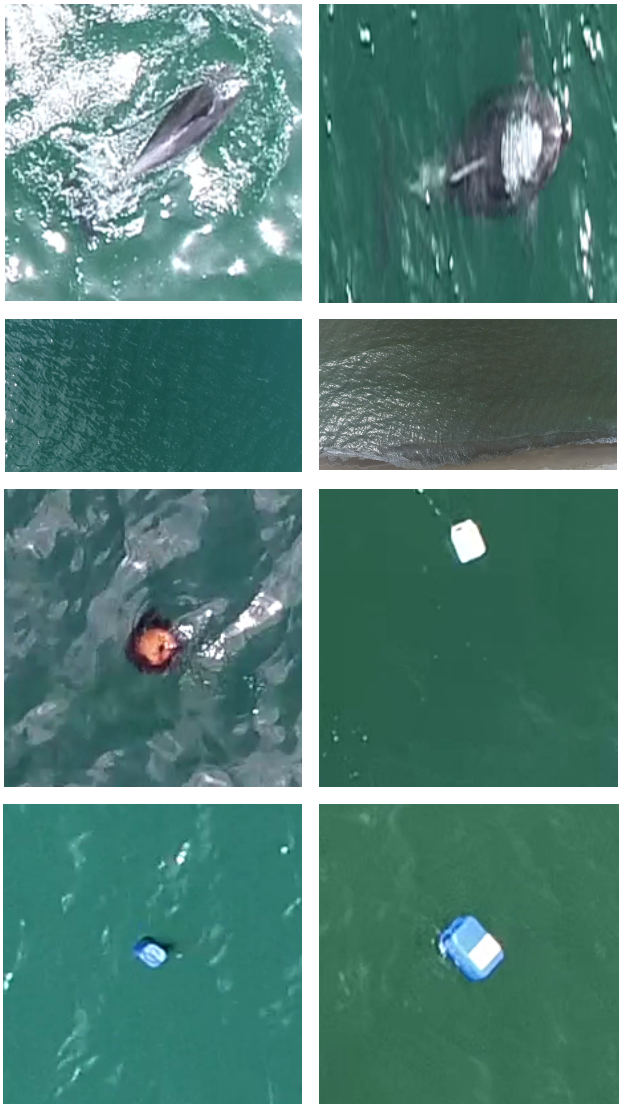

Fig. 4.2: This image shows some images of the "sea", "dolphin", and "debris" class images.

To increase intra-class diversity and improve the model's ability to generalize to different visual contexts, we integrated additional images from freely accessible public datasets. Among these, we used *SeaTurtleID2022* [1], a dataset that collects images of sea turtles from different viewpoints, environments, and interaction conditions. We also included a subset of another open-access dataset available

on the *Roboflow platform*, also containing heterogeneous turtle images. In both cases, the integration was performed selectively, including only visually relevant examples that were distinct from those already present in our private dataset. Figure 4.3 shows an example of turtles images taken from the dataset *SeaTurtleID2022* and from the *Roboflow platform*.

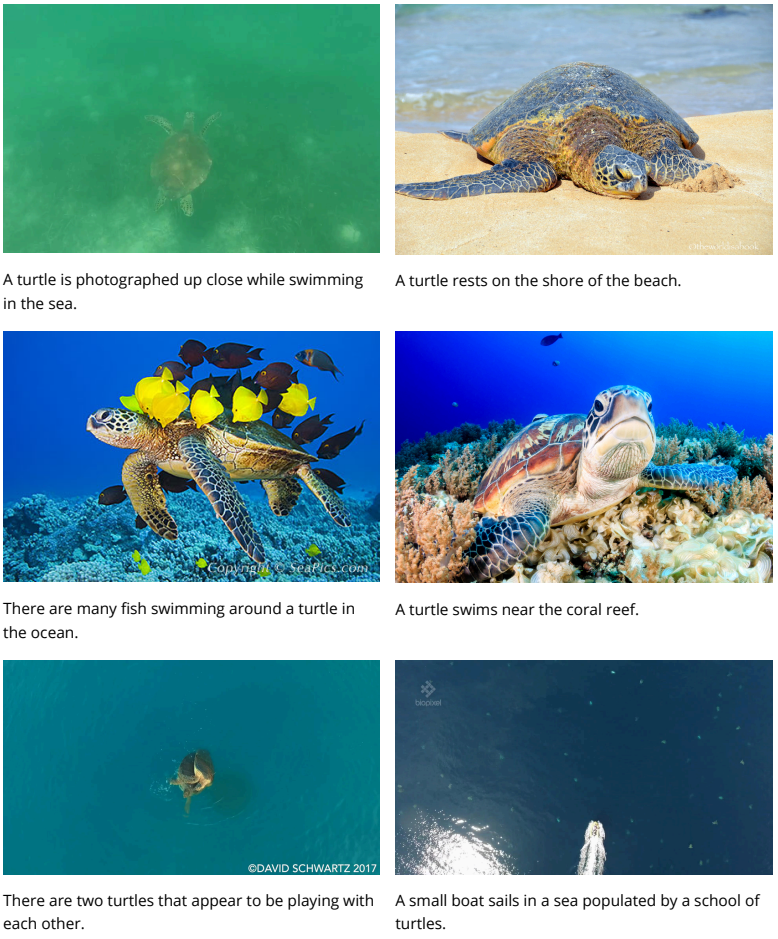

Fig. 4.3: This image shows examples of images, with their respective captions, taken from the SeaTurtleID2022 and Roboflow platform. It is possible to see how some images have noise and contrasting colors that do not help the retrieval model.

To further improve semantic discrimination during fine-tuning, it was essential to introduce a significant number of distractors. These examples, belonging to categories that were not relevant to the task but visually similar or potentially confounding, were selected from the COCO [39] validation set (COCO Val). The choice of COCO Val, rather than the training set, is motivated by the fact that the CLIP models were pre-trained on COCO Train, so using the validation set allows us to avoid leakage and introduce new distractors never seen during pre-training. Figure 4.4 shows an example of these images.

Fig. 4.4: This image shows examples of images, with their respective captions, taken from the COCO val dataset. These images are used as distractors.

Overall, the **training set** consists of approximately 24,100 images, of which approximately 17% depict turtles (including both examples from our proprietary dataset and external sources), while over 80% are images from the COCO dataset, used as distractors. The remaining images (less than 1%) represent categories such as debris, open sea, and dolphins, which serve to strengthen the model's

ability to distinguish between semantically close classes, so these are also distractors like COCO.

The **validation set** contains approximately 6,500 images, with a similar distribution: approximately 7-8% are turtles, while over 90% are distractor images from the COCO dataset. Again, a small percentage of additional images come from other minor categories (debris, sea, dolphins), present in marginal quantities but useful for evaluation.

Finally, the **test set**, used exclusively for the final evaluation of the model, includes approximately 4,500 images, of which approximately 10% belong to the target classes (turtles), while the remainder is occupied by COCO distractors and marine distractors.

The choice to include a small percentage of images from other minor categories is dictated by the fact that the primary objective of our retrieval task is focused on turtle recognition. We leverage CLIP's intrinsic ability to recognize these minority categories without the need for tuning, since their accurate classification is outside the scope of our primary objective. The pretrained CLIP model demonstrates sufficient zero-shot performance on these auxiliary classes, making additional training unnecessary for our purposes.

The dataset construction was geared not only toward including images relevant to the target category, but also, above all, toward balancing it with a large number of irrelevant or semantically close images. This is because, in the context of text-to-image retrieval, the quality of learning is strongly influenced by the presence of distractors within the batch. A dataset composed almost exclusively of positive examples (e.g., only images of turtles) would lead the model to memorize visual patterns rather than learn to generalize semantic correspondences between text and image.

The massive inclusion of distractors, particularly images from COCO, increased *intra-batch diversity and encouraged the model to more selectively and robustly identify relevant cues associated with turtles.* This approach proved crucial for the effective fine-tuning of a contrastive model like CLIP, which bases its learning on the explicit comparison of positive and negative examples.

4.1.2 Caption Generation

One of the main challenges encountered in building the dataset was the lack of textual descriptions (captions) for most images, except the COCO dataset, which provides prestructured textual annotations. The presence of distinct, semantically coherent but lexically different captions is essential for effective learning, especially in a text-to-image retrieval task, where linguistic diversity contributes to building more robust representations.

To automate the description generation process, we used a version of **BLIP** [38] (Bootstrapping Language–Image Pretraining) optimized for generic image captioning. This model combines a visual encoder with a Transformer-based linguistic decoder and can generate natural-looking descriptions from images, without the need for manual annotations (Figure 4.5 shows its architecture).

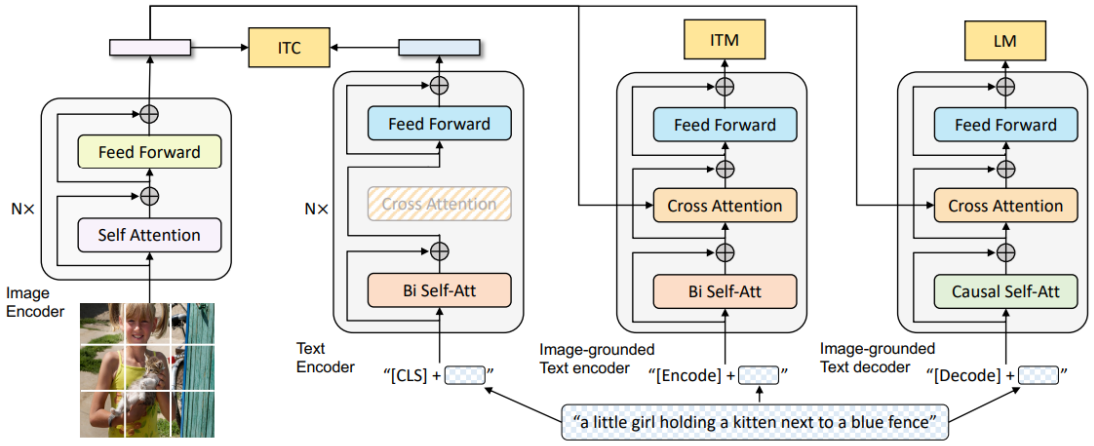

Fig. 4.5: The proposed model, *Multimodal Mixture of Encoder-Decoder*, unifies three functionalities: (1) a unimodal *encoder* trained with ITC (*Image-Text Contrastive*) loss to align visual and linguistic representations; (2) an image-based *text encoder* with cross-attention layer (trained with ITM (*Image-Text Matching*) loss) to model vision-language interactions; (3) an image-based *text decoder* with causal self-attention and LM (*Language Modeling*) loss to generate captions. Adapted from [38].

However, the direct use of BLIP presented some challenges. Specifically, semantic errors were encountered, such as the misidentification of sea turtles mistaken

for rocks, dolphins, or other marine objects, and a tendency to generate identical captions for similar images. To address these issues, an automatic post-processing phase was introduced to reduce lexical redundancy: each duplicate caption was paraphrased using a large linguistic model, **LLaMA** [76] (3.1 8B Instruct), able to produce equivalent but linguistically different semantic rewrites.
Subsequently, a manual, qualitative check was conducted on all generated captions. This step had a twofold objective:

1. Ensuring each image was associated with a unique description.
2. Correcting any residual conceptual errors resulting from the automatic process.

The final result is a fully annotated dataset, with accurate, different, and semantically consistent captions with the visual content, built through an automated but carefully supervised pipeline. This level of curation proved essential to ensure high-quality learning, laying the foundation for a model capable of fine-grained distinction between turtle images even in the presence of complex visual distractors.

4.1.3 Preprocessing

The images from above contained in our proprietary dataset feature extremely high resolution and represent marine scenes acquired from above, often characterized by complex visual conditions due to solar reflections, waves, and natural camouflage. In this context, turtle identification is difficult even for a human observer, making the direct use of the full image in training or inference processes impractical.
To address this issue, we leveraged the dataset's pre-existing annotations, originally designed for object detection and containing bounding boxes for categories such as turtle, debris, and dolphin. Starting from these annotations, we generated crops centered on the objects of interest, resized to 224x224 pixels, providing the model with a closer and semantically more informative view. This operation was applied systematically to all examples from our dataset, both in the training and in the validation and test sets.

For the auxiliary datasets, i.e., images from COCO and open sources containing sea turtles in different contexts, we applied a simple uniform resolution reduction to 224×224, without any cropping. This choice reflects both the absence of bounding box annotations in these datasets and the fact that turtles are generally clearly visible and central in the original image.
This preprocessing strategy normalized the visual content of the inputs and made the field of view more consistent across the different source domains, improving the quality of contrastive training.
In parallel with image preprocessing, a normalization process was also required for captions. Each image in the dataset is associated with a descriptive caption that must be transformed into a suitable numerical representation to be processed by the Transformer-based text encoder. For this purpose, the tokenizer provided by CLIP was used, which is based on the Byte Pair Encoding (BPE) algorithm [67]. BPE is among the most widely adopted tokenization techniques in NLP, thanks to its ability to effectively handle out-of-vocabulary words and to keep the vocabulary size manageable, a crucial aspect for large-scale models. The algorithm iteratively merges the most frequent character or byte pairs in a corpus until a predefined vocabulary size is reached. In CLIP, no additional training is needed for the tokenizer, since it comes with a pre-trained vocabulary and merging rules, ensuring consistent and stable tokenization. The text preprocessing pipeline, though largely transparent to the user, involves several important steps. Raw captions are first normalized using basic Unicode cleaning procedures, then converted into UTF-8 byte sequences. These are segmented into subword units according to the pre-learned BPE rules, producing a sequence of tokens interpretable by the model. Each token is mapped to a numerical identifier from the pre-trained vocabulary. Special tokens are then added to mark the beginning and end of the sequence and shorter captions are padded up to the model's maximum input length (77 tokens). Through this process, captions are transformed into integer sequences that can be efficiently processed by CLIP's text encoder, ensuring a coherent and semantically aligned representation with the corresponding visual modality.

4.2 Experimental Setup

Model fine-tuning was conducted on a workstation equipped with an **NVIDIA RTX 4090** with **24 GB of memory**. The training was conducted using two different pre-trained versions of CLIP available through Hugging Face: *laion/CLIP-ViT-B-32-laion2B-s34B-b79K* [3], a large-scale variant trained on the LAION-2B dataset, and *openai/clip-vit-base-patch32* [56], the original CLIP model released by OpenAI. Both models were fine-tuned separately using the same training pipeline, allowing for a comparative evaluation of performance across architectures trained on different data distributions.

To enable parameter-efficient fine-tuning, **LoRA** was applied to both models. Specifically, LoRA adapters were injected into the attention projection layers and projection heads of CLIP (i.e., `k_proj`, `q_proj`, `v_proj`, `out_proj`, `visual_projection` and `text_projection`). The rank and scaling hyperparameters were set to $r = 8$ and $\alpha = 16$, respectively, with a dropout rate of 0.1. These adapters introduce a low-rank decomposition of the weight updates, significantly reducing the number of trainable parameters while maintaining the model's expressive power.

The optimization was performed using **Adam with decoupled weight decay** [40], also called AdamW, with an initial learning rate of $3e-4$, a weight decay of $4e-4$ and a batch size of *256*, training was performed for a total of 50 epochs.

To dynamically adjust the learning rate, a scheduler with cosine annealing [41] and a warmup phase was adopted. Specifically, the warmup phase was set to 10% of the total number of steps, calculated as a function of the number of epochs and the size of the dataset.

During training, the model was optimized to learn discriminative representations through a supervised contrastive loss supporting multiple positive examples per anchor. In our setup, *only one semantic category of interest was defined explicitly: turtle*. All remaining samples, including images containing debris, dolphins, sea surface or examples from COCO, were instead treated as **independent distractor classes**, with each sample assigned to a unique class.

This means that only turtle samples benefit from having multiple positive pairs within a batch, while for all other examples, the retrieval task boils down to a

one-to-one matching between image and caption, as in the classic CLIP paradigm. This setup was specifically designed to maximize the model's ability to distinguish turtles from visually similar yet semantically irrelevant entities, such as dolphins or floating debris. Although this approach sacrifices intra-class generalization for non-turtle categories, it enables a sharper decision boundary around the concept of "turtle", which is central to our application domain. Distractor diversity thus plays a critical role in challenging the model with hard negatives, forcing it to focus on subtle but semantically meaningful visual cues.

4.2.1 Batch Sampler

To ensure balanced and optimal learning between the specific marine turtle and generic domains, we implemented a dynamic batch composition strategy with the following ratio:

- 64 samples for the target category turtle;
- 192 samples drawn from the other categories, in particular COCO, debris, sea, and dolphin. Each pair of text-image in these categories receives a unique category number.

This configuration, with a batch size of 256, was defined to optimize the dynamics of contrastive loss in a multimodal context. Batch partitioning (64 turtle examples and 192 distractors) allows the model to specialize on the class of interest while preserving its generalization ability, and is crucial for learning quality given the composition of the dataset.
Since the availability of turtle examples is lower than that of distractors (COCO, debris, sea, dolphin), the sampler implements an early stopping criterion: when the number of residual turtle examples drops below 64, the generation of new batches is stopped, and those already composed are passed on to the training process. In this way, each batch observed by the model maintains the predetermined composition (64/192), ensuring a stable balance for the entire epoch.
This choice implies that, at the end of an epoch, a small (and therefore negligible) number of unused turtle examples (less than 64) may remain, along with a larger share of unconsumed distractors. This latter case is not critical for our primary

goal, which is to maximize consistent and balanced exposure of the turtle class across batches, promoting the learning of its distinctive features without introducing imbalances in the composition required by contrastive loss.
This batch partition was designed to optimize contrastive learning dynamics in the multimodal context, allowing the model to specialize toward the target domain (turtle) while preserving the model's generalization ability across the semantic diversity of other examples in the full dataset (distractors). This ratio was also defined to ensure the quality of learning related to the type of loss used and described previously, as it is closely linked to the composition and distribution of the categories and samples in the batches.
The pseudo-code of the implemented batch sampler is reported in Algorithm 1.

Algorithm 1 Dynamic Batch Sampler

```
Initialize all data structures
turtle_n_samples ← 64
distractors_n_samples ← 192
while true do
    batch ← [ ]
    taken ← take_samples(turtle_n_samples)
    Append taken to batch
    if length(taken) < turtle_n_samples then
        break
    else
        Append take_samples(distractors_n_samples) to batch
    end if
    yield batch
end while
```

4.3 Evaluation Metrics

To evaluate the model's performance in the text-to-image retrieval task, we used **Recall@K** as the main metric, calculated in two distinct ways: one based on semantic category matching and one based on exact image-caption matching.

4.3.1 Category-level retrieval

In this first scenario, a prediction is considered correct if the model, given a textual description, is able to retrieve an image belonging to the same semantic category.

For example, a query related to a turtle will be evaluated positively if among the first K retrieved images there is at least one labeled as turtle, even if it is not the exact one associated with the query. This type of evaluation measures the model's ability to structure the multimodal space based on semantic similarity and is therefore suitable for contexts where retrieval is desired, that is, robust to visual and linguistic variability. The calculated metrics include:

- Recall @1, @5, @10 per category.
- Category Mean Rank: average of the rank of the first image with the correct category.

4.3.2 Exact instance-level retrieval

In the second, more stringent scenario, the retrieval is considered correct only if the returned image is exactly the one associated with the text query. To make this experiment more meaningful, we limited the calculation of this metric to only examples belonging to the focus categories, namely, turtles.
This type of evaluation is particularly useful for testing the model's fine-grained precision in scenarios where it is necessary to distinguish between visually very similar images, for example, several turtles seen from above in similar contexts.
The metrics calculated in this scenario are the same as those used in the previous case.
In conclusion, the combined use of these two evaluation strategies allows us to measure both the quality of the semantic structuring of the embedding space and the model's ability to accurately distinguish between similar images, a particularly critical element in our use case, where visual distractors (e.g., waves, reflections, similar marine animals) represent a significant source of ambiguity.

5. RESULTS AND DISCUSSIONS

This chapter presents and analyzes the experimental results obtained using the proposed methodology. It reports retrieval performance at both category and instance levels, discusses the observed trends, and provides an interpretation of the outcomes in relation to the objectives of this thesis.
Tables 5.1 and 5.2 show the performance of the models, with the test set, in two distinct scenarios:

- *cat_all_R@k*: Measures the model's ability to correctly retrieve images belonging to the correct category as described in 4.3.1.
- *turtle_R@k*: Evaluates the model's ability to retrieve the exact turtle instance as described in 4.3.2.

The models compared are the following:

- *CLIP_OpenAI_base*: Original version of the CLIP model with weights provided by OpenAI [56].
- *CLIP_OpenAI_tuned*: Fine-tuned version of the previous model.
- *CLIP_LAION_base*: Original version of the CLIP model with weights from the LAION-2B dataset [65].
- *CLIP_LAION_tuned*: Fine-tuned version of the previous model.

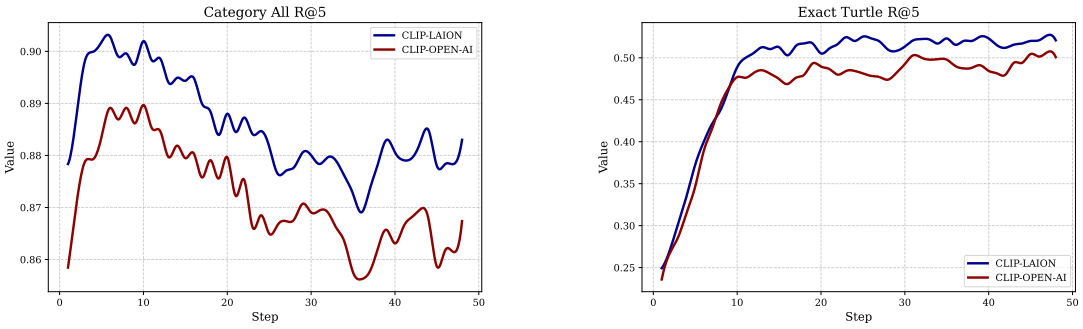

Fig. 5.1: Comparison of Recall@5 performance on the validation set when training both models. On the left, the performance is shown with respect to retrieval by category, on the right by instance.

Model Name	cat_all_R@1	cat_all_R@5	cat_all_R@10	mean_rank
CLIP_OpenAI_base	0.3737	0.6085	0.7151	19
CLIP_OpenAI_tuned	0.4531	0.6970	0.7979	12
CLIP_LAION_base	0.4821	0.7140	0.8009	13
CLIP_LAION_tuned	0.4245	0.6758	0.7733	14

Tab. 5.1: Test results considering all categories.

Model Name	turtle_R@1	turtle_R@5	turtle_R@10	mean_rank
CLIP_OpenAI_base	0.0536	0.1429	0.1984	126
CLIP_OpenAI_tuned	0.1884	0.4841	0.5853	52
CLIP_LAION_base	0.0773	0.2222	0.2896	87
CLIP_LAION_tuned	0.1488	0.3829	0.5	68

Tab. 5.2: Test results considering the "turtle" category.

5.1 Analysis of the results

5.1.1 Category-level analysis

Regarding category-level retrieval, as shown in Table 5.1, significant performance differences are observed between the different models and their fine-tuned variants. In the case of the OpenAI CLIP model, fine-tuning results in a consistent improvement across all metrics considered. Specifically, *Recall@1* increases from 0.3737 to 0.4531, *Recall@5* grows from 0.6085 to 0.6970 and *Recall@10* goes from 0.7151 to 0.7979. At the same time, the mean rank of the first correctly retrieved

image per category decreases from 19 to 12, indicating a higher ranking of relevant images in the result lists. These results suggest that fine-tuning improves the model's ability to effectively match textual descriptions to the semantic categories to which images belong, increasing accuracy and efficiency in the retrieval process. In contrast, the CLIP LAION model exhibits a different behavior: the baseline model slightly outperforms the fine-tuned model in terms of *Recall@1*, 0.4821 versus 0.4245, and also shows marginally better values for *Recall@5* and *Recall@10*. The *mean rank* deteriorates only slightly, going from 13 to 14. This trend could be related to the fact that the LAION model, trained on an extremely large and diverse dataset such as LAION-2B [65], already has a robust ability to represent semantic similarity at a general level. Consequently, fine-tuning, likely focused on a specific domain (e.g., turtles), tends to specialize the model on that domain, introducing a slight trade-off in its cross-category generalization. Two additional aspects deserve consideration. First, the reduction in mean rank from 19 to 12 for OpenAI CLIP is particularly relevant for practical retrieval scenarios, since it means that relevant images appear earlier in the ranked list, reducing the number of non-relevant images a user needs to browse. Second, the distribution of the fine-tuning dataset was heavily skewed toward the "turtle" class, which likely amplified improvements for that category while limiting gains for others (e.g., dolphins, debris, empty sea). This highlights the importance of class-aware sampling or loss weighting to achieve balanced improvements across all categories. Finally, the divergence in behavior between OpenAI CLIP and LAION CLIP suggests that the effectiveness of fine-tuning is inversely correlated with the scale and diversity of the pre-training dataset: models already optimized on billions of image-text pairs may benefit less from small, domain-specific fine-tuning and could even suffer from **overfitting** or **catastrophic forgetting** [16]. Future work could address this by adopting regularization strategies or by combining LoRA with adaptive freezing policies to preserve generalization while still incorporating domain-specific knowledge.

5.1.2 Instance-level analysis

In the context of the instance-level evaluation (table 5.2), which represents a significantly more challenging scenario than the category-level one due to the high

intra-class similarity among samples, the results show marked differences in the distinguishing capabilities between the baseline and fine-tuned models. For the OpenAI CLIP model, the effect of fine-tuning is particularly evident: *Recall@1* increases almost fourfold, from 0.0536 to 0.1884, and similarly *Recall@5* and *Recall@10* show substantial improvements (from 0.1429 to 0.4841 and from 0.1984 to 0.5853, respectively). Furthermore, the *mean rank*, an indicator of the average position of the first correctly retrieved image, dramatically decreases from 126 to 52. These data demonstrate how fine-tuning adaptation allows the model to effectively distinguish between very similar specimens of the same category, such as different sea turtles, and to be robust to visual variations such as perspective, lighting, and context. Regarding the CLIP LAION model, although the increase in performance metrics after fine-tuning is less pronounced than that of CLIP OpenAI, the improvements are still clear: *Recall@1* increases from 0.0773 to 0.1488 and the other metrics follow similar trends, accompanied by an improvement in *mean rank* from 87 to 68. This suggests that even a model originally more robust, thanks to the large LAION-2B training dataset, benefits from the specialization provided by fine-tuning, which enhances its discriminative ability in contexts with high intra-class similarity. The dynamics underlying these improvements are further clarified by the analysis of validation curves reported in Figure 5.1, which show the evolution of Recall@5 during training. Although these curves refer to validation performance, they provide meaningful insights into the learning process that led to the observed test results. When considering category-level retrieval, CLIP-LAION maintains a consistent advantage over CLIP-OpenAI across the entire training process, aligning with its superior aggregate test performance (table 5.1). However, when focusing on the instance-level scenario, a different pattern emerges: CLIP-LAION demonstrates a rapid initial improvement and dominates in the early stages of training, but its growth stabilizes relatively early. Conversely, CLIP-OpenAI, although starting from lower values, exhibits a slower but steady improvement, eventually surpassing CLIP-LAION in later stages. This behavior is consistent with the final test results (table 5.2), where the fine-tuned OpenAI model achieves the highest Recall@5 (0.4841), significantly outperforming its LAION counterpart (0.3829). These observations underline the importance of analyzing learning curves on critical classes, as they reveal adapta-

tion patterns and potential trade-offs that may remain hidden when looking only at aggregated metrics. Two additional observations are worth noting. First, the reduction in *mean rank* from 126 to 52 for OpenAI CLIP has a significant practical impact: in real-world retrieval systems, this translates into substantially fewer irrelevant images a user must navigate before finding the correct match. Second, the smaller margin of improvement observed for LAION CLIP suggests that models pre-trained on massive and highly diverse datasets already encode fine-grained features to some extent, leaving less room for improvement through small-scale fine-tuning. Nonetheless, the consistent gain across all metrics confirms that domain-specific adaptation remains beneficial, even for such strong baselines. Looking ahead, further improvements could be achieved by integrating hard negative mining and class-aware sampling strategies, which would expose the model to more challenging contrasts during training. Additionally, employing advanced contrastive losses such as Supervised Contrastive Loss (SupCon) with multiple positive pairs, combined with augmentation strategies tailored to variations in pose, lighting, and background, could further enhance instance-level discrimination and robustness.

5.1.3 Summary and Discussion

Comparing category-level and instance-level retrieval highlights the dual nature of fine-tuning's effects on pre-trained VLMs. At the category level, large-scale pre-training already provides robust generalization, while fine-tuning offers incremental but significant improvements, particularly evident in the OpenAI CLIP model. At the instance level, however, fine-tuning proves essential: it enables fine-grained discrimination between visually similar samples, resulting in substantial improvements in retrieval accuracy and ranking efficiency. The different responses of OpenAI CLIP and LAION CLIP confirm that the benefit of fine-tuning strongly depends on the scale and diversity of pre-training. Models trained on extremely large datasets, such as LAION, show smaller relative improvements but still benefit from specialization. This suggests a trade-off between broad generalization and domain-specific adaptation. Overall, the results show that fine-tuning is a necessary step when aiming for high-accuracy retrieval in semantically dense contexts. Future work should explore strategies such as hard negative mining, supervised

contrastive losses with multiple positives, and expanding domain-specific datasets to further strengthen instance-level performance while maintaining category-level generalization.

6. CONCLUSIONS AND FUTURE DEVELOPMENTS

This chapter summarizes the main contributions and findings of this work, highlighting its relevance and limitations. It also outlines possible directions for future research and discusses the practical implications of the proposed approach in real-world scenarios.

This work addresses a complex marine text-to-image retrieval task, with a particular focus on turtle recognition in images from above. The main contribution consists of the construction of a specialized dataset, obtained through an accurate image cropping process based on bounding box annotations and the automatic generation, followed by manual review, of captions using image captioning techniques and language models. An efficient fine-tuning system was implemented on two different pre-trained versions of CLIP (laion/CLIP-ViT-B-32-laion2B-s34B-b79K and openai/clip-vit-base-patch32), leveraging a combination of loss functions: CLIP loss and UniCL, in order to handle the presence of multiple positive examples within the batch without compromising intra-class variability. Furthermore, an evaluation protocol was defined that includes both category-level retrieval metrics (Recall@K), explained in 4.3.1, and a more complex variant of exact matching, focused on the class of interest, explained in 4.3.2. The results obtained confirm the validity of the proposed framework, highlighting good discriminatory capabilities of the model and significant improvements, especially in the more complex scenario of instance-level recognition.

A particularly relevant aspect that emerged during the development of this work concerns the complexity of designing highly specialized and balanced datasets for text-to-image retrieval tasks in real-world applications. The quality and consistency of textual annotations are a key factor in the success of the models, especially when working with images from above characterized by environmental variability, lighting conditions, and often challenging resolution. In this context, the combination of automatic generation and manual review has proven to be an

effective strategy for mitigating the risk of semantic ambiguity, but it requires significant human resources and time.
Furthermore, although the results show an improvement over baseline approaches, some difficulties remain in distinguishing with fine precision between visually similar instances or in conditions of spatial overlap, a frequent phenomenon in marine imagery. This highlights the need to explore further multimodal representation methodologies, for example, integrating temporal information to increase the system's robustness.

6.1 Future developments

Future developments of the project will focus primarily on improving the quality and semantic granularity of the data. A crucial first step will be the enrichment and systematic revision of captions to ensure more precise and less ambiguous image descriptions. At the same time, it will be important to introduce a cataloging of turtles by species, allowing the model to distinguish between biologically relevant subclasses.
A further extension will involve the controlled expansion of the dataset with new categories, such as **marine debris**, **sea surface**, and **dolphins**, which will no longer be treated solely as distractors, but as context-relevant semantic classes. This will allow for the development of a more comprehensive system capable of supporting broader analyses of the marine ecosystem.
From a **methodological perspective**, it is interesting to explore the integration of more recent contrastive learning techniques and next-generation multimodal models that can better handle the complexity and semantic richness of annotations, as well as the adoption of deeper multimodal architectures dedicated to these types of datasets. Furthermore, it will be possible to evaluate the effectiveness of targeted data augmentation strategies and transfer learning from related domains to improve generalization and robustness. In this context, the system could also evolve towards **multimodal integration** beyond images and text, by incorporating additional modalities such as **speech**, which would open new interaction possibilities in real-world scenarios.
From the perspective of dataset enrichment, another promising direction is the use

of **generative models** to synthetically expand the available data, for example, by generating aerial-view images or rare configurations that are underrepresented in the current collection. This would increase both the diversity and balance of the training distribution.
Regarding the **training objective**, it will be valuable to investigate weighted formulations of the loss function, assigning higher importance to **exact matching pairs**, as well as experimenting with alternative strategies for combining loss components, beyond the standard averaging scheme. Such refinements could lead to a more nuanced optimization process and improved retrieval performance.
From an **application perspective**, we plan to extend the system's use to real-world operational scenarios, in collaboration with oceanographic research and marine conservation organizations. One possible direction is to create user-friendly interfaces that allow marine biologists and researchers to directly query large archives of images to monitor environmental dynamics and protected species populations.

6.2 Practical implications

The work carried out not only provides a methodological contribution to the field of *text-to-image retrieval* but also has significant practical implications for the monitoring and protection of marine ecosystems. Accurate identification of sea turtles represents a concrete support for research and conservation activities, enabling automated and scalable population analysis based on images from above. This technology is fully aligned with the objectives of the 2030 Agenda, particularly Goal 14 *"Life Below Water"*, which promotes the conservation and sustainable use of oceans and marine resources. The integration of intelligent recognition and cataloging systems can help identify at-risk areas, monitor protected species, and assess the impact of human activities on marine habitats, providing tools to support research activities and organizations dedicated to environmental protection.
It is also important to emphasize the potential social and educational impact of these technologies, which can increase public awareness of environmental issues and stimulate citizen science processes through participatory data collection

platforms. However, it is also essential to consider the ethical and privacy implications associated with the use of data and AI, ensuring transparency, accuracy, and security in the adoption of these systems. In conclusion, this work represents an important contribution to the advancement of text-to-image retrieval techniques in complex application contexts, paving the way for further interdisciplinary developments that bring together AI, marine sciences, and environmental conservation.

7. ACKNOWLEDGMENTS

I would like to thank Professor Antonino Staiano, my thesis advisor, for his scientific guidance and availability throughout the project. I also thank Dr. Alessio Pierluigi Placitelli, my co-supervisor, for his essential technical and methodological support and valuable suggestions.

BIBLIOGRAPHY

[1] Lukáš Adam et al. "SeaTurtleID2022: A Long-Span Dataset for Reliable Sea Turtle Re-Identification". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 7146–7156.

[2] Alberto Baldrati et al. "Effective conditioned and composed image retrieval combining clip-based features". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 21466–21474.

[3] Romain Beaumont. "LAION-5B: A new era of open large-scale multi-modal datasets". In: *Laion. ai* (2022).

[4] Sara Beery, Dan Morris, and Siyu Yang. "Efficient pipeline for camera trap image review". In: *arXiv preprint arXiv:1907.06772* (2019).

[5] A. Bogisich et al. "Neural Networks for Marine Turtle Conservation: A Review". In: *Frontiers in Artificial Intelligence* 6 (2023), p. 1124090.

[6] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[7] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG]. URL: https://arxiv.org/abs/2002.05709.

[8] CITES Secretariat. *Appendices I, II and III*. Accessed: 2023-10-01. 2023. URL: https://cites.org/eng/app/appendices.php.

[9] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[10] Ritendra Datta et al. "Image retrieval: Ideas, influences, and trends of the new age". In: *ACM Computing Surveys (Csur)* 40.2 (2008), pp. 1–60.

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

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

[13] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[14] A. Dunstan et al. "Automated Facial Recognition for Sea Turtle Photo-ID". In: *Journal of Experimental Marine Biology and Ecology* 546 (2022), p. 151665.

[15] Fartash Faghri et al. *VSE++: Improving Visual-Semantic Embeddings with Hard Negatives*. 2018. arXiv: 1707.05612 [cs.LG]. URL: https://arxiv.org/abs/1707.05612.

[16] Robert M French. "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[17] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. "An algorithm for finding best matches in logarithmic expected time". In: *ACM Transactions on Mathematical Software (TOMS)* 3.3 (1977), pp. 209–226.

[18] Michael Gutmann and Aapo Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 297–304. URL: https://proceedings.mlr.press/v9/gutmann10a.html.

[19] Raia Hadsell, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[20] L.A. Hawkes et al. "Climate Change and Marine Turtles: Impacts on Sex Ratios and Nesting Sites". In: *Endangered Species Research* 7 (2009), pp. 137–144.

[21] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2020. arXiv: 1911.05722 [cs.CV]. URL: https://arxiv.org/abs/1911.05722.

[22] M.A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28. DOI: 10.1109/5254.708428.

[23] Geoffrey Hinton. *Lecture 6e: RMSProp: Divide the gradient by a running average of its recent magnitude*. CSC321: Neural Networks for Machine Learning, University of Toronto. Slide URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. 2012.

[24] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[25] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685.

[26] Yanhua Huang. "Deep Q-networks". In: *Deep reinforcement learning: fundamentals, research and applications*. Springer, 2020, pp. 135–160.

[27] Samuel Humeau et al. "Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring". In: *arXiv preprint arXiv:1905.01969* (2019).

[28] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[29] IUCN Red List. *Marine Turtle Species Assessments*. Accessed: 2023-10-01. 2023. URL: https://www.iucnredlist.org/.

[30] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Transactions on Information Systems (TOIS)* 20.4 (2002), pp. 422–446.

[31] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* 7.3 (2019), pp. 535–547.

[32] Norman P Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, pp. 1–12.

[33] Prannay Khosla et al. "Supervised contrastive learning". In: *Advances in neural information processing systems* 33 (2020), pp. 18661–18673.

[34] Katerina Kikaki et al. "Detecting Marine pollutants and Sea Surface features with Deep learning in Sentinel-2 imagery". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 210 (2024), pp. 39–54. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2024.02.017. URL: https://www.sciencedirect.com/science/article/pii/S0924271624000625.

[35] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[36] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. "Latent retrieval for weakly supervised open domain question answering". In: *arXiv preprint arXiv:1906.00300* (2019).

[37] Junnan Li et al. *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. 2023. arXiv: 2301.12597 [cs.CV]. URL: https://arxiv.org/abs/2301.12597.

[38] Junnan Li et al. "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation". In: *International conference on machine learning*. PMLR. 2022, pp. 12888–12900.

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

[40] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

[41] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983* (2016).

[42] Jiasen Lu et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks". In: *Advances in neural information processing systems* 32 (2019).

[43] Frederic Maire, Luis Mejias Alvarez, and Amanda Hodgson. "Automating marine mammal detection in aerial images captured during wildlife surveys: a deep learning approach". In: *Australasian Joint Conference on Artificial Intelligence*. Springer. 2015, pp. 379–385.

[44] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing, 2008.

[45] Anqi Mao, Mehryar Mohri, and Yutao Zhong. *Cross-Entropy Loss Functions: Theoretical Analysis and Applications*. 2023. arXiv: 2304.07288 [cs.LG]. URL: https://arxiv.org/abs/2304.07288.

[46] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[47] Ron Mokady, Amir Hertz, and Amit H. Bermano. *ClipCap: CLIP Prefix for Image Captioning*. 2021. arXiv: 2111.09734 [cs.CV]. URL: https://arxiv.org/abs/2111.09734.

[48] Mohamed Moursi, Norbert Wehn, and Bilal Hammoud. *Smart Environmental Monitoring of Marine Pollution using Edge AI*. 2025. arXiv: 2504.21759 [eess.SP]. URL: https://arxiv.org/abs/2504.21759.

[49] United Nations. *Transforming our world: The 2030 Agenda for Sustainable Development*. https://sdgs.un.org/2030agenda. 2015.

[50] Jianmo Ni et al. "Large dual encoders are generalizable retrievers". In: *arXiv preprint arXiv:2112.07899* (2021).

[51] Jia Ning et al. "The Diversity of Artificial Intelligence Applications in Marine Pollution: A Systematic Literature Review". In: *Journal of Marine Science and Engineering* 12.7 (2024). ISSN: 2077-1312. DOI: 10.3390/jmse12071181. URL: https://www.mdpi.com/2077-1312/12/7/1181.

[52] Mohammad Sadegh Norouzzadeh et al. "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning". In: *Proceedings of the National Academy of Sciences* 115.25 (2018), E5716–E5725.

[53] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE]. URL: https://arxiv.org/abs/1511.08458.

[54] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17. DOI: 10.1016/0041-5553(64)90137-5.

[55] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

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

[57] Rajat Raina, Anand Madhavan, and Andrew Y Ng. "Large-scale deep unsupervised learning using graphics processors". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 873–880.

[58] Nikhil Rasiwasia et al. "A new approach to cross-modal multimedia retrieval". In: *Proceedings of the 18th ACM international conference on Multimedia*. 2010, pp. 251–260.

[59] Scott Reed et al. *Learning Deep Representations of Fine-grained Visual Descriptions*. 2016. arXiv: 1605.05395 [cs.CV]. URL: https://arxiv.org/abs/1605.05395.

[60] A.F. Rees et al. "The Challenges of Monitoring Marine Turtle Populations". In: *Biological Conservation* 227 (2018), pp. 156–166.

[61] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[62] David E Rumelhart et al. "Backpropagation: The basic theory". In: *Backpropagation*. Psychology Press, 2013, pp. 1–34.

[63] G. Schofield et al. "Image-Based Monitoring in Marine Turtle Research: A Review". In: *Frontiers in Marine Science* 7 (2020), p. 573.

[64] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[65] Christoph Schuhmann et al. *LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs*. 2021. arXiv: 2111.02114 [cs.CV]. URL: https://arxiv.org/abs/2111.02114.

[66] Q. Schuyler et al. "Risk Analysis Reveals Global Hotspots for Marine Debris Ingestion by Sea Turtles". In: *Global Change Biology* 22.2 (2016), pp. 567–576.

[67] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units". In: *arXiv preprint arXiv:1508.07909* (2015).

[68] Junyi Shen. "Latent class model or mixed logit model? A comparison by transport mode choice data". In: *The Applied Economics of Transport*. Routledge, 2014, pp. 127–136.

[69] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. URL: http://dx.doi.org/10.1016/j.physd.2019.132306.

[70] Shoaib Ahmed Siddiqui et al. "Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data". In: *ICES Journal of Marine Science* 75.1 (2018), pp. 374–389.

[71] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[72] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for modern deep learning research". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 09. 2020, pp. 13693–13696.

[73] Richard S Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems* 12 (1999).

[74] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018.

[75] H. Sykes et al. "Drone-Based Monitoring of Marine Turtle Nests in Remote Beaches". In: *Marine Ecology Progress Series* 673 (2021), pp. 1–15.

[76] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971.

[77] UNEP. *Marine Turtles and Coastal Development*. United Nations Environment Programme, 2021. URL: https://www.unep.org/resources/report/.

[78] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2019. arXiv: 1807.03748 [cs.LG]. URL: https://arxiv.org/abs/1807.03748.

[79] Vladimir Vapnik. "Principles of risk minimization for learning theory". In: *Advances in neural information processing systems* 4 (1991).

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

[81] B.P. Wallace et al. "Global Patterns of Marine Turtle Bycatch". In: *Conservation Letters* 6.1 (2013), pp. 3–22.

[82] Qi Wang et al. "A comprehensive survey of loss functions in machine learning". In: *Annals of Data Science* 9.2 (2022), pp. 187–212.

[83] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3 (1992), pp. 279–292.

[84] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 2.1 (1987). Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists, pp. 37–52. ISSN: 0169-7439. DOI: https://doi.org/10.1016/0169-7439(87)80084-9. URL: https://www.sciencedirect.com/science/article/pii/0169743987800849.

[85] Jianwei Yang et al. "Unified Contrastive Learning in Image-Text-Label Space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 19163–19173.

[86] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[87] Peter Young et al. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". In: *Transactions of the Association for Computational Linguistics* 2 (Feb. 2014), pp. 67–78. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00166. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00166/1566848/tacl_a_00166.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00166.

[88] Xiaohua Zhai et al. *LiT: Zero-Shot Transfer with Locked-image text Tuning*. 2022. arXiv: 2111.07991 [cs.CV]. URL: https://arxiv.org/abs/2111.07991.