



UNIVERSITY OF ECHAHID HAMMA LAKHDAR - EL OUED
FACULTY OF EXACT SCIENCES



Computer Science Department

Thesis

submitted in partial fulfilment of the requirements for the degree of

ACADEMIC MASTER

Field: **Mathematics and Computer Science**

Option: **Computer Science**

Specialty: **Artificial Intelligence and Data Science**

Submitted by :

- **HENKA Ines**
- **GUENOVA Hadil**

Title

**Image Captioning using DINOv2 Vision
Transformer (ViT) model**

Defended on June 06th 2026

Examination Committee :

M. Medileh Saci	Professor	Chair
M. Hamoud Meriem	MCA	Examinator
M. Guia Sana Sahar	MCB	Supervisor

Academic year: 2025/2026

Dedication

First and foremost, we thank Almighty Allah for granting us the strength, patience, and guidance to complete this work.

We dedicate this work with sincere love and gratitude to our parents.

To our mothers, for their endless care, kindness, and constant prayers that have always accompanied us throughout our journey.

To our fathers, for their sacrifices, encouragement, and unwavering support, and for always motivating us to persevere and pursue our academic goals.

To our families, each one by name, for their love, trust, and belief in us.

To our friends and loved ones, for their support, encouragement, and for standing by us through every challenge.

To everyone who contributed to our success, near or far, and to all those who offered us a helping hand along the way.

This work is dedicated to all of you with our deepest appreciation and gratitude.

Acknowledgments

We begin by expressing our deepest gratitude to Almighty Allah, whose blessings, guidance, and mercy enabled us to successfully complete this work.

We would like to convey our sincere appreciation to our supervisor Dr. Guia Sana Sahar for her invaluable support, constructive feedback, and continuous encouragement throughout every stage of this project. Her expertise and guidance have been essential to the realization of this work.

We are also grateful to all the professors and academic staff who have contributed to our education during our university studies. Their dedication, knowledge, and commitment to teaching have provided us with the foundation necessary to accomplish this project.

Our sincere thanks are extended to the members of the examination committee for their willingness to review and evaluate this dissertation. We highly appreciate the time, effort, and expertise they devote to this task.

We would also like to acknowledge all those who have supported us in one way or another during the preparation of this work. Their assistance and encouragement have been greatly valued.

Finally, we recognize the determination, perseverance, and effort that we invested throughout this academic journey. Completing this work represents the result of continuous learning, commitment, and dedication.

Abstract

Image captioning is a multimodal task that combines Computer Vision and Natural Language Processing to automatically generate textual descriptions of images. Unlike image classification or object detection, image captioning requires understanding not only the objects present in a scene, but also their attributes, relationships, and overall context in order to produce coherent and meaningful natural language descriptions.

In this thesis, we propose an image captioning framework that combines the DINOv2 Vision Transformer with the GPT-2 language model. To effectively bridge the gap between visual and textual representations, a Visual Query Adapter is introduced to transform visual features into a representation suitable for language generation through cross-attention mechanisms. In addition, a progressive two-stage fine-tuning strategy is employed to improve cross-modal alignment while preserving pretrained knowledge.

The proposed framework was trained and evaluated on the MS COCO dataset. Experimental results demonstrate the effectiveness of the approach. In particular, the high METEOR score highlights the model's ability to generate semantically meaningful and linguistically flexible captions.

Beyond quantitative evaluation, this work explores a practical application of image captioning through Caption Quest, a gamified educational prototype that leverages AI generated captions to support English language learning. This application demonstrates how modern vision-language models can be integrated into interactive learning environments.

Overall, the results show that combining a self-supervised visual encoder with a pretrained language model through a lightweight adaptation module provides an effective and computationally efficient solution for image captioning while opening opportunities for real world educational applications.

Keywords : Image Captioning, DINOv2, Vision Transformer, GPT-2, Visual Query Adapter, Cross-Attention, Fine-Tuning, Multimodal Learning.

Résumé

La génération automatique de légendes d'images est une tâche multimodale qui combine la Vision par Ordinateur et le Traitement Automatique du Langage Naturel afin de produire automatiquement des descriptions textuelles d'images. Contrairement à la classification d'images ou à la détection d'objets, cette tâche nécessite non seulement l'identification des objets présents dans une scène, mais également la compréhension de leurs attributs, de leurs relations et du contexte global afin de générer des descriptions cohérentes et pertinentes en langage naturel.

Dans ce mémoire, nous proposons un cadre de génération de légendes d'images combinant le Vision Transformer DINOv2 et le modèle de langage GPT-2. Afin de réduire efficacement l'écart entre les représentations visuelles et textuelles, un Visual Query Adapter est introduit pour transformer les caractéristiques visuelles en une représentation adaptée à la génération de texte à travers des mécanismes de cross-attention. En complément, une stratégie progressive de fine-tuning en deux étapes est mise en œuvre afin d'améliorer l'alignement entre les modalités visuelle et textuelle tout en préservant les connaissances acquises lors du préentraînement.

Le cadre proposé a été entraîné et évalué sur la base de données MS COCO. Les résultats expérimentaux démontrent l'efficacité de l'approche. En particulier, le score élevé obtenu sur la métrique METEOR met en évidence la capacité du modèle à générer des légendes sémantiquement pertinentes et linguistiquement flexibles.

Au-delà de l'évaluation quantitative, ce travail explore une application pratique de la génération de légendes d'images à travers Caption Quest, un prototype éducatif gamifié exploitant des légendes générées par intelligence artificielle pour soutenir l'apprentissage de l'anglais. Cette application démontre comment les modèles modernes vision-langage peuvent être intégrés dans des environnements d'apprentissage interactifs.

Dans l'ensemble, les résultats montrent que la combinaison d'un encodeur visuel auto-supervisé avec un modèle de langage préentraîné, reliés par un module d'adaptation léger, constitue une solution efficace et économiquement viable en termes de calcul pour la génération de légendes d'images, tout en ouvrant de nouvelles perspectives pour des applications éducatives concrètes.

Mots clés : Génération de légendes d'images, DINOv2, Vision Transformer, GPT-2, Visual Query Adapter, Cross-Attention, Fine-Tuning, Apprentissage multimodal.

Contents

Abstract	iv
Résumé	v
List of Figures	ix
List of Tables	x
List of Abbreviations	xi
General Introduction	12
Chapter 1: Overview and Background	14
1.1 Introduction	15
1.2 Natural Language Processing	15
1.2.1 Language Modeling	15
1.2.1.1 Neural Network Language modeling	16
1.2.1.2 Recurrent Neural Networks	16
1.2.1.3 Encoder-Decoder	17
1.2.1.4 The Attention Mechanism	20
1.2.2 Transformer Architecture	22
1.2.2.1 Self-Attention	23
1.2.2.2 Multi-Head Attention	23
1.2.2.3 Position-wise Feed-Forward Networks	24
1.2.2.4 Positional Encoding	25
1.2.2.5 Residual Connections and Layer Normalization	26
1.3 Computer Vision	26
1.3.1 Taxonomy of Computer Vision Tasks	27
1.3.2 Feature Extraction	27
1.3.2.1 CNN	28
1.3.2.2 Pre-trained CNN Models	29
1.3.3 Vision Transformer	29
1.4 Conclusion	30
Chapter 2: Related Work	31
2.1 Introduction	32
2.2 Image Captioning	32
2.2.1 Formal Definition of Image Captioning	32
2.2.1.1 Image Captioning as Image to Text Generation	32
2.2.1.2 Multimodal Foundations	33
2.2.2 Core Challenges in Image Captioning	34
2.2.3 Real World Applications	35
2.3 Related Work	35

2.3.1	Traditional Methods	36
2.3.1.1	Retrieval-Based Methods	36
2.3.1.2	Template-Based Methods	36
2.3.1.3	Composition-Based Methods	36
2.3.2	State of the Art Models for Image Captioning	37
2.3.2.1	Neural-Based Methods	37
2.3.2.2	Limitations of Early Neural Models	38
2.3.2.3	Transformer-Based Methods	39
2.3.2.4	Synthesis and Research Gap	40
2.4	Benchmarking Datasets and Evaluation Metrics	41
2.4.1	Benchmark Datasets	41
2.4.2	Evaluation Metrics	42
2.5	Conclusion	43
Chapter 3:	Proposed Image Captioning Model	44
3.1	Introduction	45
3.2	Overview of the Proposed Framework	45
3.3	Visual Feature Extraction: DINOv2	45
3.3.1	Vision Transformer Backbone and Token Layout	46
3.3.2	Self-Supervised Student Teacher Distillation	47
3.3.3	DINOv2 Pretraining Strategy and Data Curation	48
3.4	Cross Modal Adaptation: Visual Query Adapter	49
3.4.1	Architecture and Mathematical Formulation	49
3.4.1.1	Linear Dimension Projection	49
3.4.1.2	Multi-Head Cross-Attention	50
3.4.1.3	Feed-Forward Network Refinement	50
3.5	Language Generation Module: GPT-2 Decoder	51
3.5.1	Pre-trained Architecture	51
3.5.2	Cross-Attention for Visual Conditioning	52
3.6	Proposed Training Strategy	53
3.6.1	Theoretical Motivations for Two Stage Training	53
3.6.2	Two Stage Fine-Tuning Strategy	53
3.6.2.1	Stage A: Adapter Training with Frozen Visual Encoder	53
3.6.2.2	Stage B: Selective Fine-Tuning of the Visual Encoder	54
3.6.3	Loss Function	55
3.7	Conclusion	56
Chapter 4:	Implementation and Experimental Setup	57
4.1	Introduction	58
4.2	Experimental Environment	58
4.2.1	Platform and Execution Environment	58
4.2.2	Hardware Configuration	58
4.2.3	Software Environment	59
4.2.4	Libraries and Frameworks	59
4.3	Datasets	61
4.3.1	Dataset Description	61
4.3.2	Dataset File Structure	61
4.3.3	Data Splitting Strategy	62
4.3.3.1	Image Level Splitting	62
4.3.3.2	Splitting Procedure	62
4.3.3.3	Dataset Specific Splits	62
4.3.3.4	Final Split Summary	63

4.3.4	Data Preprocessing	64
4.3.4.1	Image Preprocessing	64
4.3.4.2	DINOv2 Compatibility	66
4.3.4.3	Text Preprocessing	66
4.4	Implementation Details	67
4.4.1	Training Configuration	67
4.4.2	Fine-Tuning Strategy	68
4.4.3	Forward Propagation Pipeline	68
4.4.4	Model Architecture Implementation	68
4.4.5	Evaluation Metrics	69
4.4.6	Inference and Caption Generation	70
4.5	Training Process and Experimental Analysis	71
4.5.1	Training Workflow	71
4.5.2	Loss Monitoring and Training Curves	71
4.5.3	Checkpointing and Model Selection	73
4.6	Experimental Results	73
4.6.1	Quantitative Results	74
4.6.2	Qualitative Results	74
4.6.3	Comparison with State of the Art	75
4.6.4	Discussion	76
4.7	Gamification of Learning Using Image Captioning	76
4.7.1	The Gameplay Loop	77
4.8	Conclusion	78

List of Figures

1.1	General NLP Processing Pipeline	15
1.2	Architecture of a Neural Language Model	16
1.3	Unfolded Recurrent Neural Network Architecture	17
1.4	Encoder-Decoder for Machine Translation	18
1.5	Encoder Layer Architecture	18
1.6	Decoder Layer Architecture	19
1.7	Encoder-Decoder for Question Answering	19
1.8	Encoder-Decoder Architecture for Text Summarization	20
1.9	The Original Transformer Model Architecture	22
1.10	Scaled Dot-Product Attention Mechanism	23
1.11	Multi-Head Attention Architecture with Parallel Attention Layers	24
1.12	Position-wise Feed-Forward Network in a Transformer the first Encoder Layer	25
1.13	Sinusoidal Positional Encoding Matrix	26
1.14	Taxonomy of Computer Vision Tasks	27
1.15	Standard Architectural Flow of a Convolutional Neural Network	28
1.16	Conceptual Framework of Transfer Learning via Knowledge Transfer	29
1.17	The Vision Transformer Architecture	30
2.1	Illustrates examples of the image to text mapping in image captioning	32
2.2	General encoder-decoder architecture for image captioning	33
2.3	The Standard Encoder-Decoder Architecture for Image Captioning	33
2.4	Advanced Multimodal Pipeline with Object Proposals	34
3.1	Overall pipeline of the proposed image captioning framework.	45
3.2	Vision Transformer patch extraction and visual token generation process for DINOv2-Base	47
3.3	Self-supervised student-teacher distillation framework	48
3.4	DINOv2 data curation and representation scaling pipeline	49
3.5	Detailed architecture and information flow of the Visual Query Adapter	51
3.6	Architecture of the modified GPT-2 decoder layer with visual cross-attention conditioning	52
3.7	Stage A training architecture	54
3.8	Stage B training architecture	55
4.1	Kaggle logo	58
4.2	Python logo	59
4.3	PyTorch logo	59
4.4	OpenCV logo	60
4.5	NumPy logo	60
4.6	Matplotlib logo	60
4.7	Curves Across Fine-Tuning Stage A	72
4.8	Curves Across Fine-Tuning Stage B	72
4.9	a comparison between two Curves Across Fine-Tuning Stages	73
4.10	Examples of Correct Caption Generation	75
4.11	Example of Caption Quest Application Interface	77
4.12	Example of Caption Quest Application Interface	78
4.13	Example of Caption Quest Application Interface	78

List of Tables

1.1	Attention Mechanism Variants	22
1.2	Hierarchical Feature Representation in Deep Networks	28
1.3	Architectural Specifications of ViT Variants	30
2.1	Summary of neural-based image captioning models	39
2.2	Summary of transformer-based components and models relevant to image captioning. . .	40
2.3	Comparison of widely used image captioning datasets.	42
2.4	Overview of automated evaluation metrics used in image captioning.	43
4.1	MS COCO Dataset Statistics After Preprocessing	61
4.2	Summary of Dataset Splitting Configurations	64
4.3	Summary of Image-Level Split (Unique Images)	64
4.4	Summary of Caption-Level Samples (After Expansion)	64
4.5	Training Hyperparameters Configuration	67
4.6	Trainable Components per Fine-Tuning Stage	68
4.7	Evaluation Metrics Used in Experiments	70
4.8	Inference Configuration for Caption Generation	70
4.9	Stage A vs Stage B Performance Comparison	74
4.10	Final Test Results on MS COCO	74
4.11	Comparison with state of the art image captioning models on MS COCO.	75

List of Abbreviations

Abbreviation	Definition
NLP	Natural Language Processing
RNNs	Recurrent Neural Networks
NNLM	Neural Network Language Modeling
QA	Question Answering
FFN	Feed-Forward Network
CV	Computer Vision
CNNs	Convolutional Neural Networks
SIFT	Scale-Invariant Feature Transform
HOG	Histogram of Oriented Gradients
ReLU	Rectified Linear Unit
ViT	Vision Transformer
AI	Artificial Intelligence
alt-text	Alternative Text
SEO	Search Engine Optimization
MatMul	Matrix Multiplication
MSA	Multi-Head Self-Attention
MHCA	Multi-Head Cross-Attention
GPU	Graphics Processing Unit

General Introduction

Introduction

Artificial Intelligence (AI) has witnessed remarkable progress in recent years, enabling systems capable of processing and understanding multiple data modalities such as images, text, and audio. Within this broad field, Computer Vision (CV) focuses on enabling machines to interpret and analyze visual content, while Natural Language Processing (NLP) aims to understand and generate human language.

The growing need for systems that can jointly understand vision and language has led to the emergence of multimodal AI. One of the most representative tasks in this domain is Image Captioning, which establishes a direct link between visual perception and natural language generation.

Image Captioning refers to the task of automatically generating a natural language description of an image. Unlike image classification, which assigns a single label, or object detection, which localizes objects, image captioning requires a deeper semantic understanding of the scene. The model must identify objects, capture their attributes, model relationships between entities, and express this information in a coherent and grammatically correct sentence.

Despite significant advances, image captioning remains a challenging problem. Many existing models successfully recognize dominant objects but struggle to produce detailed, context-aware, and linguistically rich descriptions, particularly in complex scenes involving multiple interacting elements. A key challenge lies in bridging the semantic gap between visual and textual representations, as these modalities are inherently different in structure and encoding.

Recent developments in Vision Transformers and large language models have opened new opportunities for multimodal learning. In particular, DINOv2 provides strong self-supervised visual representations, while GPT-2 offers powerful language generation capabilities. However, effectively aligning these pretrained components for image captioning remains an open research challenge.

Problem Statement

Despite the availability of powerful pretrained vision and language models, several challenges still limit the performance of image captioning systems:

- **Cross-modal alignment:** Visual and textual representations lie in different embedding spaces, requiring effective adaptation mechanisms.
- **Catastrophic forgetting:** Full fine-tuning of pretrained models may degrade general purpose representations.
- **Fine-grained scene understanding:** Many models fail to capture secondary objects, attributes, and complex relationships in cluttered scenes.

-
- **Computational cost:** End to end training of large multimodal architectures is resource-intensive.

These limitations highlight the need for efficient architectures that balance performance, adaptability, and computational feasibility.

Hypothesis and Proposed Approach

This thesis is based on the hypothesis that a modular architecture combining a self-supervised Vision Transformer with a pretrained language model can achieve strong captioning performance while maintaining computational efficiency.

The proposed framework consists of three main components:

- A **DINOv2 Vision Transformer** for robust visual feature extraction.
- A **Visual Query Adapter** that bridges the semantic gap between vision and language using learned projections and cross-attention.
- A **GPT-2 decoder** that generates natural language captions conditioned on visual information.

To improve training stability and preserve pretrained knowledge, a progressive two-stage fine-tuning strategy is employed, allowing selective adaptation of model components rather than full end to end training.

Practical Application

Beyond the core model, this work introduces **Caption Quest**, a gamified educational application that leverages image captioning to support English language learning. The system enables users to write image descriptions and compare them with AI generated captions, transforming captioning into an interactive learning experience.

Thesis Structure

This thesis is organized as follows :

- **Chapter 1** presents foundational concepts in Computer Vision, Natural Language Processing, Transformers, Vision Transformers, and image captioning.
- **Chapter 2** reviews related work, including traditional and deep learning-based approaches, transformer architectures, datasets, evaluation metrics, and research gaps.
- **Chapter 3** describes the proposed model architecture, including DINOv2, the Visual Query Adapter, GPT-2, and the training strategy.
- **Chapter 4** presents the implementation details, experimental setup, results analysis, comparisons with state-of-the-art methods, and the Caption Quest application.

This thesis investigates the effectiveness of combining DINOv2 with GPT-2 for image captioning and explores how such multimodal systems can be effectively applied in real-world educational scenarios.

Chapter 1

Overview and Background

1.1 Introduction

Image captioning uses computer vision and natural language processing to describe a photo in a full sentence. Unlike object detection, which only lists what is in the image, captioning needs to understand how objects relate to each other and tell the story of the scene.

Modern captioning depends on deep learning. On the vision side, models like CNNs and Vision Transformers pull out visual details from an image. On the text side, Transformer models turn those details into clear and natural sentences.

This chapter builds the technical background for our work. We start with language modeling, the Encoder-Decoder setup, and the Attention mechanism. Then we see how the Transformer changed the field. Finally, we look at how computer vision moved from old manual feature extraction to modern deep learning methods.

1.2 Natural Language Processing

Natural Language Processing teaches machines to understand and write human language. Today, deep learning models learn directly from text and speech. In image captioning, NLP is very important. The decoder acts like a language model that builds sentences one word at a time. It does not work alone, it is guided by the visual information coming from the encoder. This way, the text sounds natural and also describes the image correctly [1].

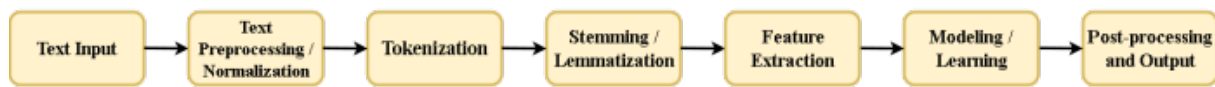


Figure 1.1: General NLP Processing Pipeline

- **Text Input:** The raw text (sentences, documents or captions) that the system will learn from.
- **Text Preprocessing / Normalization:** Cleaning the text by fixing uppercase/lowercase, removing punctuation, and handling noise like URLs or special characters.
- **Tokenization:** Splitting the text into smaller pieces like words or sub words that the model can process.
- **Stemming / Lemmatization:** Reducing words to their root form (e.g., "running" becomes "run") so different forms of the same word are treated the same.
- **Feature Extraction:** Turning the processed text into numbers that a computer can work with.
- **Modeling/Learning:** The model learns patterns, meaning, and relationships in the data using machine learning algorithms.
- **Post-processing and Output:** Turning the model's raw predictions back into clean, readable text.

1.2.1 Language Modeling

Language modeling means guessing how likely a sequence of words is, so machines can understand and produce human language. By predicting the most probable next word given the preceding context, language models understand grammar and meaning in a language. Formally, the probability of a sequence $w_1, \dots, w_T$ is expressed as a product of conditional probabilities:

$$P(w_1, \dots, w_T) = \prod_{t=1}^T P(w_t \mid w_1, \dots, w_{t-1}) \quad (1.1)$$

This fundamental concept serves as the basis for autoregressive language generation, where each word is predicted sequentially based on previous context.

1.2.1.1 Neural Network Language modeling

Neural Network Language Modeling uses neural networks to guess word sequences by learning word meanings as numbers (vectors). Unlike traditional statistical models, neural approaches capture semantic similarities and generalize better to unseen data by learning word embeddings and how to predict the next word at the same time. These models compute the probability of the next word given the previous context and are commonly implemented using architectures such as feedforward networks, RNNs, and Transformers [2].

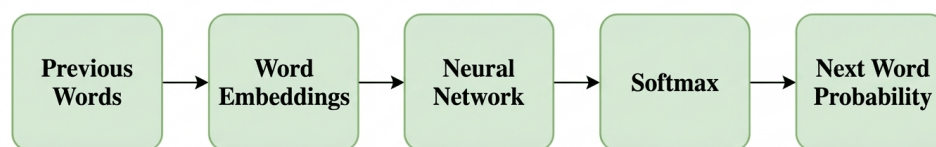


Figure 1.2: Architecture of a Neural Language Model

- **Previous Words:** The words that come before the current one. They give context so the model knows what has been said so far.
- **Word Embeddings:** Numbers that represent each word. They capture the meaning of words and how they relate to each other (e.g., "king" and "queen" are close in meaning).
- **Neural Network:** A set of layers that process these word numbers to find patterns in the language.
- **Softmax:** A function that turns the model's raw scores into probabilities. This means the model's guesses for all possible next words add up to 100%.
- **Next Word Probability:** The model's final guess for the most likely next word, based on everything it has seen so far.

1.2.1.2 Recurrent Neural Networks

Recurrent Neural Networks are designed to process sequential data by maintaining a "hidden state" that acts as a form of internal memory. Unlike standard feed-forward networks, they reuse the same parameters throughout a sequence, which allows them to capture temporal dependencies, the way earlier words influence those that follow. By blending the current input with the previous hidden state at every step, the model creates a continuous flow of information, enabling it to predict the next word with the context of everything it has "seen" so far [3].

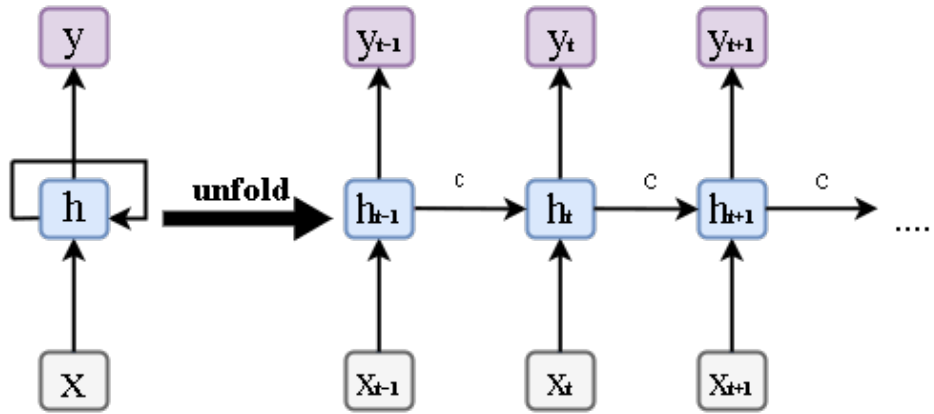


Figure 1.3: Unfolded Recurrent Neural Network Architecture

This diagram 1.3 illustrates the RNN architecture in both its compact form (left) and its temporally unfolded state (right). As the network processes data through time steps $t - 1$, t , and $t + 1$, the hidden states h_t act as a continuous bridge, propagating information forward. This sequential flow allows the model to map the input layer to the output layer while maintaining a temporal memory of previous steps.

1.2.1.3 Encoder-Decoder

The Encoder–Decoder architecture is a neural framework designed for sequence to sequence tasks, where an input sequence is transformed into a corresponding output sequence through two main components[4].

The Encoder:

The encoder processes the input data and extracts meaningful features, converting it into a contextual representation often a set of hidden states or embeddings that captures the semantic information of the input[5].

The Decoder:

The decoder generates the target sequence using the representation produced by the encoder. Acting as a conditional language model, it predicts the output token by token while considering both the encoded context and the previously generated tokens to produce coherent and relevant outputs. [6]

In multimodal tasks such as image captioning, the encoder extracts visual features from the input image, while the decoder generates a natural language description based on these features.

Encoder-Decoder Architecture for Text to Text Generation:

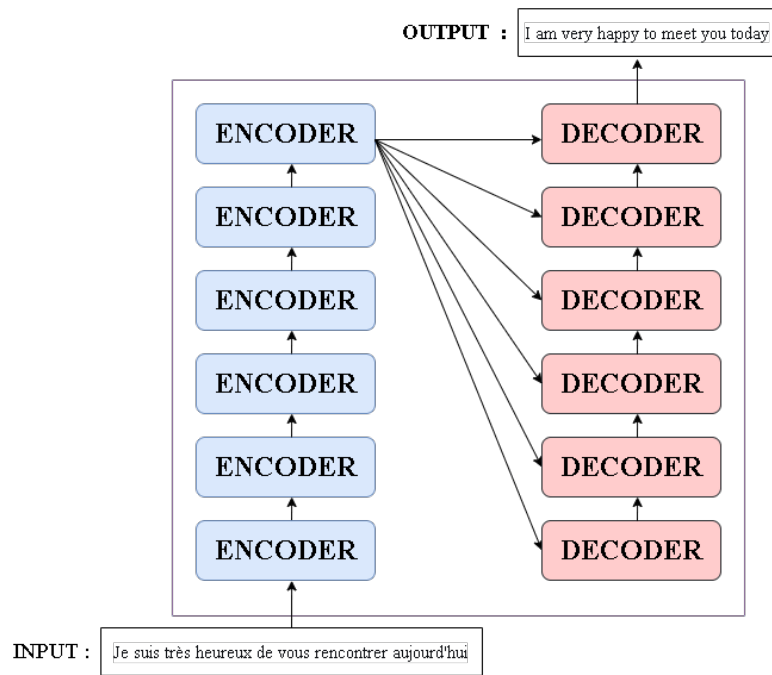


Figure 1.4: Encoder-Decoder for Machine Translation

This figure 1.4 shows the high level workflow of a stacked Encoder–Decoder. The encoder processes the source sentence ("I am very happy to meet you today") through several layers to build a rich representation of its meaning. The decoder then uses this representation to generate the target sentence ("Je suis très heureux de vous rencontrer aujourd'hui") one word at a time. Each word it produces helps predict the next one.

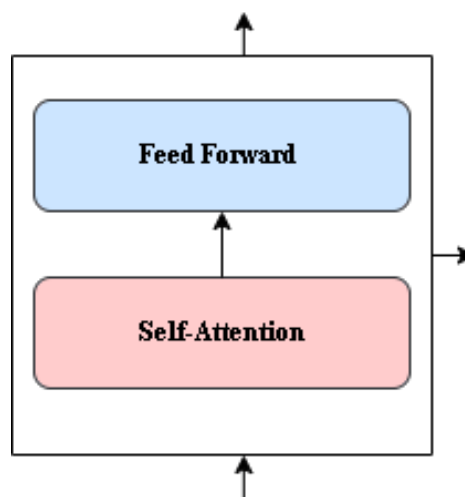


Figure 1.5: Encoder Layer Architecture

This figure 1.5 shows the inside of one encoder block. It has two main parts: self-attention, which looks at how words relate to each other, and a feed-forward network, which adjusts the features to make them more useful.

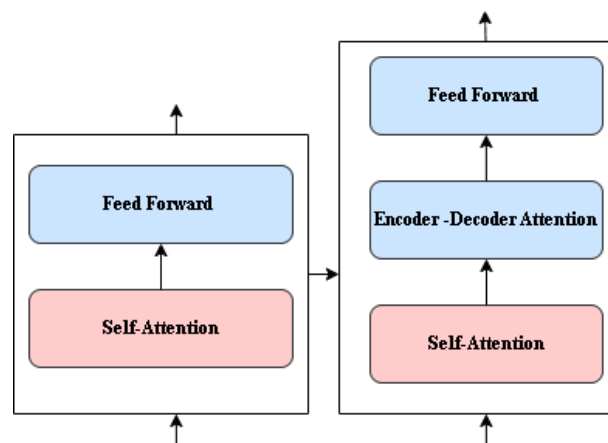


Figure 1.6: Decoder Layer Architecture

This figure 1.6 shows the decoder block. It includes two important components: masked self-attention, which ensures the model only uses previously generated words when predicting the next word, and cross-attention, which lets the decoder access information from the encoder output during text generation.

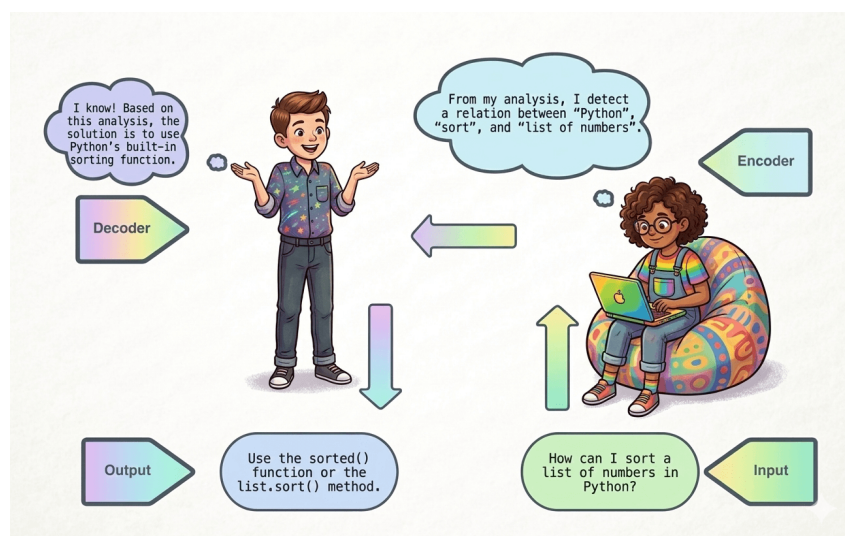


Figure 1.7: Encoder-Decoder for Question Answering

This figure 1.7 shows how the Encoder-Decoder framework can be applied to question answering. First, the encoder reads the input question and analyzes the relationships between key words to understand what is being asked. Then, the decoder uses this information to generate a precise and relevant answer, one word at a time.

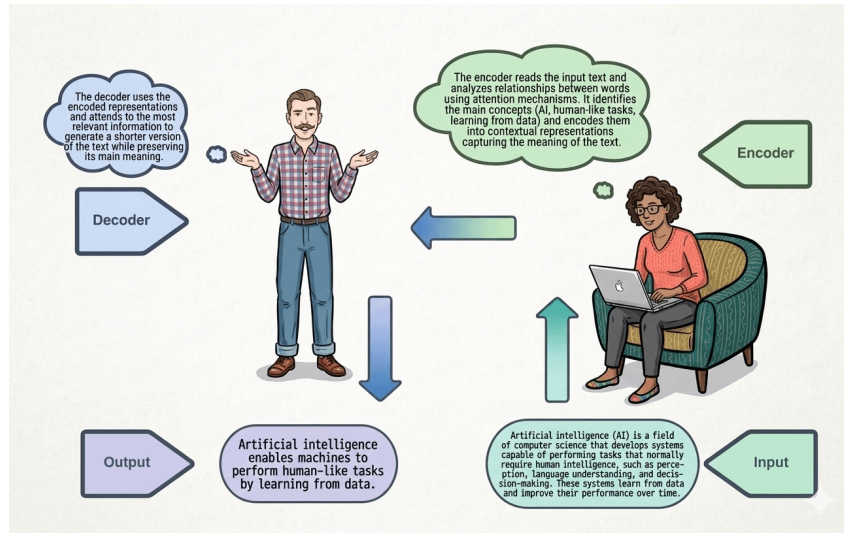


Figure 1.8: Encoder-Decoder Architecture for Text Summarization

This figure 1.8 shows the Encoder–Decoder used for text summarization. The encoder takes a long text and compresses it into a short representation. The decoder then picks the most important parts to create a shorter version that keeps the original meaning.

1.2.1.4 The Attention Mechanism

In early sequence to sequence models, the encoder compressed the entire input into a single vector. This often created a bottleneck, causing the model to lose important information, especially when dealing with long sequences.

The Attention mechanism was introduced to overcome this limitation. Instead of relying on one fixed representation, the decoder can access all encoder hidden states ($h_1, h_2, \dots, h_n$) during each decoding step. It assigns higher weights to the parts of the input that are most relevant to the current prediction.

This allows the model to focus on the most important information when generating each output word, leading to better performance and improved handling of long sequences.[7].

The Mathematical Workflow:

At each decoding step t , the mechanism follows a three-step process to generate a dynamic context vector:

Alignment Score: The model computes a score $e_{t,i}$ between the current decoder hidden state s_t and each encoder hidden state h_i . This represents the compatibility between the input and the potential output:

$$e_{t,i} = \text{score}(s_t, h_i) \quad (1.2)$$

Attention Weights: These scores are normalized using the softmax function to produce a probability distribution $\alpha_{t,i}$, ensuring the weights sum to one:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^n \exp(e_{t,k})} \quad (1.3)$$

Context Vector: A dynamic context vector c_t is computed as a weighted sum of all encoder hidden states. This vector provides the decoder with a focused "glimpse" of the input:

$$c_t = \sum_{i=1}^n \alpha_{t,i} h_i \quad (1.4)$$

This dynamic context vector provides the decoder with the most relevant information from the input sequence when generating the next token.

Types of Attention Scoring Functions

There are three common ways to compute attention scores:

1. Additive Attention (Bahdanau Attention): This method uses a small feedforward neural network with a nonlinear activation (tanh) to compute the score. It is highly flexible and effective for modeling complex relationships between states of different dimensions, though it is computationally more expensive [8].

The alignment score is computed as:

$$\text{score}(s_t, h_i) = v^T \tanh(W_s s_t + W_h h_i) \quad (1.5)$$

where:

- s_t represents the current hidden state of the decoder
- h_i represents the hidden state of the encoder at position i
- W_s and W_h are learnable weight matrices
- v is a trainable parameter vector
- $\tanh$ introduces non-linearity in the scoring process

2. Multiplicative Attention (Luong Attention): This approach utilizes matrix multiplication to measure interaction. It is more computationally efficient than additive attention and scales well for larger models [9].

The alignment score is computed as:

$$\text{score}(s_t, h_i) = s_t^T W h_i \quad (1.6)$$

where:

- s_t is the decoder hidden state
- h_i is the encoder hidden state
- W is a learnable weight matrix

3. Dot-Product Attention: The simplest form of attention, which directly calculates the similarity between states using a dot product. While extremely efficient and parameter free, it assumes that the encoder and decoder hidden states share the same dimensionality [10].

The alignment score is computed as:

$$\text{score}(s_t, h_i) = s_t^T h_i \quad (1.7)$$

Summary of Key Milestones The following table 1.1 compares the mathematical formulations used to calculate the relevance between different elements in a sequence.

Type	Formula	Description
Additive Attention (Bahdanau)	$\text{score}(s_t, h_i) = v^T \tanh(W_s s_t + W_h h_i)$	Uses a neural network to compute alignment scores.
Multiplicative Attention (Luong)	$\text{score}(s_t, h_i) = s_t^T W h_i$	Uses a weighted dot product for alignment.
Dot-Product Attention	$\text{score}(s_t, h_i) = s_t^T h_i$	Uses a simple dot product for alignment.

Table 1.1: Attention Mechanism Variants

1.2.2 Transformer Architecture

The Transformer redefined sequence to sequence modeling by replacing recurrence with attention mechanisms. Unlike RNNs, which process words one by one, the Transformer works in parallel. This makes training much faster and helps the model capture long range dependencies better.[10].

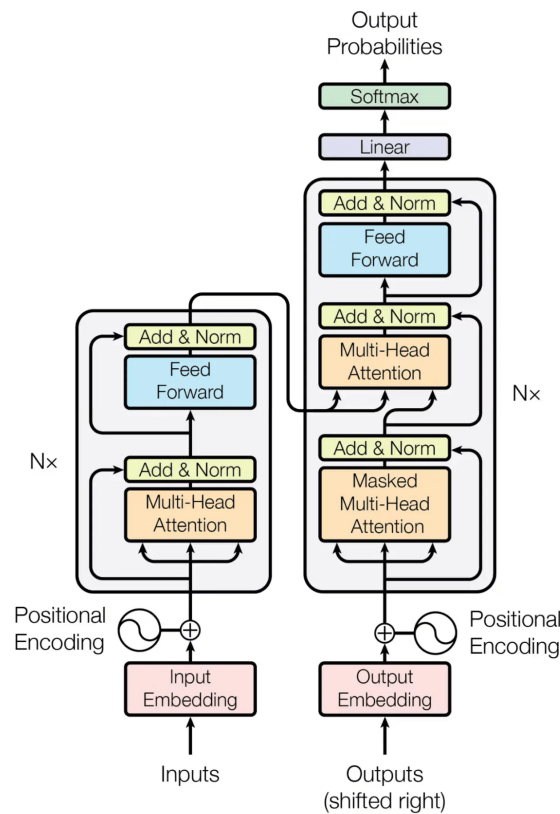


Figure 1.9: The Original Transformer Model Architecture [11]

This figure 1.9 shows the Transformer architecture. It consists of an encoder (left) and a decoder (right). The encoder converts the input into contextual representations using self-attention and feed-forward layers, while the decoder generates the output by attending to both the encoder's representations and the previously generated tokens. Both the encoder and decoder use residual “Add & Norm”

connections to stabilize training and improve gradient flow in deep networks.

1.2.2.1 Self-Attention

Self-attention is the main idea behind the Transformer architecture. It allows each word in a sentence to look at all the other words at the same time in order to understand its meaning in context. This is different from older models that process words one after another [10].

To achieve this, each input token is transformed into three vectors: Query (Q), Key (K), and Value (V) using learned linear layers. The model then computes attention using the following formula [10]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1.8)$$

Here, d_k is the dimension of the Key vectors. The scaling factor $1/\sqrt{d_k}$ —helps keep the values stable during training and prevents very large dot-product values that could affect learning.

The final output gives each word a new representation that combines information from the entire sentence, not just the word itself.

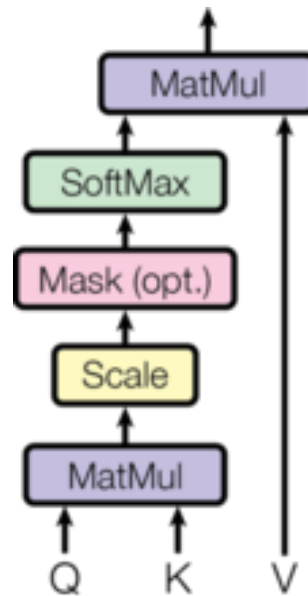


Figure 1.10: Scaled Dot-Product Attention Mechanism [12]

This figure 1.10 illustrates the operations of a single attention head. The Query (Q) and Key (K) vectors are multiplied to compute similarity scores, which are scaled and optionally masked, then normalized via Softmax. These weights are applied to the Value (V) vectors to produce an output that focuses on the most important tokens.

1.2.2.2 Multi-Head Attention

Instead of using a single attention mechanism, multi-head attention performs several attention operations in parallel [10]. This allows the model to focus on different types of relationships in the input at the same time, such as syntactic and semantic information.

The input is first projected into multiple sets of Queries, Keys, and Values. Each set is processed independently using scaled dot product attention, forming multiple attention heads. Each head learns to focus on different aspects of the input.

The outputs of all heads are then concatenated and passed through a final linear layer to produce the final representation.

Formally, the multi-head attention operation is defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (1.9)$$

where each head is computed as :

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (1.10)$$

By combining multiple heads, the model can capture different relationships in the input that a single attention mechanism might miss.

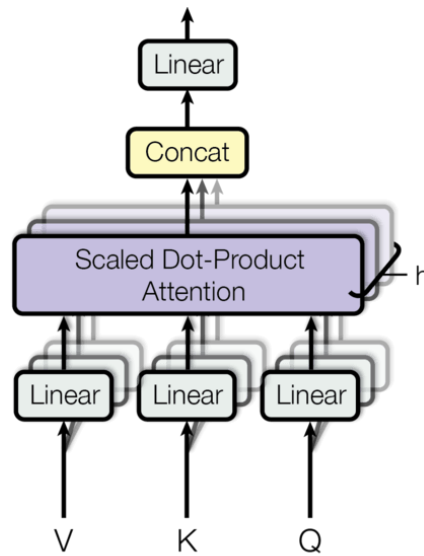


Figure 1.11: Multi-Head Attention Architecture with Parallel Attention Layers [12]

This diagram 1.11 shows multi-head attention. The input goes through several parallel linear layers to create multiple subspaces. Each head computes attention on its own. Then, their outputs are combined and transformed to produce the final result. This allows the model to capture different types of relationships at the same time.

1.2.2.3 Position-wise Feed-Forward Networks

In addition to attention sublayers, each Transformer layer contains a position-wise FFN that applies nonlinear transformations to the representations. While attention mechanisms model dependencies between tokens, the FFN operates independently and identically on each position in the sequence. As illustrated in Figure 1.12, the representation for each token (e.g., z_1, z_2) is processed through an identical neural network to produce a refined output (r_1, r_2) [12].

Formally, this component consists of two linear transformations with a ReLU activation in between:

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}W_1 + b_1)W_2 + b_2 \quad (1.11)$$

where W_1, W_2 are learned weights and b_1, b_2 are biases. This non linear transformation helps the

model learn more complex patterns beyond what attention alone can capture.

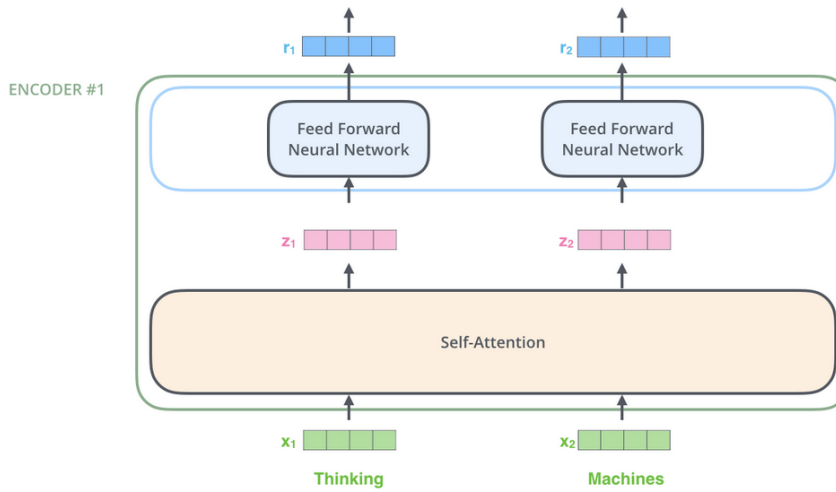


Figure 1.12: Position-wise Feed-Forward Network in a Transformer the first Encoder Layer [13]

This figure 1.12 shows the feed-forward network inside a Transformer encoder layer. After self-attention produces intermediate representations (z_1, z_2) , the same small network processes each one independently. It applies the same transformation to every position, giving refined outputs (r_1, r_2) . The figure highlights that the FFN works on each token separately but in the same way, which helps the model become more powerful after the attention step.

1.2.2.4 Positional Encoding

Unlike RNNs, the Transformer does not naturally capture the order of words because it has no recurrence or convolution. To solve this issue, positional encodings are added to the input embeddings at the beginning of both the encoder and decoder [10].

These encodings are based on sine and cosine functions with different frequencies:

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (1.12)$$

$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (1.13)$$

Here pos represents the position of a word in the sequence, and i refers to the embedding dimension.

This approach allows the model to understand both absolute and relative positions of words. It also helps the Transformer generalize better to sequences longer than those seen during training. Finally, the positional encoding has the same dimension as the word embeddings, so it can be directly added to them before entering the first Transformer layer.

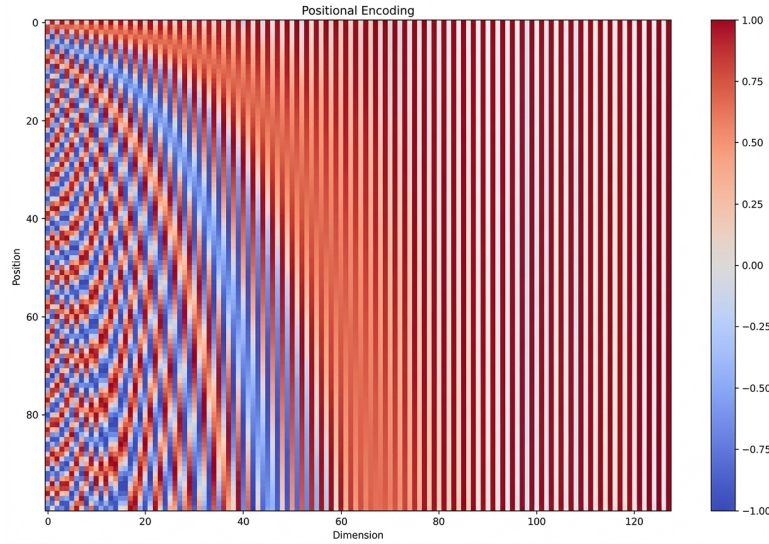


Figure 1.13: Sinusoidal Positional Encoding Matrix [14]

This figure 1.13 visualizes sinusoidal positional encoding in the Transformer. Each row shows a position in the sentence, and each column is a dimension in the embedding space. The colors represent values from sine and cosine functions, creating unique patterns that help the model understand the order of tokens.

1.2.2.5 Residual Connections and Layer Normalization

To improve training stability in deep Transformer models, residual connections are applied around each sublayer, such as the attention and feed-forward blocks. This means that the output of each sublayer is added back to its input before applying normalization.

The operation can be written as:

$$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x)) \quad (1.14)$$

In this equation, x is the input to the sublayer. This design does two main things:

Residual connections : work like a shortcut for gradients. They help avoid the vanishing gradient problem, which makes training deep networks easier.

Layer normalization : keeps the values stable by re-centering and re scaling them. This stops them from becoming too large or shifting too much.

Together, these two techniques help the Transformer train faster and go deeper without losing performance.

1.3 Computer Vision

Computer Vision (CV) is a branch of artificial intelligence that enables machines to understand images and videos. It transforms raw pixel data into meaningful information that can be used for interpretation and decision-making [15].

Early approaches relied on hand crafted features and manually designed algorithms, but they struggled in complex real world environments. With the rise of deep learning, the field has significantly

improved. CNN learn visual features automatically in a hierarchical way, while ViT capture relationships between different regions of an image [15].

Today, computer vision plays an essential role in multimodal systems such as image captioning. It is also widely used in applications like medical imaging, autonomous driving, and intelligent monitoring systems [15].

1.3.1 Taxonomy of Computer Vision Tasks

To better understand the scope of computer vision, tasks can be categorized based on label complexity and spatial density, as shown in Figure 1.14. This taxonomy ranges from simple global prediction to complex structured descriptions.

- **Image Classification:** Assigns a single label to the entire image (e.g., "Dog"), without considering object location [16].
- **Object Detection:** Extends classification by identifying and localizing multiple objects using bounding boxes (e.g., "Dog", "Red Ball") [17].
- **Image Captioning:** Increases complexity by generating a natural language description of the scene (e.g., "A fluffy puppy pushing a red ball"), capturing objects and their relationships [18].
- **Dense Captioning:** Combines localization and description by generating captions for multiple regions within the image, representing the most advanced level of visual understanding [19].

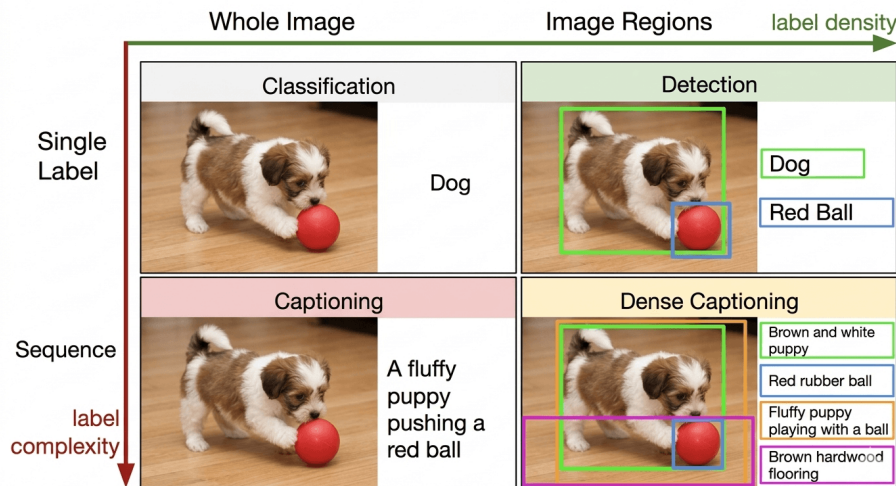


Figure 1.14: Taxonomy of Computer Vision Tasks

1.3.2 Feature Extraction

Feature extraction is the process of turning raw pixels into a smaller set of useful information, known as "features," that represent the most important aspects of an image. This step reduces the amount of data the model needs to handle, so the model can ignore noise and focus on what matters and focus on the data necessary for decision-making [20].

This has changed over time, with two main approaches :

Traditional (Hand-crafted): Early computer vision relied on manually designed algorithms, such as Scale Invariant Feature Transform or Histogram of Oriented Gradients, where researchers wrote the

rules by hand for detecting edges, textures, and shapes [21].

Modern (Automated): With the advent of deep learning, models like CNNs and Transformers learn to represent images on their own. They automatically learn features at different levels starting from basic edges in the initial layers to complex semantic objects (like eyes or wheels) in the deeper layers [22].

As detailed in Table 1.2, deep architectures process visual information through a hierarchical pipeline, moving from primitive edges to complex semantic concepts as the depth increases:

Layer Level	Feature Type
Early Layers	Edges, Corners, Colors
Middle Layers	Textures, Patterns, Simple Shapes
Late Layers	Object Parts, Semantic Components
Final Layers	Complete Objects, High-Level Concepts

Table 1.2: Hierarchical Feature Representation in Deep Networks

1.3.2.1 CNN

A Convolutional Neural Network is a specialized deep learning architecture designed for processing grid structured data such as images. By exploiting spatial relationships between neighboring pixels, CNNs can automatically learn relevant visual features directly from raw image data, eliminating the need for manual feature engineering [23].

A typical CNN architecture consists of several fundamental components [23]:

- **Convolutional Layers:** These layers apply learnable filters (kernels) that slide across the input image to detect local patterns such as edges, textures, and shapes.
- **Activation Functions:** Non-linear functions, commonly the Rectified Linear Unit (ReLU), are applied to introduce non linearity, enabling the network to learn complex representations.
- **Pooling Layers:** These layers reduce the spatial dimensions of feature maps through downsampling operations, which decreases computational cost and improves robustness to small variations in the input.
- **Fully Connected Layers:** Located near the end of the network, these layers combine the extracted high level features to produce the final prediction, such as an image classification label.

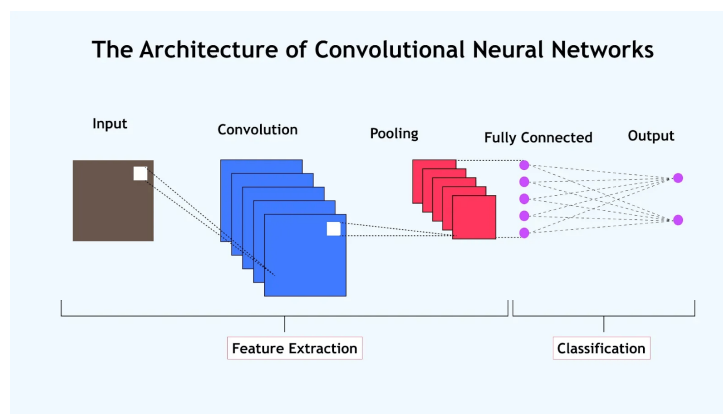


Figure 1.15: Standard Architectural Flow of a Convolutional Neural Network [24]

This figure 1.15 shows the CNN pipeline. First, convolution and pooling layers extract features, turning raw pixels into high-level information. Then, fully connected layers classify the image and output probabilities for each possible class.

1.3.2.2 Pre-trained CNN Models

Pretrained models are deep neural networks that have been previously trained on massive, high quality datasets (most notably ImageNet) to learn a rich hierarchy of visual features. Through Transfer Learning, these models can be reused for new, specialized tasks such as medical imaging without the need for the immense computational power or data required to train a model from scratch [25].

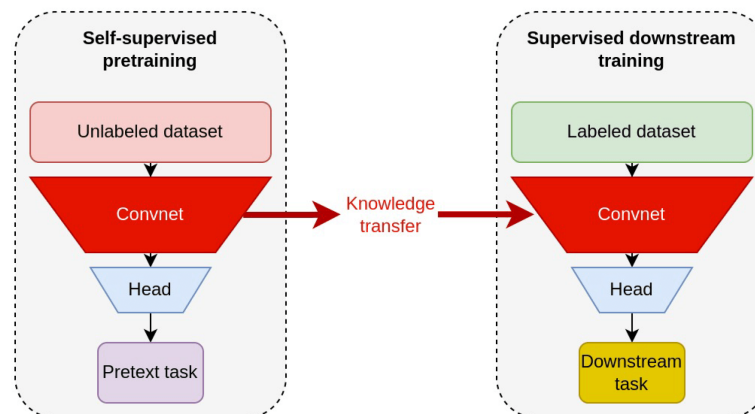


Figure 1.16: Conceptual Framework of Transfer Learning via Knowledge Transfer [26]

Several architectures have become industry standards for their ability to act as powerful "backbone" feature extractors:

- **VGGNet (University of Oxford):** Known for its elegant simplicity, it uses a uniform stack of small 3×3 filters. While computationally heavy, its straightforward design makes it a reliable baseline for understanding spatial hierarchies [27].
- **ResNet (Microsoft Research):** Revolutionized deep learning by introducing Residual (Skip) Connections. These allow gradients to bypass certain layers, solving the "vanishing gradient" problem and enabling the training of extremely deep networks (e.g., 50, 101, or 152 layers) [28].
- **Inception Network (Google):** Focuses on efficiency by using Inception Modules. These modules perform parallel convolutions of different sizes (1×1 , 3×3 , 5×5) within the same layer, allowing the network to capture multi-scale features simultaneously [29].

1.3.3 Vision Transformer

The Vision Transformer (ViT) brought a big change to computer vision. It took the Transformer, which was originally made for text, and applied it to images. Unlike CNNs, which look at small local areas, ViT treats an image as a sequence of small patches. This lets it see how different parts of the image relate to each other from the very beginning [30].

The ViT pipeline involves several key steps:

- **Patch Partitioning:** The input image is divided into fixed size patches (e.g., 16×16 pixels).
- **Linear Projection:** Each patch is flattened and projected into a lower dimensional embedding space.

- **Positional Encoding:** Since Transformers lack an inherent understanding of spatial order, positional embeddings are added to the patch representations to preserve spatial structure.
- **Transformer Encoder:** The sequence of embeddings goes through a Transformer encoder with Multi-Head Self-Attention and Feed-Forward Networks. This lets each patch look at all others, capturing long range relationships in the image.

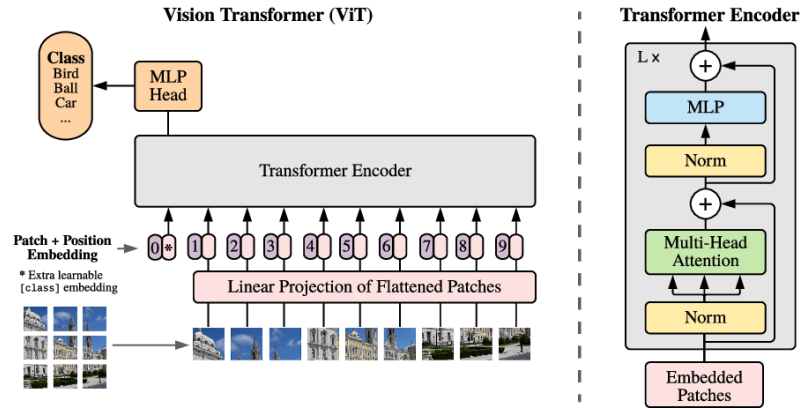


Figure 1.17: The Vision Transformer Architecture [31]

Vision Transformer Configurations:

The Vision Transformer architecture is designed to scale across different computational budgets. Table 1.3 details the specific hyperparameters for the three standard variants, which vary primarily in depth and the dimensionality of the embedding space.

Model	Layers	Hidden size D	MLP size	Heads
ViT-Base	12	768	3072	12
ViT-Large	24	1024	4096	16
ViT-Huge	32	1280	5120	16

Table 1.3: Architectural Specifications of ViT Variants [32]

1.4 Conclusion

In conclusion, this chapter established the theoretical foundation of image captioning by covering key concepts in Natural Language Processing, Encoder-Decoder architectures, Attention mechanisms, and Transformer models. It also presented essential techniques in Computer Vision, along with the main challenges and real world applications, providing a basis for the proposed approach in the following chapters.

Chapter 2

Related Work

2.1 Introduction

Image captioning has become an important research topic in artificial intelligence and computer vision due to its ability to connect visual understanding with natural language generation. Generating accurate and coherent image descriptions remains a challenging task because it requires models to capture both visual content and contextual relationships within a scene.

Over the years, image captioning methods have evolved from early rule based techniques to advanced deep learning architectures. Understanding this evolution is important for identifying the strengths and limitations of existing approaches.

This chapter introduces the main concepts, challenges, and applications of image captioning before reviewing traditional, neural based, and transformer based methods that form the foundation of the proposed work.

2.2 Image Captioning

Image captioning combines computer vision and natural language processing to generate textual descriptions of images. Computer vision extracts information from the image, while natural language processing converts this information into a readable sentence.

Instead of simply detecting objects, image captioning requires understanding the whole scene, including what is happening, how objects interact, and the overall context. The final output must reflect this understanding in clear and natural language.

This makes image captioning a challenging task. A good model must generate correct and fluent sentences while staying grounded in the visual content and avoiding incorrect or imagined details [18].

2.2.1 Formal Definition of Image Captioning

This section defines image captioning and highlights its main characteristics compared to other computer vision tasks.

2.2.1.1 Image Captioning as Image to Text Generation

Image captioning is the task of generating a natural language sentence that describes the content of an image. Unlike image classification, which assigns a label, or object detection, which locates objects, image captioning requires a deeper understanding of the scene.

The model must recognize objects, understand their relationships, and express this information in a coherent sentence. This makes the task more complex, as it combines visual understanding with natural language generation.

For this reason, image captioning is considered an important task in computer vision and natural language processing.

A boy at the beach dumps a bucket of water on another boy's head.



A black-and-white dog bounds off the ground, all feet in the air, of a yellow field.



Figure 2.1: Illustrates examples of the image to text mapping in image captioning [33]

2.2.1.2 Multimodal Foundations

Most image captioning models use an encoder-decoder architecture. The encoder (CNN or ViT) extracts visual features from the image, and the decoder generates a caption word by word.

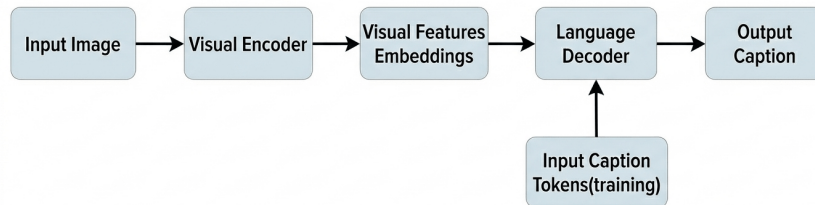


Figure 2.2: General encoder-decoder architecture for image captioning

- **Input Image:** The image to be described.
- **Visual Encoder:** Extracts visual features (e.g., objects, textures, spatial context).
- **Visual Features Embeddings:** A numerical representation of the image.
- **Language Decoder:** Generates the caption one word at a time.
- **Input Caption Tokens (training):** Used only during training (ground truth captions).
- **Output Caption:** The final generated description.

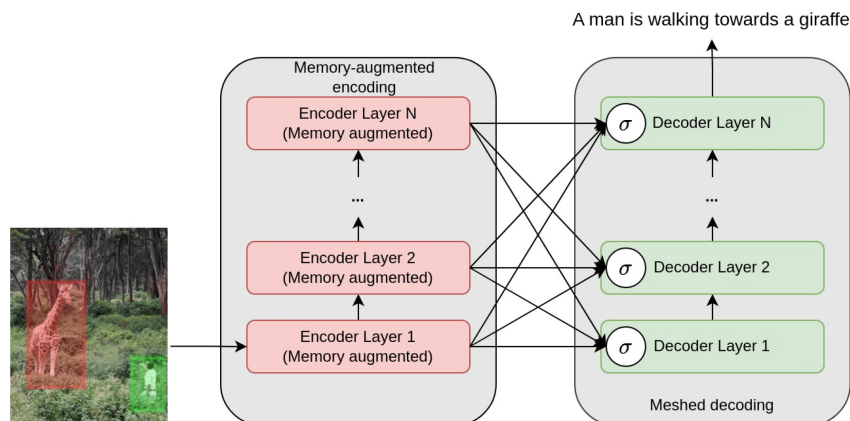


Figure 2.3: The Standard Encoder-Decoder Architecture for Image Captioning [25]

This figure 2.3 illustrates the fundamental pipeline of image captioning. The process begins with a Visual Encoder (CNN or ViT) that extracts semantic embeddings from the input image. These features are then passed to a Language Decoder (RNN or Transformer), which generates the corresponding natural language description token by token.

Modern advancements in this framework often incorporate additional components, such as object detectors for fine-grained feature extraction or memory augmented layers to improve contextual understanding.

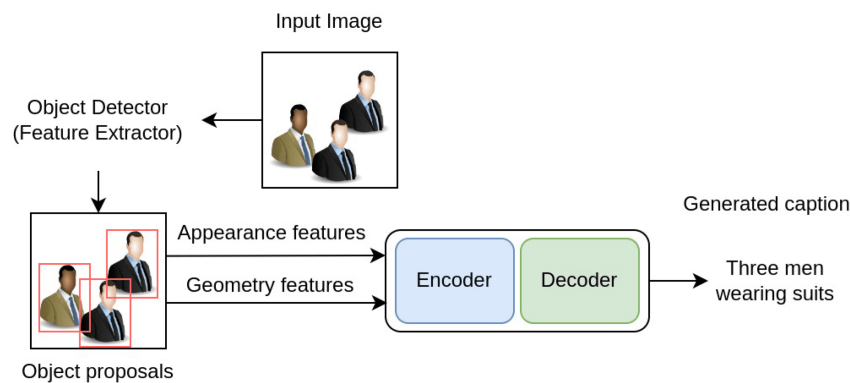


Figure 2.4: Advanced Multimodal Pipeline with Object Proposals [25]

This figure 2.4 illustrates an expanded view of the multimodal foundation, where an Object Detector is utilized to provide localized appearance and geometry features. This allows the encoder-decoder framework to focus on specific regions, leading to more detailed and accurate "Three men wearing suits" style captions.

2.2.2 Core Challenges in Image Captioning

Image captioning combines computer vision and NLP. Deep learning has helped a lot, but there are still basic problems that stop models from describing images as well as humans.

1. Semantic Gap and Visual Ambiguity

The biggest challenge is the semantic gap: turning pixels into words is hard. Models get confused by similar objects (like a dog vs a wolf) or when things are partly hidden. Understanding "a cat sitting on a mat" vs "a cat hiding under a mat" needs more than just detecting objects; it needs to understand space and relationships [34].

2. Object Grounding and Hallucination

Sometimes models describe objects that are not there ("hallucination"). This happens when the language model relies too much on common word patterns (e.g., "coffee" and "table" often go together) instead of actually looking at the image [35].

3. Fine-grained Detail and Multi-scale Features

Small details (like text on a sign or a person's expression) are hard to capture. Standard encoders often shrink images into a fixed size vector and lose these details.

4. Evaluation Metrics and Subjectivity

There is no single "right" caption for an image. Metrics like BLEU, METEOR, and ROUGE measure word overlap and sentence structure, while CIDEr and SPICE are designed specifically for image captioning. However, these metrics don't always capture whether a caption is actually good or sounds natural.

5. Bias and Dataset Limitations

Models learn biases from training data. For example, they might assume that anyone in a kitchen is female. This is unfair and limits how well the model works in the real world [36].

6. Lack of Commonsense Reasoning

Models can recognize objects and actions, but they don't understand common sense. They might see "a person holding an umbrella" but not realize that "it is raining."

7. Generalization and Domain Shift

A model trained on one type of data may fail on new data (different lighting, angles, or unusual scenes). Making models work well in real, unpredictable environments is still an open problem [37].

These challenges show why we need better, more robust multimodal models for image captioning.

2.2.3 Real World Applications

Image captioning has become an important research area in artificial intelligence. By transforming visual content into natural language descriptions, it enables machines to better connect visual and textual information.

1. Digital Accessibility and Inclusion

Image captioning can be used to generate textual descriptions of images. When combined with assistive technologies such as screen readers, these descriptions can help visually impaired users access visual content more easily [38].

2. Content Management and Intelligent Retrieval

Artificial Intelligence has significantly improved information retrieval and classification in large-scale data management systems [39]. In the context of image collections, generated captions can support efficient search and indexing by providing semantic information about image content. This allows users to retrieve images using natural language queries instead of relying only on manually assigned tags [40].

3. E-commerce and Automated Metadata

Image captioning can support e-commerce platforms by assisting in the generation of basic product descriptions from product images. These descriptions may help reduce manual annotation effort and improve the organization of large product catalogs, which can contribute to better search and retrieval of items.

4. Healthcare and Clinical Decision Support

In medical imaging, image captioning has been explored as a supportive tool that can generate simple textual descriptions of scans such as X-rays or MRIs. These descriptions may help highlight general visual patterns and assist clinicians in understanding images more efficiently, especially in preliminary analysis tasks. However, such systems are not intended to replace medical experts, but rather to act as an auxiliary tool that can support decision-making.

5. Autonomous Systems and Scene Understanding

In autonomous systems such as robots or self-driving cars, understanding the surrounding environment is essential for perception and decision-making. Instead of relying only on object detection, vision-language approaches, including captioning-like models, can provide a higher-level description of the scene by identifying objects and describing their relationships. This can contribute to improved situational awareness, especially in complex environments.

2.3 Related Work

Image captioning has been approached through various methods over the years, ranging from traditional rule-based systems to modern deep learning architectures. This section reviews the main categories of methods developed for image captioning, starting with traditional approaches.

2.3.1 Traditional Methods

Traditional methods represent the earliest approaches to image captioning, relying on predefined rules and structured pipelines rather than learning from data. These methods can be broadly categorized into retrieval based, template based, and composition based techniques. Traditional methods are simple and interpretable, but they lack flexibility and generalization capability, particularly when dealing with complex or unseen visual scenes. As a result, their limitations motivated the transition toward learning-based methods.

2.3.1.1 Retrieval-Based Methods

Retrieval based methods are among the earliest approaches to image captioning. Rather than generating new descriptions, they retrieve captions from visually similar images and reuse them for the input image.

A typical retrieval based system extracts visual features, searches a reference dataset for similar images, and selects one or more associated captions. Im2Text [40] is a representative example of this approach, retrieving nearest neighbor images from large scale collections and transferring their captions directly. While this produces grammatically correct descriptions, the results are often generic and may overlook important details in the image.

Several extensions sought to increase flexibility by combining captions from multiple retrieved images or by operating at the phrase level instead of retrieving complete sentences [41, 42]. Retrieval based methods can also benefit significantly from large and diverse datasets [43].

While retrieval-based methods are straightforward and easy to interpret, they remain constrained by the captions available in the reference dataset and cannot generate truly novel descriptions. In addition, visual similarity does not always correspond to semantic similarity, which can lead to inaccurate captions. These shortcomings encouraged the development of generation based approaches, particularly template-based methods.

2.3.1.2 Template-Based Methods

Unlike retrieval based approaches, template based methods generate captions by filling predefined sentence structures with detected visual elements such as objects, attributes, and spatial relations.

These methods typically rely on extracting a structured representation of the image and then mapping it into natural language using fixed templates. This intermediate representation acts as a bridge between visual analysis and sentence generation.

BabyTalk [44] is a representative example, where images are decomposed into objects, attributes, and relationships, and then combined into candidate descriptions using a probabilistic selection process. While this improves interpretability, the generated captions remain limited by the rigidity of the templates.

Mitchell et al. [45] introduced linguistic constraints to improve syntactic validity and reduce invalid word combinations, resulting in more consistent descriptions. However, the method still depends heavily on predefined structures and accurate visual detection.

Building on BabyTalk, Kulkarni et al. [46] incorporated statistical language models to improve fluency by ranking multiple candidate sentences using language statistics. Although this leads to more natural outputs, the diversity of generated captions remains limited by the underlying template structure.

Overall, template based methods establish a stronger connection between visual content and language compared to retrieval based approaches. However, their reliance on fixed patterns limits flexibility and leads to repetitive descriptions. Such restrictions encouraged the exploration of more flexible compositional strategies.

2.3.1.3 Composition-Based Methods

Composition based methods address the limitations of retrieval-based and template based approaches by generating captions through the combination of smaller visual and linguistic units. Instead of reusing

full sentences or fixed templates, they construct descriptions from detected words or phrases in a more flexible manner.

Yang et al. [47] formulate caption generation as a structured prediction problem, where visual elements such as objects, scenes, and relationships are detected and then combined using a language model to produce a coherent sentence. This approach improves flexibility compared to earlier methods but remains dependent on the accuracy of visual detection.

Kuznetsova et al. [41] propose an Integer Linear Programming (ILP) formulation to select and combine candidate phrases while maximizing global consistency. This approach increases diversity but may introduce inconsistencies when combining fragments from different sources. From a compositional perspective, this method focuses on assembling linguistic units rather than retrieving complete captions.

Even with these improvements, composition-based methods still suffer from limitations related to grammatical consistency and error propagation from earlier vision modules. Their reliance on intermediate representations also restricts their ability to generalize to complex or unseen visual scenes.

The difficulty of managing handcrafted intermediate representations ultimately accelerated the transition to end-to-end neural approaches, where visual understanding and language generation are learned jointly.

2.3.2 State of the Art Models for Image Captioning

The limitations of traditional image captioning approaches have driven the development of data-driven methods based on deep learning. Unlike earlier techniques, these models learn a direct mapping between visual inputs and natural language descriptions.

Early neural based approaches introduced the encoder-decoder framework, which improved caption generation through end to end learning. More recently, transformer based architectures [10] have further advanced the field by replacing sequential processing with attention mechanisms. The following sections discuss these developments in more detail.

2.3.2.1 Neural-Based Methods

Neural based methods represent a fundamental shift in image captioning by replacing modular, hand-crafted pipelines with end to end learned models. Rather than relying on predefined templates or explicit fragment selection, these approaches learn a direct mapping from visual inputs to natural language descriptions through the encoder–decoder architecture.

This formulation enables the joint modeling of visual understanding and language generation, allowing models to produce more flexible and contextually appropriate descriptions than traditional methods. The following subsections trace the evolution of this paradigm, from early CNN RNN models to attention-based extensions that allow the model to selectively focus on relevant visual regions during caption generation.

- CNN-RNN Encoder-Decoder Models

A representative implementation of the encoder–decoder framework was introduced by Vinyals et al. [48] in the Show and Tell model. In this approach, a convolutional neural network (CNN) is used to encode the input image into a fixed length feature vector that captures high level semantic information about the scene.

This visual representation is then provided as input to a Long Short Term Memory (LSTM) network, which acts as a decoder to generate the caption sequentially. At each time step, the model predicts the next word conditioned on the encoded image features and the previously generated words.

This model achieved strong performance and established a new baseline for image captioning. Nevertheless, it suffers from a key limitation: compressing the entire image into a single fixed length vector can lead to the loss of important spatial details, particularly in complex scenes.

- Region Based Representations and Alignment

Early encoder–decoder models rely on a single global representation of the image, which may fail to capture fine grained spatial details. To overcome this limitation, subsequent approaches introduced region based representations that model different parts of the image separately.

In this context, Karpathy and Fei-Fei [49] proposed a model that focuses on learning alignments between image regions and corresponding textual elements. Instead of encoding the entire image as a single vector, the model extracts region proposals using R-CNN and represents each region independently. These region level features are then projected into a shared embedding space, where they are aligned with words or phrases in the caption.

While the primary contribution of this approach lies in visual textual alignment rather than full caption generation, it highlights the importance of associating specific image regions with linguistic components. This formulation represents a shift from global description to region-level grounding, where individual words are explicitly associated with specific visual regions. However, the method remains dependent on region proposals and does not dynamically adapt its focus during generation.

As a result, attention mechanisms were introduced to enable more flexible and context dependent focus during caption generation.

- Attention Mechanisms

Addressing the limitations of global image representations and fixed region alignments, attention mechanisms were introduced to allow models to dynamically focus on relevant parts of the image during caption generation.

Xu et al. [50] proposed the Show, Attend and Tell model, which integrates a spatial attention mechanism into the encoder–decoder framework. In this approach, the image is represented as a set of spatial feature vectors, and at each step of the generation process, the model assigns different importance weights to these features when predicting the next word.

The concept was further extended by You et al. [51], who introduced semantic attention mechanisms that allow the model to focus on high level visual concepts such as objects and attributes, in addition to spatial regions. Such extensions improve the model’s ability to select not only where to focus in the image, but also what semantic information is most relevant for generating the next word.

By enabling dynamic and context-dependent selection of visual information, attention mechanisms improve the quality of generated captions and strengthen the alignment between visual features and linguistic elements. This capability allows the model to better capture interactions between multiple objects and their context, particularly in complex scenes.

More importantly, attention removes the constraint of fixed representations by allowing the model to adapt its focus at each time step. This shift represents a key conceptual advance in image captioning and laid the foundation for transformer-based architectures, which rely entirely on attention mechanisms to model dependencies within visual and textual data.

2.3.2.2 Limitations of Early Neural Models

Early neural image captioning models significantly improved caption generation through end-to-end learning, region-level alignment, and attention mechanisms. Most of these approaches still relied on recurrent architectures for sequence generation.

Their sequential design limits parallelization, making training slower and less scalable. Recurrent models such as LSTMs also struggle to capture long-range dependencies, which can affect the coherence of generated captions, particularly in complex scenes.

These constraints gradually led researchers toward transformer-based architectures [10]. Self-attention mechanisms remove the need for recurrence and provide a more effective way to model global relationships within sequences.

Differences among these models can be observed in their architectures and overall design (Table 2.1).

Model	Architecture	Key Contribution	Limitation
Show and Tell [48]	CNN + LSTM	First end to end neural framework for image captioning	Single fixed length visual vector loses spatial information
Show, Attend and Tell [50]	CNN + Attention + LSTM	Introduces spatial attention; model learns to focus on relevant regions dynamically	Attention is applied only at decoder level; still uses RNN
Karpathy & Fei-Fei [49]	R-CNN + RNN	Aligns image regions with textual fragments for region-word grounding	Not a full captioning system; depends on external region proposals
You et al. [51]	CNN + Attention + LSTM	Incorporates high level semantic attributes into attention	Requires predefined attribute detectors; still RNN-based

Table 2.1: Summary of neural-based image captioning models

2.3.2.3 Transformer-Based Methods

Limitations observed in earlier neural approaches led to the adoption of transformer-based architectures as a more effective paradigm for image captioning. Relying on attention mechanisms rather than sequential processing, these models offer improved capacity for modeling complex dependencies between visual content and generated text [10].

Research in this direction has explored transformer-based methods at multiple levels of the image captioning pipeline, including sequence modeling, visual representation learning, language generation, and multimodal alignment. The following subsections examine these developments, with particular attention to visual representation learning and language generation, which form the foundation of the proposed approach.

- Transformer Encoder-Decoder Models

Limitations of recurrent architectures in modeling long-range dependencies and complex visual-textual interactions have led to the adoption of transformer-based encoder-decoder models for image captioning. Unlike RNN-based approaches, which process information sequentially, transformers rely on attention mechanisms that enable direct modeling of relationships across the entire input. Through self-attention and cross-attention, visual regions can be dynamically aligned with generated words, enabling more explicit interaction between visual content and language.

Several works have explored this paradigm. The Meshed-Memory Transformer (M^2) [52] introduces memory vectors within the encoder to aggregate multi level visual information, capturing richer semantic representations across layers. Similarly, the Object Relation Transformer [53] models interactions between detected objects using self-attention, allowing explicit reasoning over object relationships rather than isolated features.

Even with these improvements, these models still rely heavily on the quality of the underlying visual features, which are often extracted using CNN-based encoders. Such features primarily capture local patterns and may not fully represent global spatial structure, limiting the overall effectiveness of the model.

This limitation has motivated the exploration of vision transformers and self-supervised visual representations, which aim to provide more globally consistent and semantically rich features for image captioning.

- Multimodal Alignment and Pretrained Vision-Language Models

Transformer-based language models in image captioning rely on effective integration of visual information, which has motivated multimodal alignment methods that bridge vision and language representations.

A common approach is to condition a pre-trained language model on visual features through a lightweight mapping. ClipCap [54] projects image embeddings into a prefix sequence compatible with GPT-2 while keeping the language model frozen, enabling efficient adaptation.

However, fixed visual features may struggle with fine-grained details. BLIP [55] addresses this by jointly modeling vision and language and improving training through bootstrapped image–text filtering.

More recent methods decouple the visual encoder and language model while introducing lightweight bridging modules. Flamingo [56] uses a Perceiver Resampler to compress visual features into a fixed set of tokens for a frozen language model, enabling strong few-shot performance.

BLIP-2 [57] introduces a Q-Former that connects a frozen image encoder and a frozen language model through a two-stage training strategy, achieving strong performance with fewer trainable parameters.

Taken together, these developments highlight the diversity of transformer based image captioning systems in terms of their components, roles, and design choices within the overall pipeline, as reflected in Table 2.2.

Model	Architectural Role	Key Contribution	Limitation / Gap
M ² Transformer [52]	Encoder–Decoder	Memory-augmented encoder for multi level visual feature interaction	Relies on pre-extracted CNN features; does not learn visual representations
Object Relation Transformer [53]	Encoder–Decoder	Explicitly models object to object relationships via self-attention	Depends on the quality of an external object detector
ClipCap [54]	Multimodal Bridge	Lightweight mapping network projects visual features into text prefix space	Limited to fixed visual features; lacks fine-grained grounding
BLIP [55]	Vision–Language Model	Joint training with bootstrapping for noisy data filtering	Computationally expensive; requires large-scale paired data
Flamingo [56]	Vision–Language Model	Perceiver Resampler compresses visual features for frozen LLM; supports few-shot learning	Extremely computationally expensive; requires massive interleaved training data
BLIP-2 [57]	Vision–Language Model	Q-Former bridges frozen image encoder and frozen LLM via two-stage pre-training	Complex training pipeline; two-stage training may limit few-shot adaptability

Table 2.2: Summary of transformer-based components and models relevant to image captioning.

2.3.2.4 Synthesis and Research Gap

The transformer-based methods surveyed here have pushed image captioning forward on two fronts: richer visual representations and more fluent language generation. However, a recurring limitation across top-performing systems is their dependence on large-scale architectures and resource-intensive training pipelines, which makes replication and adaptation costly.

This motivates the exploration of more lightweight designs that reduce computational requirements while preserving the effectiveness of pretrained components. In this context, this work proposes a

framework that connects a self-supervised DINOv2 encoder [58] with GPT-2 [59] through a compact Visual Query Adapter.

This design aims to reduce training complexity and reliance on large annotated datasets while preserving the semantic richness of pretrained models.

2.4 Benchmarking Datasets and Evaluation Metrics

2.4.1 Benchmark Datasets

Image captioning models are typically trained on datasets that associate images with human-written descriptions. These resources are fundamental for both learning and evaluation, as they determine the diversity of visual content and linguistic expressions available to the model. The presence of multiple captions per image is particularly important, as it captures the variability of natural language, where the same scene can be described in several valid ways.

MS COCO

The Microsoft Common Objects in Context (MS COCO) dataset [60] is the most widely used benchmark for image captioning. The commonly adopted 2014 version contains approximately 82,000 images for training and 40,000 for validation, each paired with five human-written captions, resulting in over 600,000 annotations for these two splits.

One of the defining characteristics of MS COCO is its emphasis on realistic scenes containing multiple interacting objects. With 91 object categories, the dataset prioritizes contextual understanding over isolated object detection. The captions, collected via crowd-sourcing, reflect a broad range of descriptive styles and linguistic variations.

Owing to its scale and diversity, MS COCO is extensively used for training deep learning models as well as for standardized evaluation. Nevertheless, working with this dataset can be computationally intensive, especially when training large transformer based architectures.

Flickr30k

Flickr30k [61] consists of approximately 31,783 images sourced from Flickr, each accompanied by five human-generated captions. Compared to MS COCO, it places greater emphasis on human actions and interactions, often leading to more detailed and expressive descriptions.

This makes Flickr30k particularly suitable for assessing the fluency and richness of generated captions. It is frequently used as a complementary benchmark, especially in studies that focus on language quality rather than large-scale visual generalization.

Due to its moderate size, Flickr30k is often employed for evaluation or fine-tuning rather than full scale training. However, its relatively limited coverage compared to MS COCO can restrict the generalization capabilities of models trained solely on this dataset.

Flickr8k

Flickr8k [62] contains 8,000 images, each annotated with five captions. As one of the earliest benchmarks in image captioning, it is mainly used for rapid experimentation and proof of concept studies.

Its compact size allows for faster training and easier debugging, making it useful in scenarios with limited computational resources. However, it lacks the scale and variability required for training modern models, which typically rely on large and diverse datasets to achieve strong performance.

These datasets differ in scale and usage, which directly influences both training strategies and evaluation practices in image captioning (Table 2.3).

Dataset	Images	Captions per Image	Total Captions	Primary Use
MS COCO [60]	123K	5	615K	Training and evaluation
Flickr30k [61]	31.8K	5	158K	Evaluation and fine-tuning
Flickr8k [62]	8K	5	40K	Prototyping

Table 2.3: Comparison of widely used image captioning datasets.

2.4.2 Evaluation Metrics

Evaluating the quality of generated captions is more complex than standard prediction tasks, as a single image can be described in multiple valid ways using different vocabularies and structures. For this reason, evaluation in image captioning relies on comparing a generated caption with several human-written references using automated metrics that capture both statistical and semantic similarities.

BLEU

BLEU, introduced by Papineni et al. [63], is based on n-gram precision and measures the proportion of word sequences in the generated caption that also appear in the reference captions. The final score is typically computed as a geometric mean over multiple n-gram levels.

It provides a computationally efficient baseline for evaluating lexical similarity. However, it is highly sensitive to exact wording and does not account for synonymy or semantic equivalence, which can lead to low scores for valid paraphrases.

METEOR

METEOR, proposed by Banerjee and Lavie [64], was developed to address the limitations of BLEU by incorporating both precision and recall, along with flexible word matching.

Alignment between candidate and reference captions is performed using exact matches, stemming, and synonym matching (e.g., via WordNet). This allows the metric to better capture semantic similarity and results in a higher correlation with human judgment.

A limitation of METEOR lies in its dependence on external linguistic resources, which may restrict its applicability across different languages.

ROUGE-L

ROUGE-L, introduced by Lin [65], is based on the longest common subsequence (LCS) between the generated and reference captions. This approach captures sentence-level structure by identifying the longest sequence of words that appears in both sentences in the same order.

It is effective for evaluating the coherence and overall structure of generated captions. However, like BLEU, it does not fully capture semantic correctness and may overlook meaning differences.

CIDEr-D

CIDEr-D, proposed by Vedantam et al. [66], was specifically designed for image captioning. It applies TF-IDF weighting to n-grams, giving less importance to common words and emphasizing distinctive and informative terms.

This metric evaluates how well a generated caption aligns with the consensus of multiple human annotations, making it particularly suitable for assessing descriptive relevance.

However, reliance on word frequency statistics may introduce bias toward certain keywords, even when overall sentence quality is suboptimal.

SPICE

SPICE, introduced by Anderson et al. [67], evaluates captions at the semantic level by converting them into structured representations known as scene graphs. These graphs encode objects, attributes, and relationships, allowing the comparison to focus on meaning rather than surface text.

As a result, SPICE is particularly effective for assessing whether a caption accurately reflects the content of an image. However, it is computationally expensive and does not evaluate linguistic fluency.

Different evaluation metrics capture complementary aspects of caption quality, as summarized in Table 2.4.

Metric	Original Task	Role in Captioning	Key Limitation
BLEU [63]	Machine Translation	Measures lexical overlap using n-grams	Does not capture semantic meaning
METEOR [64]	Machine Translation	Evaluates semantic similarity with flexible matching	Still partially dependent on lexical overlap
ROUGE-L [65]	Text Summarization	Captures sentence structure via sequence matching	Limited ability to reflect semantic meaning
CIDEr-D [66]	Image Captioning	Measures agreement with human descriptions	Sensitive to reference distribution
SPICE [67]	Image Captioning	Evaluates semantic content using scene graphs	Computationally expensive

Table 2.4: Overview of automated evaluation metrics used in image captioning.

Each metric captures a different dimension of caption quality. For this reason, they are typically used together to provide a balanced evaluation that accounts for lexical similarity, structural coherence, and semantic correctness.

2.5 Conclusion

This chapter reviewed the evolution of image captioning, from traditional methods to modern transformer-based approaches. Recent advances have improved both visual understanding and language generation, yet effective vision-language alignment remains a key challenge for image captioning systems. The next chapter addresses this gap by introducing a Visual Query Adapter that connects a self-supervised visual encoder (DINOv2) with a generative language model (GPT-2).

Chapter 3

Proposed Image Captioning Model

3.1 Introduction

Following the review of image captioning methods in Chapter 2, this chapter presents the proposed model for generating textual descriptions from images. Many existing approaches rely on supervised visual features or tightly coupled encoder–decoder architectures, which reduce flexibility and make adaptation to new tasks difficult.

Building on these observations, the proposed framework adopts a modular design that combines a Vision Transformer trained using DINOv2 self-supervised learning [58] with a pre-trained GPT-2 decoder [59]. A key component of the framework is the Visual Query Adapter, which bridges the visual and textual modalities by transforming visual features into a form that can be incorporated into the decoder through cross-attention.

3.2 Overview of the Proposed Framework

This section provides an overview of the proposed framework, outlining its main components and the flow of information from the input image to the generated caption.

Figure 3.1 illustrates the complete pipeline of the proposed image captioning framework.

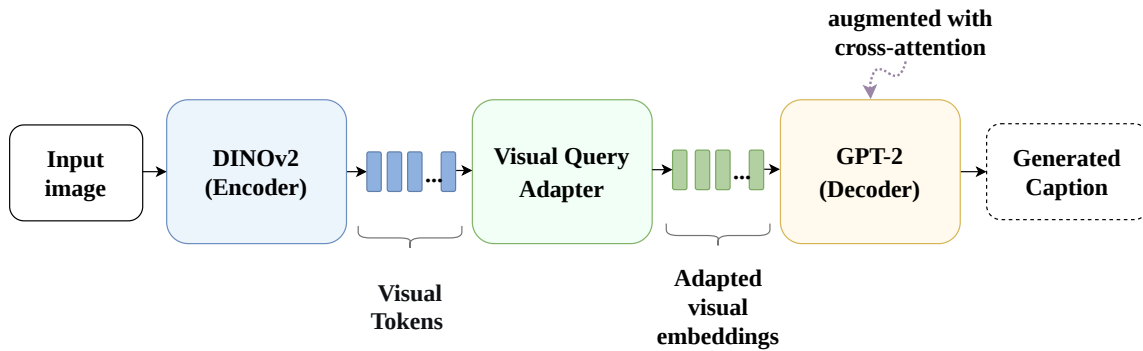


Figure 3.1: Overall pipeline of the proposed image captioning framework.

As shown in Figure 3.1, the processing pipeline consists of three main stages:

- **Encoding:** The input image is passed through the DINOv2 encoder, which extracts a sequence of visual tokens.
- **Adaptation:** The visual tokens are processed by the Visual Query Adapter, which produces a compact set of adapted visual embeddings.
- **Decoding:** The GPT-2 decoder, augmented with cross-attention layers, receives the adapted visual embeddings and generates the caption autoregressively.

3.3 Visual Feature Extraction: DINOv2

The visual encoding component of our framework uses the pre-trained DINOv2-Base architecture [58]. DINOv2 is a self-supervised Vision Transformer that learns rich semantic visual representations without requiring manually annotated image text pairs during pretraining.

Unlike conventional convolutional neural networks, Vision Transformers capture long range dependencies between image regions through self-attention mechanisms. This enables the encoder to understand both local visual details and the overall scene context, which is essential for image captioning tasks. To generate meaningful captions, the model must first understand object relationships, spatial composition, and contextual scene information.

We selected DINOv2 as the visual backbone because it provides strong semantic representations, generalizes effectively across different vision tasks, and learns meaningful visual structures from large scale unlabeled datasets.

The encoder converts the input image into a sequence of contextual visual embeddings that preserve both spatial and semantic information. Instead of relying only on a single global image representation, the proposed framework retains patch level visual tokens, enabling the subsequent adaptation module to access fine-grained visual information during caption generation.

3.3.1 Vision Transformer Backbone and Token Layout

DINOv2 is built on the Vision Transformer (ViT) architecture [68], which follows the standard Transformer design [10]. In this architecture, the input image is first divided into fixed-size non-overlapping patches. Each patch is then linearly projected into a learnable embedding vector.

In the proposed framework, the input image is resized to:

$$224 \times 224 \times 3$$

The image is divided into patches of size:

$$14 \times 14$$

This results in a total number of spatial patches:

$$N = \left(\frac{224}{14} \right) \times \left(\frac{224}{14} \right) = 16 \times 16 = 256$$

Each patch embedding is represented as a 768 dimensional feature vector.

A learnable classification token ([CLS]) is appended to the patch sequence to capture global image semantics. Positional embeddings are then added to all tokens to preserve spatial information and maintain the positional relationships between image patches.

The resulting token sequence passes through 12 Transformer encoder blocks. Following the original Transformer architecture [10], each block contains:

- Multi-head self-attention (12 attention heads)
- Feed-forward networks (FFN)
- Residual connections
- Layer normalization

Through self-attention, each visual token can interact with all other image regions. This allows the model to capture semantic relationships between distant objects and better understand the global structure of the scene.

The final visual representation generated by the encoder is expressed as:

$$H_{\text{visual}} \in \mathbb{R}^{257 \times 768} \quad (3.1)$$

where the sequence length of 257 tokens consists of one global [CLS] token and 256 spatial patch tokens.

These visual embeddings are later refined by the proposed Visual Query Adapter before being forwarded to the GPT-2 language decoder.

The patch extraction and visual token generation process is presented in Figure 3.2.

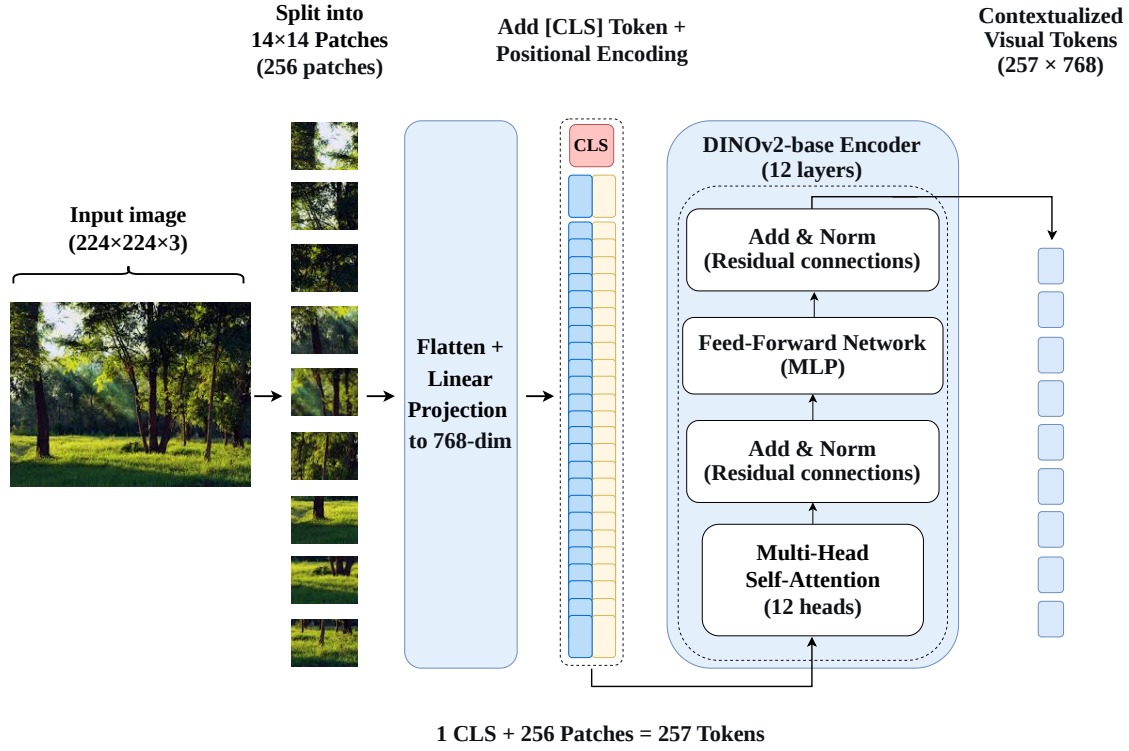


Figure 3.2: Vision Transformer patch extraction and visual token generation process for DINOv2-Base

3.3.2 Self-Supervised Student Teacher Distillation

DINOv2 is trained using a self-supervised student teacher distillation framework [58]. This training strategy enables the model to learn meaningful visual representations directly from image data without requiring manual annotations.

During training, the framework maintains two neural networks: a Student network and a Teacher network. Multiple augmented views of the same image are generated through random cropping and geometric transformations. The Student network processes both global and local image crops, while the Teacher network processes only global crops.

The Student network learns to produce representations that match those generated by the Teacher network despite differences in image augmentations. While the Student network is updated using gradient descent, the Teacher network parameters are updated using an Exponential Moving Average (EMA) of the Student parameters:

$$\theta_t \leftarrow \lambda \theta_t + (1 - \lambda) \theta_s \quad (3.2)$$

where:

- θ_t represents the Teacher parameters.
- θ_s represents the Student parameters.
- λ is the momentum coefficient.

This update mechanism stabilizes training and helps prevent representation collapse during self-supervised learning.

The overall student–teacher learning framework is shown in Figure 3.3.

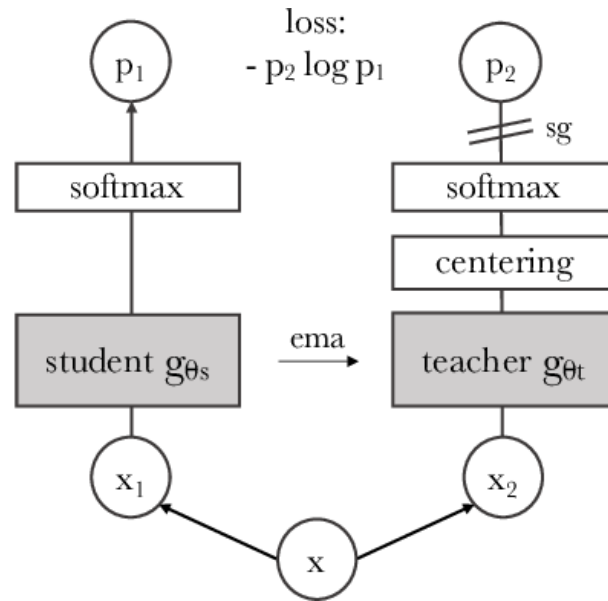


Figure 3.3: Self-supervised student-teacher distillation framework [69]

3.3.3 DINOv2 Pretraining Strategy and Data Curation

One of the main strengths of DINOv2 comes from its large scale curated dataset. Instead of relying solely on raw internet images, DINOv2 applies automated data filtering, retrieval, and deduplication pipelines to construct the LVD-142M dataset, significantly improving training quality and semantic diversity.

As shown in Figure 3.4, this pipeline processes a massive pool of uncured web data by mapping images into a shared embedding space, executing strict copy detection (deduplication) to remove redundancies, and utilizing a high quality curated seed dataset for semantic retrieval.

The pretraining process optimizes the student backbone using two complementary loss functions:

- **Global DINO Loss:** Computes cross entropy between the student and teacher [CLS] token outputs to align global image-level representations.
- **Local iBOT Loss:** Implements masked image modeling, where the student network receives randomly masked spatial patches and predicts the corresponding unmasked patch embeddings generated by the teacher network.

This combination of global and local training objectives enables the encoder to capture fine spatial details, object boundaries, and semantic relationships across different visual contexts. These properties are particularly beneficial for image captioning tasks, where the encoder must understand scene structure and visual relationships before generating natural language descriptions.

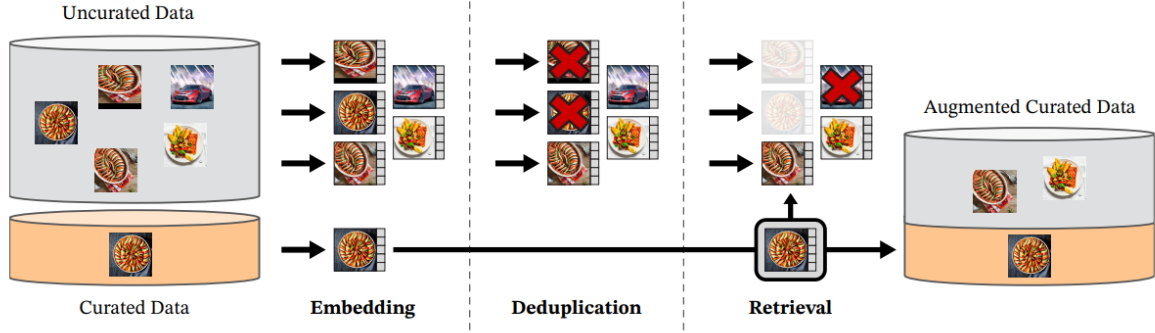


Figure 3.4: DINOv2 data curation and representation scaling pipeline [58]

The contextual visual embeddings extracted by DINOv2 are subsequently refined by the proposed Visual Query Adapter to facilitate efficient interaction with the GPT-2 language decoder.

3.4 Cross Modal Adaptation: Visual Query Adapter

The Visual Query Adapter bridges the representational gap between DINOv2’s continuous visual feature space and GPT-2’s textual embedding space. Standard language decoders are not designed to interpret raw visual sequences directly. Therefore, the primary objective of this module is to compress and project the dense visual token sequence into a fixed-size set of tokens that effectively conditions autoregressive text generation.

Instead of propagating the entire raw sequence directly to the language model which would increase computational overhead and inject redundant background noise the adapter selectively condenses the sequence into 16 learnable query embeddings. This bottleneck structure preserves the essential semantic structure of the scene while filtering redundant visual information.

3.4.1 Architecture and Mathematical Formulation

The structural pipeline of the proposed Visual Query Adapter consists of three sequential stages:

1. Linear Dimension Projection
2. Multi-Head Cross-Attention (MHCA)
3. Feed-Forward Network (FFN) Feature Refinement

3.4.1.1 Linear Dimension Projection

The contextual visual embeddings extracted from the DINOv2 encoder consisting of one global [CLS] token and 256 spatial patch tokens are represented as:

$$H_{\text{visual}} \in \mathbb{R}^{257 \times 768} \quad (3.3)$$

To align these features with the latent representation space of the cross-attention layers, they are passed through a trainable linear projection layer. Let $W_p \in \mathbb{R}^{768 \times 768}$ denote the weight matrix and $b_p \in \mathbb{R}^{768}$ denote the bias vector. The projected visual tensor is formulated as:

$$X_{\text{projected}} = H_{\text{visual}}W_p + b_p \quad (3.4)$$

where $X_{\text{projected}} \in \mathbb{R}^{257 \times 768}$ preserves the target hidden dimensionality.

3.4.1.2 Multi-Head Cross-Attention

To selectively aggregate spatial semantics, we introduce a set of $Q_{\text{learnable}} \in \mathbb{R}^{16 \times 768}$ learnable query tokens.

These queries interact dynamically with the projected visual features through a Multi-Head Cross-Attention mechanism. In this configuration, the learnable query tokens define the Queries (Q), while the projected visual sequence serves as both the Keys (K) and Values (V).

The corresponding linear transformations per attention head are expressed as:

$$\begin{aligned} Q &= Q_{\text{learnable}} W_Q \\ K &= X_{\text{projected}} W_K \\ V &= X_{\text{projected}} W_V \end{aligned} \quad (3.5)$$

where $W_Q, W_K, W_V \in \mathbb{R}^{768 \times d_k}$ represent the projection matrices.

The dimensionality per attention head is $d_k = 768/h = 96$, with $h = 8$ attention heads. The scaled dot-product attention for an individual head is formulated as:

$$\text{head}_i = \text{softmax} \left(\frac{Q_i K_i^\top}{\sqrt{d_k}} \right) V_i \quad (3.6)$$

The final multi-head cross-attention output A is obtained by concatenating all attention heads followed by a linear output projection $W^O \in \mathbb{R}^{768 \times 768}$:

$$A = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h) W^O \quad (3.7)$$

A residual connection and layer normalization [70] are subsequently applied to facilitate stable gradient propagation:

$$\tilde{A} = \text{LayerNorm}(Q_{\text{learnable}} + A) \quad (3.8)$$

where $\tilde{A} \in \mathbb{R}^{16 \times 768}$.

Through this mechanism, each learnable query token systematically probes the entire visual token space, capturing complementary semantic information from different regions of the image.

3.4.1.3 Feed-Forward Network Refinement

Following information aggregation, the intermediate representations are forwarded through a Position-Wise Feed-Forward Network (FFN) to perform non-linear feature transformation. The FFN consists of two linear layers separated by a Gaussian Error Linear Unit (GELU) activation function [71]:

$$\text{FFN}(\tilde{A}) = \text{GELU}(\tilde{A} W_1 + b_1) W_2 + b_2 \quad (3.9)$$

where $W_1 \in \mathbb{R}^{768 \times 3072}$, $W_2 \in \mathbb{R}^{3072 \times 768}$, and b_1, b_2 denote the corresponding bias vectors.

The final compressed, vision conditioned token block Z_{adapter} is obtained through a secondary residual connection and normalization step:

$$Z_{\text{adapter}} = \text{LayerNorm}(\tilde{A} + \text{FFN}(\tilde{A})) \quad (3.10)$$

where $Z_{\text{adapter}} \in \mathbb{R}^{16 \times 768}$.

This condensed feature representation is subsequently fed as the cross-attention context represen-

tation into the modified cross-attention layers of the GPT-2 decoder to drive autoregressive caption generation.

Figure 3.5 illustrates the complete internal architecture of the proposed Visual Query Adapter.

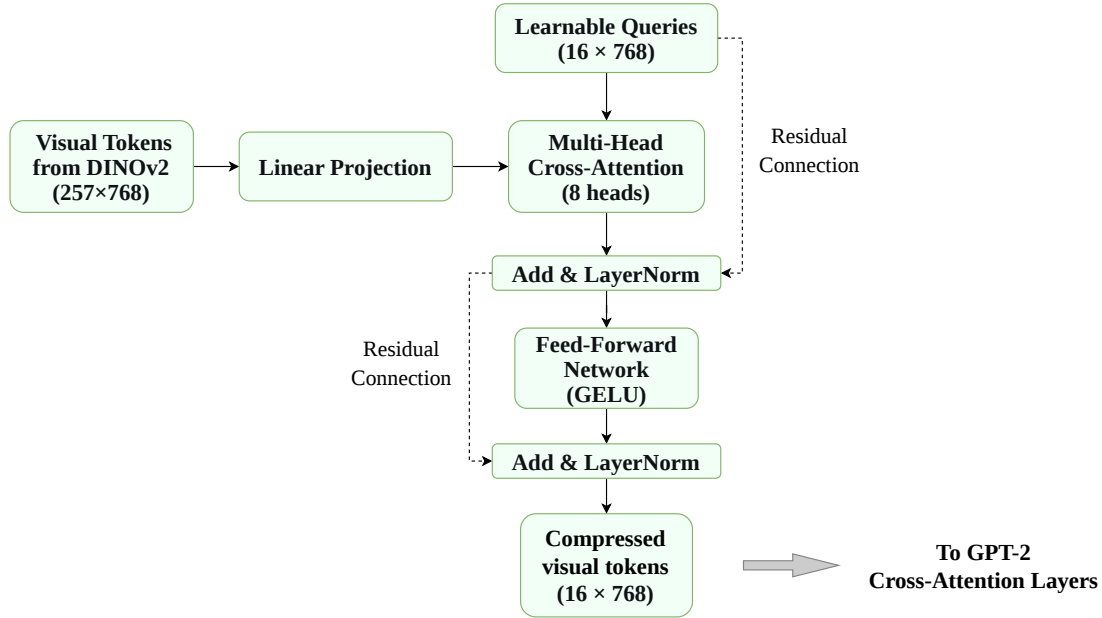


Figure 3.5: Detailed architecture and information flow of the Visual Query Adapter

3.5 Language Generation Module: GPT-2 Decoder

We use GPT-2 [59] as our language decoder. GPT-2 is a Transformer decoder that generates text autoregressively: each new token is predicted based on the tokens generated so far. We chose GPT-2 because it is pretrained on large text corpora, which gives it strong linguistic fluency even before adaptation to the image captioning task.

The original GPT-2 architecture does not accept visual input. To enable image-conditioned generation, we modify the decoder by activating cross attention layers within each Transformer block. This allows the model to attend to visual features while generating each word.

3.5.1 Pre-trained Architecture

GPT-2 consists of 12 stacked Transformer decoder layers [59]. Each layer contains three main components:

- Masked self-attention (causal attention with 12 heads)
- Feed-forward network (FFN) with GELU activation
- Layer normalization and residual connections around each sub-layer

The masked self-attention mechanism enforces causality, ensuring that each token can only attend to previous tokens in the sequence. This is essential for autoregressive generation, where the model predicts one word at a time from left to right.

The hidden dimension of GPT-2 Base is 768, matching the output dimension of our Visual Query Adapter. This dimensional alignment allows the visual embeddings to be directly integrated into the decoder’s cross-attention layers.

3.5.2 Cross-Attention for Visual Conditioning

To condition text generation on visual information, we augment each GPT-2 decoder block with a cross-attention layer. This layer is inserted between the masked self-attention and the feed-forward network.

The cross-attention mechanism follows the scaled dot-product formulation introduced in the Transformer architecture [10]. It operates as follows: the decoder's hidden states serve as the Queries (Q), while the visual embeddings from the Visual Query Adapter serve as the Keys (K) and Values (V).

Formally, for a single attention head:

$$\text{head}_i = \text{softmax} \left(\frac{Q_i K_i^T}{\sqrt{d_k}} \right) V_i \quad (3.11)$$

The multi-head output combines all heads (12 heads in GPT-2) and projects them back to the hidden dimension. This enables the decoder to dynamically focus on different visual regions at each generation step.

During training, the model uses teacher forcing [72] with cross-entropy loss. The decoder receives the ground-truth token sequence shifted by one position and predicts the next token at each step, conditioned on the visual context.

Three special tokens are introduced to manage the generation process:

- $\langle \text{bos} \rangle$ (beginning of sequence): signals the start of caption generation
- $\langle \text{eos} \rangle$ (end of sequence): marks the completion of the generated caption
- $\langle \text{pad} \rangle$: used for padding sequences to a fixed length within a batch

The overall generation process begins with the $\langle \text{bos} \rangle$ token. At each step, the decoder computes the probability distribution over the vocabulary and selects the next token. The process continues until the $\langle \text{eos} \rangle$ token is generated or the maximum length is reached.

The architecture of the modified GPT-2 decoder layer is illustrated in Figure 3.6.

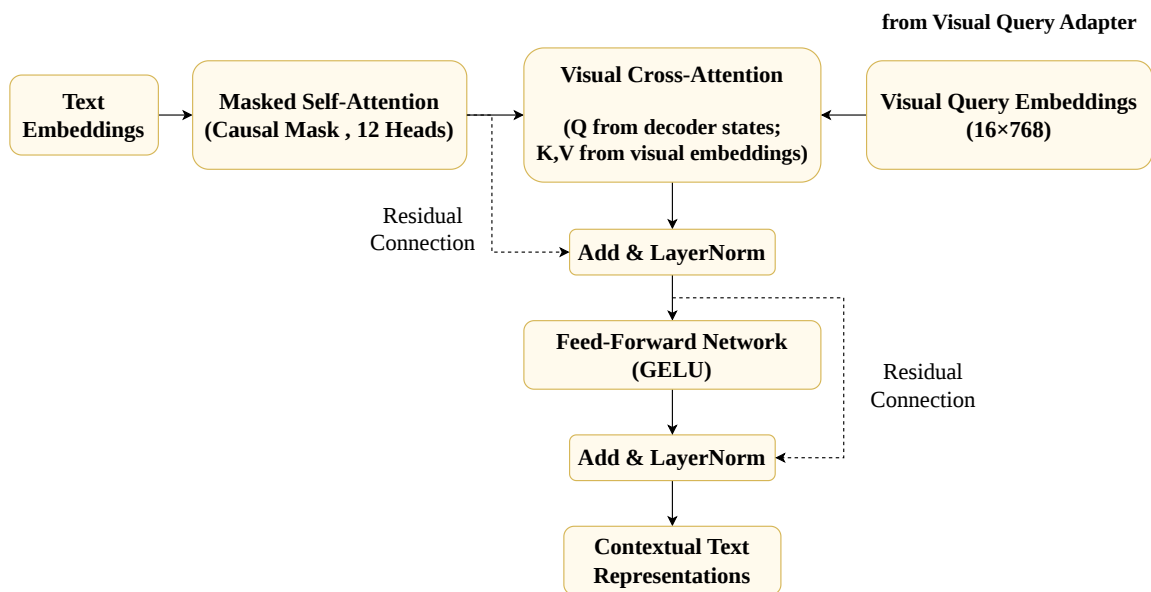


Figure 3.6: Architecture of the modified GPT-2 decoder layer with visual cross-attention conditioning

3.6 Proposed Training Strategy

This section presents the theoretical foundation of the proposed training methodology. First, the theoretical motivations for the proposed strategy are presented. Then, the proposed two stage fine-tuning strategy is introduced, followed by the adopted loss function.

3.6.1 Theoretical Motivations for Two Stage Training

To avoid the well documented risks of catastrophic forgetting [73, 74] and optimization instability [75] associated with full end to end fine-tuning of large vision language architectures, we adopt a progressive two stage fine-tuning strategy.

The primary theoretical motivations for this design choice are threefold:

Catastrophic Forgetting Prevention

Full end-to-end fine-tuning of pretrained models has been shown to overwrite general purpose representations acquired during large-scale pretraining [74]. In the context of image captioning, simultaneously updating all parameters of DINOv2 and GPT-2 risks degrading both visual feature extraction and linguistic fluency.

Computational Efficiency

End to end training of large vision language models requires substantial GPU memory and computational resources [76]. A two stage strategy reduces memory footprint by freezing the visual encoder during the initial phase, enabling stable training in resource-constrained environments.

Optimization Stability

Jointly optimizing two large pretrained networks with different representational structures can lead to unstable gradients and slow convergence [75]. By first establishing a stable cross modal alignment with a frozen encoder, the optimization problem is simplified, allowing the model to learn modality mapping before refining visual representations.

3.6.2 Two Stage Fine-Tuning Strategy

The core principle of the proposed strategy is to first establish a stable cross modal alignment while preserving the pretrained visual representations. Once this alignment is learned, selective fine-tuning is performed to refine both the visual and linguistic representations for the captioning task.

3.6.2.1 Stage A: Adapter Training with Frozen Visual Encoder

Objective: The objective of Stage A is to train the Visual Query Adapter together with the upper layers of the GPT-2 decoder while keeping the visual encoder completely frozen.

Trainable Components:

- Visual Query Adapter: Fully trainable
- GPT-2 Decoder: Last 4 transformer blocks are trainable
- DINOv2 Encoder: Completely frozen

Theoretical Rationale: The DINOv2 encoder, pretrained on large-scale image data [58], already provides robust and highly generalizable semantic visual features. Freezing the encoder during the initial

phase prevents catastrophic forgetting and preserves these stable visual representations.

With static visual features, the optimization process becomes significantly simpler because the model only needs to learn how to map visual embeddings into the textual generation space through the adapter and decoder layers. Training only the upper transformer blocks of GPT-2 enables the decoder to specialize its higher-level representations for image conditioned generation while preserving the lower level linguistic knowledge acquired during pretraining.

Figure 3.7 illustrates the training setup for Stage A, where the DINOv2 encoder is frozen, while the Visual Query Adapter and the last four blocks of GPT-2 decoder are trainable.

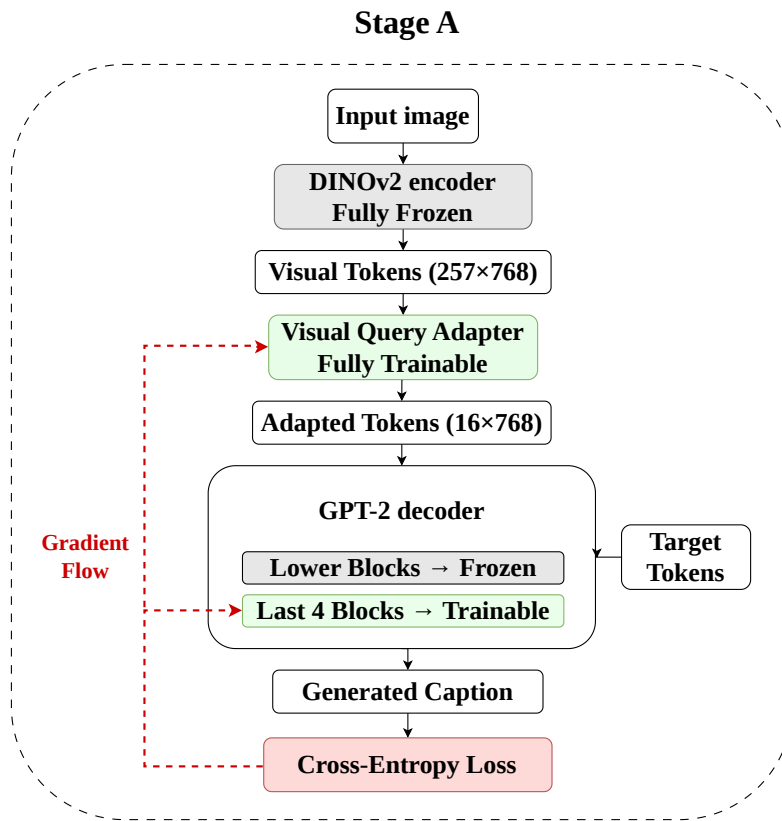


Figure 3.7: Stage A training architecture

3.6.2.2 Stage B: Selective Fine-Tuning of the Visual Encoder

Objective: The objective of Stage B is to refine the learned visual and linguistic representations using the stable cross modal alignment established during Stage A.

Trainable Components:

- Visual Query Adapter: Fully trainable
- GPT-2 Decoder: Last 8 transformer blocks are trainable
- DINOv2 Encoder: Last 2 transformer blocks are trainable

Theoretical Rationale: Once the adapter and decoder have learned a stable mapping between visual and textual modalities, selective fine-tuning of the visual encoder becomes feasible. Instead of updating

the entire encoder, only the final transformer blocks are unfrozen.

This selective adaptation strategy enables the model to refine high level task specific visual semantics relevant to image captioning while preserving the more general low level visual representations learned during large scale pretraining. Restricting trainable parameters to the final layers also stabilizes optimization and reduces the risk of catastrophic forgetting.

Figure 3.8 illustrates the training setup for Stage B, where the last two blocks of DINOv2 and the last eight blocks of GPT-2 are selectively unfrozen.

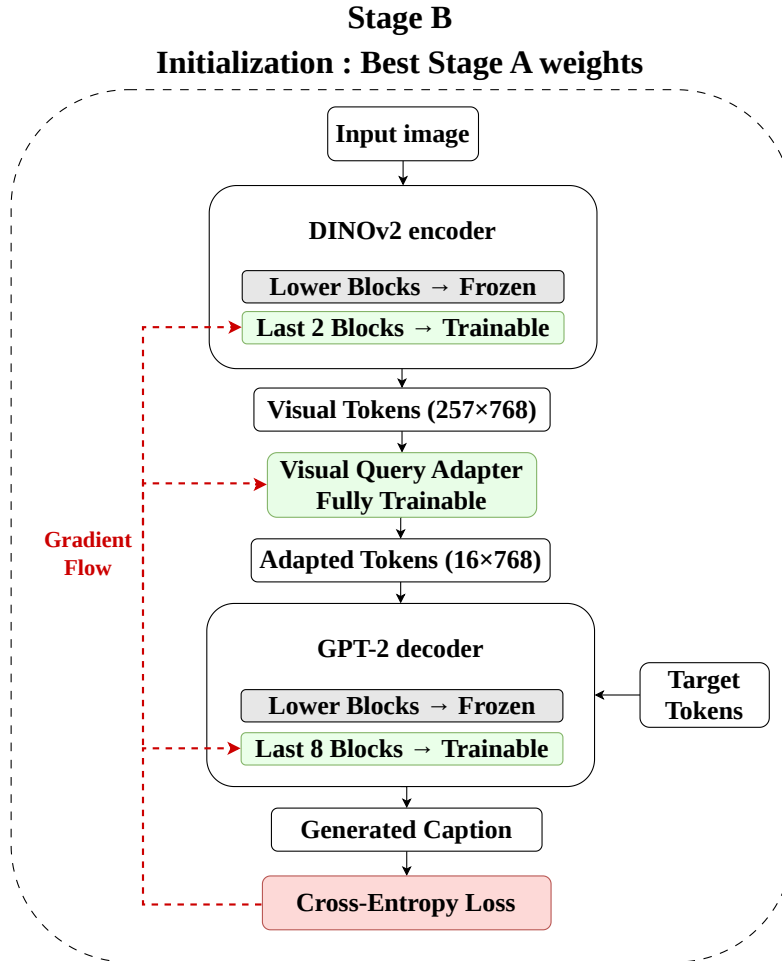


Figure 3.8: Stage B training architecture

3.6.3 Loss Function

The model is trained using a standard autoregressive language modeling objective based on categorical cross-entropy loss.

For a target caption represented as a sequence of tokens:

$$y = \{y_1, y_2, \dots, y_T\} \quad (3.12)$$

the model predicts a probability distribution over the vocabulary at each generation step. The objective is to minimize the negative log likelihood of the correct target tokens:

$$\mathcal{L}_{LM} = -\frac{1}{T} \sum_{t=1}^T \log P(y_t | y_{1:t-1}, I) \quad (3.13)$$

where:

- T denotes the length of the target caption.
- y_t represents the ground-truth token at position t .
- $y_{1:t-1}$ denotes all previous ground-truth tokens.
- I represents the encoded image features.
- $P(y_t | y_{1:t-1}, I)$ is the probability assigned by the model to the correct token conditioned on the previous tokens and the visual context.

During training, teacher forcing [72] is employed, where the decoder receives the ground-truth previous token rather than its own predicted token at each generation step. This improves optimization stability and accelerates convergence.

The cross-entropy objective is minimized during both training stages using gradient-based optimization.

3.7 Conclusion

In this chapter, we presented the architecture of the proposed image captioning model. We described the DINOv2 visual encoder, the Visual Query Adapter, and the GPT-2 decoder with cross-attention. We also explained the progressive two stage fine-tuning strategy used to preserve pre-trained knowledge while improving vision language alignment.

The model generates meaningful captions, but it still struggles with fine details in complex scenes. Even so, the proposed framework provides a solid foundation for the experimental evaluation presented in the next chapter.

Chapter 4

Implementation and Experimental Setup

4.1 Introduction

This chapter focuses on the practical execution of our project: turning our design into working code and analyzing its performance.

We begin by detailing the hardware, software, and datasets that formed the backbone of our implementation. We then walk through the core engineering steps, from visual feature extraction to full model training. Next, we present our experimental results, showcasing both quantitative metrics and qualitative examples of the generated captions.

Finally, we discuss the model's strengths and its current limitations. To conclude the chapter, we explore a real world application of this technology through a prototype language learning game, demonstrating how the model functions in a practical setting.

4.2 Experimental Environment

In the following, we detail the hardware and software configuration used to build, train, and evaluate our models.

4.2.1 Platform and Execution Environment

Kaggle Notebooks

All experiments were conducted on Kaggle, a cloud-based machine learning platform. This environment provided a streamlined workspace for dataset management and interactive notebook execution. To meet the intensive computational demands of the model, GPU acceleration was utilized, significantly reducing both training and inference times [77].



Figure 4.1: Kaggle logo

4.2.2 Hardware Configuration

Graphics Processing Unit (GPU)

Since deep learning needs a lot of computing power, the training phase leveraged GPU T4. This specialized processor was important for accelerating tensor operations and managing the large scale matrix multiplications inherent in Transformer based architectures [78].

4.2.3 Software Environment

Python

All experiments were implemented using Python, chosen for its robust ecosystem of scientific libraries and its standing as the industry standard for artificial intelligence research [79].



Figure 4.2: Python logo

4.2.4 Libraries and Frameworks

The implementation relied on a specialized stack of open-source libraries to handle distinct stages of the pipeline:

PyTorch

Served as the primary deep learning framework, providing the necessary flexibility for building, backpropagating, and optimizing the neural network architecture [80].



Figure 4.3: PyTorch logo

Transformers (Hugging Face)

Utilized to integrate state of the art pretrained components for both NLP and multimodal tasks, ensuring high performance feature extraction [81].

OpenCV

Employed for essential image preprocessing tasks, such as resizing and normalizing visual inputs before they were fed into the encoder [82].



Figure 4.4: OpenCV logo

NumPy

Used for efficient numerical operations and the management of multi-dimensional arrays during data manipulation [83].



Figure 4.5: NumPy logo

Matplotlib

Facilitated data visualization, allowing for the clear presentation of training loss curves and the qualitative analysis of generated captions [84].



Figure 4.6: Matplotlib logo

4.3 Datasets

To train our model, we used the MS COCO 2017 dataset, which is one of the most widely used benchmarks for image captioning. The dataset was used for large-scale training to learn the mapping between visual features and natural language captions.

4.3.1 Dataset Description

MS COCO Dataset

From the full MS COCO dataset, we randomly selected 45,000 images. These images were used as the main subset for training and evaluation.

The selected subset was split at the image level into three partitions:

- 80% for training → approximately 36,000 images
- 10% for validation → approximately 4,500 images
- 10% for testing → approximately 4,500 images

After mapping each image to its associated captions (caption level expansion), the dataset size increased significantly because each image contains multiple textual annotations (typically 5 captions per image, after filtering and preprocessing).

The final dataset statistics after preprocessing are summarized in Table 4.1.

Split	Number of Image Caption Pairs
Training	179,987
Validation	22,505
Testing	22,506

Table 4.1: MS COCO Dataset Statistics After Preprocessing

4.3.2 Dataset File Structure

Each dataset was organized in a specific directory structure to facilitate data loading and preprocessing. The following file structures were used:

MS COCO 2017 Dataset Structure :

```
coco2017/
  annotations/
    captions_train2017.json
    captions_val2017.json
    instances_train2017.json
    instances_val2017.json
    person_keypoints_train2017.json
    person_keypoints_val2017.json
  train2017/
    (training images: .jpg)
  val2017/
    (validation images: .jpg)
```

For this study, only the captions_train2017.json annotation file and the train2017 image directory were used, as the model was trained exclusively on the training split of MS COCO.

4.3.3 Data Splitting Strategy

To make sure the evaluation is fair, we split the dataset at the image level. This means each image and all its captions stay together in the same set (training, validation, or testing). This prevents data leakage, where the model might accidentally see test images during training and give misleading results.

4.3.3.1 Image Level Splitting

In COCO, every image has several captions. If we split by captions directly, the same image could appear in multiple sets, and the model would already know the image before testing. That would make the evaluation unfair.

To avoid this, we first split the unique image names into three sets. Then, we attached all captions back to their images. Finally, each image-caption pair was treated as a separate training sample, but the original split between the three sets stayed clean.

4.3.3.2 Splitting Procedure

We split the data in two simple steps:

1. **Split images first:** We took all unique image names and split them into training, validation, and test sets. Each image went to only one set. To make sure we can repeat the same split later, we used a fixed random seed (SEED = 42). The split was done using the `train_test_split` function from `scikit-learn`.
2. **Add captions back:** After splitting the images, we attached all captions back to their images. This turned each image-caption pair into one training example, but the original split between the three sets stayed untouched. This two-step method keeps the test images completely separate from training while still letting the model learn from every available caption.

```
expanded_df = grouped_df.explode("captions").reset_index(drop=True)
train_df = expanded_df[expanded_df["image_name"].isin(train_names)]
val_df = expanded_df[expanded_df["image_name"].isin(val_names)]
test_df = expanded_df[expanded_df["image_name"].isin(test_names)]
```

Listing 4.1: Caption-Level Expansion After Image-Level Split

4.3.3.3 Dataset Specific Splits

We applied the same image-level splitting logic to our MS COCO subset, but we adjusted the steps to fit the size of the data.

MS COCO

Since COCO is big, we first reduced it to a workable size. After cleaning, we ended up with a subset of 45,000 images taken from the original train2017 split.

We split these 45,000 images using a fixed random seed (SEED = 42) so the process can be repeated later. The split was:

- 80% for training → about 36,000 images
- 10% for validation → about 4,500 images
- 10% for testing → about 4,500 images

After that, we expanded each image back to all of its captions. Since each image comes with several descriptions, this turned our dataset into many more (image, caption) pairs.

The final numbers were:

- Training samples: 179,987
- Validation samples: 22,505
- Test samples: 22,506

This way, the model learns from every available caption, but the training, validation, and test sets stay completely separate.

Sampling :

```
sampled_images = coco_grouped_df["image_name"].sample(  
    n=min(MAX_SAMPLES, len(coco_grouped_df)),  
    random_state=SEED  
)
```

Listing 4.2: Sampling and Single-Stage Split for MS COCO

Splitting:

```
coco_train_names, coco_val_names = train_test_split(  
    coco_image_names,  
    test_size=0.1,  
    random_state=SEED  
)
```

Listing 4.3: Caption-Level Expansion for MS COCO

4.3.3.4 Final Split Summary

Tables 4.2, 4.3, and 4.4 below show how we split the data. We used a fixed random seed and kept each image in only one set. Then we expanded each image into its captions, so every (image, caption) pair became a separate training sample.

Property	MS COCO
Splitting Method	train_test split
Train Ratio	80%
Validation Ratio	10%
Test Ratio	10%
Random Seed	42
Splitting Level	Image level
Final Dataset Size	45,000 images

Table 4.2: Summary of Dataset Splitting Configurations

Dataset	Train Images	Validation Images	Test Images
MS COCO	36,000	4,500	4,500

Table 4.3: Summary of Image-Level Split (Unique Images)

Dataset	Train Samples	Validation Samples	Test Samples
MS COCO	179,987	22,505	22,506

Table 4.4: Summary of Caption-Level Samples (After Expansion)

4.3.4 Data Preprocessing

We prepared all images to match what DINOv2 expects. The same steps were applied to every image so the model sees consistent input.

4.3.4.1 Image Preprocessing

We resized, converted, normalized, and (during training) augmented the images.

Resizing and Cropping

All images ended up at 224×224 pixels, but the way we got there changed depending on the phase:

- **Training:** We used RandomResizedCrop. It picks a random part of the image and resizes it. This helps the model handle different object sizes and positions.

```
train_transform = transforms.Compose([
    transforms.RandomResizedCrop(
        IMAGE_SIZE,
        scale=(0.8, 1.0)
    ),
])
```

Listing 4.4: RandomResizedCrop Transformation for Training Images

- **Evaluation:** We used a fixed method. We resized the image to 256 pixels, then cropped the center to 224×224. This keeps results consistent and repeatable.

```
eval_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(IMAGE_SIZE),
])
```

Listing 4.5: Deterministic Resize and CenterCrop for Evaluation

Tensor Conversion

After resizing, we turned the image into a PyTorch tensor. This converts pixels into numbers the model can process.

```
transforms.ToTensor(),
```

Listing 4.6: Tensor Conversion for Model Input

Image Normalization

We adjusted the image values to match what DINOv2 expects. We used the official mean and standard deviation from AutoImageProcessor, or fell back to standard ImageNet values. This keeps training stable.

```
dino_processor = AutoImageProcessor.from_pretrained(DINO_MODEL_ID)
DINO_IMAGE_MEAN = list(dino_processor.image_mean)
DINO_IMAGE_STD = list(dino_processor.image_std)
```

Listing 4.7: Loading DINOv2 Normalization Statistics

```
transforms.Normalize(
    mean=DINO_IMAGE_MEAN,
    std=DINO_IMAGE_STD
),
```

Listing 4.8: Image Normalization Using DINOv2 Statistics

Data Augmentation

During training, we used several tricks to help the model generalize better and avoid just "memorizing" the training data. This included:

- Horizontal Flipping: Mirroring the image.
- Color Jittering: Slightly changing brightness and contrast.
- Random Erasing: Hiding small parts of the image.

We only used these during training. For testing and evaluation, we kept the images as they were (besides resizing) to get a fair and consistent result.

```

transforms.RandomHorizontalFlip(p=0.5),
transforms.ColorJitter(
    brightness=0.2,
    contrast=0.2,
    saturation=0.2,
    hue=0.05
),
transforms.RandomErasing(p=0.15),
]

```

Listing 4.9: Data Augmentation Transformations for Training

4.3.4.2 DINOv2 Compatibility

We picked the image settings (224×224 size, normalization values, and tensor format) to match what DINOv2 expects. This ensures the model receives inputs in the same format it was pretrained on, which improves stability and performance.

4.3.4.3 Text Preprocessing

We prepared the captions so they can be properly used by the GPT-2 model.

Cleaning: We converted all text to lowercase, removed URLs and symbols, kept only captions between 5 and 45 words, and removed duplicates.

```

def advanced_text_clean(text):
    text = str(text).strip().lower()
    text = re.sub(r"http\S+|www\.\S+", " ", text)
    text = re.sub(r"^[a-z0-9\s\.\,\!\@\?\'\-]", " ", text)
    text = re.sub(r"\s+", " ", text)
    return text.strip()

```

Listing 4.10: Caption Cleaning Function

Tokenization: We used the GPT-2 tokenizer and added three special tokens: <bos>(start), <eos>(end), and <pad>(empty space).

```

tokenizer.add_special_tokens({
    "bos_token": "<bos>",
    "eos_token": "<eos>",
    "pad_token": "<pad>"
})

```

Listing 4.11: Adding Special Tokens to GPT-2 Tokenizer

Attention masks: These are used to tell the model which parts are real words and which are padding. Padding tokens were set to 100 so they are ignored during loss calculation.

```

ids = [BOS_TOKEN_ID] + body[:max_len - 2] + [EOS_TOKEN_ID]
labels = [tok if tok != PAD_TOKEN_ID else -100 for tok in ids]

```

Listing 4.12: Caption Encoding and Label Masking

4.4 Implementation Details

We built and trained the model step by step.

4.4.1 Training Configuration

We used AdamW. Each part had its own learning rate:

- DINOv2: 1×10^{-6} (very small, so it keeps what it already learned)
- GPT-2: 1×10^{-5}
- Adapter: 5×10^{-7} (highest, because it starts from zero)

GPU memory was tight, so we used a batch size of 8. To help training, we used gradient accumulation over 4 steps. This gave us an effective batch size of 4.

We set weight decay to 0.01 to avoid overfitting.

The learning rate slowly increased at the start (warmup), then went down following a cosine curve. This made training smoother and helped the model end at a better spot.

The complete training configuration is summarized in Table 4.5.

Parameter	Value
Base learning rate	1×10^{-5}
Fine-tuning learning rate (GPT-2 decoder)	1×10^{-5}
Fine-tuning learning rate (adapter)	5×10^{-7}
Fine-tuning learning rate (DINOv2 encoder)	1×10^{-6}
Batch size	8
Gradient accumulation steps	4
Effective batch size	32
Weight decay	0.05
Optimizer	AdamW
Learning rate scheduler	Linear warmup + cosine decay
Warmup steps	20% of total steps
Base epochs	6

Table 4.5: Training Hyperparameters Configuration

```

BASE_LR = 1e-5
FINE_DINO_LR = 1e-6
FINE_GPT2_LR = 1e-5
FINE_ADAPTER_LR = 5e-7
TRAIN_BATCH_SIZE = 8
GRAD_ACCUM_STEPS = 4
WEIGHT_DECAY = 0.05

```

Listing 4.13: Training Hyperparameters Configuration

4.4.2 Fine-Tuning Strategy

We trained in two stages.

Stage A: We froze DINOv2 and trained only the adapter and the last four blocks of GPT-2. This taught the model the basic link between images and text without breaking what the model already knew.

```
caption_model.freeze_dino()
caption_model.freeze_decoder()
caption_model.unfreeze_decoder_last_blocks(4)
```

Listing 4.14: Stage A Configuration (Partial Fine-Tuning)

Stage B: We started from Stage A's best checkpoint. This time, we unfroze the last two blocks of DINOv2 and the last eight blocks of GPT-2. The adapter stayed trainable. This helped the model fine-tune the details of captioning while still keeping most pretrained knowledge safe.

```
caption_model.unfreeze_dino_last_blocks(2)
caption_model.unfreeze_decoder_last_blocks(8)
```

Listing 4.15: Stage B Configuration (Selective Fine-Tuning)

Table 4.6 summarizes which components were trainable at each stage.

Stage	DINOv2 Encoder	Visual Query Adapter	GPT-2 Decoder
Stage A	Frozen	Trainable	Last 4 blocks
Stage B	Last 2 blocks trainable	Trainable	Last 8 blocks

Table 4.6: Trainable Components per Fine-Tuning Stage

4.4.3 Forward Propagation Pipeline

Whether we are training or testing, the model follows the same steps to turn an image into a caption.

First, we resize and normalize the image, then pass it to DINOv2. This encoder pulls out high level visual features from the image.

Next, the Visual Query Adapter takes these features and reshapes them so GPT-2 can understand them. Finally, GPT-2 generates the caption one word at a time. It looks at both the image and the words it has already written to decide what comes next.

During training, we compare the generated caption with the real one to calculate the loss and improve the model.

4.4.4 Model Architecture Implementation

The implemented architecture combines a DINOv2 visual encoder, a Visual Query Adapter, and a GPT-2 decoder. The adapter transforms visual embeddings into a representation compatible with the language model through cross-attention mechanisms. The simplified implementation of the main architecture is shown below.

VisualQueryAdapter:

```
class VisualQueryAdapter(nn.Module):
    def __init__(self, visual_dim, hidden_dim, num_queries=16, nhead=8, dropout=0.1):
        super().__init__()

        self.input_proj = nn.Linear(visual_dim, hidden_dim)
        self.input_norm = nn.LayerNorm(hidden_dim)

        self.query_tokens = nn.Parameter(
            torch.randn(1, num_queries, hidden_dim) * 0.02
        )

        self.cross_attn = nn.MultiheadAttention(
            embed_dim=hidden_dim,
            num_heads=nhead,
            dropout=dropout,
            batch_first=True,
            bias=False
        )
```

Listing 4.16: Simplified Implementation of VisualQueryAdapter

CaptionModel:

```
class CaptionModel(nn.Module):
    def __init__(self, dino, decoder, num_queries=16):
        super().__init__()

        self.dino = dino
        self.decoder = decoder

        self.dino_hidden = self.dino.config.hidden_size
        self.gpt_hidden = self.decoder.config.n_embd

        self.adapter = VisualQueryAdapter(
            self.dino_hidden,
            self.gpt_hidden,
            num_queries=num_queries
        )
```

Listing 4.17: Simplified Implementation of CaptionModel

4.4.5 Evaluation Metrics

Measuring caption quality is tricky because there is rarely a single "correct" description for an image. We used multiple metrics to assess different aspects of the results, from basic word accuracy to deeper semantic meaning.

BLEU (1-4): Measures the overlap of word groups (n-grams) between the generated and reference captions.

METEOR & ROUGE-L: These go beyond exact word matching by considering synonyms and overall sentence structure, providing a better sense of readability.

CIDEr & SPICE: Made for image captioning. CIDEr checks agreement with human captions. SPICE checks if the meaning is correct.

Throughout the training process, we monitored BLEU-4 and CIDEr as our primary benchmarks for selecting the best model checkpoints and comparing different experimental runs.

The evaluation strategy is summarized in Table 4.7.

Metric	Evaluation Aspect	Usage in This Work
BLEU-1/2/3	N-gram overlap	General caption quality evaluation
BLEU-4	Higher-order sentence matching	Main validation metric
METEOR	Semantic and lexical similarity	Complementary evaluation
ROUGE-L	Sentence structure similarity	Fluency evaluation
CIDEr	Consensus with human captions	Main image captioning metric

Table 4.7: Evaluation Metrics Used in Experiments

4.4.6 Inference and Caption Generation

At test time, DINOv2 pulls features from the image. The Adapter reshapes them for GPT-2, which then writes the caption word by word, starting with a BOS token and stopping at EOS or when it hits the length limit.

We used beam search to pick better words. Beam size was 5 during validation and 5 during final testing.

We also added a few rules to avoid repetition and keep captions clean: a repetition penalty, a no repeat n-gram size, and min/max length limits.

The complete inference configuration is summarized in Table 4.8.

Parameter	Validation	Testing	Description
Beam size	5	5	Number of beams used in beam search
Repetition penalty	1.1	1.1	Penalizes repeated tokens
No-repeat n-gram size	2	2	Prevents repeated 3-gram sequences
Length penalty	1.2	1.2	Controls preference for caption length
Min new tokens	5	5	Minimum generated caption length
Max new tokens	25	25	Maximum generated caption length

Table 4.8: Inference Configuration for Caption Generation

```
gen_kwargs = dict(
    num_beams=5,
    do_sample=False,
    early_stopping=True,
    no_repeat_ngram_size=2,
    repetition_penalty=1.1,
    length_penalty=1.2,
    min_new_tokens=5,
    max_new_tokens=25,
    pad_token_id=PAD_TOKEN_ID,
    eos_token_id=EOS_TOKEN_ID,
)
```

Listing 4.18: Inference Generation Configuration

4.5 Training Process and Experimental Analysis

This section describes how the model was trained and how its performance was evaluated.

4.5.1 Training Workflow

The training process followed a clear sequence: loading the data, preprocessing images and text, running the forward pass, calculating the loss, then backpropagation, validation, and checkpoint saving. We used gradient accumulation (4 steps) and Automatic Mixed Precision (AMP) to reduce GPU memory usage and speed up training.

We also enabled checkpoint resuming so we could pick up exactly where we left off if a session was interrupted.

4.5.2 Loss Monitoring and Training Curves

During training, we monitored both training loss and validation loss to check how well the model was learning. We also tracked BLEU-4, METEOR, ROUGE-L, and CIDEr to evaluate the quality of the generated captions.

In Stage A, the model learned gradually and the scores improved from one epoch to another. Training loss decreased clearly, while BLEU-4, METEOR, and ROUGE-L increased over time. However, the validation loss remained relatively high, which showed that the model was still limited by the frozen visual encoder.

The evolution of losses and evaluation metrics during Stage A is shown in Figure 4.7

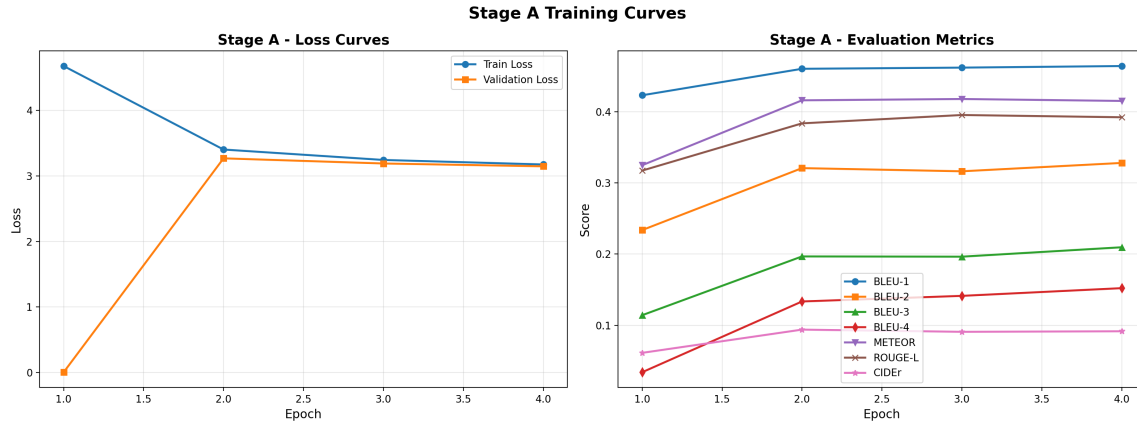


Figure 4.7: Curves Across Fine-Tuning Stage A

In Stage B, the results improved significantly after partially unfreezing the last blocks of DINOv2 and expanding GPT-2 fine-tuning. The training and validation losses became lower and more stable, while all captioning metrics achieved much higher scores compared to Stage A. BLEU-4 increased to around 0.33, METEOR reached about 0.50, ROUGE-L exceeded 0.57, and CIDEr achieved its best performance during this stage.

The evolution of losses and evaluation metrics during Stage B is shown in Figure 4.8

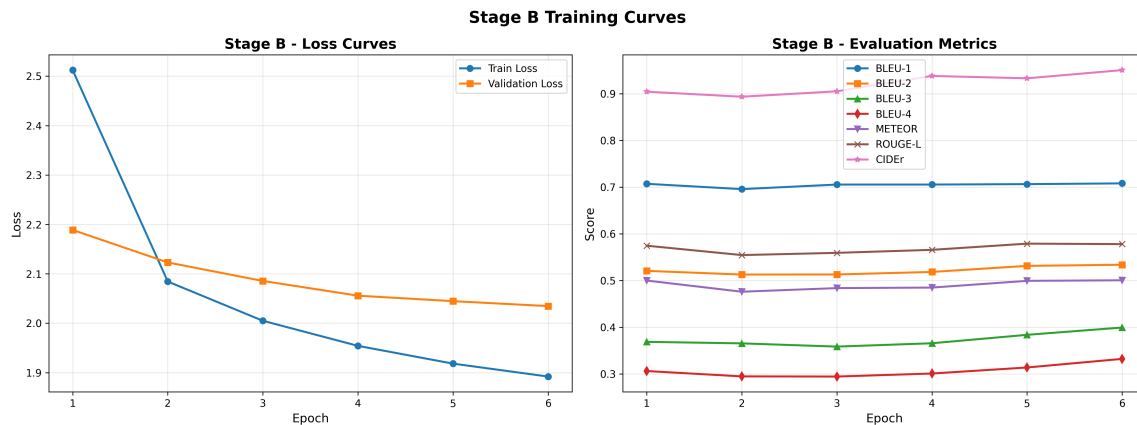


Figure 4.8: Curves Across Fine-Tuning Stage B

Figure 4.2 shows the evolution of the losses and evaluation metrics across both training stages.

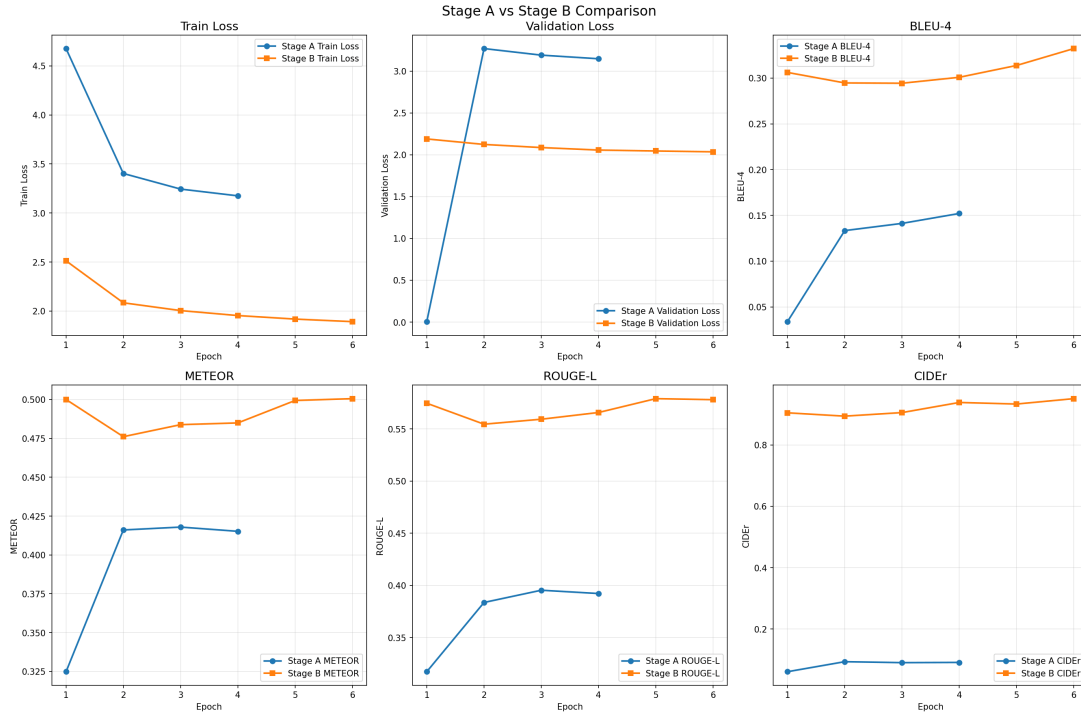


Figure 4.9: a comparison between two Curves Across Fine-Tuning Stages

Overall, the curves indicate that the two stage fine-tuning strategy improved caption quality effectively without showing strong signs of overfitting.

4.5.3 Checkpointing and Model Selection

During training, we saved checkpoints regularly to avoid losing progress and to keep the best model version.

Two types of checkpoints were used:

- Last checkpoint: saved after every epoch so training could continue later if interrupted.
- Best checkpoint: saved only when the validation BLEU-4 score improved.

BLEU-4 was used as the main metric for selecting the best model, while CIDEr was also monitored to confirm caption quality.

We also used early stopping with a patience of four epochs. If BLEU-4 did not improve for four consecutive evaluations, training stopped automatically to avoid unnecessary computation and overfitting.

Each checkpoint stored the model weights, optimizer state, scheduler state, scaler state, and epoch number, which allowed training to resume exactly from the last saved step.

4.6 Experimental Results

We tested our model on the MS COCO test set using standard image captioning metrics: BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR, ROUGE-L, CIDEr, and SPICE.

The evaluation was done after Stage B (selective fine-tuning) using beam search decoding. We mainly watched BLEU-4 and CIDEr to compare results.

The numbers show that our two-stage training helped. Unfreezing the last two blocks of DINOv2 and the last eight blocks of GPT-2 in Stage B gave better captions than Stage A alone.

The results are shown in Table 4.9 below.

Metric	Stage A	Stage B
Train Loss	3.1750	1.8921
Val Loss	3.1467	2.0347
Best BLEU-1	0.4643	0.7082
Best BLEU-2	0.3281	0.5339
Best BLEU-3	0.2096	0.3994
Best BLEU-4	0.1521	0.3323
Best METEOR	0.4152	0.5006
Best ROUGE-L	0.3923	0.5781
Best CIDEr	0.0915	0.9509

Table 4.9: Stage A vs Stage B Performance Comparison

4.6.1 Quantitative Results

After finishing the two stage training, we evaluated the final model on the test set. These are images the model had never seen before.

The results in Table 4.10 show that the model works well overall. It generates correct descriptions for most images. However, it still misses some fine details, especially in busy or complex scenes. The SPICE score (0.2038) was calculated in the Kaggle environment, and it reflects that the model captures the main meaning but does not always get every detail right.

Metric	Score
Loss	1.9499
BLEU-1	0.7162
BLEU-2	0.5519
BLEU-3	0.4109
BLEU-4	0.3090
METEOR	0.4846
ROUGE-L	0.5637
CIDEr	1.0715
SPICE	0.2038

Table 4.10: Final Test Results on MS COCO

4.6.2 Qualitative Results

While quantitative metrics provide a useful benchmark, qualitative analysis reveals how well the model actually understands visual content.

Figure 4.10 shows representative examples of successfully generated captions from our model on the MS COCO test set.

Overall, our model performs well on clear, simple scenes, producing fluent and accurate descriptions that correctly identify main objects and actions. When the image contains a single prominent subject, the generated captions closely resemble human-written references.

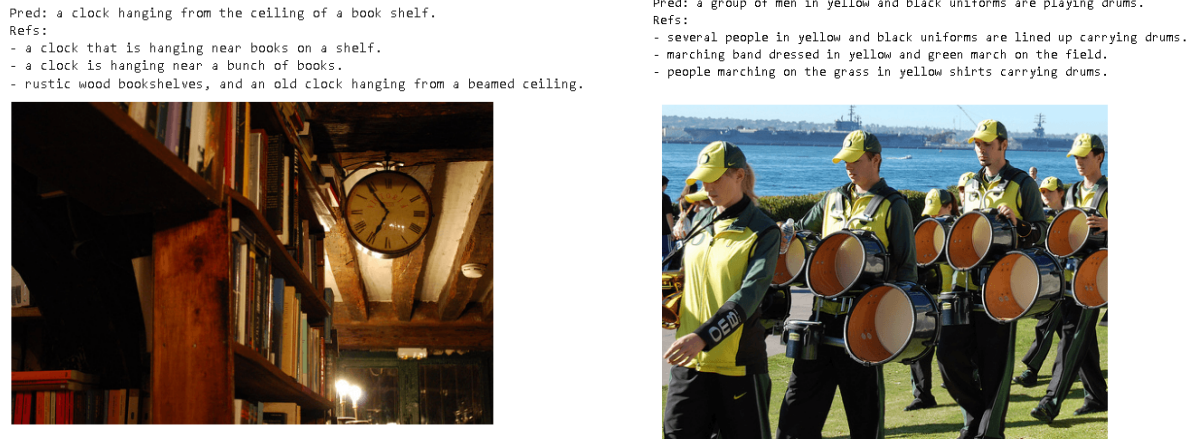


Figure 4.10: Examples of Correct Caption Generation

4.6.3 Comparison with State of the Art

To contextualize our model’s performance, we compare against several influential image captioning methods on the MS COCO dataset.

Table 4.11 summarizes key metrics from the literature alongside our results.

Model	BLEU-4	METEOR	CIDEr	SPICE
Att2in [85]	0.332	0.272	1.085	0.205
Up-Down [86]	0.363	0.277	1.200	0.214
AoANet [87]	0.394	0.292	1.231	0.220
M2 Transformer [52]	0.391	0.299	1.281	0.221
BLIP [55]	0.394	0.306	1.312	0.227
GIT [88]	0.408	0.309	1.388	—
BLIP-2 [57]	0.401	0.312	1.346	0.232
Ours (Final Test)	0.309	0.485	1.072	0.204

Table 4.11: Comparison with state of the art image captioning models on MS COCO.

Compared with established image captioning models on the MS COCO benchmark, our framework demonstrates a distinctive performance profile. The most notable result is the METEOR score of 0.485, which is substantially higher than all compared methods. Since METEOR rewards synonym matching, stemming, and semantic similarity, this result suggests that the generated captions remain semantically aligned with the reference descriptions even when different wording is used.

However, the model achieves lower BLEU-4 and CIDEr scores (0.309 and 1.072, respectively) than recent state of the art approaches such as BLIP, BLIP-2, and GIT. These metrics place greater emphasis on exact n-gram matching and agreement with human reference captions. This indicates that while the generated captions are often semantically meaningful, they do not always follow the most common lexical patterns found in the reference annotations.

The SPICE score of 0.204 is also slightly lower than several competing methods. Since SPICE evaluates objects, attributes, and relationships within a scene, this result suggests that the model may occasionally miss fine-grained visual details, particularly in complex images containing multiple interacting objects.

Overall, the proposed framework demonstrates strong semantic caption generation capabilities, as reflected by its high METEOR score, while still leaving room for improvement in capturing detailed scene relationships and increasing agreement with reference captions according to BLEU, CIDEr, and SPICE metrics.

4.6.4 Discussion

The experimental results show that combining DINOv2, GPT-2, and the proposed Visual Query Adapter forms an effective framework for image captioning. The improvement from Stage A to Stage B confirms that selectively fine-tuning the visual encoder enhances visual-text alignment while preserving pretrained knowledge.

A key contribution of the model is the Visual Query Adapter, which bridges the gap between visual features and the language model through projection and cross-attention. This enables stronger multimodal interactions and improves semantic consistency in the generated captions.

The high METEOR score (0.485) reflects strong semantic alignment between generated captions and reference descriptions, indicating that the model produces flexible and meaningful paraphrases rather than fixed expressions. This is largely due to the rich visual representations of DINOv2 and the linguistic capabilities of GPT-2.

In contrast, the lower BLEU-4 score (0.309) compared to models such as BLIP and GIT suggests reduced exact n-gram overlap, as the model tends to prioritize semantic correctness over strict textual matching. However, CIDEr (1.072) remains competitive, indicating reasonable agreement with human annotations.

Despite these strengths, the model still struggles with fine-grained details, especially in cluttered scenes with multiple objects. This is reflected in the SPICE score (0.204), which is slightly lower than some recent approaches. The generated captions also tend to be concise, which may be influenced by both the MS COCO dataset characteristics and the decoder's tendency to focus on the most salient visual elements.

These limitations are also related to the relatively small size of GPT-2, the partial fine-tuning strategy, and the lack of explicit object-level reasoning mechanisms.

4.7 Gamification of Learning Using Image Captioning

To demonstrate a practical use case of the proposed image captioning framework, we developed a prototype web application called **Caption Quest**. Rather than limiting the work to offline evaluation metrics, the application integrates the trained DINOv2 and GPT-2 model into an interactive environment where users can directly interact with AI generated captions.

The application was implemented using Streamlit and provides a complete end to end workflow.

A user uploads an image, writes a short description in English, and receives a caption generated by the model. The system then compares both descriptions and provides immediate feedback based on lexical overlap, sentence length, and vocabulary usage.

4.7.1 The Gameplay Loop

The interaction process consists of four main steps:

1. The user uploads an image.
2. The user writes a textual description in English.
3. The proposed DINOv2 and GPT-2 framework generates a reference caption.
4. The application evaluates the user description and assigns experience points (XP).

To increase engagement, several gamification mechanisms were integrated into the platform:

- **Word by Word Feedback:** highlights similarities and differences between the learner's description and the generated caption.
- **Word of the Day:** encourages the use of new vocabulary through bonus rewards.
- **Daily Challenges:** provide simple language related objectives that motivate regular practice.
- **XP and Achievement System:** tracks user progress through levels and milestones.
- **Streak Tracking:** records consecutive days of activity to encourage consistency.

From a technical perspective, Caption Quest serves as a practical validation of the proposed image captioning model. It demonstrates that the generated captions can be integrated into a real-time application and used as the basis for interactive educational activities rather than only benchmark evaluation.

Although the current version is a prototype, it highlights the potential of combining image captioning with educational gamification. Future improvements may include multilingual support, more advanced feedback mechanisms, learner performance analytics, and deployment as an online educational platform accessible to a wider audience.

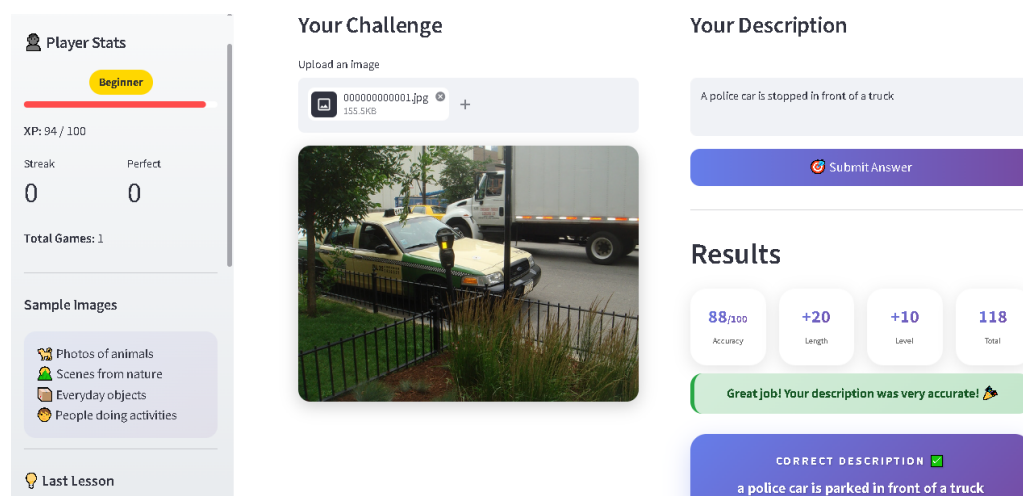


Figure 4.11: Example of Caption Quest Application Interface

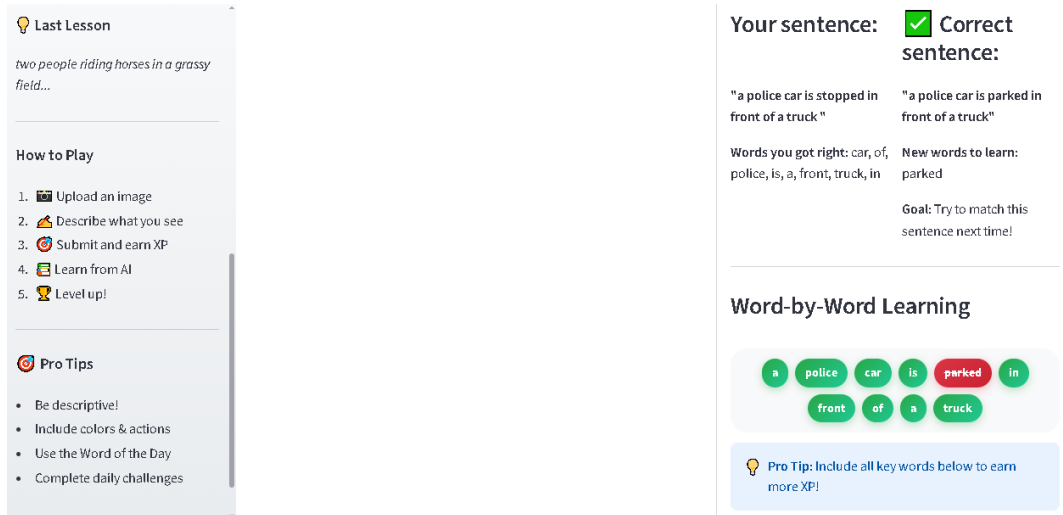


Figure 4.12: Example of Caption Quest Application Interface

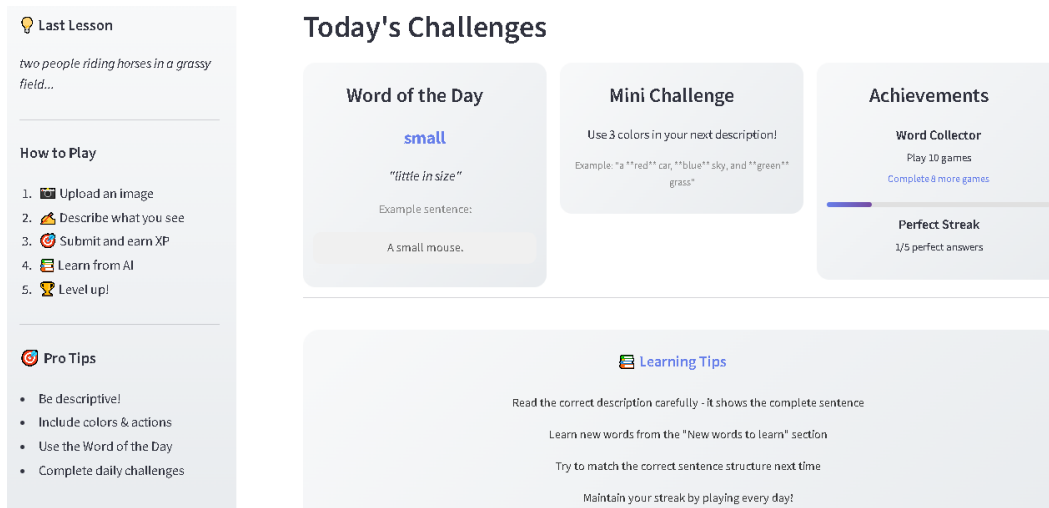


Figure 4.13: Example of Caption Quest Application Interface

4.8 Conclusion

This chapter presented the implementation and experimental evaluation of the proposed image captioning framework based on DINOv2, GPT-2, and the Visual Query Adapter. Results demonstrated consistent improvements from Stage A to Stage B across all metrics, confirming the effectiveness of the progressive fine-tuning strategy. The final model achieved competitive performance on MS COCO, particularly in METEOR and CIDEr scores, indicating strong semantic alignment between generated and reference captions. Qualitative analysis showed coherent descriptions, though fine-grained details in complex scenes remain challenging.

Beyond evaluation, the framework was validated through Caption Quest, a prototype educational application demonstrating practical applicability. Overall, the results confirm the effectiveness of combining a self-supervised vision encoder with a pretrained language model through a lightweight adapter, with future work focused on improving fine-grained understanding and reducing computational cost.

General Conclusion

In this thesis, we investigated the problem of image captioning by developing a multimodal framework that combines a DINOv2 Vision Transformer, a GPT-2 language model, and a proposed Visual Query Adapter designed to bridge the gap between visual and textual representations. The framework was trained and evaluated on the MS COCO dataset using a progressive two stage fine-tuning strategy aimed at preserving pretrained knowledge while enabling task-specific adaptation.

The experimental results demonstrated the effectiveness of the proposed approach. In particular, the transition from Stage A to Stage B consistently improved performance across all evaluation metrics, confirming the benefits of selectively fine-tuning the visual encoder. The final model achieved a BLEU-4 score of 0.309, a METEOR score of 0.485, a CIDEr score of 1.072, and a SPICE score of 0.204. Among these results, the exceptionally high METEOR score highlights the model's ability to generate semantically meaningful and linguistically flexible captions, indicating strong alignment between visual understanding and natural language generation.

One of the main contributions of this work is the introduction of the Visual Query Adapter, which enables effective interaction between DINOv2 visual features and the GPT-2 decoder through learned projections and cross-attention mechanisms. Combined with the progressive fine-tuning strategy, this design provides an efficient alternative to full end-to-end training while maintaining competitive captioning performance.

The qualitative analysis further showed that the model is capable of producing coherent and contextually relevant descriptions across a wide variety of images. Nevertheless, several limitations remain. The model may overlook fine-grained attributes, secondary objects, or complex relationships in crowded scenes, and generated captions tend to be relatively concise. These limitations suggest that detailed scene understanding remains a challenging aspect of image captioning despite recent advances in vision language models.

Beyond benchmark evaluation, this thesis also explored the practical applicability of image captioning through Caption Quest, a gamified educational prototype that transforms AI generated captions into an interactive language-learning experience. This application demonstrates that image captioning models can serve not only as research tools but also as the foundation for real world educational systems that encourage vocabulary acquisition, descriptive writing, and visual literacy.

Future work may focus on extending the framework through larger and more diverse datasets, multilingual caption generation, more powerful language models, and enhanced visual reasoning mechanisms. Additional research could also investigate reinforcement learning approaches and semantic optimization techniques to improve fine-grained caption quality and scene comprehension.

Overall, this work demonstrates that combining DINOv2 with GPT-2 through a lightweight adaptation module and a progressive training strategy provides an effective solution for image captioning. The proposed framework successfully bridges visual and linguistic representations, produces semantically rich captions, and illustrates how modern multimodal AI systems can be translated into practical applications beyond traditional benchmark evaluation.

Bibliography

- [1] Stanford University. Cs224n: Natural language processing with deep learning, 2024. URL <https://web.stanford.edu/class/cs224n/>.
- [2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Stanford University, 3rd edition, 2023. URL <https://web.stanford.edu/~jurafsky/slp3/>. Draft version.
- [3] Christopher Olah. Understanding lstm networks, 2015. URL <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [4] Hugging Face. Encoder-decoder models, 2024. URL <https://huggingface.co/blog/encoder-decoder>.
- [5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. URL <https://arxiv.org/abs/1406.1078>.
- [6] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014. URL <https://arxiv.org/abs/1409.3215>.
- [7] Jay Alammar. Visualizing neural machine translation mechanics of seq2seq models with attention, 2018. URL <https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>.
- [8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 2015. URL <https://arxiv.org/abs/1409.0473>.
- [9] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015. URL <https://arxiv.org/abs/1508.04025>.
- [10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. URL <https://arxiv.org/abs/1706.03762>.
- [11] Deepgram. Visualizing and explaining transformer models from the ground up, 2023. URL <https://deepgram.com/learn/visualizing-and-explaining-transformer-models-from-the-ground-up>.
- [12] Harvard NLP. The annotated transformer, 2018. URL <https://nlp.seas.harvard.edu/2018/04/03/attention.html>.

- [13] Stack Overflow. How is position-wise feed-forward neural network implemented for transformers?, 2023. URL <https://stackoverflow.com/questions/74979359/how-is-position-wise-feed-forward-neural-network-implemented-for-transformers>.
- [14] The Deep Hub. Positional encoding explained: A deep dive into transformer pe, 2023. URL <https://medium.com/thedeephub/positional-encoding-explained-a-deep-dive-into-transformer-pe-65cfe8cfe10b>.
- [15] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018(1):7068349, 2018.
- [16] Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*, 13(22):4712, 2021.
- [17] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.
- [18] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.
- [19] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Denscap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4565–4574, 2016.
- [20] Wamidh K. Mutlag, Shaker K. Ali, Zahoor M. Aydam, and Bahaa H. Taher. Feature extraction methods: A review. In *Journal of Physics: Conference Series*, volume 1591, page 012028. IOP Publishing, 2020. doi: 10.1088/1742-6596/1591/1/012028.
- [21] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [23] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1):53, 2021.
- [24] upGrad. Basic cnn architecture, 2023. URL <https://www.upgrad.com/blog/basic-cnn-architecture/>.
- [25] Abdulrahman Alqahtani et al. A review of transformer-based approaches for image captioning. *Applied Sciences*, 13(19):11103, 2023. URL <https://www.mdpi.com/2076-3417/13/19/11103>.
- [26] MDPI. Image captioning: A review, 2023. URL <https://www.mdpi.com/2076-3417/13/19/11103>.
- [27] Andrea Vedaldi and Andrew Zisserman. Vgg convolutional neural networks practical. *Department of Engineering Science, University of Oxford*, 66:27, 2016.
- [28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

- [29] Christian Szegedy, Wei Liu, Yangqing Jia, et al. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- [30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2021. URL <https://arxiv.org/abs/2010.11929>.
- [31] Alexey Dosovitskiy et al. Vision transformer architecture figure, 2020. URL <https://www.semanticscholar.org/paper/An-Image-is-Worth-16x16-Words%3A-Transformers-for-a-t-Dosovitskiy-Beyer/268d347e8a55b5eb82fb5e7d2f800e33c75ab18a/figure/0>.
- [32] Alexey Dosovitskiy et al. Vision transformer model configurations, 2020. URL <https://www.semanticscholar.org/paper/An-Image-is-Worth-16x16-Words%3A-Transformers-for-at-Dosovitskiy-Beyer/268d347e8a55b5eb82fb5e7d2f800e33c75ab18a>.
- [33] Aditya Jain. Flickr8k dataset for image captioning, 2020. URL <https://www.kaggle.com/datasets/adityajn105/flickr8k>.
- [34] Leslie F. Sikos. The semantic gap. In *Description Logics in Multimedia Reasoning*, pages 23–33. Springer, 2017. doi: 10.1007/978-3-319-54066-5_3.
- [35] Alessandro Favero, Luca Zancato, Matthew Trager, Siddharth Choudhary, Pramuditha Perera, Alessandro Achille, Ashwin Swaminathan, and Stefano Soatto. Multi-modal hallucination control by visual information grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14303–14312, June 2024.
- [36] Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. Women also snowboard: Overcoming bias in captioning models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 771–787, 2018.
- [37] Jindong Wang, Charley Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Hua Zeng, and Philip S. Yu. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2022. doi: 10.1109/TPAMI.2022.3199849.
- [38] Mukta Kulkarni. Digital accessibility: Challenges and opportunities. *IIMB Management Review*, 31(1):91–98, 2019.
- [39] Azratul Ain Nadiyah Ismail, Saiful Farik Mat Yatin, Nur Imana Mustakin, Aina Musfirah Mustafa Kamal, and Roslaizam Daud. Artificial intelligence in information management: Transforming information retrieval, classification, and management systems. *Journal of Information and Knowledge Management (JIKM)*, 15(SI2), 2025. doi: 10.24191/jikm.v15iSI2.7681.
- [40] Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images using 1 million captioned photographs. In *NIPS*, 2011.
- [41] Polina Kuznetsova, Vicente Ordonez, Tamara L. Berg, and Alexander C. Berg. Collective generation of natural image descriptions. In *ACL*, 2012.
- [42] Rebecca Mason and Eugene Charniak. Nonparametric method for data-driven image captioning. In *ACL*, 2014.
- [43] Jacob Devlin, Saurabh Gupta, Ross Girshick, and Margaret Mitchell. Exploring nearest neighbor approaches for image captioning. In *EMNLP*, 2015.
- [44] Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, and Tamara L. Berg. Baby talk: Understanding and generating simple image descriptions. In *CVPR*, 2011.

- [45] Margaret Mitchell, Jesse Dodge, Amit Goyal, Kota Yamaguchi, Karl Stratos, Xufeng Han, Arthur Mensch, Alexander C. Berg, and Tamara L. Berg. Midge: Generating image descriptions from computer vision detections. In *EACL*, 2012.
- [46] Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, and Tamara L. Berg. Baby talk: Understanding and generating image descriptions with improved language modeling. In *ACL*, 2013.
- [47] Yezhou Yang, Ching Lik Teo, Hal Daume, and Yiannis Aloimonos. Corpus-guided sentence generation of natural images. In *EMNLP*, 2011.
- [48] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.
- [49] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.
- [50] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.
- [51] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *CVPR*, 2016.
- [52] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *CVPR*, 2020.
- [53] Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. In *NeurIPS*, 2019.
- [54] Ron Mokady, Amir Hertz, and Amit H. Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.
- [55] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, 2022.
- [56] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [57] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, pages 19730–19742. PMLR, 2023.
- [58] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision, 2023.
- [59] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Report*, 2019.
- [60] Tsung-Yi Lin et al. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [61] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. doi: 10.1162/tacl_a_00166.

- [62] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013. doi: 10.1613/jair.3994.
- [63] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.
- [64] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL Workshop*, 2005.
- [65] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop*, 2004.
- [66] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.
- [67] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *ECCV*, 2016.
- [68] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.
- [69] Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.
- [70] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [71] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [72] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.
- [73] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks. *Psychology of Learning and Motivation*, 24:109–165, 1989.
- [74] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.
- [75] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2019.
- [76] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(9):13693–13696, 2019.
- [77] Kaggle / Google LLC. Kaggle: Data science platform, 2025. URL <https://www.kaggle.com/>. Cloud-based machine learning platform with GPU acceleration.
- [78] NVIDIA Corporation. Nvidia tesla t4 gpu architecture, 2025. URL <https://www.nvidia.com/en-us/data-center/tesla-t4/>. Turing architecture GPU with Tensor Cores for deep learning acceleration.

- [79] Python Software Foundation. Python programming language, 2025. URL <https://www.python.org/>. High-level programming language for scientific computing and AI.
- [80] Meta AI. Pytorch: Deep learning framework, 2025. URL <https://pytorch.org/>. Open source machine learning framework with GPU acceleration.
- [81] Hugging Face. Hugging face transformers: State-of-the-art machine learning, 2025. URL <https://huggingface.co/docs/transformers>. Library for pretrained models in NLP, vision, and audio.
- [82] OpenCV Team. Opencv: Open source computer vision library, 2025. URL <https://opencv.org/>. Computer vision and machine learning software library.
- [83] NumPy Developers. Numpy: Fundamental package for scientific computing, 2025. URL <https://numpy.org/>. N-dimensional array computing library for Python.
- [84] Matplotlib Development Team. Matplotlib: Visualization with python, 2025. URL <https://matplotlib.org/>. Comprehensive library for creating static and interactive visualizations.
- [85] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 375–383, 2017.
- [86] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6077–6086, 2018.
- [87] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4634–4643, 2019.
- [88] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*, 2022.