Faculty of Media Engineering and Technology
German University in Cairo

Image Captioning

Bachelor Thesis

by

Michael Maged Farouk Weesa

Supervised by

Prof. Dr. Hossam El-Din Hassan Abd El Munim

Date: May 2025

Declaration

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor of Science (B.Sc.) in Computer Engineering at the German University in Cairo (GUC),

(ii) due acknowledgement has been made in the text to all other material used

---

Name: Michael Maged Farouk Weesa

Date: 29/05/2025

Acknowledgement

I would like to thank Prof. Dr. Hossam El-Din Hassan Abd El Munim for his support, guidance, and advice during my thesis. His feedback helped me improve my work and stay on track.

I am also very grateful to my parents and friends for always being there for me, encouraging me, and believing in me while I worked on this project.

Thank you all.

Contents

Abstract

This thesis investigates image and video captioning using deep learning. Two models are developed and compared. A traditional CNN-LSTM architecture, and a more advanced CNN-Transformer architecture. Both models are trained and evaluated on the Flickr30k dataset. It contains over 31,000 images and 158,000 human-written captions.

The CNN-LSTM model uses a pre-trained ResNet50 for visual feature extraction and an LSTM network for sequential caption generation. The CNN-Transformer model combines EfficientNetB0 with a Transformer-based decoder, using self-attention for improved context and language modeling. Data preprocessing steps include image augmentation, text cleaning, tokenization, and embedding with GloVe vectors.

Model performance is evaluated using BLEU scores and human assessment. The CNN-Transformer consistently produces more natural and accurate captions. It achieves an average BLEU-1 score of 0.55, while the CNN-LSTM model peaks at a BLEU-1 score of 0.57. However, human evaluation shows that the Transformer-based model provides more satisfying and descriptive captions in practice.

This work demonstrates the strengths and trade-offs of each approach and highlights the benefits of self-attention in image and video captioning. The thesis provides a full pipeline, from data collection to model deployment, and discusses challenges, limitations, and directions for future improvement in end-to-end vision-language models.

Chapter 1

Introduction

1.1 Problem Definition

In the past few years, making machines understand visual media has been one of the main goals of computer vision and artificial intelligence. One significant challenge was Image Captioning, where a computer receives and image, and creates a meaningful caption for it. The main problem was the combination of computer vision and natural language processing, which requires a system to express objects and scenes in a clear and concise manner, giving a grammatically correct sentence.

Generating accurate and contextually relevant captions remains a difficult challenge even with great progress toward image understanding and language modeling. The work is challenging because of variances in image content, uncertainty in visual scenes, and the variety of possible true descriptions. Moreover, the model has to be able to generate natural language output in line with human-like descriptions by generalizing over a broad spectrum of new and unseen images.

This thesis proposes a simplified encoder-decoder architecture to solve the image captioning challenge. The aim is to create a system able to generate coherent captions for fresh images by learning from a dataset of image-caption pairs.

1.2 Importance and Impact

Image captioning is important because it connects visual data and natural language, which makes machines understand and communicate visual content in a way that's

understandable by humans. This task has a lot of applications and importance in a lot of domains.

One of the most important benefits of image captioning is how it helps visually impaired people. Generating written and spoken descriptions of images helps them understand the world around them, whether it's a scene in their environment, or any type of content on the web.

Image Captioning also plays an important role in content management and accessibility, making it easier to tag, organize, and summarize big amounts of visual content, in an automated way. From social media platforms to digital libraries, it's helping organize information in ways that make it more accessible and user friendly.

Moreover, image captioning contributes to advancements between humans and computers, for example self driving vehicles that are now able to identify roads, surveillance systems for CCTV and Security, and e-commerce that can describe product images. It also helps contribute to more complex vision and language tasks, like visual question answering (VQA) and Video Captioning (which was also done in this Thesis).

Solving this problem advances the fields of AI and computer vision. It also has the potential to create technologies to improve user experiences across many real world applications.

1.3 Applications

Image captioning has many practical and useful applications in a lot of industries and sectors, for example:

- Assistive Technology: Using tools for visually impaired people. Image captioning can provide spoken descriptions of the person's environment or media on their phone, which improves their accessibility and independence.

- Social Media and Content Management: Automatic caption generation for images helps in content tagging, organization, as well as optimizing the search engine.

For example, platforms like Facebook and Instagram benefit from text generation for accessibility.

- E-commerce: Online shopping platforms can use image captioning to describe product images & generate descriptions from uploaded photos, helping improve the customer's experience and improve automation.

- Surveillance and Security: Generating text from camera feed images (CCTV) in security systems can help in surveillance and reviewing big quantities of data in an efficient way.

- Autonomous Vehicles: Understanding and describing environments can contribute to the systems of self driving cars, by enabling them to interpret complex roads and act in difficult situations.

- Medical Imaging: Image captioning can help automate the process of describing and annotating diagnostic images (X-Rays, Cancer) in healthcare, assisting doctors in clinics and hospitals.

- Education and Research: Image Captioning can help build educational and academic tools that describe images, benefiting schools, universities, students, and professors.

Those are just some of the applications that show the importance of developing accurate and efficient image captioning systems, which is why this thesis is being written in the first place.

1.4 Previous Work

Image captioning has evolved through several key stages. It started with rule based and retrieval approaches. It then reached advanced deep learning architectures.

Early Approaches (Pre-Deep Learning)

The first attempts to address image captioning used template-based or retrieval-based techniques. Template-based methods generate captions by filling predefined sentence structures with detected visual elements. Retrieval-based methods find images that resemble each other in a database, and reuse their captions. For example, in 2010, Farhadi et al. mapped visual scenes to intermediate semantic representations (detected objects, actions, and scenes), then selected matching sentences from a database [1]. Similarly, Ordonez et al. introduced Im2Text in 2011, which retrieves captions by finding similar images from a 1 million image dataset [2]. These approaches were innovative but lacked flexibility and struggled with image compositions. Kulkarni et al. addressed this limitation in 2013 by explicitly detecting objects, attributes, and relationships, then inserting these into linguistic templates to produce more flexible and descriptive captions [3].

Figure 1.1: Example of a retrieval-based image captioning system. The input image is matched to similar images in a database. The caption from the most similar image is reused for the new image. This method is simple and efficient, but it depends on the variety and quality of captions in the database.
Image adapted from Automatic image caption generation using deep learning.

Deep Learning Revolution (CNN + RNN)

The introduction of deep learning was around 2015. It brought a paradigm shift, where deep learning replaced hand crafted pipelines with learned representations. Vinyals

et al. proposed the landmark "Show and Tell" model, combining a CNN image encoder with an LSTM language decoder to generate captions end-to-end [4]. This model directly maps images to sentences using neural networks. Around the same time, Karpathy and Fei-Fei introduced a model that aligned image regions with words using a joint embedding space [5]. Their approach linked parts of an image to corresponding words, improving alignment and interpretability.

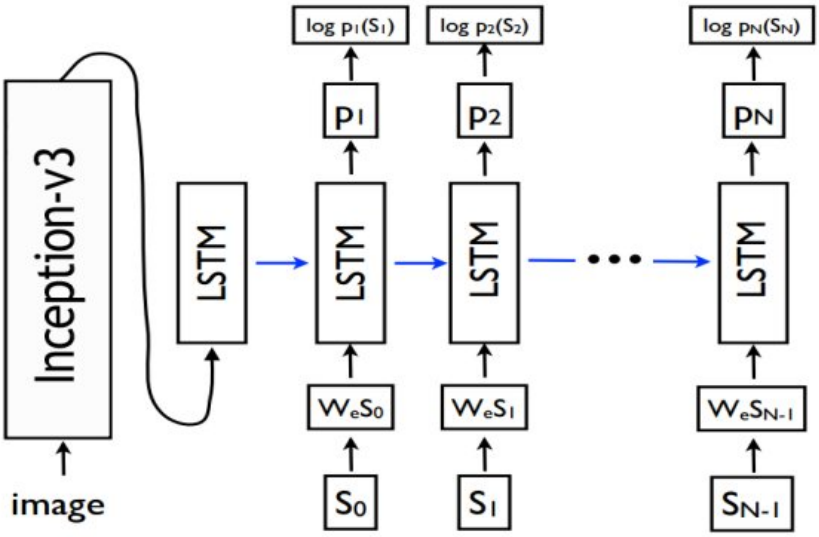


Figure 1.2: The "Show and Tell" model architecture. An image is encoded by a CNN (Inception-v3) and then passed to an LSTM, which generates the caption word by word. Each word prediction depends on both the image features and all previous words. *Image adapted from* Image Captioning using Deep Neural Architectures.

Following this, several models expanded the encoder-decoder framework, building on the idea of jointly learning vision and language. Mao et al. proposed m-RNN (Multimodal Recurrent Neural Networks) in 2015 as well, which fused image and language representations at every time step [6]. This model allowed visual features to influence each word in the caption as it was generated. Donahue et al. introduced LRCN (Long-term Recurrent Convolutional Networks), using CNNs and RNNs for video and image description tasks [7]. Their model handled temporal sequences, enabling captioning of not just static images but also video frames over time.

A major milestone came also in 2015 with Xu et al.'s "Show, Attend and Tell", which

introduced attention mechanisms, allowing the model to focus on specific image regions when generating each word [8]. This technique mimicked human visual focus, like when they look at specific objects while describing a scene, significantly improving alignment between image content and generated captions. A year later, You et al. expanded this concept with semantic attention (meaningful attention), guiding generation using detected visual attributes [9]. Their method used high level concepts like object labels to influence word selection, making captions more semantically accurate.

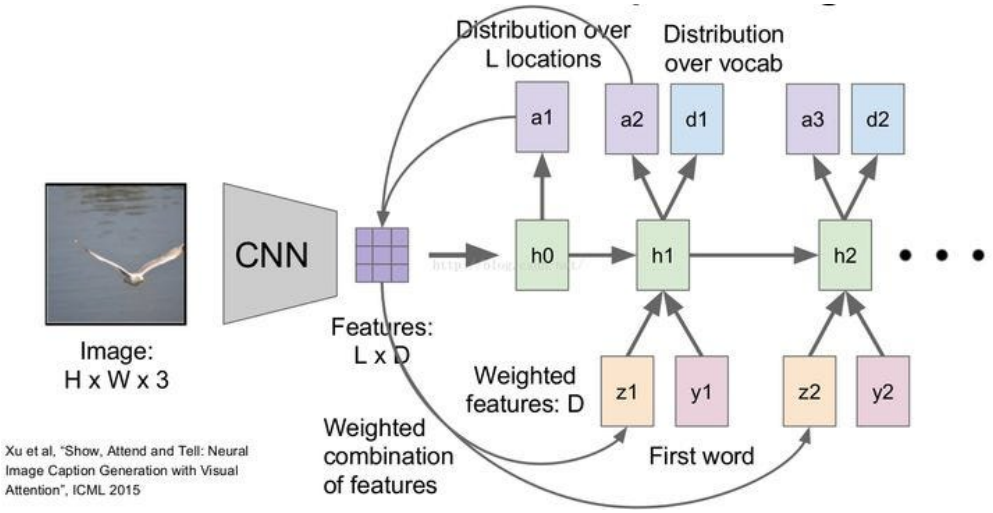


Figure 1.3: The "Show, Attend and Tell" model introduces visual attention to image captioning. Instead of using the whole image at once, the model learns to focus on specific parts of the image when generating each word. At each step, the attention mechanism produces a weighted combination of features from different regions, helping the model describe important objects more accurately.
Image adapted from Xu et al., 2015 [8], see also Image Captioning using Deep Neural Architectures.

Attention-Based Models

Anderson et al. advanced attention modeling in 2018 with the "Bottom-Up and Top-Down" approach, using object detectors to extract semantically meaningful regions and then applying attention over them [10]. Bottom-Up attention uses the image content to suggest important regions, while Top-Down attention uses the caption being generated to decide where to focus. This allowed the model to attend to actual objects instead of uniform grid features, improving interpretability. The model became a benchmark

and improved caption relevance and precision, setting a new standard for accuracy in image captioning tasks.

Zhang et al. introduced AoANet in 2019, which refined attention maps by applying a second layer of attention, enhancing focus on relevant visual regions [11]. This mechanism re-weighted the original attention scores, helping the model better prioritize important features in the image. Cornia et al. proposed the Meshed-Memory Transformer, leveraging multi-layer attention and meshed connections to improve long-range dependencies and generation fluency [12]. Their architecture enabled deeper contextual reasoning across image and language tokens, resulting in more coherent and descriptive captions.

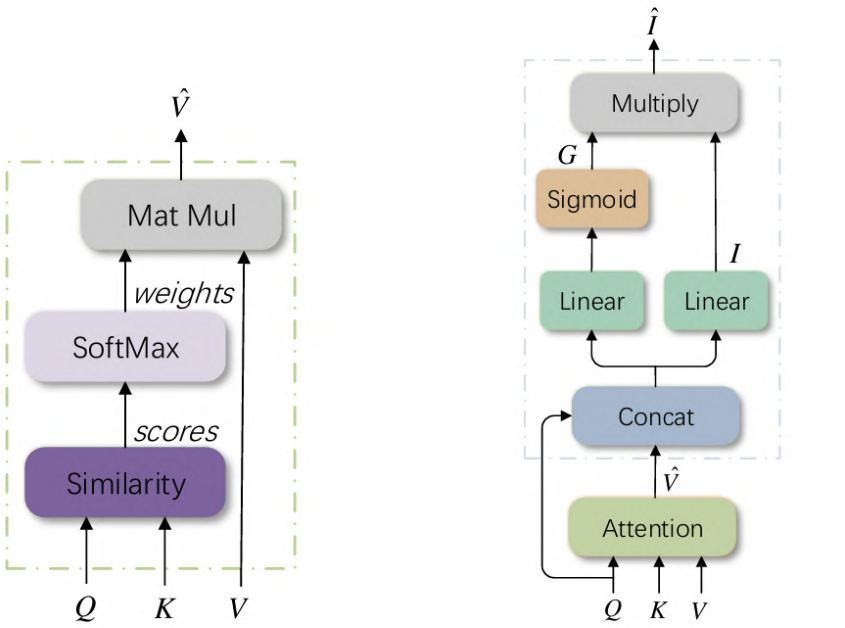


Figure 1.4: Visual illustration of the standard attention mechanism (left) and the "Attention on Attention" (AoA) module (right). (a) Standard attention computes a weighted combination of values based on similarity between queries and keys. (b) AoA builds on this by creating an information vector and an attention gate, combining them with another attention step for improved focus.
Image adapted from Huang et al., 2019 [11].

Pre-trained and Transformer-Based Models

With the rise of large-scale pretraining, image captioning advanced significantly. Li et al. introduced OSCAR (Object-Semantics Aligned Pre-training for Vision-Language Tasks), which improves vision-language alignment by pairing object labels with text tokens during training [13]. Specifically, detected object tags (e.g., "dog," "ball") are inserted as special tokens alongside text descriptions, helping the model learn stronger associations between visual content and language. Around the same time, Radford et al. proposed CLIP, a contrastive learning model trained on 400 million image-text pairs (from multiple publicly available sources). Although not designed for captioning, CLIP's ability to connect images and text embeddings enabled zero-shot and prompt-based captioning approaches [14].

Mokady et al. introduced CLIPCap, a lightweight captioning model based on CLIP, that generates captions by conditioning a Transformer decoder directly on CLIP's image embeddings [15]. Instead of training a new visual encoder, the model uses CLIP's fixed image features as a starting prefix for the Transformer, which then allows it to generate captions effectively without extra visual training.

Zhang et al. further enhanced captioning performance with VinVL (Visual Representations in Vision-Language Models), which improved object detection by retraining the detector on larger and more diverse datasets, yielding richer and better visual features [16]. Unlike standard feature extractors, VinVL provides dense region-level representations that capture more object attributes and relationships, boosting downstream models like OSCAR.

In 2022, Li et al. proposed BLIP, a unified vision-language model that learns from large amounts of image-text data by combining contrastive learning, caption generation, and image-text matching, using auto-generated captions to train without full supervision. [17]. In this context, bootstrapping refers to a self-training process where a captioning model generates captions for unlabeled images, and a filtering mechanism removes noisy or low-quality outputs before using them as training data. BLIP in-

troduces a vision-language encoder-decoder and a dual-stream architecture to support both retrieval-based and generative tasks within a single model.

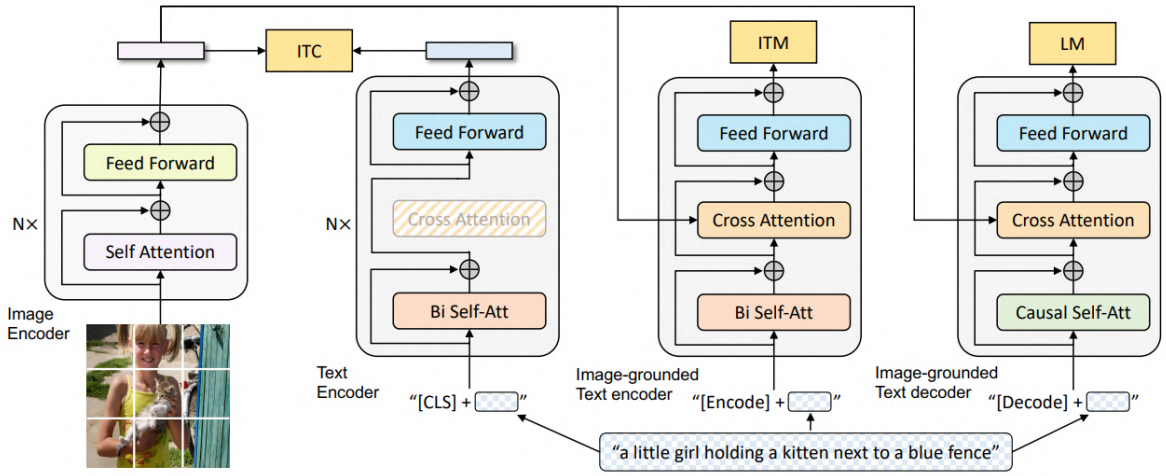


Figure 1.5: The BLIP model architecture combines an image encoder and a text encoder to process images and text together. The image encoder uses self-attention layers to extract visual features, while the text encoder uses similar layers for text. BLIP matches images and text using three main tasks: image-text contrastive learning (ITC), image-text matching (ITM), and language modeling (LM). Cross-attention layers connect image and text features, helping the model understand how they relate. The model can encode images and text, then decode and generate captions or answers based on both, allowing it to create detailed image descriptions like "a little girl holding a kitten next to a blue fence".

In 2023, Li et al. introduced BLIP-2, a vision-language pretraining framework that connects frozen image encoders and large language models (LLMs) using a lightweight module called the Querying Transformer (Q-Former) [18]. The model is trained in two stages. During the first stage, it aligns image features with text features; then, it learns to generate captions by linking the Q-Former outputs to a frozen LLM in the second stage. The Querying Transformer is a small transformer that learns a set of query tokens to extract the most useful information from the image features. Here, "frozen" means that the model's weights are kept unchanged during training, which lets BLIP-2 reuse powerful pre-trained models without extra fine-tuning. This setup allows for efficient zero-shot image-to-text generation while reducing the number of trainable parameters.

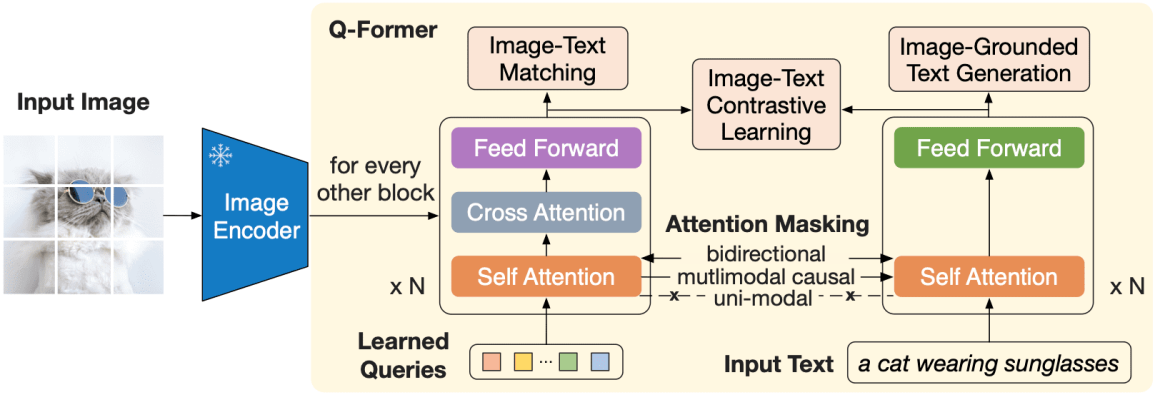


Figure 1.6: The BLIP-2 model processes images and text together using an image encoder and a special module called Q-Former. The image encoder turns the input image into features the model can use. The Q-Former uses self-attention and cross-attention layers to connect learned queries to these image features and the input text. BLIP-2 is trained to perform image-text matching, image-text contrastive learning, and image-grounded text generation, allowing it to understand and describe images in detail. Attention masking helps the model handle information from images and text in different ways, making BLIP-2 effective at tasks like generating captions, answering questions about images, and matching images to descriptions.

1.5 Gap in Image Captioning and What is Missing

Hallucination of Content

One major issue in image captioning is hallucination, where the model describes objects or actions that don't exist in the image. Even strong models like OSCAR and BLIP [13], [17] can add plausible but incorrect details due to biased training data. This is especially problematic in fields like healthcare or security, where accuracy is very important [19].

Out-of-Domain Generalization

Most image captioning models perform well on datasets like MSCOCO but struggle with unfamiliar domains, such as wildlife photos or art. While models like CLIP improve visual generalization [14], generating full, fluent captions in a zero-shot setup remains challenging [20]. Domain shift still causes errors in object recognition and caption

quality [21].

Handling of Rare Objects

Many models have trouble describing new or rare objects not seen during training. Models like OSCAR [13] and VinVL [16] try to fix this using object detectors and large-scale data, but they still rely on known vocabulary. Zero-shot methods like ZeroCap [20] show promise. But they can generate awkward or vague captions.

Lack of Reasoning and Inference

Today's models describe what they see but rarely understand deeper meaning. They can say "a person holding an umbrella" but not infer why (it's raining). Reasoning about cause, time, or intent is mostly missing. Even attention-based models like Show, Attend and Tell [8] focus on what's visible rather than making logical conclusions.

Weak Visual Grounding

Another gap is that captions often mention things not clearly grounded in the image. Grounding refers to the model's ability to link words in the caption to specific regions or objects in the image. Some models rely on co-occurrence patterns in training data instead of actual visual evidence. While models like Anderson et al.'s Bottom-Up attention help [10], grounding is still weak. This limits trust in the captions, especially when they mention objects not actually visible.

Limited Commonsense Knowledge

Models often lack commonsense understanding. For example, they might describe people eating but not recognize it's likely a meal or a celebration. They don't integrate background knowledge or understand cultural context. Although language models help a bit, there's still a big gap in embedding real-world logic into captions [22].

Multilingual and Non-English Limitations

Most captioning systems work only in English due to limited multilingual datasets. Non-English benchmarks like Multi30K [23] are rare and small. As a result, captioning models often perform poorly in other languages or can't handle language-specific grammar and cultural nuances. Recent work like LMCap [24] addresses this by enabling few-shot multilingual image captioning without requiring large-scale parallel datasets, showing that retrieval-augmented prompting can help bridge the multilingual gap.

Specialized Domains like Medical Images

In domains like medical or satellite imagery, captioning remains underdeveloped. These fields need precise, technical language and can't tolerate errors. However, datasets like MedICaT [25] are limited and expensive to annotate. General captioning models trained on everyday photos are not suitable for these expert tasks [26].

Zero-Shot and Few-Shot Captioning

Zero-shot captioning is where the model generates the image description without being explicitly trained or fine tuned on the data. For example, models like ZeroCap [20] show that this is possible by using pretrained vision-language models, but the captions are usually less accurate and detailed than those from models trained with labeled data. Few-shot learning, where a model learns to perform a task using only a small number of labeled examples, has also been underexplored, despite its importance for building flexible and scalable captioning systems.

Interactive and Controllable Captioning

Most captioning tools generate a single static sentence. But real users may want to tweak or guide the caption. Early tools like iCap [27] allow user interaction, but this is still a new area. There's also growing interest in controllable captioning where users set the tone, focus, or style of captions. But few models support it yet.

1.6 Thesis Contributions

This thesis presents and explains 2 simple and effective approaches to the image captioning problem. It focuses on clarity, ease of access, and educational value. The main work contributions of this thesis are:

- Implementation: A complete pipeline is implemented. It consists of preprocessing the data, tokenizing, and extracting the image features with a CNN. It then generates captions and evaluates metrics.

- Lightweight Encoder-Decoder Architecture: The model uses a pre-trained CNN (like ResNet-50 and EfficientNetB0) as the encoder, and a basic LSTM (and Transformer) based decoder to generate captions. This design avoids complex architectures, making the model suitable for training on users' computers and laptops.

- Customized Data Preparation: A preprocessing pipeline is developed. We resize images, tokenize captions, build a vocabulary, and prepare sequences for training on the Flickr30k dataset.

- Caption Generation Mechanism: Beam search algorithms are used to generate captions for the images.

- Model Evaluation and Visualization: Evaluation is performed using BLEU metrics. Visual outputs are provided to demonstrate the model's capability in generating meaningful captions for unseen images.

- Notebook-Based Modularity: The project is implemented using Jupyter Notebook inside a Conda environment. Each notebook contrains all the steps for data preprocessing, model training, caption generation, and evaluation, making the workflow easy to follow, test, and modify.

- User Interface: An interface is created using a python web interface called Streamlit. It allows users to test the different models with custom images, as well as videos.

Overall, this work provides a well-documented example of image captioning using deep learning, suitable for students, professors, and developers with limited resources.

1.7 Chapter Summary

This first chapter talked about the image captioning problem, as well as its importance and its many applications. It also mentioned who contributed in that domain, while identifying existing gaps in the literature and what is missing or lacking. Finally, it ended with the main contributions of this thesis.

The following chapters are organized as follows:

- Chapter 2: Literature Survey: Provides definitions and background on important concepts like Artificial Intelligence, Machine Learning, Deep Learning, and Computer Vision. It explores recent advancements in image classification as well as image captioning techniques.

- Chapter 3: Methodology and Solution Description: This chapter explains the technicality used to solve the image captioning problem, like details on the model architectures, data preprocessing and tokenizing, training, and evaluation metrics.

- Chapter 4: Experimental Results and Validation: Presents the dataset used, the development environment, and a detailed analysis of the model's performance using evaluation metrics and visual examples.

- Chapter 5: Conclusions and Future Directions: Summarizes the work created in this thesis, important findings, lessons learned, and directions for future improvement.

- Chapter 6: References and Bibliography: Lists all academic papers, tools, and datasets cited throughout the report.

These chapters provide a comprehensive overview of the research process, from conceptual understanding to practical implementation and evaluation.

Chapter 2

Literature Survey

In this chapter, we cover the main background needed for understanding image captioning. We will define six important topics: Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL), and Computer Vision (CV). After that, we will discuss the key milestones in image classification. Finally we will review the major advances in image captioning, including who contributed what to each field.

2.1 Artificial Intelligence

Artificial Intelligence (AI) is a computer science field that aims to make machines smart and think on their own. It makes them do anything that needs human intelligence, like solving problems, understanding human language, and learning patterns. The term "Artificial Intelligence" was first used in 1956 by an American computer scientist called John McCarthy. He defined it as "the science and engineering of making intelligent machines" [28]. Ever since, AI has become one of the fastest evolving fields of technology.

AI is divided into two types: narrow AI and general AI [29]. Narrow AI is designed to do one task very well, like recognizing faces in pictures, translating languages, or even Image Captioning. Most of today's AI systems belong to this category. They're highly skilled in certain areas, but don't have human intelligence. On the other hand, general AI is the idea of a machine that can think and learn like a human across many different areas. It can perform any intellectual task a human can; like learning, reasoning, adapting, and solving problems across a wide variety of domains without needing specific programming for each task. This kind of AI is still theoretical and

doesn't exist yet [30].

There are many different areas within AI, known as subfields. This thesis focuses primarily on the intersection of Computer Vision and Natural Language Processing, where AI enables systems to understand visual content and describe it using human language, which is Image Captioning.

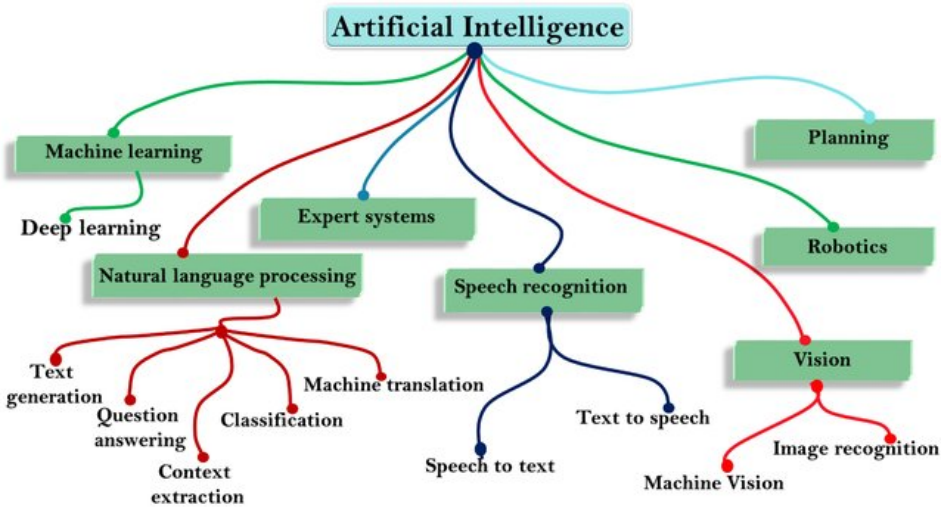


Figure 2.1: Illustration showing Artificial Intelligence and its main domains, such as Machine Learning, Vision, and Natural Language Processing. This type of diagram gives a quick overview of AI's broad reach, with connections to many subfields relevant for modern technology.
Image adapted from The role of Artificial Intelligence in future technology (ResearchGate).

2.2 Machine Learning

Machine Learning (ML) is a branch of artificial intelligence that focuses on teaching computers to learn from data and improve at tasks through experience, rather than following hardcoded rules. ML enables systems to find patterns in data and use those patterns to make predictions or decisions. As Mitchell defines it, a program learns from experience when its performance improves with more data and practice [31].

Every Machine Learning system generally involves three parts:

- Data — The system is trained on labeled data that contains input-output pairs.

- Model — A mathematical way to make predictions and capture relationships in the data (like linear regression, decision trees, or neural networks).
- Training — The model is optimized using algorithms like gradient descent like Adam optimizer, to minimize the error between predictions and actual outputs, as well as to do better based on past mistakes.

Machine Learning is often divided into four types:

- Supervised learning: The model is trained using labeled data; for example, image-caption pairs, classification, and regression tasks (predict continuous numerical values).
- Unsupervised learning: The model explores the structure of unlabeled data. For example; clustering, dimensionality reduction, or grouping similar customers together.
- Semi-supervised learning: This approach combines a small amount of labeled data with a large amount of unlabeled data. It is useful when labeled data is scarce or expensive to obtain, while unlabeled data is abundant [32].
- Reinforcement learning: The model learns through trial and error by interacting with an environment, and receiving feedback in the form of rewards or penalties [33].

In this thesis, Machine Learning is used in the context of image captioning, where supervised learning trains models on a dataset of image–caption pairs to generate descriptive sentences for new images [4], [8]. The model learns to map visual features to language patterns, forming the basis of the caption generation process.

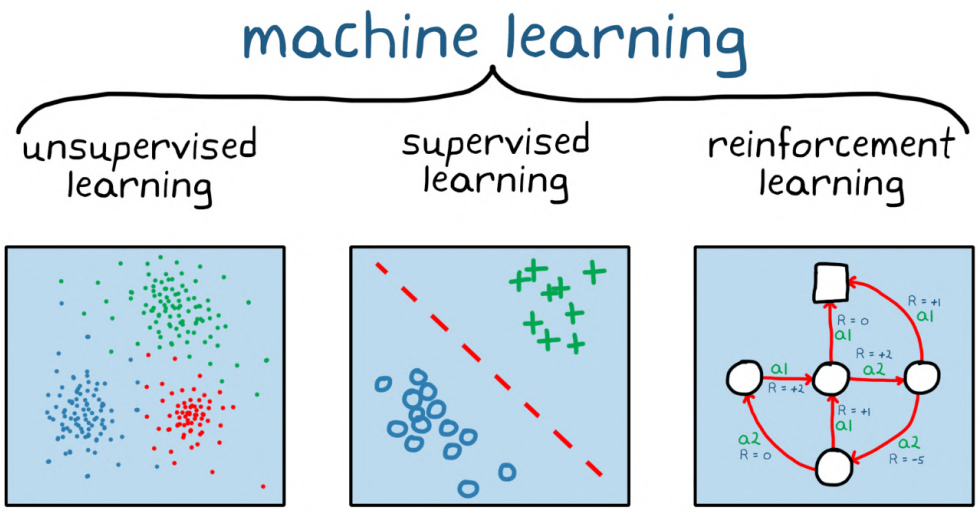


Figure 2.2: Visual comparison of supervised, unsupervised, and reinforcement learning. The diagram shows how each type of machine learning uses data: supervised learning is based on labeled examples, unsupervised learning finds patterns in unlabeled data, and reinforcement learning learns by interacting with an environment.
Image adapted from The AIOps.

2.3 Deep Learning

Deep learning is a branch of machine learning that uses multi-layered artificial neural networks to learn from data. These networks, called deep neural networks, are made up of many layers of simple processing units, often called "neurons." Deep learning has enabled major advances in complex tasks, especially in computer vision and natural language processing. As explained by El-Amir and Hamdy [34], deep learning models automatically extract useful features and patterns from raw data by passing it through multiple hidden layers. This ability to learn directly from data makes them very effective at handling high-dimensional information like images, audio, and text.

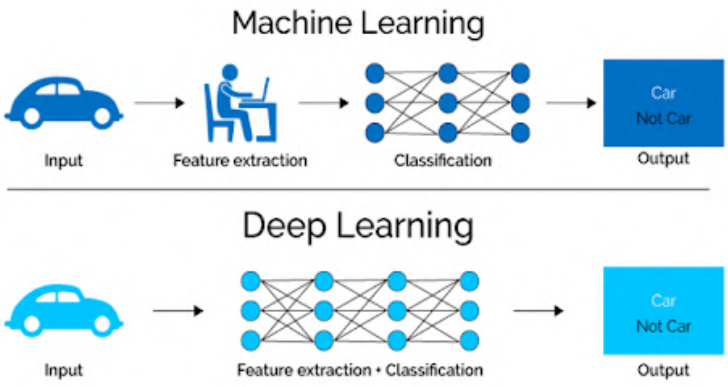


Figure 2.3: Comparison between classic machine learning and deep learning. In traditional machine learning, feature extraction is done by humans before classification. In deep learning, neural networks learn to extract features and classify data end-to-end. *Image adapted from* Built In.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of deep learning architecture designed for sequential data, such as sentences or time series. They use an internal state (memory) to capture information from earlier steps in a sequence. One of the earliest forms was the Elman network [35]. However, standard RNNs have trouble learning long-term dependencies because of the vanishing gradient problem. This happens when the gradients (a measure of how much a function changes when its inputs change) used for learning, become very small as they are passed backward through many time steps, causing the model to stop learning effectively. The Long Short-Term Memory (LSTM) network, introduced by Hochreiter and Schmidhuber [36], solved this by adding gated memory cells. These gates control the flow of information, allowing LSTMs to learn long-range patterns and making them good for tasks like language modeling and speech recognition.

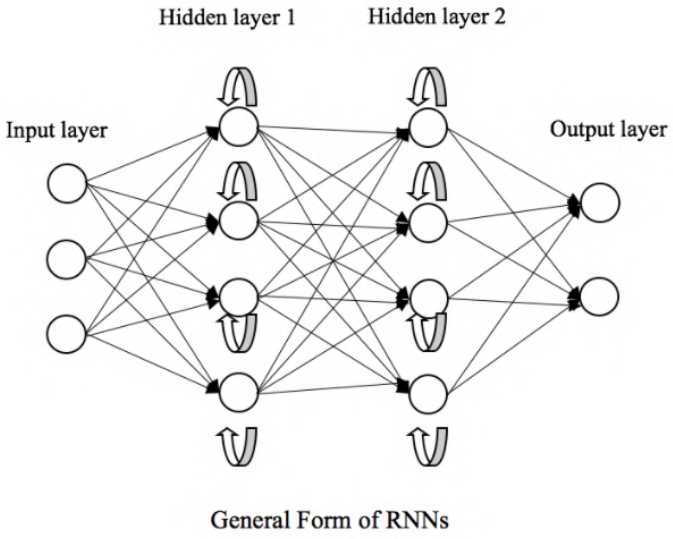


Figure 2.4: General form of a Recurrent Neural Network (RNN). The curved arrows show the "recurrent" connections, meaning each neuron passes information forward to itself in the next time step. This lets the network remember previous inputs and handle sequence data, like sentences or time series. RNNs use these loops to learn patterns across time, making them powerful for tasks where order and context matter.
Image adapted from Open Data Science.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of neural networks first popularized by Yann LeCun in the 1980s for digit recognition tasks, through the LeNet-5 architecture [37]. CNNs are specifically designed to process grid-like data such as images, by exploiting their spatial structure. The core building block of a CNN is the convolutional Layer, which applies a set of learnable filters (also called kernels) that slide over the input, to detect local patterns like edges, textures, or shapes. These filters produce feature maps that preserve spatial relationships, allowing the model to learn increasingly abstract visual features across multiple layers.

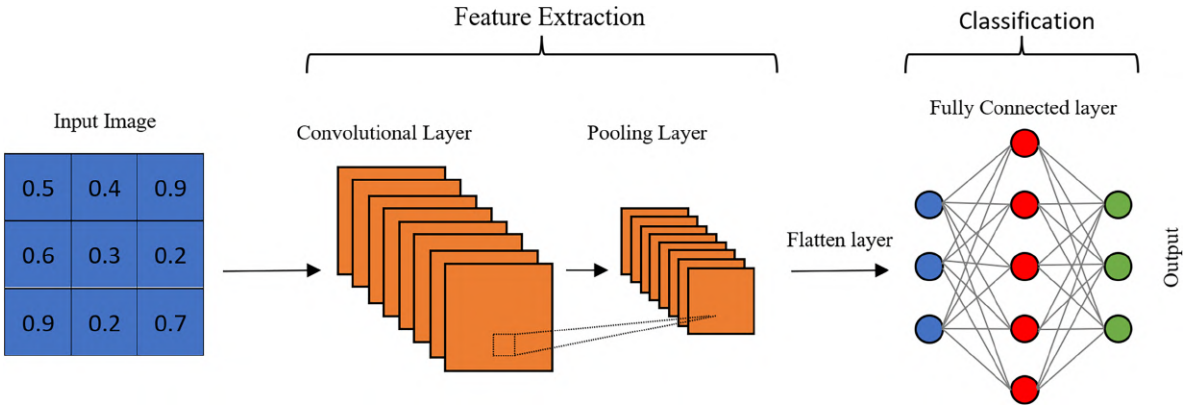


Figure 2.5: Structure of a basic Convolutional Neural Network (CNN). **Input Image:** The process begins with a grid of pixel values representing the image. **Convolutional Layer:** Multiple filters slide over the image to detect patterns like edges or shapes, creating feature maps. **Pooling Layer:** These feature maps are downsampled, reducing their size and helping the network focus on the most important features. **Flatten Layer:** The feature maps are flattened into a single long vector. **Fully Connected Layer:** The vector is passed to a dense neural network layer, where each node is connected to every node in the next layer. **Output:** The final predictions are produced, such as classifying the image as "Car" or "Not Car".
Image adapted from PeerJ Publishing.

A major breakthrough came with the AlexNet model by Krizhevsky et al. [38], which drastically improved performance on the ImageNet classification challenge by a large margin. The model achieved this by increasing network depth, introducing the use of Rectified Linear Units (ReLU) as activation functions, applying Dropout (neuron deactivation during training) to reduce overfitting, and leveraging GPU acceleration to train on large datasets. ReLU, defined as $f(x) = \max(0, x)$, outputs the input value if it's positive, and zero otherwise. This lets the network introduce non-linearity while avoiding issues like the vanishing gradient problem that occurs with older activation functions, like sigmoid or tanh. This success demonstrated that deep CNNs, when trained at scale with the right architectural choices, could surpass traditional computer vision techniques. They marked the beginning of deep learning's dominance in image recognition.

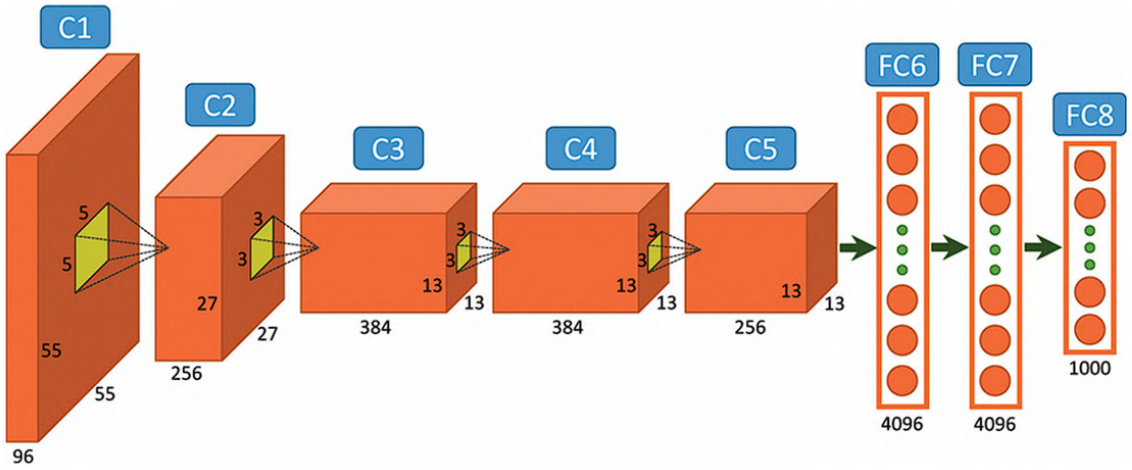


Figure 2.6: Overview of the AlexNet architecture. The network begins with the input image, which passes through five convolutional layers (C1–C5) that use learnable filters to extract features of increasing complexity. Each layer reduces the spatial size and increases the number of channels, capturing more abstract visual patterns. After the convolutional blocks, the feature maps are flattened and fed into three fully connected layers (FC6–FC8). The fully connected layers serve as the classifier. They combine the features extracted by the convolutional layers and enabling the network to learn complex, high-level representations for making final predictions. The first two fully connected layers each have 4096 neurons, and the final layer (FC8) outputs the class probabilities for 1000 categories. This structure allows AlexNet to learn rich, hierarchical representations from raw image pixels and achieve high accuracy on image classification tasks.
Image adapted from Visio AI.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) introduced a groundbreaking approach to generative modeling. They are a type of generative model used to create new images. They were proposed by Goodfellow et al. [39]. GANs have two neural networks trained in opposition. A generator, which tries to create realistic synthetic data, and a discriminator, which learns to distinguish between real and generated data. The generator's goal is to fool the discriminator, but the discriminator improves its ability to detect fakes. This results in a dynamic adversarial process. Over time, this competition pushes the generator to give more realistic outputs. GANs have been used in tasks such as high-resolution image synthesis, style transfer, and data augmentation. They are still one of the most influential important in generative modeling.

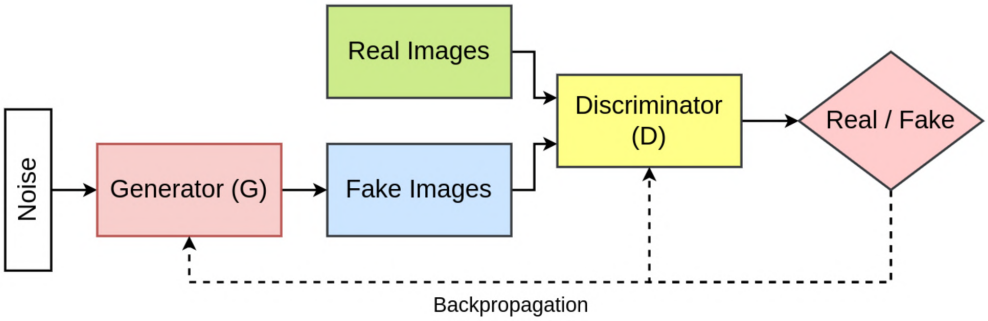


Figure 2.7: Diagram of a Generative Adversarial Network (GAN). Random noise is fed to the Generator (G), which tries to create realistic fake images. Both real images and fake images are passed to the Discriminator (D), typically a CNN, which attempts to tell real from fake by extracting features from each image. The discriminator outputs a real/fake decision. During training, both networks are improved by backpropagation: the generator learns to create more convincing images, and the discriminator learns to better distinguish them.
Image adapted from Idiot Developer: What is DCGAN?

Transformers

Transformers were introduced in 2017. They are a deep learning architecture. Recurrent Neural Networks (RNNs) process sequences step by step. But on the other hand, Transformers use a mechanism called self-attention to examine all parts of the input sequence at the same time. Self-attention allows the model to assign different weights to different elements in the sequence. This helps it focus on the most relevant parts for a given task. This also makes Transformers especially effective at capturing long-range dependencies and processing sequences in parallel. Introduced by Vaswani et al. [40], the Transformer architecture brought major improvements in natural language processing tasks. For example, machine translation. Due to its scalability and strong performance, it has also been adapted for vision-language applications, like image captioning.

All these innovations are the foundation for many modern AI systems. They enable models to handle complex multimodal tasks such as image captioning. In this thesis, a combination of CNNs and RNNs are used for the image captioning task: the CNN encodes the visual content of an image and extracts its features, while the RNN generates

a natural language caption based on the extracted features.

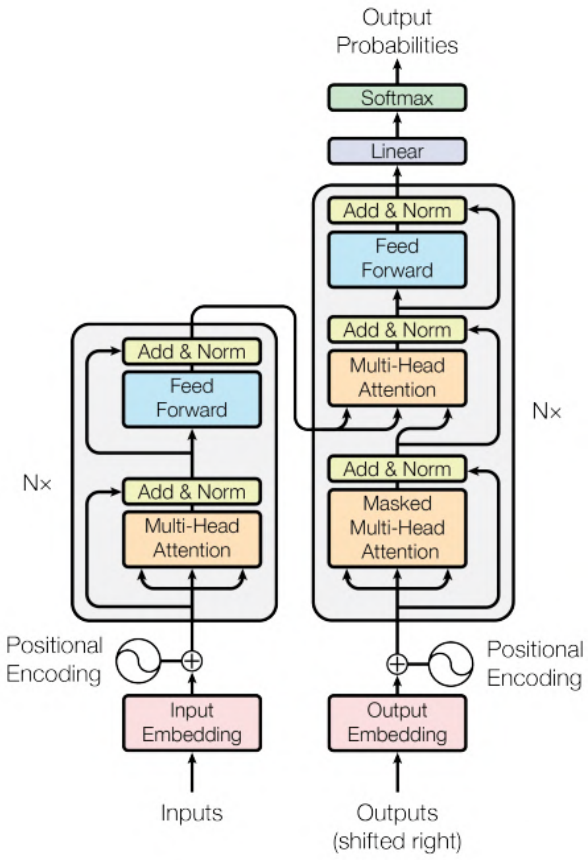


Figure 2.8: The Transformer model architecture, introduced by Vaswani et al. [40]. It is based on stacked encoder and decoder blocks. Each encoder (left) contains multi-head self-attention and feed-forward layers, with layer normalization and residual connections (Add & Norm). The decoder (right) also includes masked multi-head attention to prevent access to future tokens during training. It also contains cross-attention layers that connect the decoder to the encoder's output. Both encoder and decoder use positional encoding to provide sequence order information, since the model does not process inputs sequentially. The output is generated token by token, using the final softmax layer to predict probabilities over the vocabulary.
Image adapted from Attention Is All You Need explained.

2.4 Computer Vision

Computer vision is another field of artificial intelligence. It helps computers understand and analyze visual information from images or videos. Inspired by the human vision system, computer vision methods allow machines to recognize objects, detect patterns, and interpret scenes from visual data. Early computer vision techniques relied on man-

ually created features. But today, most systems use deep learning methods, especially Convolutional Neural Networks (CNNs); which are very effective at learning important visual patterns from large image collections [37], [38]. Improvements in computing power, GPUs, and the availability of large, labeled datasets(COCO, Imagenet) have made computer vision models very accurate and largely used in fields like self-driving cars, medical imaging, and augmented reality.

Computer vision involves many important tasks. Each task is aimed at different aspects of understanding visual content.

- Image Classification: This task involves assigning one category label to an entire image, based on the primary object or scene. Modern classification models use deep CNN architectures to automatically learn discriminative visual features. This achieves very good accuracy on benchmarks like ImageNet [38].

- Object Detection: Unlike classification, object detection aims to identify multiple objects within an image, specifying both their categories and spatial locations using bounding boxes. Contemporary approaches such as Faster R-CNN combine region proposal networks and CNN-based classification to achieve precise and efficient detection, widely applicable in surveillance, autonomous navigation, and image retrieval tasks [41].

- Semantic Segmentation: This task classifies every individual pixel of an image into predefined categories, providing detailed insights into object boundaries and scene structure. Fully convolutional networks (FCNs) enable accurate pixel-level prediction, significantly improving upon classical image segmentation methods and proving crucial for scene understanding and analysis [42].

- Instance Segmentation: Extending semantic segmentation, instance segmentation differentiates between distinct instances of the same object category by assigning unique masks to each instance. State-of-the-art models, such as Mask R-CNN,

simultaneously perform object detection and segmentation, enabling precise delineation of individual objects within complex scenes [43].

- Image Captioning: This multimodal task generates descriptive natural-language sentences to summarize the visual content of images. Current approaches employ encoder-decoder frameworks where a CNN extracts visual features, and a recurrent neural network (RNN) or Transformer generates coherent captions, demonstrating sophisticated integration of vision and language processing capabilities [44].

- Facial Recognition: Facial recognition systems automatically identify or verify individuals from facial features extracted in images or video frames. Modern facial recognition methods, such as FaceNet, leverage deep embedding techniques, where faces are represented as points in high-dimensional feature spaces, enabling accurate identity verification through distance metrics [45].

- Optical Character Recognition (OCR): OCR techniques convert printed or handwritten text within images into editable, machine-readable text. Advanced OCR systems utilize end-to-end deep learning architectures that integrate CNNs for visual feature extraction and sequence modeling, significantly improving recognition accuracy, especially under challenging real-world conditions [46].

- Scene Understanding: This comprehensive task involves interpreting and integrating various visual elements within an image to achieve an overall semantic understanding of the depicted scene. Models trained on large-scale scene-centric datasets, like Places, leverage CNNs and contextual modeling to classify scenes, recognize constituent objects, infer relationships, and predict scene layouts, facilitating deeper comprehension of complex visual environments [47].

To perform those tasks, modern computer vision systems use deep learning methods, like Convolutional Neural Networks (CNNs). They have significantly advanced the field, achieving performance comparable to, or even surpassing, human accuracy on many

visual benchmarks. In this thesis, computer vision is a core component of the image captioning pipeline. A CNN analyzes the input image and extracts visual features, which helps generate accurate and descriptive textual captions.

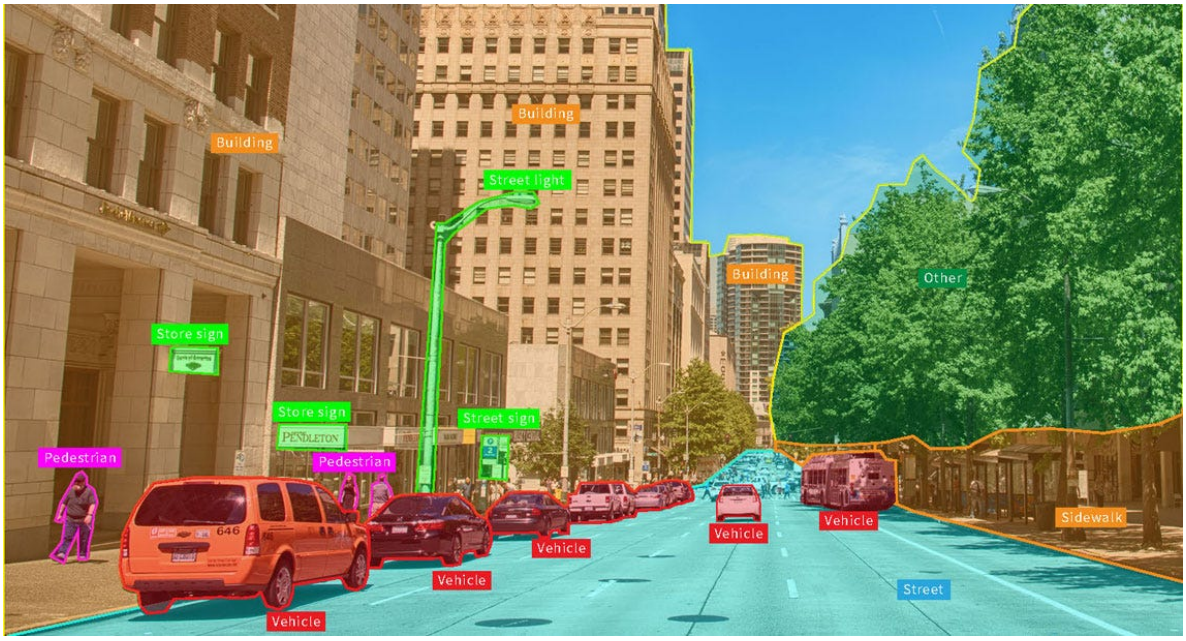


Figure 2.9: Example of computer vision tasks on a city street scene. The image shows semantic segmentation (colored regions for building, sidewalk, street, etc.), object detection (vehicles and pedestrians with labeled boxes), and scene understanding. *Image adapted from* "What is Computer Vision?

2.5 Recent Techniques in Image Classification

Image classification is a very important task in computer vision. It aims to assign an accurate label to an image, based on its visual content. Over the past decade, the rise of deep learning, like CNNs, has enhanced both the accuracy and efficiency of image classification systems, mainly due to architectural innovations and improvements in computational resources and GPUs.

Image classification is essential in computer vision. It assigns labels to images based on their visual content. Deep learning has improved image classification accuracy and efficiency. CNNs play a key role in this progress. Advances in architecture and GPUs have boosted performance.

Key CNN Models: AlexNet and VGGNet

The evolution of CNN architectures produced several influential models. A significant early breakthrough was AlexNet, proposed by Krizhevsky et al.[38], which won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The ILSVRC is an annual computer vision competition that evaluates algorithms on a massive dataset of over 1 Million labeled images, spanning 1,000 object categories, and serves as a benchmark for progress in image classification and object detection. AlexNet demonstrated the importance of GPU acceleration, dropout regularization, and the Rectified Linear Unit (ReLU) activation function, which led to better classification performance. Following AlexNet, Simonyan and Zisserman introduced VGGNet [48]. It was characterized by its deeper structure and uniform use of small (3×3) convolutional filters. VGGNet is simple, but it showed that deeper CNN architectures could capture richer visual representations, resulting in better accuracy.

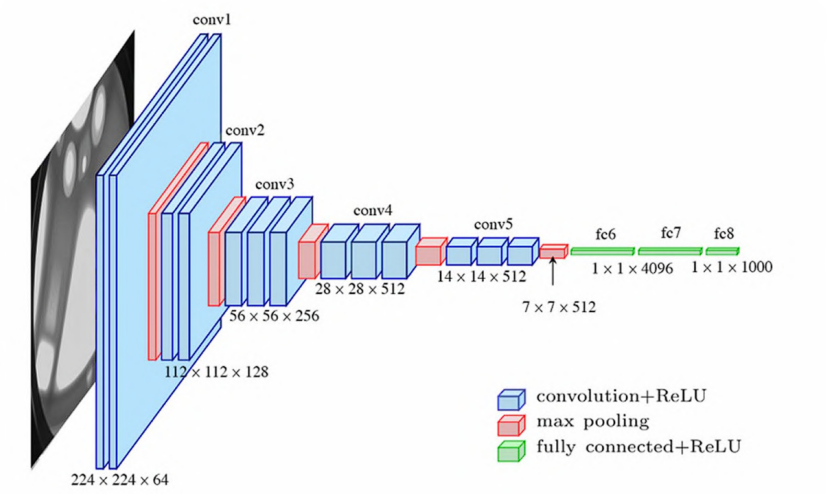


Figure 2.10: The VGG-16 convolutional neural network architecture, used for image classification tasks. The model processes a 224×224 RGB input image through a stack of convolutional layers (blue, each followed by a ReLU activation), interleaved with max pooling layers (red) to reduce spatial dimensions. After the convolutional stages, the output feature maps are flattened and passed through three fully connected layers (green, with ReLU activation), ending in a softmax output of 1,000 categories. VGG-16 is characterized by its use of small (3×3) filters and deep architecture. *Image adapted from* Viso AI.

Advancements: Inception, ResNet, and DenseNet

Subsequently, Szegedy et al. proposed the GoogLeNet architecture [49]. It introduced "Inception modules". These modules process visual information at multiple scales simultaneously. This greatly enhances computational efficiency and accuracy. Another crucial development was the introduction of residual connections by He et al. in their ResNet architecture [50]. ResNet's innovative skip connections significantly improved gradient propagation, enabling the effective training of very deep networks (e.g., 50–152 layers). Later, Huang et al. introduced DenseNet [51]. It further enhanced gradient flow and feature reuse by directly connecting each layer to every other layer in a dense feed forward manner.

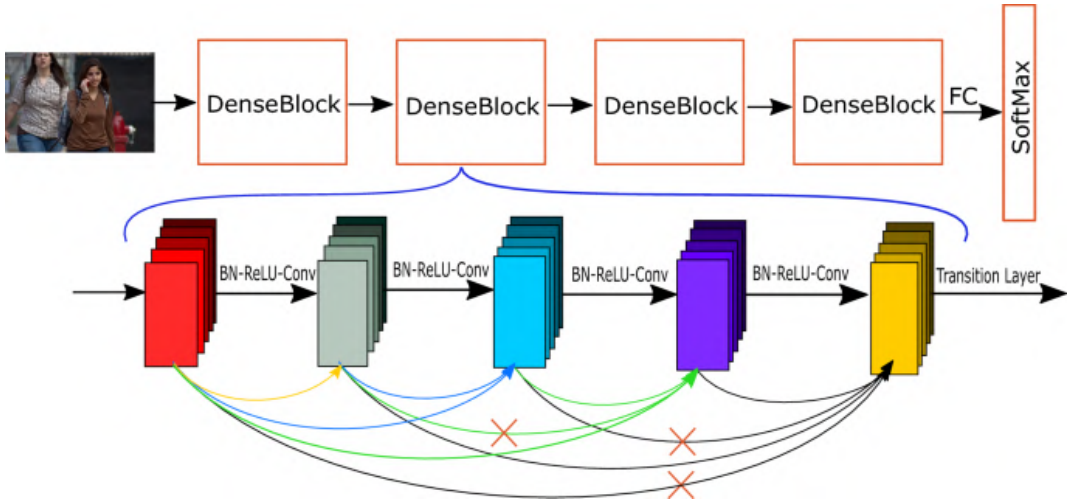


Figure 2.11: DenseNet architecture as proposed by Huang et al. [51]. The network consists of several DenseBlocks, each containing multiple convolutional layers. Unlike traditional CNNs, DenseNet connects each layer to every other layer within the same block, passing feature maps forward (shown as multi-colored connections). This design maximizes information flow and enables efficient gradient propagation. Each arrow labeled BN-ReLU-Conv represents a sequence of Batch Normalization, ReLU activation, and Convolution. Transition layers reduce the size of feature maps between blocks. Dense connectivity encourages feature reuse, reduces parameters, and improves training for deep networks.
Image adapted from Huang et al., 2017.

Transfer Learning in Image Classification

Beyond architectural innovations, transfer learning has emerged as a widely used and effective strategy in image classification tasks [52]. In transfer learning, a convolutional neural network (CNN) model is first pre-trained on a large-scale dataset such as ImageNet, where it learns general visual features. These learned weights and filters are then reused as the starting point for new, specialized classification tasks with much smaller datasets. Developers can either fine-tune all layers or freeze early layers and only train the final layers on the new data. This approach significantly reduces computational resources and training time, avoids overfitting, and usually leads to improved performance, especially when labeled data is limited. In this thesis, transfer learning plays a key role in using state-of-the-art vision models (like ResNet50 using ImageNet weights) for tasks like image captioning.

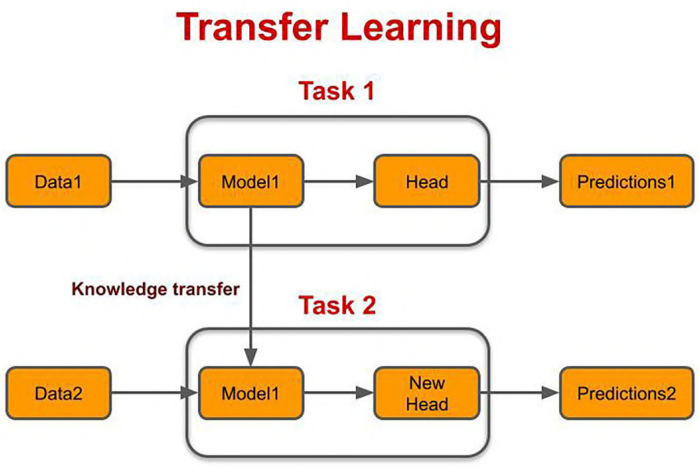


Figure 2.12: A visual explanation of transfer learning. In Task 1, a model is trained on Data1, producing predictions for its specific problem. In Task 2, instead of training a new model from scratch, the previously learned model (Model1) is reused and connected to a new "head" (output layer) for a different dataset (Data2). This process, called *knowledge transfer*, allows the model to leverage previously learned features, enabling faster convergence and better performance with less data for the new task.
Image adapted from topbots.com.

Commercial Use: NVIDIA DLSS

One example of applying deep learning in commercial image processing is NVIDIA's DLSS (Deep Learning Super Sampling). DLSS uses convolutional neural networks trained on pairs of high-resolution and low-resolution game frames to predict and reconstruct lower-resolution images in real time, enhancing visual quality while boosting frame rates [53]. By rendering fewer pixels, and using AI to upscale the output, DLSS reduces computational load on the GPU, making high-resolution, high-FPS gameplay possible even on demanding games. This shows how deep learning is increasingly used beyond academic tasks like classification, into high performance, real-world graphics systems.

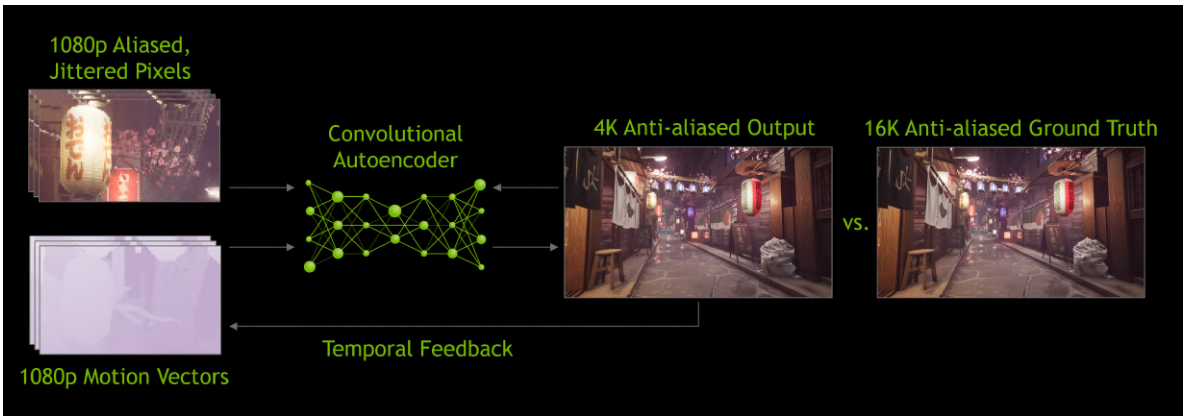


Figure 2.13: Illustration of NVIDIA's Deep Learning Super Sampling (DLSS) pipeline. The system uses a convolutional autoencoder neural network to upscale 1080p aliased game frames into high-quality, anti-aliased 4K outputs. The network takes as input low-resolution, jittered images and motion vectors, and incorporates temporal feedback from previous frames. The model is trained to reconstruct sharp images by minimizing the difference between its output and a 16K ground truth image. This deep learning approach enables real-time rendering of high-resolution frames, improving both visual quality and GPU performance in modern games.
Image adapted from NVIDIA DLSS 2.0: A Big Leap In AI Rendering.

Advancements in image classification contribute to image captioning tasks. Pre-trained CNNs, particularly architectures like VGGNet and ResNet, are employed as feature extractors within image captioning pipelines. They provide robust visual representations that are foundational for generating accurate and relevant captions.

These advances in image classification directly benefit image captioning, as pre-trained CNNs like VGG16 or ResNet-50 are often used as feature extractors in the encoder component of Image Captioning models, like in this thesis.

2.6 Recent Techniques in Image Captioning

Image captioning is a challenging task that combines computer vision and natural language processing. It requires models to understand visual information and generate clear, meaningful textual descriptions. Recent advancements, driven by deep learning methods, have improved caption quality, coherence, and relevance.

Encoder-Decoder Architecture

The most widely adopted framework for image captioning is the encoder-decoder architecture. It is inspired by sequence-to-sequence modeling approaches. Typically, a Convolutional Neural Network (CNN), such as VGGNet [48], Inception [49], or ResNet [50], is used as the encoder. This encoder processes the image through multiple convolutional and pooling layers, extracting spatial patterns and hierarchical visual features. The resulting feature representations are then flattened into fixed-length vectors, often 2048-dimensional (a vector of 2048 elements). Each element corresponds to a specific abstract visual feature that captures the content of the image. These vectors are passed to a Recurrent Neural Network (RNN) usually a Long Short-Term Memory (LSTM) [36] or Gated Recurrent Unit (GRU) [54]. The RNN acts as a decoder, sequentially generating captions one word at a time. At each step, the RNN takes as input the visual features from the encoder, combined with the previously generated words, to produce a probability distribution over the next possible word. It then generates a coherent, contextually relevant description of the image.

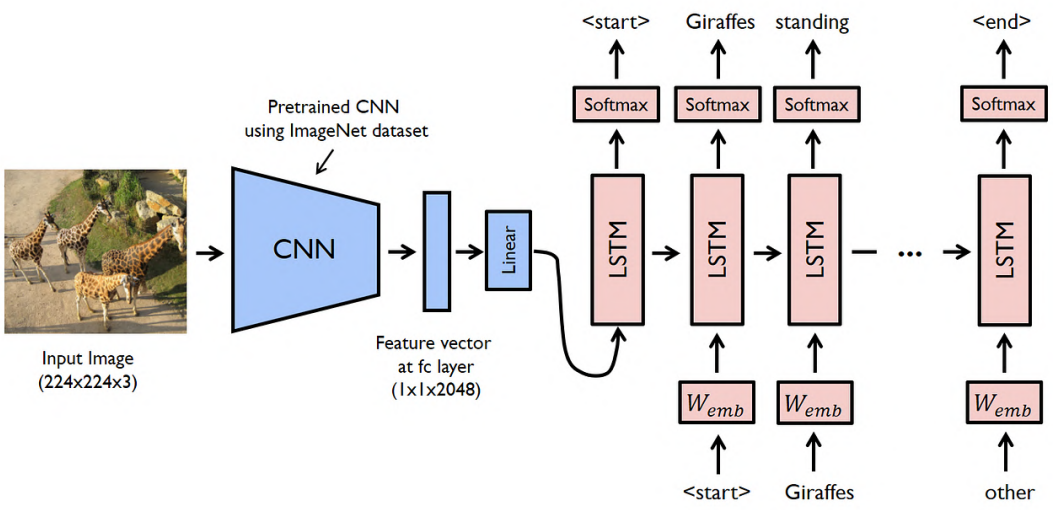


Figure 2.14: Schematic of the classic encoder-decoder architecture for image captioning. A pre-trained Convolutional Neural Network (CNN, like ResNet50 or VGG16), typically trained on ImageNet, encodes the input image ($224 \times 224 \times 3$) into a fixed-length feature vector (here $1 \times 1 \times 2048$). This visual embedding is then passed to a Long Short-Term Memory (LSTM) network, which acts as a decoder. The LSTM generates the caption one word at a time. It is conditioned on the image features and previously generated words. At each step, word embeddings (W_{emb}) represent the current input word, and the LSTM predicts a probability distribution over the vocabulary via a Softmax layer. This produces coherent sentences until an `<end>` token is reached.
Image adapted from Medium: CNN+LSTM Image Captioning.

Attention Mechanisms

An influential enhancement to the standard encoder-decoder approach was the introduction of attention mechanisms. Proposed by Xu et al. in the "Show, Attend and Tell" model in 2015 [8], attention enables the decoder to dynamically assign different levels of importance (attention weights) to specific spatial regions or features of the image during caption generation. This mechanism is called "attention" because the model explicitly "attends", or selectively focuses on different parts of the visual input at each decoding step. Similar to how humans selectively concentrate on relevant details while describing a scene. By allowing the decoder to adaptively highlight relevant visual information aligned with the current context, attention mechanisms significantly enhance the accuracy, detail, and coherence of generated captions.

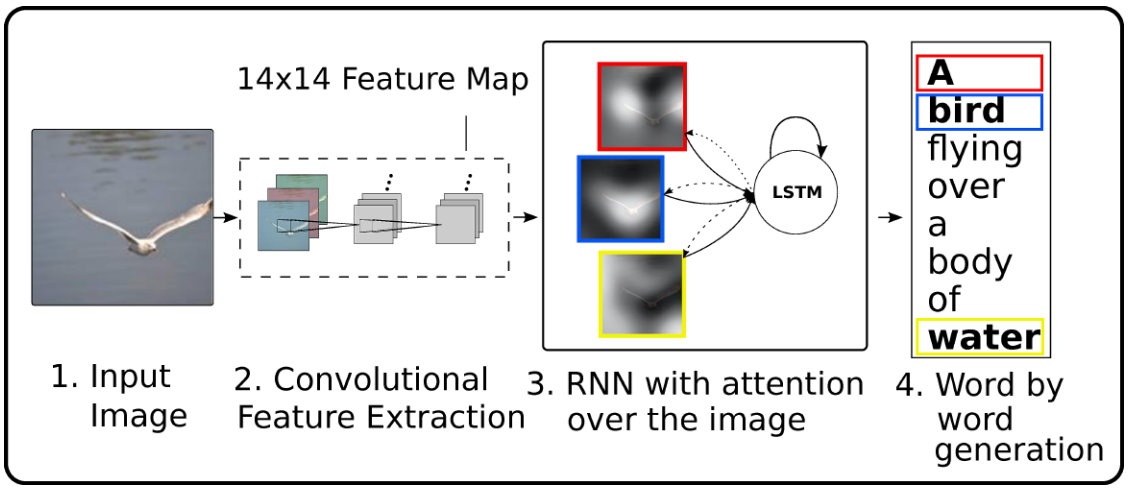
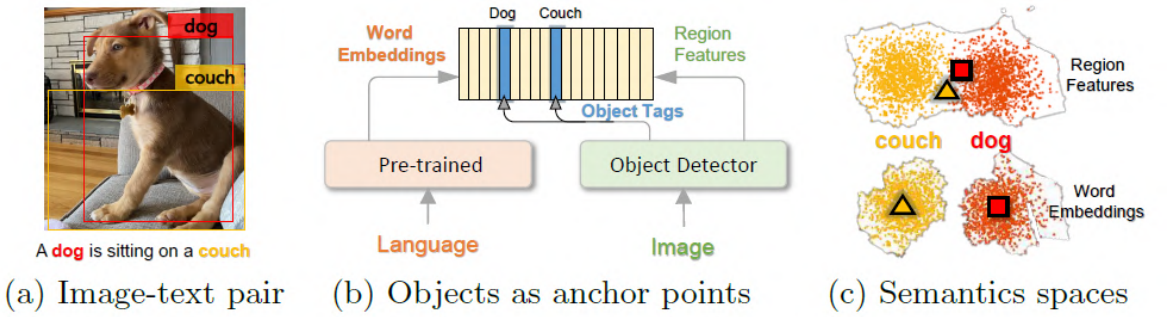


Figure 2.15: Example of an image captioning system with attention, adapted from Xu et al.'s "Show, Attend and Tell" [8]. **1)** The input image is first processed by a CNN, which extracts a 14×14 spatial feature map. **2)** These features represent different regions of the image. **3)** At each word generation step, the LSTM decoder uses an attention mechanism to focus on different spatial regions, as visualized by the attention heatmaps (colored boxes). **4)** This allows the model to select relevant regions when predicting each word. For example, focusing on the bird for "bird" and on the water for "water". Attention mechanisms make generated captions more accurate and descriptive by allowing dynamic focus on important image details.
Image adapted from Show, Attend and Tell.

Transformer-Based Models

More recently, Transformer-based architectures have become popular in image captioning. Transformers were first introduced for language tasks by Vaswani et al.[40]. They do not use recurrence (like RNNs or LSTMs). Instead, they rely entirely on attention mechanisms, like self-attention. Self-attention means that the model looks at all positions in the input (words or objects) at once and learns how important each part is for every other part. It learns complex relationships across the whole sequence On the other hand, regular "attention" usually means focusing on certain parts of another sequence (for example, image features when generating text). Image captioning models such as OSCAR[13], ViLT [55], and the Image Transformer [56] use these attention-based models to find deep connections between images and words. This approach allows for faster, parallel processing and usually leads to better and more accurate captions.



(a) Image-text pair (b) Objects as anchor points (c) Semantics spaces

Figure 2.16: Visualization of the OSCAR model's approach for image captioning with Transformers[13]. **(a)** An input image is paired with a descriptive caption. Detected objects (such as "dog" and "couch") are highlighted with bounding boxes. **(b)** OSCAR uses an object detector to extract "region features" and "object tags" from the image, while also creating word embeddings from the caption. These tags act as anchor points, connecting visual objects directly to corresponding words. All this information is fed into a pre-trained Transformer. Inside the Transformer, *self-attention* allows the model to look at every part of the combined image and text sequence at once, and to decide which parts (objects, words, or regions) are most important for generating the correct caption. **(c)** This approach brings visual features and language features into a shared "semantic space." Words like "dog" and "couch" are closely linked to their visual regions, helping the model generate captions that are both accurate and grounded in the image. *Image adapted from* OSCAR: Object-Semantics Aligned Pre-training for Vision-Language Tasks.

Multimodal Pretraining

Another significant trend is multimodal pretraining, where large models are pretrained jointly on massive datasets consisting of paired images and text. Models such as CLIP [14] and BLIP [17] benefit from large-scale data, learning rich, general-purpose representations useful for a wide variety of vision-language tasks. Although these pretrained models achieve state-of-the-art captioning results, they often demand extensive computational resources and complex training procedures.

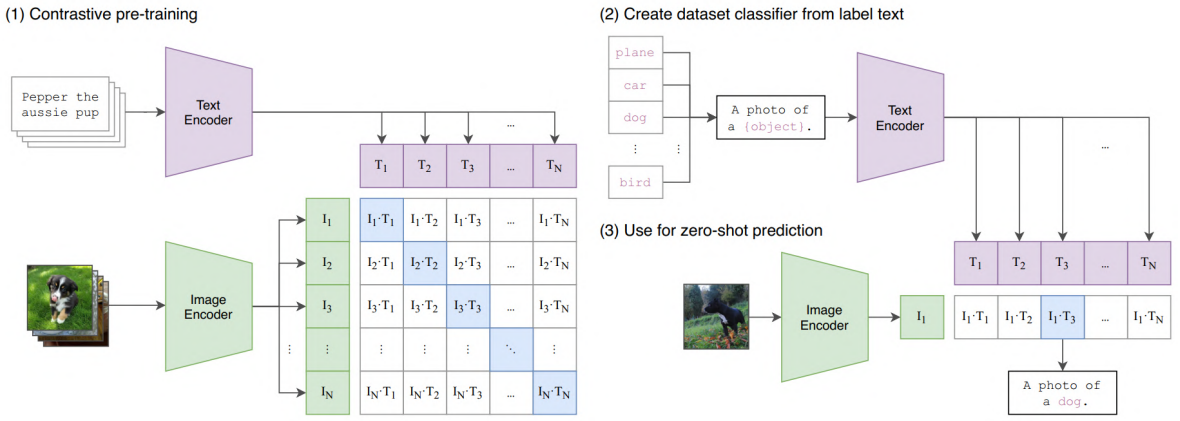


Figure 2.17: Overview of the CLIP model [14]. (1) During contrastive pre-training, the model learns to match images with their corresponding captions by encoding both images and text into a shared embedding space. Each image and text pair is processed by separate encoders, and similarity scores are computed between all possible image-text pairs in a batch. (2) For classification, label names (like "dog," "plane") are converted to text prompts and encoded with the text encoder. The model then compares the image embedding to all text embeddings to select the most relevant label. (3) For zero-shot prediction, CLIP can assign a label to an image without any additional training on that task, simply by matching the image encoding to the closest text encoding. This flexible framework enables CLIP to perform a variety of vision-language tasks by using large-scale pretraining on image-text pairs.
Image adapted from Radford et al., CLIP (2021).

Used Approaches in this Thesis

State-of-the-art captioning models can be hard to train and need a lot of computing power. Simpler methods, like using a standard CNN with an RNN decoder, are still useful, especially for learning, or when resources are limited. These models are easy to reproduce and train quickly. That makes them perfect for exploring the basics of image captioning. Two simple models were used in this thesis to demonstrate the main ideas, while keeping things practical: CNN-LSTM and CNN-Transformer.

Chapter 3

Methodology

In this section, we explain how both models were built and how they work. We show how the Flickr30k dataset is cleaned, tokenized, and prepared. We describe how features are extracted from images using CNNs, and how captions are generated, either with an LSTM or a Transformer that uses self-attention. Each step, from data processing to caption generation, is covered in detail. We also mention the Pros and Cons of each method.

3.1 Technical Details of the Solutions

3.1.1 CNN+LSTM Model

The first model that was built is based on a guide from the Paperspace Blog. It combines a Convolutional Neural Network (CNN, which is ResNet50) with a Long Short-Term Memory (LSTM) network to generate captions for images, as well as Videos.

Data Preparation:

The process begins by collecting a large set of images (31,783 images) and their corresponding captions (158,915 captions). The dataset used was Flickr30k. Each image is associated with exactly 5 human-written descriptions. The captions are cleaned by converting all text to lowercase, removing punctuation, and adding special start and end tokens to each sentence. The dataset is then split into 70% training, 15% validation, and 15% test sets to ensure proper model evaluation. Only words that occur

frequently enough in the training captions are included in the vocabulary, helping the model focus on the most important words.

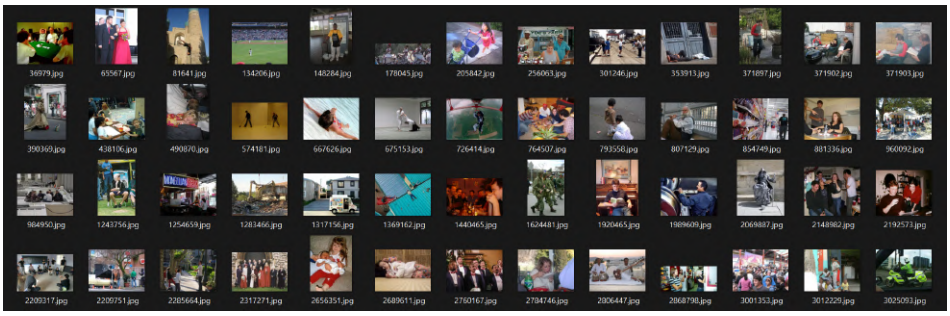


Figure 3.1: A screenshot showing a sample of images from the Flickr30k dataset. Each image in this dataset is paired with five human-written captions describing its content. The dataset covers a wide variety of everyday scenes and activities. This provides a diverse and challenging set of examples for image captioning tasks.

Image Feature Extraction:

Each image is processed using a ResNet50 model that is pre-trained on the ImageNet dataset. The top layer is removed, so the network does not output class predictions but instead produces a 2048-dimensional feature vector for each image. These vectors summarize the main content and are saved to disk for fast access during training.

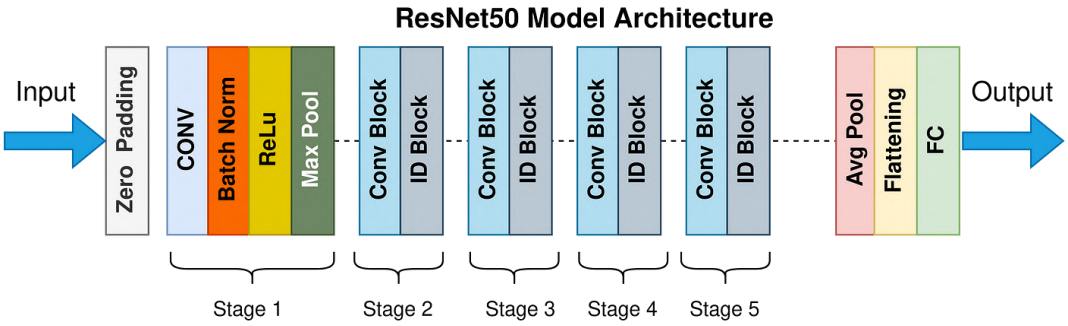


Figure 3.2: Overview of the ResNet-50 architecture used for image feature extraction. The network begins with initial convolutional processing (zero padding, convolution, batch normalization, ReLU, and max pooling), followed by a series of convolutional and identity blocks with residual connections to enable deep feature learning. In our setup, the top classification layer (the green Fully Connected (FC) layer in the figure) is removed, so the output is a 2048-dimensional feature vector representing the main content of the image. These extracted features are then used as input for the caption generation model (LSTM), instead of class predictions. This pipeline allows for efficient extraction of visual features from the dataset's images. Figure adapted from The Annotated ResNet-50.

Text Processing:

All captions are converted into sequences of word indices, based on the vocabulary built from the training set. Shorter captions are padded with zeros, and longer ones are truncated to a fixed length. Two mapping files are created: the "word-to-index" file converts words to their corresponding numerical indices for model input, while the "index-to-word" file does the reverse, allowing generated indices to be translated back into readable captions. This numerical representation is required because neural networks can only process numbers, not raw text.

```
2757779501 <start> a shirtless person with jeans is climbing a rocky mountainside <end>
2757779501 <start> a man climbing the rocky face of a mountain <end>
2757779501 <start> extreme rock wall climbing <end>
2757779501 <start> a climber on some rocks <end>
2757779501 <start> a man rock climbing <end>
4445556540 <start> several african children looking toward the blackboard in school  while two look back toward the camera <end>
4445556540 <start> a group of black children sits behind red desks in a classroom <end>
4445556540 <start> school children reading and listening in africa <end>
4445556540 <start> a crowded classroom in a third world country <end>
4445556540 <start> a classroom full of kids of different ages <end>
```

Figure 3.3: Example of preprocessed captions from the Flickr30k dataset. Each image is associated with five human-written descriptions, with special `<start>` and `<end>` tokens added to mark the boundaries of each caption. These cleaned captions are later converted into sequences of word indices for neural network processing.

Embedding Layer:

To help the model understand the meaning of words, pre-trained GloVe word embeddings are used. GloVe (Global Vectors for Word Representation) provides a 300-dimensional dense vector for each word in the vocabulary, learned from large text corpora. These embeddings capture semantic (with meaning) relationships between words. The embedding layer is initialized with these GloVe vectors and kept frozen during training to preserve their learned knowledge.

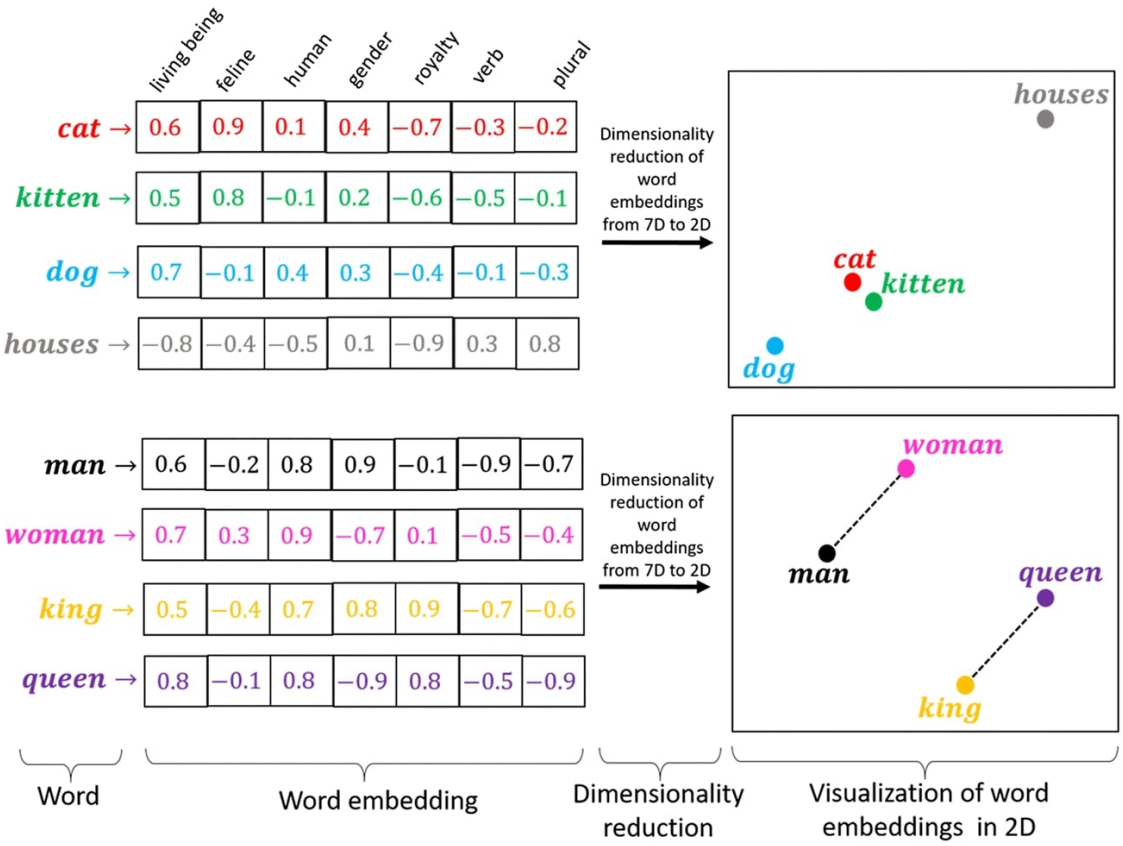


Figure 3.4: Visualization of how pre-trained GloVe word embeddings represent words as dense vectors. In our model, each word in the vocabulary is mapped to a high-dimensional vector that captures semantic relationships, as illustrated on the left. The right side shows these embeddings projected into 2D space, where similar words are clustered together. This embedding layer is initialized with GloVe vectors and kept frozen during training, allowing the model to leverage prior knowledge of word meanings and relationships for improved understanding of input captions. Figure adapted from CoderzColumn: GloVe Embeddings.

Model Architecture:

The model uses two main branches. The first branch processes the image features, passing them through dense layers with batch normalization and dropout for stability and to prevent overfitting. The second branch handles the input caption, using an embedding layer followed by a bidirectional LSTM. The bidirectional LSTM helps the model learn context from both directions in the sequence. This improves its understanding of language structure. The outputs of both branches are combined, further refined, and passed through a final layer that predicts the probability of each word in the vocabulary

at every step of caption generation. Batch normalization layers are used throughout the model to stabilize and speed up training by normalizing the outputs of each layer.

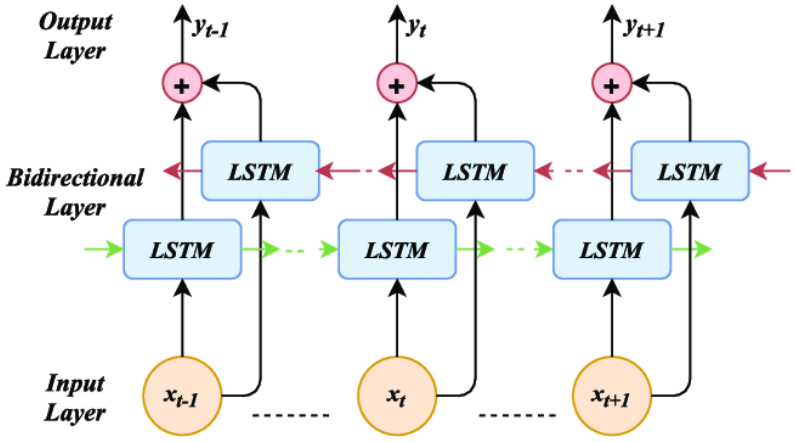
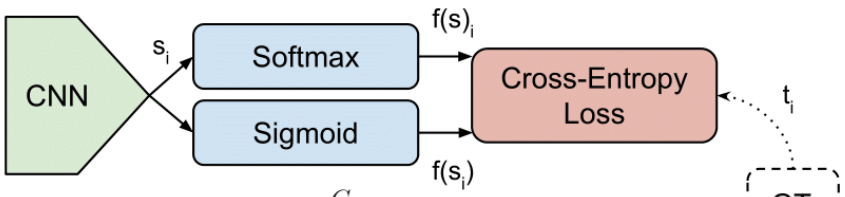


Figure 3.5: Illustration of a Bidirectional LSTM (BiLSTM) network, used in our model to process input captions. The BiLSTM captures context from both past and future words in the sequence, helping the model better understand language structure for caption generation. Figure adapted from Medium: bidirectional LSTM.

Model Training:

The model is trained using the Adam optimizer, which adjusts the learning rate for each parameter automatically. This gets us more efficient and stable training. The loss function used is categorical cross-entropy. It compares the predicted probability distribution for the next word with the true word at each step. Training is performed in batches over multiple epochs, with early stopping to prevent overfitting if validation loss stops improving. The best-performing model is saved during training.



$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$

Figure 3.6: Diagram illustrating how cross-entropy loss is computed during model training. The CNN produces scores s_i, which are converted to predicted probabilities $f(s)_i$ using either Softmax (for multi-class classification) or Sigmoid (for binary classification). These predictions are compared with the ground truth labels (t_i, shown as GT) using the cross-entropy loss. The general cross-entropy formula for multi-class classification is given by $CE = -\sum_i^C t_i \log(f(s)_i)$, where t_i is the true label and $f(s)_i$ is the predicted probability for class i. For binary classification, the loss simplifies to $CE = -t_1 \log(f(s_1)) - (1 - t_1) \log(1 - f(s_1))$, penalizing incorrect predictions more heavily. This loss function guides the optimization process, helping the model improve its predictions over time. Figure adapted from gombru.github.io.

Image Captioning:

After training, the model can generate captions for new images. First, the image is processed through the CNN to extract its features. Then, a caption is generated one word at a time using beam search, which keeps track of several possible sequences and chooses the most likely one. This method produces more accurate and natural-sounding captions compared to simple greedy selection.

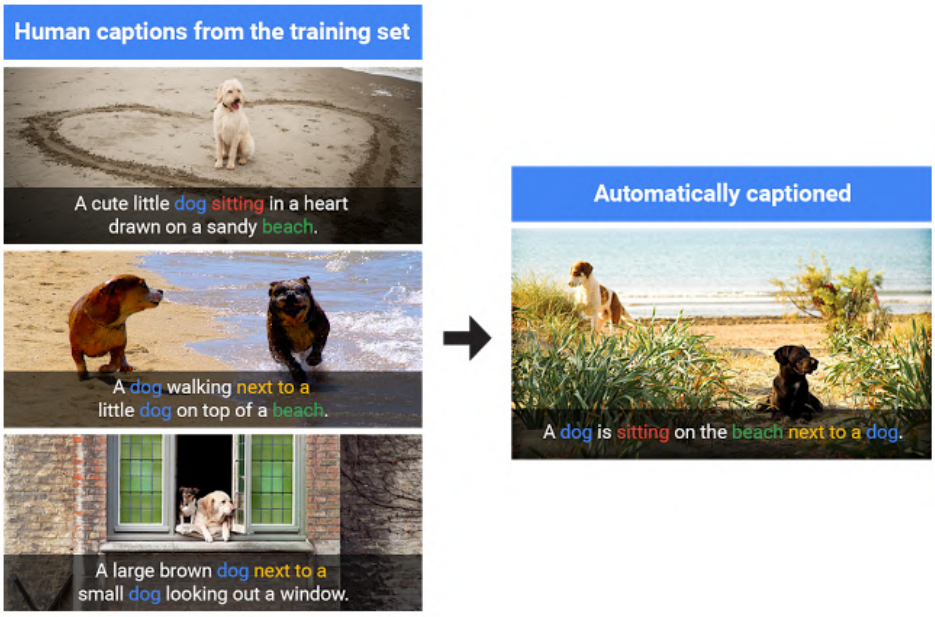


Figure 3.7: Example of automatic image captioning: The left shows human-written training captions for images, while the right displays a caption generated by the trained model for a new image. The model uses visual features and context to create natural-sounding descriptions. Figure adapted from Popular Mechanics.

Video Captioning:

The same model can be used for videos by extracting frames at regular intervals, usually every 30 to 60 frames. A caption is generated for each selected frame, and these captions are joined together in 1 large caption. This one large caption can be summarized into a single sentence using an external language model API, providing a concise summary of the video's content.

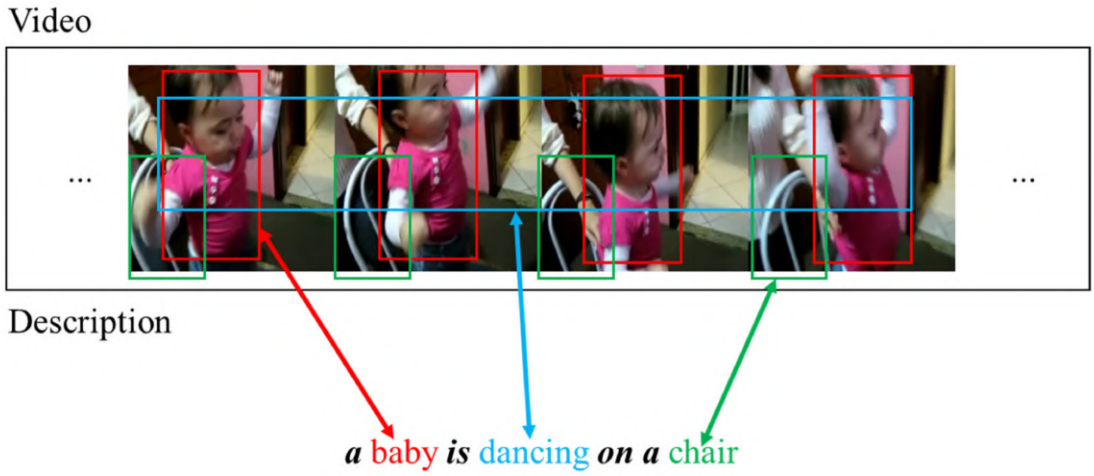


Figure 3.8: Example of video captioning: Key frames are extracted from the video, and image captioning is performed in each frame. The detected actions and objects are used to generate a descriptive sentence summarizing the video's content. Figure adapted from MDPI.

CNN+LSTM Conclusion:

This CNN+LSTM solution forms the foundation for generating descriptive captions from visual content. It provides a strong baseline for comparison with more advanced methods.

3.1.2 CNN+Transformer Model

The second model that was built is based on a guide from Kaggle. It combines a Convolutional Neural Network (CNN, which is EfficientNetB0) with a Transformer, where the Transformer is responsible for generating captions by attending to both the image features and the previously generated words. The Transformer uses self-attention, a mechanism that allows the model to weigh and relate different parts of the input sequence when generating each word in the caption.

Data Preparation

The dataset consists of 31,783 images and their 158,915 captions from Flickr30k. Each image has 5 human-written captions. Captions are cleaned by removing punctuation,

converting text to lowercase, and adding special start and end tokens. Very short or very long captions are filtered out to improve data quality.

Train-Validation Split

The data is randomly split into training (90%) and validation (10%) sets. This allows the model to be trained and evaluated on separate samples for fair evaluation.

```
image,caption
1000092795.jpg, Two young guys with shaggy hair look at their hands while hanging out in the yard .
1000092795.jpg," Two young , White males are outside near many bushes ."
1000092795.jpg, Two men in green shirts are standing in a yard .
1000092795.jpg, A man in a blue shirt standing in a garden .
1000092795.jpg, Two friends enjoy time spent together .
10002456.jpg, Several men in hard hats are operating a giant pulley system .
10002456.jpg, Workers look down from up above on a piece of equipment .
10002456.jpg, Two men working on a machine wearing hard hats .
10002456.jpg, Four men on top of a tall structure .
10002456.jpg, Three men on a large rig .
```

Figure 3.9: Example of image-caption pairs from the Flickr30k dataset. Each image filename is associated with five different human-written captions. This illustrates the diversity of descriptions provided for each image. These captions are cleaned and filtered as part of the data preparation process.

Text Vectorization

A vocabulary is built by selecting the 10,000 most frequent words from the cleaned captions. Each caption is tokenized, meaning it is split into individual words (tokens). Each token is mapped to a unique integer index from the vocabulary. Sequences shorter than 25 tokens are padded with zeros, and those longer than 25 tokens are truncated. This process produces fixed-length numeric arrays, which can be efficiently processed by the neural network. If a caption contains a rare word that is not in the top 10,000, that word is replaced with a special “unknown” token. For example, if the model sees a picture of a zebra, but “zebra” is not in the vocabulary, it might use the word “horse” or even say “striped horse” instead. That;s because those words are more common, and part of its vocabulary.

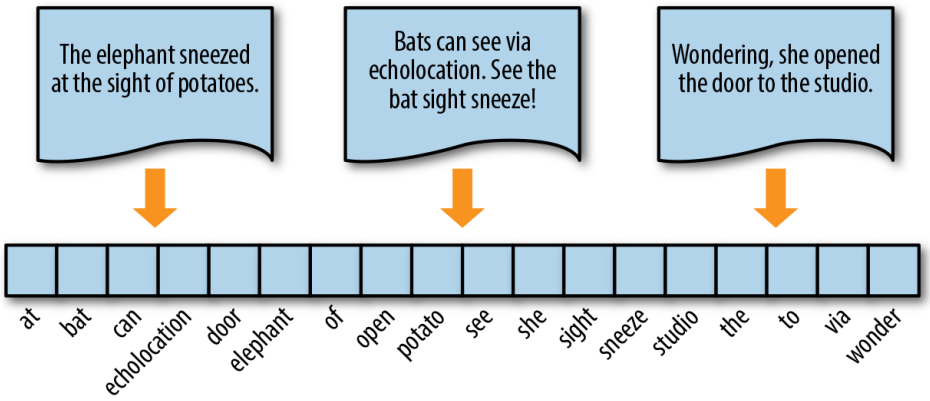


Figure 3.10: Illustration of the text vectorization process: Each sentence, like "The elephant sneezed at the sight of potatoes.", is split into individual words (tokens), resulting in a sequence like ["the", "elephant", "sneezed", "at", "the", "sight", "of", "potatoes"]. Each token is then mapped to a unique index in the vocabulary, transforming the original text into a structured sequence of integers. This step converts variable-length sentences into fixed-length numeric arrays, through padding or truncation, which can be efficiently handled by neural networks for tasks like image captioning. Figure adapted from O'Reilly, Applied Text Analysis with Python.

Image Augmentation

During training, images are randomly flipped, rotated, and have their contrast changed.

This data augmentation helps the model generalize better and prevents overfitting.

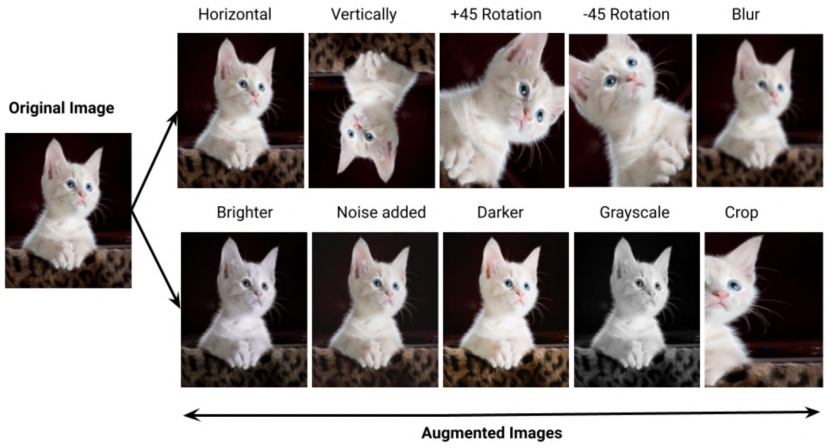


Figure 3.11: Example of image augmentation techniques used during training. Starting from the original image, various transformations such as flipping (horizontal and vertical), rotation, brightness adjustment, noise addition, blurring, grayscale conversion, cropping, and contrast changes are applied to create multiple augmented versions. These diverse samples help the model generalize better and reduce overfitting. Figure adapted from ubiai.tools.

Image Feature Extraction (with EfficientNetB0)

EfficientNetB0, pre-trained on ImageNet, is used as a feature extractor. EfficientNetB0 is designed using a compound scaling method that uniformly scales the network's depth (number of layers), width (number of channels), and input resolution. This architecture achieves high accuracy with less parameters compared to traditional CNNs. The model's top layers are removed, and its parameters are frozen to preserve the learned visual features. Each image is converted to a fixed-size embedding suitable for the Transformer model.

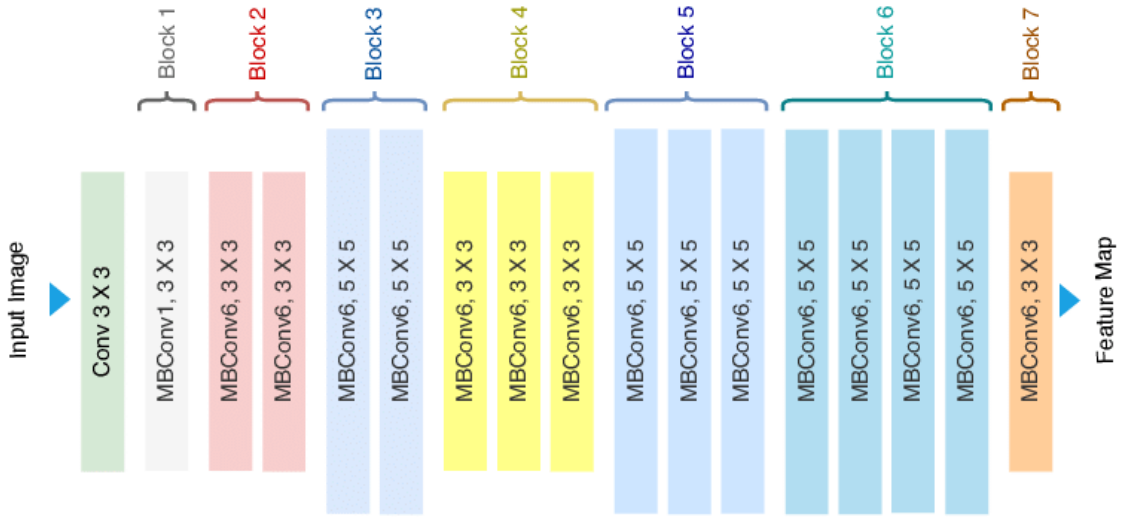


Figure 3.12: Overview of the EfficientNetB0 feature extractor. The model starts with a regular convolution, then processes the image through seven main blocks. Each block contains MBConv layers ("Mobile Inverted Bottleneck" blocks), which are efficient building blocks that help the network learn complex features while using fewer parameters. Some MBConv layers use 3x3 filters, others use 5x5 filters, allowing the model to recognize patterns of different sizes. At the end, the network creates a compact feature map for each image. Figure adapted from ResearchGate.

Transformer Encoder Block

The encoder block takes the image embeddings and applies multi-head self-attention, which allows the model to consider multiple relationships between different parts of the image features in parallel. Each attention head learns to focus on different aspects or regions of the image. Layer normalization is used to standardize activations and speed

up convergence. A feed-forward dense layer transforms the attended features to capture complex patterns in the visual data.

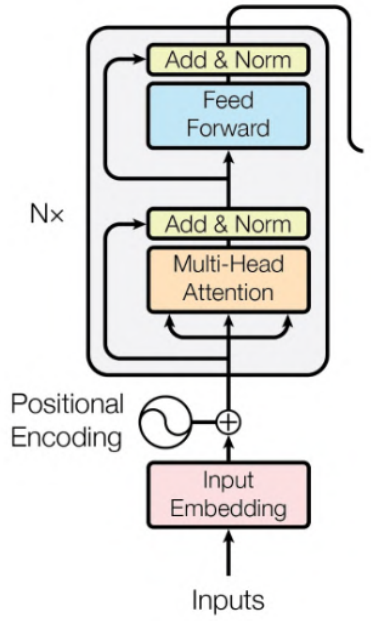


Figure 3.13: The encoder block starts by turning each input (such as an image embedding or word embedding) into a fixed-size vector. Positional encoding is added so the model knows the order of inputs. Then, multi-head self-attention helps the model look at different parts of the input at the same time, learning how features relate to each other. Layer normalization keeps training stable. Finally, a feed-forward network transforms these features, and the process repeats for better learning. This structure helps the transformer understand complex relationships in data before generating captions.

Positional Embedding

This layer adds information about the position of each token in the input caption sequence. Since transformers do not process data sequentially like RNNs, positional embeddings are necessary for the model to capture word order. The layer creates a learnable vector for each possible position in the input sequence of words. It then adds it to the corresponding word embedding. This allows the model to distinguish, for example, which word is first, second, or last in the caption.

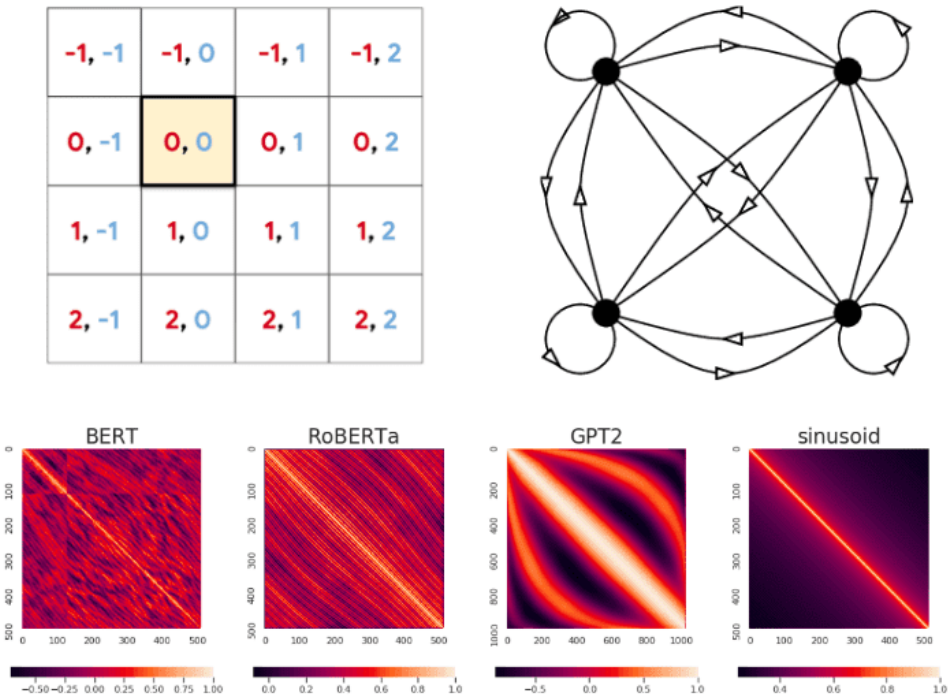


Figure 3.14: Visualization of how transformers use positional information. **Top left:** The table shows relative positions of words in a sequence, with each cell representing a different (row, column) offset. These offsets help the model understand how far apart words are, which is important for encoding word order. **Top right:** The directed graph demonstrates the self-attention mechanism in transformers, where every token (node) can attend to all others, regardless of their position in the sequence. This flexibility is why positional embeddings are needed. Otherwise, the model would treat the input as a bag of words, losing order information. **Bottom:** The four colored heatmaps are actual attention patterns from four different models (BERT, RoBERTa, GPT-2, and sinusoidal embeddings). Each colored pixel shows how strongly a word attends to another word in the sequence. Brighter colors mean stronger attention. The diagonal lines in the sinusoidal embedding indicate attention focused on nearby words, while other models show more complex or global patterns. Together, these parts illustrate how positional encoding and attention work together so transformers can model both word order and complex relationships for tasks like caption generation. Figure inspired by and adapted from The AI Summer.

Transformer Decoder Block

The decoder generates the output caption, one word at a time, by attending to both the encoded image features and the sequence of words generated so far. It uses masked multi-head self-attention. This prevents the model from accessing future words in the sequence during training. That ensures predictions are made based only on known

words. Cross-attention layers then allow the decoder to integrate relevant information from the image features at each decoding step. Layer normalization and feed-forward networks further refine the output representations before predicting the next word.

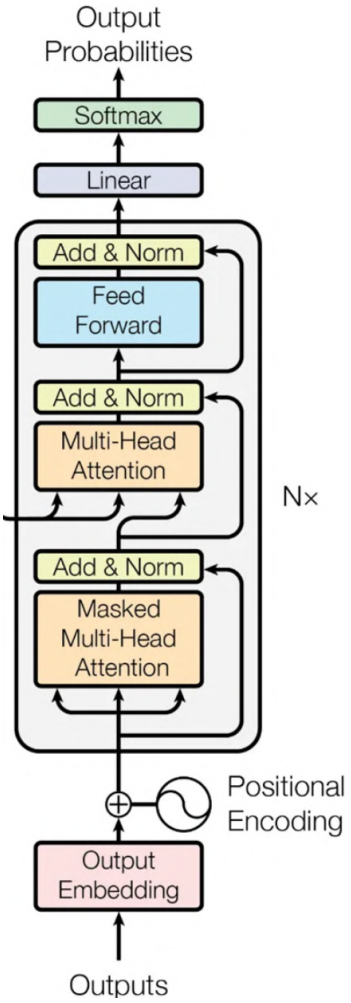


Figure 3.15: The decoder block generates captions one word at a time. It starts with output embeddings plus positional encoding, just like the encoder. The first attention layer is masked, so each word can only see the words before it, never the future words—this keeps the generation process realistic. The next attention layer (cross-attention) lets the decoder use information from the encoder (for example, image features). Layer normalization and a feed-forward network refine the output at each step. The final result is a probability distribution over possible next words, used to pick the most likely caption.

ImageCaptioningModel

This custom model class integrates the CNN feature extractor, the Transformer encoder, and the Transformer decoder into a single architecture. It defines custom training

and validation steps, calculates masked loss and accuracy for caption sequences, and supports training with multiple captions per image. The class handles batching, applies image augmentations, and manages gradient updates across all trainable components.

Loss Function (SparseCategoricalCrossentropy)

The model uses sparse categorical cross-entropy loss. This loss compares the predicted probability distribution at each time step with the true next word in the caption, using only the word index. It is efficient for large vocabularies and multi-class output.

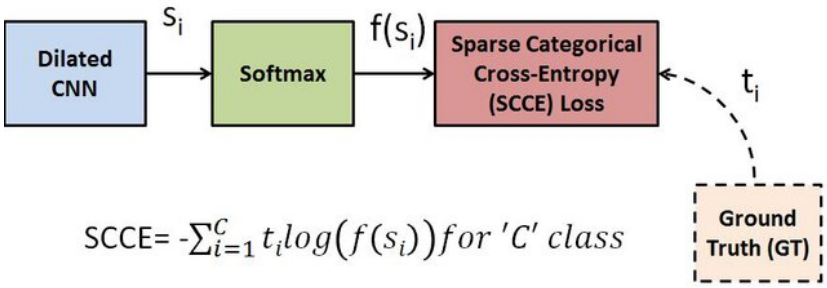


Figure 3.16: Diagram illustrating how sparse categorical cross-entropy (SCCE) loss is used in training. The model outputs a score for each possible class (such as words in a vocabulary) through the CNN. These scores (s_i) are passed through a Softmax function to get predicted probabilities ($f(s)_i$). The SCCE loss then compares these predictions to the ground truth label (t_i), which is the correct class index, not a one-hot vector. The formula shows how the loss is computed for all classes: only the probability for the true class contributes to the loss, making it efficient for large vocabularies and multi-class problems. This guides the model to make better predictions over time.

Learning Rate Scheduling

A custom learning rate schedule with warmup steps is used. The learning rate controls how much the model weights are updated during each step of training. At the start, a warmup phase increases the learning rate gradually from zero to the target value of 1×10^{-4} over several steps. That helps the model avoid unstable updates early in training. After warmup, the learning rate is kept constant. This strategy allows smoother and more stable convergence.

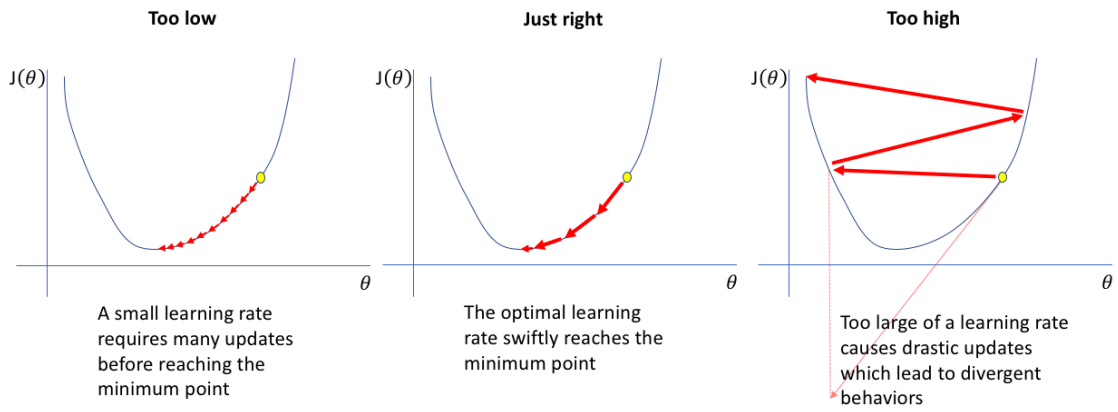


Figure 3.17: Illustration of how different learning rates affect model training. The left plot shows that a low learning rate makes progress slow, requiring many updates to reach the minimum. The middle plot shows an optimal learning rate, which allows the model to reach the minimum efficiently. This is the goal when scheduling the learning rate up to a value such as 1×10^{-4}, as used in our model. The right plot shows that a high learning rate can cause updates to overshoot and diverge. A learning rate schedule with a warmup phase helps avoid instability at the start of training and supports smoother, more stable convergence. Figure adapted from Jeremy Jordan.

Model Training

The model is compiled with the Adam optimizer, which combines momentum (using information from past gradients to speed up training) and adaptive learning rates (automatically adjusting the step size for each parameter). This helps the optimizer efficiently update model weights. Training runs for a fixed number of epochs with early stopping to prevent overfitting. The best weights are saved for inference.

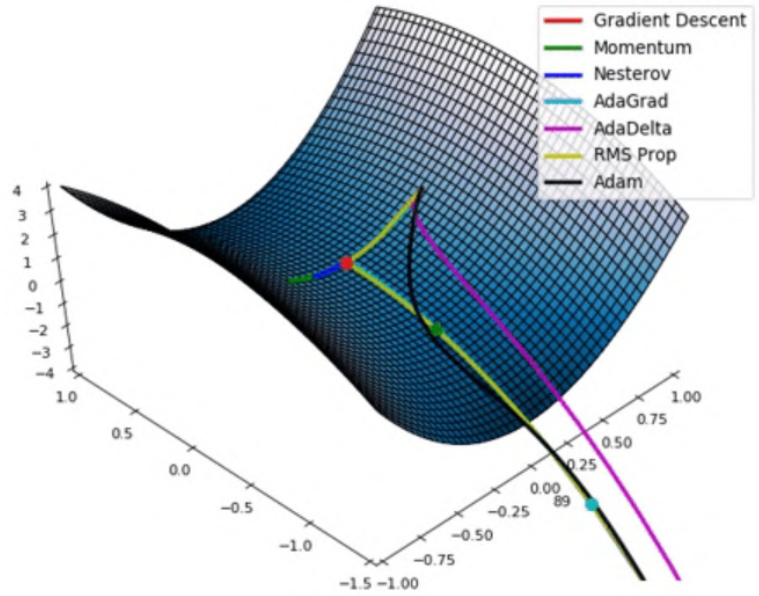


Figure 3.18: Visualization of different optimization algorithms navigating a loss surface. The Adam optimizer (shown in black) combines momentum, which smooths out gradient updates, and adaptive learning rates, which adjust the update size for each parameter based on past gradients. This allows Adam to converge faster and more reliably than standard gradient descent (red) and other optimizers like RMSProp (yellow). In our model, Adam's adaptive approach ensures stable and efficient training, helping the model reach optimal weights for generating accurate captions. Figure adapted from DeepLearning.AI.

Image and Video Captioning

After training, the model generates captions for new images using beam search to select the most likely sequence of words. For videos, frames are sampled (for example, every 30 or 60 frames), each frame is captioned, and the resulting sentences are joined or summarized using an external language model for a final description.

3.2 Pros and Cons of Each Method

CNN+LSTM Model

The CNN+LSTM model is a classic approach for image captioning. It uses a convolutional neural network like ResNet50 to extract visual features from images. Then a Long

Short-Term Memory (LSTM) network generates the caption, word by word. This setup is simple, reliable, and supported in most deep learning libraries. One big advantage is that it works well with smaller datasets (Flickr8k or Flickr30k) and is less demanding on hardware, making it practical for many projects and for educational use. However, LSTM-based models generate captions sequentially, so training and inference can be slower because there is no parallelization across sequence steps. LSTMs also sometimes struggle with very long sentences, as they may forget earlier words or lose context, even with their memory cells. As a result, these models can sometimes miss subtle details or relationships in complex scenes, and generally, their caption quality is not as high as more modern methods like Transformers. Adding an attention mechanism can help, but it also increases complexity and computation time.

CNN+Transformer Model

On the other hand, the CNN+Transformer model represents a more recent and powerful approach to image captioning. Here, a CNN such as EfficientNetB0 extracts the visual features, which are then passed to a Transformer network for caption generation. The Transformer's self-attention mechanism allows the model to consider all words and image regions at once. This helps it generate more accurate and context-aware captions. This parallel processing also makes Transformers much faster to train, especially on large datasets (Flickr30k or MSCOCO). Technically, Transformers can better capture long-range dependencies and complex sentence structures, which is important for generating fluent and detailed captions. However, the added power comes with higher memory requirements and greater computational demands. Training Transformers usually needs larger datasets and more careful tuning of hyperparameters to avoid overfitting or instability. The architecture is more complex, making implementation and debugging more challenging for beginners. Still, with enough resources and data, the CNN+Transformer model sets a new standard for image captioning quality.

Chapter 4

Experimental Results and Validation

4.1 Dataset description

The dataset used in both the CNN+LSTM and CNN+Transformer models that were developed in this thesis is Flickr30k. It is a popular and widely-used dataset in image captioning research. Flickr30k contains 31,783 images collected from the Flickr photo-sharing platform. What makes it special is that each image is paired with five different captions, resulting in a total of 158,915 captions. These captions are written by humans and describe the image from multiple perspectives. This variety helps models learn to generate richer and more natural descriptions.

The captions are stored in a file usually called `captions.txt`, and sometimes `captions.csv`. This file is formatted as a simple table with two columns: the image filename and its corresponding caption. Here is a small sample of what the `captions.txt` file looks like:

Image	Caption
...	...
1000092795.jpg	Two young guys with shaggy hair look at their hands while hanging out in the yard.
1000092795.jpg	Two young, White males are outside near many bushes.
1000092795.jpg	Two men in green shirts are standing in a yard.
1000092795.jpg	A man in a blue shirt standing in a garden.
1000092795.jpg	Two friends enjoy time spent together.
10002456.jpg	Several men in hard hats are operating a giant pulley system.
10002456.jpg	Workers look down from up above on a piece of equipment.
10002456.jpg	Two men working on a machine wearing hard hats.
10002456.jpg	Four men on top of a tall structure.
10002456.jpg	Three men on a large rig.
...	...

Flickr30k was created in 2014 by Young and colleagues. Their goal was to provide a large and diverse dataset for training and evaluating image captioning models. Before Flickr30k, datasets were smaller or less varied, limiting progress in this field. Thanks to its size and rich captions, Flickr30k has become a standard benchmark. It helps researchers develop models that understand images better and generate human-like captions. This makes Flickr30k an ideal choice for the experiments and models in this thesis [57].

Although this thesis primarily uses the Flickr30k dataset for training image captioning models, the CNN encoder for the CNN+LSTM model (ResNet50) is initialized with weights pretrained on the ImageNet dataset [58]. ImageNet is a large-scale image classification dataset containing over 14,197,122 images labeled across thousands of categories. Pretraining on ImageNet allows convolutional networks to learn general visual features that transfer well to other tasks, improving accuracy and reducing training time on smaller datasets such as Flickr30k.

4.2 Software tools and involved libraries

Several important Python libraries and tools were used for this thesis. Below is a summary of each, along with a short description:

Table 4.1: Key Python libraries and tools used in the project.

Library/Tool	Description
`TensorFlow & Keras`	Main deep learning framework for building, training, and testing CNN+LSTM and CNN+Transformer models. Keras offers a high-level neural network API.
`NumPy`	Arrays, matrices, numerical computations, and data manipulation.
`Pandas`	Handling tabular data, e.g., CSV files with captions.
`Matplotlib`	Plotting graphs, displaying images, and model outputs.
`OpenCV (cv2)`	Loading, processing, and manipulating images/videos.
`Pillow (PIL)`	Opening, editing, and saving images.
`tqdm`	Progress bars for loops during feature extraction or captioning.
`scikit-learn`	Splitting data into train, validation, and test sets.
`nltk`	Caption quality evaluation using BLEU scores.
`glob`	File pattern matching in directories.
`pickle`	Saving/loading Python objects, e.g., image features.
`EfficientNet`	Pre-trained CNN for efficient image feature extraction.
`ResNet50`	Pre-trained CNN for visual feature extraction.
`requests`	HTTP requests to external APIs, e.g., for captions.
`heapq`	Efficiently finds largest/smallest items, e.g., in beam search.
`os`	File system operations.
`json`	Handling JSON data, especially with APIs.
`re`	Cleans captions via regular expressions.
`random`	Shuffling data, setting seeds for reproducibility.

These libraries were all used to handle all stages of the project, including data loading and cleaning, image and video processing, deep learning, evaluation, and presentation of results.

4.3 Hardware specifications used

Both the CNN+LSTM and CNN+Transformer models were trained on a 2021 Lenovo Legion 5 Pro laptop. It has an NVIDIA RTX 3060 graphics card with 6GB of VRAM.

The CPU is an AMD Ryzen 7 5800H, and the system has 32GB of RAM, with additional virtual memory automatically managed by the operating system.

Training was done locally using a conda environment in Jupyter Notebook. Although Google Colab offers better GPUs (T4, L4, A100), the dataset files on Colab are deleted when sessions time out, so the whole dataset would have to be re-uploaded every time. This made a local setup more reliable for this thesis.

4.4 Training, testing, and validation results

In this section, we show the training pipeline, how we tested both models using the BLEU evaluation metric, and finally results are compared for both models.

4.4.1 CNN-LSTM

Training

The CNN-LSTM model was trained using the Adam optimizer with a learning rate of 0.0005 and categorical cross-entropy loss, which is suitable for multi-class classification. Training was performed for up to 30 epochs with early stopping (patience = 8) to prevent overfitting and a model checkpoint to save the best-performing weights based on validation loss. Each epoch processed batches of 8 samples, with steps per epoch and validation steps determined by the dataset size.

During training, the model's loss and accuracy improved steadily. For example, the training loss decreased from 3.78 to 2.62, and accuracy increased from 0.32 to 0.42 over several epochs. The validation accuracy similarly improved from 0.36 to 0.39, with the best model being saved whenever the validation loss reached a new minimum.

Here's a part of the training loop:

```
1 Epoch 16: val_loss improved from 3.04064 to 3.03167, saving model to best_model_6.h5
2 2781/2781 [==============================] - 1891s 680ms/step - loss: 2.8177 -
      accuracy: 0.4018 - val_loss: 3.0317 - val_accuracy: 0.3837
3
```

```
Epoch 17: val_loss did not improve from 3.03167
2781/2781 [==============================] - 1864s 670ms/step - loss: 2.7084 -
    accuracy: 0.4131 - val_loss: 3.0398 - val_accuracy: 0.3847

Epoch 18: val_loss did not improve from 3.03167
2781/2781 [==============================] - 2039s 733ms/step - loss: 2.6179 -
    accuracy: 0.4234 - val_loss: 3.0542 - val_accuracy: 0.3855
```

Testing

The image captioning model is evaluated using the following procedure:

1. Feature Extraction: A pre-trained ResNet50 model (with ImageNet weights and without the classification layer) is used to extract a 2048-dimensional feature vector from each test image.

2. Loading the Captioning Model: The trained CNN-LSTM model, along with word-to-index and index-to-word mappings, are loaded from disk for inference.

3. Caption Generation (Beam Search): Captions are generated word by word using beam search.

4. Caption Enhancement with LLM: The raw generated caption is refined using a Large Language Model via an API.

5. Visualization: The test image is displayed together with both the raw and enhanced captions.

The model is also tested on videos using the following steps:

1. Frame Extraction: Video frames are sampled at fixed intervals using OpenCV.

2. Feature Extraction per Frame: Each extracted frame is processed through ResNet50.

3. Caption Generation per Frame: A caption is generated every 30 or 60 frames (the user can modify the value to their liking).

4. Caption Summarization: All captions are concatenated, then summarized by an LLM API.

5. Visualization and Output: The captions and summarized output are displayed.

This approach enables automated captioning for both images and videos, with additional LLM post-processing for enhanced output quality.

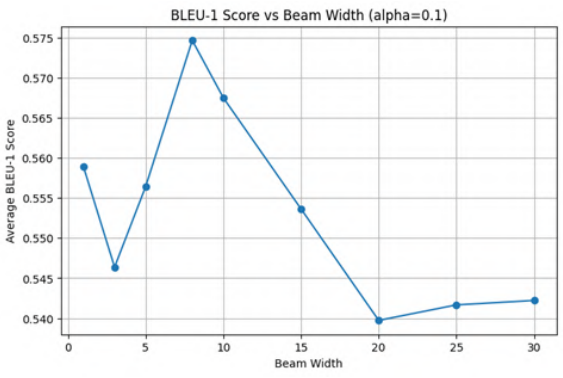
Results

To evaluate the performance of the CNN-LSTM model, BLEU-1 scores were computed on a sample of 100 test images using various beam widths and two different values for the length normalization parameter α (0.1 and 1.5). The BLEU-1 score measures the unigram (word-level) overlap between the generated captions and the reference captions.

The results are summarized in the tables and plots below.

Beam Width	BLEU-1 Score
1	0.5589
3	0.5464
5	0.5564
8	0.5747
10	0.5675
15	0.5536
20	0.5397
25	0.5417
30	0.5422

(a) BLEU-1 Scores (Alpha = 0.1)

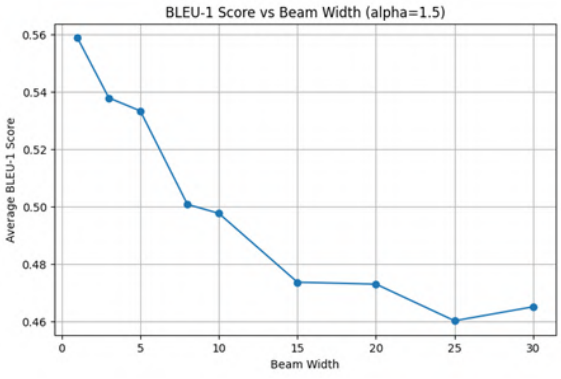


(b) BLEU-1 vs Beam Width ($\alpha = 0.1$)

Figure 4.1: BLEU-1 score as a function of beam width for $\alpha = 0.1$. The table (left) lists the scores for each beam width, while the plot (right) shows the trend, peaking at beam width 8.

Beam Width	BLEU-1 Score
1	0.5589
3	0.5379
5	0.5334
8	0.5008
10	0.4977
15	0.4736
20	0.4730
25	0.4602
30	0.4651

(a) BLEU-1 Scores (Alpha = 1.5)

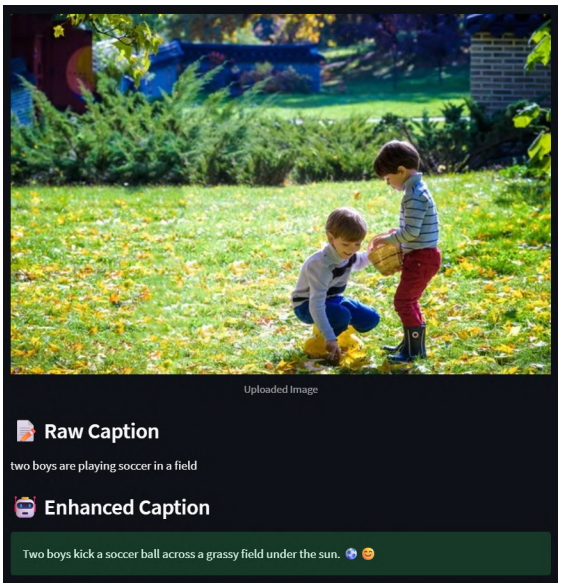


(b) BLEU-1 vs Beam Width ($\alpha = 1.5$)

Figure 4.2: BLEU-1 score as a function of beam width for $\alpha = 1.5$. The table (left) lists the scores for each beam width, while the plot (right) shows that increasing beam width leads to steadily decreasing BLEU-1 scores. This suggests that strong length normalization can reduce caption accuracy with wider beams.

Interpretation:

For $\alpha = 0.1$, BLEU-1 scores peak at a beam width of 8 (0.5747), suggesting that moderate beam widths can improve caption diversity and accuracy without over-penalizing longer sequences. For $\alpha = 1.5$, the BLEU-1 scores decrease steadily as beam width increases, indicating that excessive length normalization may hurt performance with wider beams. The results suggest that careful tuning of beam width and normalization is important for optimal caption generation quality.



Description: Two young boys are in a grassy field, seemingly playing or gardening.

Generated Caption: "Two boys are playing soccer in a field."

Analysis: The model hallucinated a soccer ball that does not exist in the image. This shows that while the scene context is mostly correct (boys, field), specific details can be fabricated.

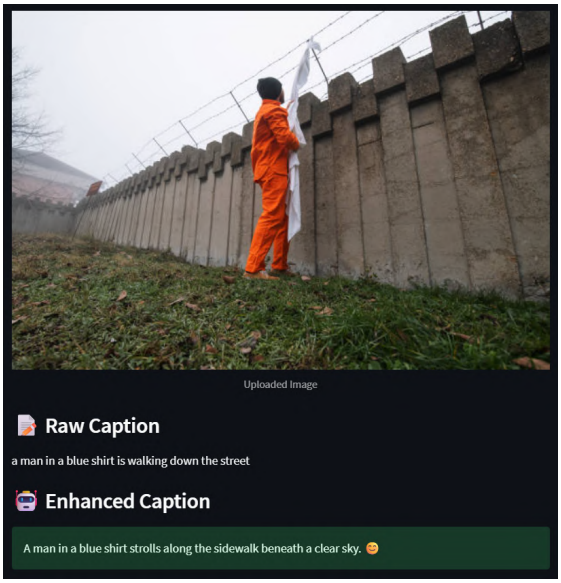
Figure 4.3: CNN-LSTM Example 1

Description: An Asian girl stands on a grassy hill, with the ocean and a rainbow behind her.

Generated Caption: "A young girl in a pink shirt running through a field of tall grass."

Analysis: The caption is mostly accurate, correctly identifying the girl and her location. But it incorrectly states she is running, doesn't mention background elements like the ocean and rainbow.

Figure 4.4: CNN-LSTM Example 2



Description: A prisoner in an orange jumpsuit is attempting to climb a barbed wire fence inside a prison.

Generated Caption: "A man in a blue shirt is walking down the street."

Analysis: The model misidentifies both clothing color and context, describing a street and blue shirt not present in the image. But it still correctly recognizes there is a man.

Figure 4.5: CNN-LSTM Example 3

4.4.2 CNN-Transformer

Training

The CNN-Transformer model was trained using a custom workflow. This included a learning rate schedule with a warmup phase, the Adam optimizer, and sparse categorical cross-entropy loss. Early stopping was used to avoid overfitting and to restore the best weights if validation loss stopped improving.

First, the loss function was set up using Keras' SparseCategoricalCrossentropy. An early stopping callback was added, set to monitor validation loss and trigger after three stagnant epochs. A learning rate scheduler was created to gradually increase the learning rate during the initial steps and then keep it stable.

Training and validation datasets were prepared. The model was compiled with the Adam optimizer, configured to use the custom learning rate schedule. The .fit() method handled the training, passing in the early stopping callback and validation data. Training was allowed up to 20 epochs, but early stopping halted the process at epoch 11.

Below is the log for some key epochs during training:

```
Epoch 1/20
377/377 [==========================] - 140s 344ms/step - loss: 19.3299 - acc: 0.3222 -
    val_loss: 17.3777 - val_acc: 0.3681
Epoch 7/20
377/377 [==========================] - 165s 439ms/step - loss: 13.6409 - acc: 0.4298 -
    val_loss: 14.9108 - val_acc: 0.4086
Epoch 11/20
377/377 [==========================] - 179s 476ms/step - loss: 12.3763 - acc: 0.4621 -
    val_loss: 15.0929 - val_acc: 0.4080
```

As the log shows, both loss and accuracy improved steadily. Early stopping ensured the best validation loss was restored. The final model weights were saved for later use.

Testing

Before inference, the model and all layers (including data augmentation) are initialized. Trained weights are loaded from disk.

```
dummy_images = tf.random.uniform((1, *IMAGE_SIZE, 3))
dummy_sequences = tf.random.uniform((1, SEQ_LENGTH), maxval=VOCAB_SIZE,
    dtype=tf.int32)
_ = image_augmentation(dummy_images)
_ = caption_model((dummy_images, dummy_sequences), training=True)
caption_model.load_weights('best_transformer_1_weights.h5')
print("Weights loaded successfully!")
```

For image captioning, each image is encoded. Captions are generated using beam search, allowing tuning of beam width and length normalization. For video captioning, frames are sampled from the video. Each frame is captioned, and all captions are then summarized into a single description using an LLM API. This is similar to the process used for the CNN-LSTM model.

This method allows the CNN-Transformer to produce accurate and context-aware captions for both images and video frames. It benefits from advanced sequence modeling and attention mechanisms, leading to strong performance.

Results

Even though the model often generates captions that make sense and match the images, the BLEU scores can look a bit low. This is because BLEU only checks how many exact words and word pairs match the reference captions. There are many ways to describe the same image, so a good caption can still get a modest BLEU score if the wording is different.

Below are the average BLEU-1 to BLEU-4 scores for different beam width and α settings:

Table 4.2: BLEU Scores (Beam width = 5, $\alpha = 0.7$)

Score	**Value**
BLEU-1	0.5471
BLEU-2	0.3606
BLEU-3	0.2378
BLEU-4	0.1607

Table 4.3: BLEU Scores (Beam width = 5, $\alpha = 10$)

Score	**Value**
BLEU-1	0.5527
BLEU-2	0.3719
BLEU-3	0.2365
BLEU-4	0.1501

Table 4.4: BLEU Scores (Beam width = 10, $\alpha = 20$)

Score	**Value**
BLEU-1	0.5505
BLEU-2	0.3736
BLEU-3	0.2417
BLEU-4	0.1563

Overall, these results are typical for image captioning with a Transformer. BLEU scores alone do not tell the whole story. Many generated captions are actually very good, even if the score is not high. We can see a few examples in the following pictures:

Figure 4.6: In this image, the predicted caption was "a young man in a white shirt is sitting on the floor". He is in fact a young man wearing a white shirt, even though the actual captions didn't have either of those 2. Little issue is that he's sitting on a toy giraffe not the floor, but there is an actual floor around him, which is sort of valid.

Figure 4.7: The Predicted Caption for this image was "a man in a suit walks down the street" which is correct. But the BLEU scores aren't the best, since it measures how many sequences of words in the generated caption match those in the actual captions

Figure 4.8: Again for this image, the predicted caption "a little girl in a red shirt is standing in the woods" is sort of correct, except it doesn't mention the deer in the background nor what's in the girl's hand.

Figure 4.9: Here, the predicted caption "a man in a yellow shirt and blue jeans is walking down the street" is correct, even if the caption doesn't mention him posing for a photo, or wearing colorful accessories.

4.4.3 Comparing the 2 models

The CNN-LSTM and CNN-Transformer models both generate image and video captions, but in different ways. The CNN-LSTM relies on a sequential LSTM decoder, which is simple and fast, but sometimes misses global context in the caption. The Transformer, on the other hand, uses self-attention to look at all words and image features at once, capturing more complex relationships. In practice, the Transformer model produced slightly better captions, especially for longer and more complicated images, although both models reached similar BLEU-1 scores. Overall, the Transformer is more flexible and accurate, but requires more computation and careful tuning.

For human evaluation on the captions, several random images outside the dataset were tested. In the CNN-LSTM model, around 40–60% of the generated captions were judged as satisfying, meaning they made sense and matched the images well. In comparison, the CNN-Transformer model achieved a higher satisfaction rate, with approximately 60–80% of captions considered satisfactory. This highlights the Transformer's ability to better understand new and unseen images. While both models are capable of producing good results, the Transformer's advanced attention mechanism clearly helps it generalize more reliably. Still, the LSTM is not far behind and works surprisingly well, especially for straightforward images. Overall, the Transformer is better at producing natural and accurate captions, but the LSTM remains a solid, lightweight choice.

Here are 6 comparison examples between the two models, with each model's generated raw caption, and a small comparison:

CNN-LSTM: a little boy in a bathrobe plays with vacuum cleaner vacuum cleaner

CNN-Transformer: a little girl is taking a picture

Comparison: The LSTM model got confused and thought the girl was a boy, playing with a vacuum cleaner. The Transformer did much better. It correctly said a little girl is taking a picture, even though it's a camera tripod. This shows the Transformer understands the scene more accurately.

Figure 4.10: Comparison of generated captions for Example 1.

CNN-LSTM: a young woman in a flower dress is dancing in front of flowers flowers

CNN-Transformer: a girl in a pink dress is standing in front of flowers

Comparison: The LSTM model repeated the word "flowers" and guessed a "flower dress," which isn't accurate. It's just a pink dress. The Transformer got it right by mentioning the girl in a pink dress and the flowers, making its caption much more natural. Nevertheless, the LSTM didn't do a bad job at captioning.

Figure 4.11: Comparison of generated captions for Example 2.

CNN-LSTM: a man in a black and white striped shirt is driving a car on a road
CNN-Transformer: a man in a blue shirt is standing next to a white car

Figure 4.12: Both models spotted the car, but neither one mentioned the woman or the dog. The LSTM got the idea of a man and a car, but added details that weren't there, like "driving" and "striped shirt". The Transformer caption was closer: it noticed a man and a white car, but it's the woman wearing blue. Still, both missed key parts of the scene.

CNN-LSTM: a group of people are watching a baseball game in a park
CNN-Transformer: a group of men are riding a red car

Figure 4.13: Neither model really described what's happening. The LSTM made up a baseball game that isn't there, but at least noticed the park. The Transformer thought people were riding in the car, when actually they're just standing in front of it. But the Transformer noticed the red car.

CNN-LSTM: a young boy playing with a soccer ball
CNN-Transformer: a young girl is playing with a green toy

Figure 4.14: Both captions are sort of correct, but miss some scene elements. The LSTM focused only on the boy and the soccer ball, but ignored the blocks and other kids. The Transformer mentioned a green toy and one of the girls, but missed the rest.

CNN-LSTM: a white dog runs through the grass with a white ball in its mouth
CNN-Transformer: a white dog is looking at a red ball

Figure 4.15: The LSTM model made up some action, and got the ball color wrong. It's not white and the dog isn't running. The Transformer did better, catching that the dog is just looking at a red ball, even though it didn't say the dog was chewing it. Overall, the Transformer was closer to reality.

Chapter 5

Conclusion & Future Directions

5.1 What was done in this thesis?

In this thesis, two deep learning models were designed and implemented for automatic image and video captioning: a CNN-LSTM model and a CNN-Transformer model. The Flickr30k dataset was collected, cleaned, and preprocessed. Visual features were extracted from images using pre-trained CNNs (ResNet50 and EfficientNetB0). Captions were generated using both LSTM and Transformer-based decoders. The models were trained, validated, and tested using standard machine learning workflows, including data augmentation and early stopping. Performance was evaluated with BLEU scores and additional human assessment on unseen images and videos. The full pipeline was developed to support both image and video captioning, providing a comprehensive comparison between the two approaches.

5.2 Results in brief

Both models successfully generated captions for images and videos from the Flickr30k dataset. The CNN-LSTM model performed well on simple scenes, reaching a BLEU-1 score of 0.57 with beam width 8 and $\alpha = 0.1$. The CNN-Transformer model produced slightly better and more descriptive captions, especially for complex or longer sequences, with comparable BLEU scores to the CNN-LSTM. Human evaluation showed that the Transformer model generated more accurate and natural sounding captions on most test images. Overall, the Transformer-based approach was more flexible and robust,

while the LSTM model remained easier to train.

5.3 What has been learned

Throughout this thesis, a deep understanding was gained in every stage of the image captioning pipeline. This includes data collection, cleaning, and preprocessing for large datasets. Key concepts in convolutional neural networks (CNNs), long short-term memory networks (LSTMs), and Transformer architectures were learned and applied. The training process for deep learning models was explored in detail, along with practical skills in hyperparameter tuning, loss functions, and evaluation metrics like BLEU. Experience was also gained in model deployment, software tools, and integrating external APIs for enhanced captioning. Overall, the project provided hands-on knowledge of modern computer vision and natural language processing techniques.

5.4 Future Work

There are many ways to extend this project in the future:

- Using advanced vision models: Newer models like Vision Transformers (ViT) or CLIP can be tried instead of ResNet and EfficientNet. For example, ViT treats images as sequences, similar to text, and has shown strong performance on various vision tasks. The only problem is that they're computationally demanding, which would require a better GPU.

- Larger and more diverse datasets: Training on datasets bigger than Flickr30k, such as MSCOCO or Google Conceptual Captions, can help the models handle more types of images and captions, and reduce overfitting.

- Combine with reinforcement learning: After initial training, reinforcement learning (a technique where the model learns by receiving feedback or "rewards" for

good outputs) can be used to directly optimize for caption quality metrics like BLEU or for human feedback, which may produce more natural captions.

- Interpretability: Adding attention map visualizations can help show which parts of an image or caption the model focuses on when generating each word, making the process easier to understand.

- Real-world deployment: Models can be deployed in mobile apps or on edge devices for real-time captioning, such as assisting visually impaired users with live image descriptions.

- Multilingual and domain adaptation: Training or fine-tuning the models to work in different languages, or on medical images, security footage, or social media content, could make the solution useful in more fields.

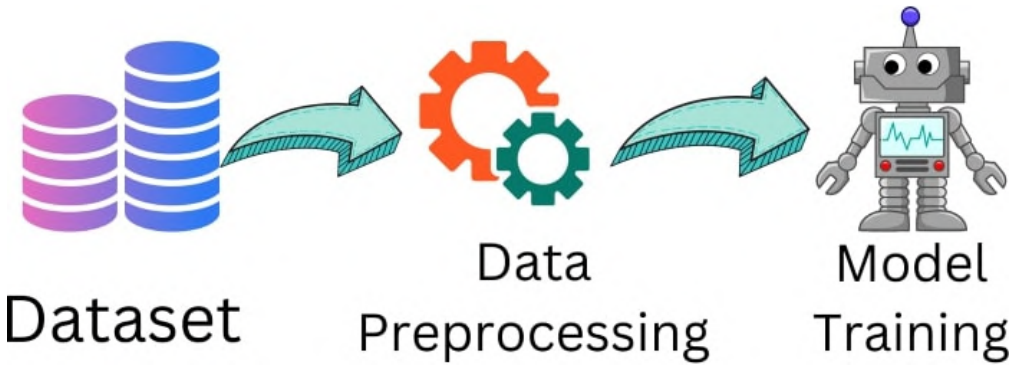


Figure 5.1: Overview of the main pipeline used in this thesis: starting from a dataset, followed by data preprocessing, and finishing with model training and evaluation.
Figure adapted from AirByte

Chapter 6

Appendix

This section of the thesis contains the Github Repository with all of this thesis' files and codes. This section also contains the LLM API link used for caption Enhancement & Summarization for the Video Captioning, as well as the list of all Figures and Tables.

Table 6.1: Project Resources

Resource	**Link**
Github Repository	https://github.com/MichaelWeesa/Image-Captioning
DeepSeek R1 API	https://openrouter.ai/deepseek/deepseek-r1:free

List of Figures

List of Tables

Bibliography

[1] A. Farhadi, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth, "Every picture tells a story: Generating sentences from images," in *European Conference on Computer Vision (ECCV)*, 2010. [Online]. Available: https://www.cs.cmu.edu/~afarhadi/papers/sentence.pdf.

[2] V. Ordonez, G. Kulkarni, and T. L. Berg, "Im2text: Describing images using 1 million captioned photographs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2011. [Online]. Available: http://www.tamaraberg.com/papers/generation_nips2011.pdf.

[3] G. Kulkarni, V. Ordonez, S. Dhar, S. Li, Y. Choi, and T. L. Berg, "Babytalk: Understanding and generating simple image descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2891–2903, 2013. [Online]. Available: http://www.tamaraberg.com/papers/babytalk_pami.pdf.

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2015/html/Vinyals_Show_and_Tell_2015_CVPR_paper.html.

[5] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2015/html/Karpathy_Deep_Visual-Semantic_Alignments_2015_CVPR_paper.html.

[6] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," in *International Conference*

on Learning Representations (ICLR), 2015. [Online]. Available: https://arxiv.org/abs/1412.6632.

[7] J. Donahue, L. A. Hendricks, S. Guadarrama, *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2015/html/Donahue_Long-Term_Recurrent_Convolutional_2015_CVPR_paper.html.

[8] K. Xu, J. Ba, R. Kiros, *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 2048–2057. [Online]. Available: https://arxiv.org/abs/1502.03044.

[9] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/You_Image_Captioning_With_CVPR_2016_paper.html.

[10] P. Anderson, X. He, C. Buehler, *et al.*, "Bottom-up and top-down attention for image captioning and visual question answering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Anderson_Bottom-Up_and_Top-Down_CVPR_2018_paper.html.

[11] L. Huang, W. Wang, J. Chen, and X.-Y. Wei, "Attention on attention for image captioning," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/papers/Huang_Attention_on_Attention_for_Image_Captioning_ICCV_2019_paper.pdf.

[12] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara, "Meshed-memory transformer for image captioning," in *IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), 2020. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Cornia_Meshed-Memory_Transformer_for_Image_Captioning_CVPR_2020_paper.html.

[13] X. Li, X. Yin, C. Li, *et al.*, "Oscar: Object-semantics aligned pre-training for vision-language tasks," in *European Conference on Computer Vision (ECCV)*, 2020. [Online]. Available: https://arxiv.org/abs/2004.06165.

[14] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020.

[15] R. Mokady, A. Hertz, and A. H. Bermano, "Clipcap: Clip prefix for image captioning," *arXiv preprint arXiv:2111.09734*, 2021. [Online]. Available: https://arxiv.org/abs/2111.09734.

[16] X. Zhang, X. Li, L. Zhang, *et al.*, "Vinvl: Revisiting visual representations in vision-language models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Zhang_VinVL_Revisiting_Visual_Representations_in_Vision-Language_Models_CVPR_2021_paper.html.

[17] J. Li, D. Li, C. Xiong, and S. C. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International Conference on Machine Learning (ICML)*, 2022. [Online]. Available: https://arxiv.org/abs/2201.12086.

[18] J. Li, D. Li, S. Savarese, and S. C. H. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023. [Online]. Available: https://arxiv.org/abs/2301.12597.

[19] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko, "Object hallucination in image captioning," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. [Online]. Available: https://aclanthology.org/D18-1437.

[20] Y. Tewel, Y. Shalev, I. Schwartz, and L. Wolf, "Zerocap: Zero-shot image-to-text generation for visual-semantic arithmetic," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. [Online]. Available: https://arxiv.org/abs/2111.14447.

[21] H. Agrawal, K. Desai, Y. Wang, *et al.*, "Nocaps: Novel object captioning at scale," in *arXiv preprint arXiv:1812.08658*, 2019. [Online]. Available: https://arxiv.org/abs/1812.08658.

[22] H. Zhou, N. Wang, L. Zhang, *et al.*, "Knowledge enhanced multimodal bart for visual commonsense generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 1801–1811. [Online]. Available: https://aclanthology.org/2020.acl-main.165.

[23] D. Elliott, S. Frank, K. Sima'an, and L. Specia, "Multi30k: Multilingual english-german image descriptions," in *Proceedings of the 5th Workshop on Vision and Language*, 2016. [Online]. Available: https://aclanthology.org/W16-3210.

[24] R. Ramos, B. Martins, and D. Elliott, "Lmcap: Few-shot multilingual image captioning by retrieval augmented language model prompting," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 1635–1651. DOI: 10.18653/v1/2023.findings-acl.104. [Online]. Available: https://aclanthology.org/2023.findings-acl.104.

[25] S. Subramanian, L. L. Wang, B. Bogin, *et al.*, "MedICaT: A Dataset of Medical Images, Captions, and Textual References," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 2112–2120. DOI: 10.18653/

v1/2020.findings-emnlp.191. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.191/.

[26] B. Jing, P. Xie, and E. Xing, "On the automatic generation of medical imaging reports," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2577–2586. DOI: 10.18653/v1/P18-1240. [Online]. Available: https://aclanthology.org/P18-1240.

[27] Z. Jia and X. Li, "Icap: Interactive image captioning with predictive text," in *Proceedings of the 2020 International Conference on Multimedia Retrieval (ICMR)*, 2020, pp. 249–257. DOI: 10.1145/3372278.3390697. [Online]. Available: https://arxiv.org/abs/2001.11782.

[28] J. McCarthy, "What is artificial intelligence?" In *Formalizing Common Sense: Papers by John McCarthy*, Ablex Publishing, 2007. [Online]. Available: http://jmc.stanford.edu/articles/whatisai/whatisai.pdf.

[29] M. Mitchell, *Artificial Intelligence: A Guide for Thinking Humans*. New York: Farrar, Straus and Giroux, 2019, ISBN: 9780374257835. [Online]. Available: https://melaniemitchell.me/aibook/.

[30] J. Kaplan, *Artificial Intelligence: What Everyone Needs to Know*. Oxford University Press, 2016, ISBN: 9780190602390. [Online]. Available: https://books.google.com/books/about/Artificial_Intelligence.html?id=7y_KDAAAQBAJ.

[31] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997, ISBN: 9780070428072. [Online]. Available: https://www.cs.cmu.edu/~tom/mlbook.html.

[32] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020. DOI: 10.1007/s10994-019-05855-6. [Online]. Available: https://doi.org/10.1007/s10994-019-05855-6.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd. MIT Press, 2018, ISBN: 9780262039246. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html.

[34] H. El-Amir and M. Hamdy, *Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow*. Berkeley, CA: Apress, 2020, ISBN: 978-1-4842-5349-6. DOI: 10.1007/978-1-4842-5349-6. [Online]. Available: https://doi.org/10.1007/978-1-4842-5349-6.

[35] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990. DOI: 10.1207/s15516709cog1402_1. [Online]. Available: https://doi.org/10.1207/s15516709cog1402_1.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://www.bioinf.jku.at/publications/older/2604.pdf.

[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791. [Online]. Available: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25 (NeurIPS 2012)*, 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *arXiv preprint arXiv:1406.2661*, 2014. [Online]. Available: https://arxiv.org/pdf/1406.2661.

[40] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762.

[41] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. [Online]. Available: https://arxiv.org/abs/1506.01497.

[42] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: https://arxiv.org/abs/1411.4038.

[43] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://arxiv.org/abs/1703.06870.

[44] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: https://arxiv.org/abs/1411.4555.

[45] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: https://arxiv.org/abs/1503.03832.

[46] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 11, pp. 2298–2304, 2016. [Online]. Available: https://arxiv.org/abs/1507.05717.

[47] B. Zhou, A. Khosla, À. Lapedriza, A. Torralba, and A. Oliva, "Places: An image database for deep scene understanding," *arXiv preprint arXiv:1610.02055*, 2016. [Online]. Available: https://arxiv.org/abs/1610.02055.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556.

[49] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9. [Online]. Available: https://arxiv.org/abs/1409.4842.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. [Online]. Available: https://arxiv.org/abs/1512.03385.

[51] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708. [Online]. Available: https://arxiv.org/abs/1608.06993.

[52] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 3320–3328. [Online]. Available: https://arxiv.org/abs/1411.1792.

[53] NVIDIA Corporation, *NVIDIA DLSS (Deep Learning Super Sampling)*, https://developer.nvidia.com/rtx/dlss, Accessed: 2025-05-13, 2023.

[54] K. Cho, B. van Merrienboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734. [Online]. Available: https://arxiv.org/abs/1406.1078.

[55] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," *arXiv preprint arXiv:2102.03334*, 2021. [Online]. Available: https://arxiv.org/abs/2102.03334.

[56] N. Parmar, A. Vaswani, J. Uszkoreit, *et al.*, "Image transformer," *arXiv preprint arXiv:1802.05751*, 2018. [Online]. Available: https://arxiv.org/abs/1802.05751.

[57] Young et al., *Flickr30k dataset*, https://paperswithcode.com/dataset/flickr30k, Accessed: 2025-05-20.

[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255. [Online]. Available: https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf.