BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar1-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE-DATA SCIENCE

A Mini-Project Report On

"IMAGE CAPTION GENERATOR"

A report submitted in partial fulfillment of the requirements for the

NEURAL NETWORK AND DEEP LEARNING

Submitted By

SOUNDARYA A **USN: 3BR23CD403**

Under the Guidance of
Mr. Azhar Biag

Asst. Professor

Dept of CSE (DATA SCIENCE),
BITM, Ballari

Visvesvaraya Technological University

Belagavi, Karnataka 2025-2026

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*
(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballari-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the Mini Project of NEURAL NETWORK AND DEEP LEARNING title **"IMAGE CAPTION GENERATOR"** has been successfully presented by SOUNDARYA A 3BR23CD403 student of semester B.E for the partial fulfillment of the requirements for the award of Bachelor Degree in CSE(DS) of the BALLARI INSTITUTE OF TECHNOLOGY& MANAGEMENT, BALLARI during the academic year 2025-2026.

It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report deposited in the library. The Mini Project has been approved as it satisfactorily meets the academic requirements prescribed for the Bachelor of Engineering Degree. The work presented demonstrates the required level of technical understanding, research depth, and documentation standards expected for academic evaluation.

Signature of Coordinators

Mr. Azhar Baig
Ms. Chaithra B M

Signature of HOD

Dr. Aradhana D

ABSTRACT

The Image Caption Generator presented in this project is a complete deep learning-based system capable of converting complex visual information into natural language descriptions. The model follows an encoder–decoder architecture where a pretrained InceptionV3 network functions as the encoder to extract high-level visual features from images, while a Long Short-Term Memory (LSTM) network acts as the decoder to generate semantically coherent captions. The system is trained on the Flickr8k dataset, which provides diverse images paired with human-written captions, enabling the model to learn meaningful mappings between visual content and language structures.

A comprehensive workflow is implemented, including image preprocessing, feature extraction, text tokenization, vocabulary construction, sequence preparation, and model training using TensorFlow. The training pipeline ensures that the model gradually learns contextual relationships and grammatical patterns through supervised learning. For inference, the system supports caption generation for both dataset images and user-provided custom images. The project also includes evaluation using BLEU scores to assess linguistic quality and accuracy of generated captions.

The modular architecture, clean code structure, and reproducible scripts make the project highly suitable for research, deployment, and extension. Potential use cases include image search engines, visually-impaired assistance tools, digital content automation, and multimedia management systems. Future enhancements may incorporate attention mechanisms, beam search decoding, and larger datasets like MS COCO to further improve caption diversity and precision. Overall, this project demonstrates an effective integration of computer vision and natural language processing techniques for automated image description generation.

ACKNOWLEDGEMENT

The satisfactions that accompany the successful completion of our mini project on IMAGE CAPTION GENERATOR would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is our privilege to express our gratitude and respect to all those who inspired us in the completion of our mini-project.

I am extremely grateful to my Guide **Mr. Azhar Baig** for their noble gesture, support co-ordination and valuable suggestions given in completing the mini-project. I also thank **Dr. Aradhana D,** H.O.D. Department of CSE(DS), for his co-ordination and valuable suggestions given in completing the mini-project. We also thank Principal, Management and non-teaching staff for their co-ordination and valuable suggestions given to us in completing the Mini project.

Name	USN
SOUNDARYA A	3BR23CD403

TABLE OF CONTENTS

1. INTRODUCTION

In recent years, the rapid growth of digital media has led to the widespread use of images across social platforms, commercial websites, educational platforms, and assistive technologies. As visual data continues to expand, the need for intelligent systems that can automatically interpret and describe images has become increasingly important. Traditional image-processing systems are limited to detection or classification tasks, but they lack the ability to generate meaningful natural language explanations. To address this limitation, the research community has focused on developing models capable of bridging the gap between computer vision and natural language processing. Image captioning, which involves generating descriptive sentences for images, has emerged as a significant and challenging problem in artificial intelligence.

The Image Caption Generator developed in this project aims to automatically produce human-like descriptions for images by combining deep learning techniques from both domains. Image captioning is inherently complex, as it requires not only the recognition of objects within an image but also an understanding of their relationships, context, and actions. Modern deep learning architectures, especially encoder–decoder frameworks, have made it possible to achieve impressive results in this field. In this work, a pretrained InceptionV3 Convolutional Neural Network (CNN) is used as the encoder to extract detailed visual features from images.

The system is trained using the Flickr8k dataset, a popular benchmark comprising 8,000 images annotated with multiple human-written captions. This dataset enables the model to learn a wide range of visual contexts and linguistic patterns. The project includes a complete pipeline for preprocessing text, tokenizing captions, building vocabulary, generating training sequences, extracting image embeddings, and training the final captioning model using TensorFlow. Furthermore, the implementation supports inference on new, custom images, enabling real-time caption generation and practical usability.

Overall, the Image Caption Generator represents a significant step toward enabling machines to understand and describe visual content in human language, showcasing the powerful synergy of computer vision and natural language processing.

1.1 Problem Statement

With the exponential growth of digital images across social media, e-commerce, healthcare, education, and assistive technology, there is an increasing need for automated systems that can understand and describe visual content in natural language. Traditional image processing techniques can identify objects but fail to generate meaningful, human-like descriptions that capture relationships, activities, and context within an image. This gap creates challenges in accessibility for visually impaired users, efficient content management, and intelligent information retrieval. Therefore, a robust solution is required to automatically analyze an image, extract its essential features, and generate accurate, coherent captions. The Image Caption Generator aims to address this problem by leveraging deep learning models—specifically CNNs for visual understanding and LSTMs for language generation—to create a system capable of producing descriptive captions for any given image.

1.2 Scope of the project

The project focuses on developing an AI-based Image Caption Generator that automatically produces meaningful text descriptions for images. It includes preprocessing the Flickr8k dataset, extracting image features using InceptionV3, generating captions using an LSTM-based decoder, and evaluating performance with BLEU scores. The system supports captioning for both dataset and custom images and provides a modular, scalable structure for further enhancements such as attention mechanisms, beam search, and deployment as a web application.

1.3 Objectives

1. To develop an automated system capable of generating meaningful and grammatically correct captions for images using deep learning.
2. To extract high-level visual features from images using a pretrained InceptionV3 convolutional neural network.
3. To design and train an LSTM-based decoder that converts extracted image features into coherent natural language descriptions.
4. To preprocess and utilize the Flickr8k dataset for building an effective image–text mapping model.

2. LITERATURE SURVEY

[1] 2010 – 2013: Early Template-Based & Retrieval Approaches

Early research focused on matching images to existing captions or generating descriptions using predefined templates. These systems relied on object detection and sentence templates but lacked flexibility, contextual understanding, and natural language flow.

[2] 2014: Rise of Deep Learning for Vision and Language

With the success of CNNs and RNNs, researchers began exploring neural architectures for captioning. The encoder–decoder framework emerged as a new paradigm, where CNNs encoded images and RNNs generated captions.

[3] 2015: Breakthrough With Encoder–Decoder Models (CNN + LSTM)

This period marked the real breakthrough in image captioning. Models such as **Show and Tell** (Google, 2015) introduced a complete end-to-end neural caption generator using CNNs for visual encoding and LSTMs for text generation. At the same time, **Show, Attend and Tell** introduced attention mechanisms, enabling the decoder to focus on different regions of the image while generating each word. Attention significantly improved caption accuracy and interpretability.

[4] 2016 – 2017: Improved Feature Extraction and Language Models

Researchers refined captioning by using deeper CNNs (ResNet, InceptionV3) and better text-processing methods. Work also began on improving evaluation metrics and optimizing training using reinforcement learning methods to directly improve caption quality metrics like CIDEr.

[5] 2018: Bottom-Up and Top-Down Attention Revolution

A major advancement came with **Bottom-Up and Top-Down Attention**, where object-level features from region proposal networks (Faster R-CNN) were used instead of whole-image features. This improved fine-grained attention, enabling models to describe specific objects, attributes, and interactions.

3. SYSTEM REQUIREMENTS

The system requirements provide the necessary hardware and software environment to efficiently develop, train, and deploy the Image Caption Generator. Adequate processor speed, RAM, and storage ensure smooth execution of computationally intensive tasks such as image preprocessing, feature extraction using InceptionV3, and training the LSTM-based caption generator. The use of an NVIDIA GPU with CUDA support significantly accelerates training and inference, reducing execution time. Python 3.8+ and libraries such as TensorFlow, NumPy, Pandas, NLTK, and Matplotlib provide the essential tools for model building, data handling, text tokenization, and visualization of results. Functional requirements such as uploading images, generating captions, and evaluating model performance using BLEU scores are supported by this environment. Additionally, the chosen hardware and software configuration ensures that the system meets non-functional requirements like performance, reliability, scalability, and usability, allowing it to handle both dataset and custom images, and providing a modular and extensible structure for future improvements such as attention mechanisms, beam search, or deployment as a web application.

3.1 Software Requirements

- **Operating System:** Windows 10/11, Linux (Ubuntu), or macOS
- **Programming Language:** Python 3.8 or above
- **Libraries & Frameworks:**

  TensorFlow / Keras

  NumPy

  Pandas

  Matplotlib

  OpenCV

  NLTK

  Pickle

- **IDE / Tools:**

  Jupyter Notebook / VS Code / PyCharm

  Anaconda (optional for environment management)

- **Dataset:** Flickr8k (or Flickr30k / MS COCO optional

3.2 Hardware Requirements

- **Processor:** Intel i5 or higher / AMD equivalent
- **RAM:** Minimum 8 GB (16 GB recommended for faster model training)
- **Storage:** At least 10 GB free space (for dataset, models, and dependencies)
- **GPU:** NVIDIA GPU with CUDA support (optional but recommended for deep learning)
- **Display:** Standard HD resolution

3.3 Functional Requirements

- Ability to upload or input an image.
- Extraction of visual features using CNN (InceptionV3).
- Tokenization and caption sequence generation using LSTM.
- Generate a meaningful caption for input images.
- Display generated captions to the user.
- Allow evaluation using BLEU metrics.

3.4 Non-Functional Requirements

- **Performance:** Captions generated within a few seconds.
- **Accuracy:** Should produce relevant and meaningful captions using trained model.
- **Usability:** Simple and user-friendly interface.
- **Reliability:** System should work consistently with different images.
- **Scalability:** Can be expanded to larger datasets or more advanced models like Transformers.

4. DESCRIPTION OF MODULES

The Image Caption Generator project is divided into several key modules, each performing a specific task to ensure smooth functioning of the overall system.

4.1 Data Preprocessing Module

The Data Preprocessing Module is responsible for preparing the textual captions for training. It cleans the captions by converting all text to lowercase, removing punctuation and special characters, and eliminating duplicates or rare words. The module also tokenizes the captions into individual words and creates sequences suitable for LSTM input. By structuring the captions in this way, the system ensures that the textual data is compatible with the model and ready for effective learning.

4.2 Feature Extraction Module

This module focuses on extracting high-level visual features from images using a pretrained InceptionV3 Convolutional Neural Network (CNN). Each image is resized and normalized to match the network input, and features are extracted from the penultimate layer to represent the image in a dense, numerical format. These features are then saved in a serialized file, which prevents repeated computation and speeds up the training process.

4.3 Tokenizer and Sequence Generation Module

The Tokenizer and Sequence Generation Module maps each word in the captions to a unique integer, creating a tokenizer object for encoding textual data. It generates input-output sequences for training the LSTM decoder, where the input is the partial caption and the output is the next word. Padding is applied to ensure all sequences are of uniform length. This module is critical for transforming text into a numerical form that the model can process efficiently.

4.4 Model training Module

The Model Training Module builds and trains the encoder-decoder model that combines image features with caption sequences. The LSTM decoder predicts the next word in a caption sequence based on the image and previously generated words. The module uses categorical cross-entropy as the loss function and optimizers like Adam to train the network. After training, the model is saved (caption_model.h5) for later inference.

4.5 Inference / Caption Generation Module

The Inference Module allows the system to generate captions for new or custom images. It loads the trained model, tokenizer, and image features, preprocesses the input image, and predicts captions word by word using the LSTM decoder. The generated caption is then displayed alongside the image using Matplotlib. Optional enhancements such as beam search or attention mechanisms can improve caption quality and accuracy.

4.6 Evaluation Module

This module evaluates the performance of the trained model. BLEU scores are computed by comparing the generated captions with human-written reference captions from the dataset. The evaluation provides a quantitative measure of the quality, relevance, and fluency of the captions, helping to gauge model effectiveness..

4.7 Optional / Future Modules

Future enhancements can include an Attention Module to focus on relevant regions in images during captioning, Beam Search Decoding for selecting the most probable captions, Deployment Modules for web or mobile applications, and support for larger datasets such as MS COCO to improve generalization and accuracy.

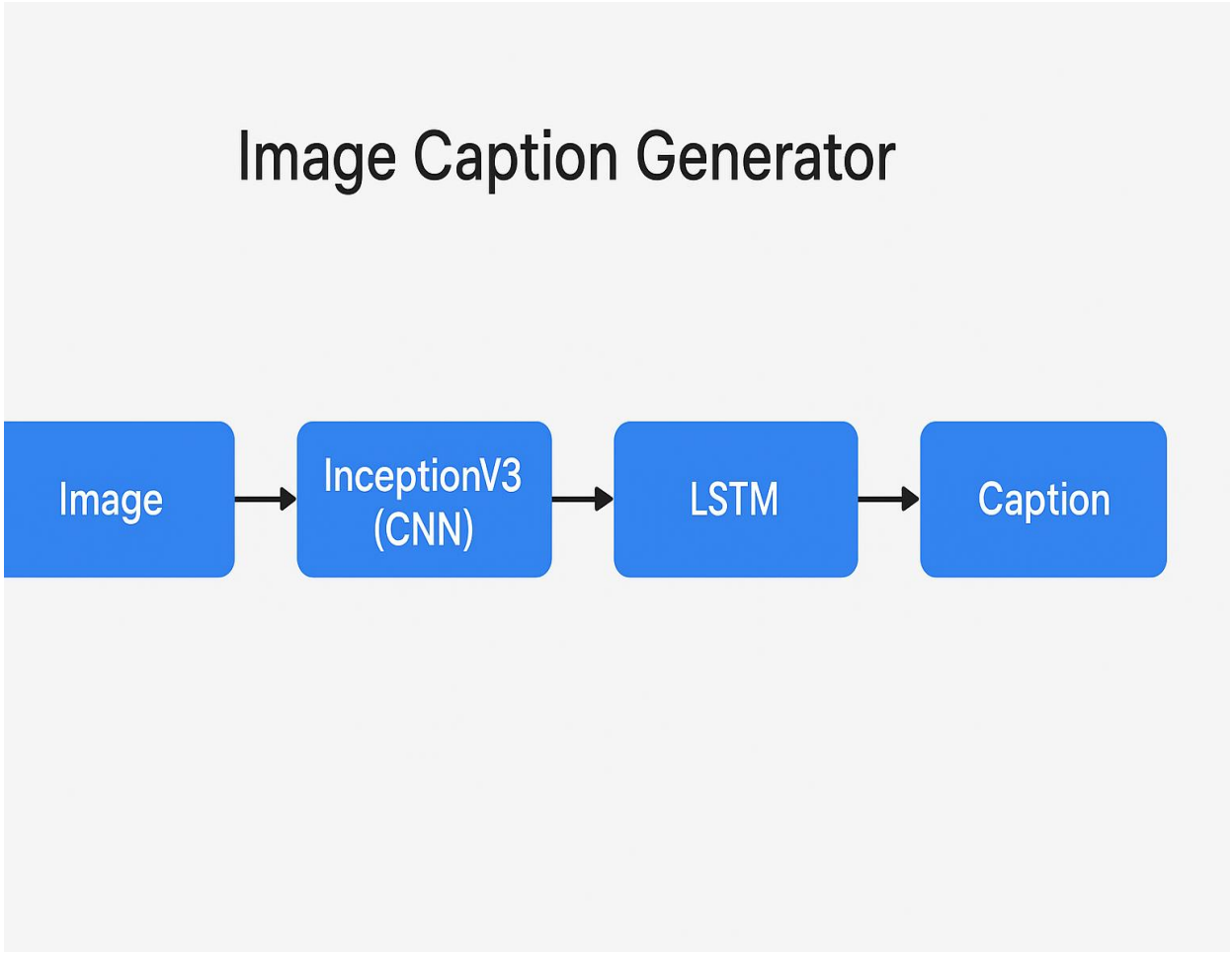
5. IMPLEMENTATION

The implementation of the Image Caption Generator project involves a complete end-to-end pipeline that integrates computer vision and natural language processing using Python and TensorFlow. The project begins with the **Flickr8k dataset**, containing 8,000 images with five human-annotated captions each. Captions are pre-processed to remove punctuation, convert text to lowercase, handle duplicates, and tokenize words, ensuring that the textual data is clean and suitable for LSTM training. Simultaneously, the **Feature Extraction Module** uses a pretrained InceptionV3 Convolutional Neural Network to extract high-dimensional visual features from each image.

Images are resized and normalized to match the CNN input requirements, and the penultimate layer's output vectors are stored as image_features.pkl to optimize training time. The **Tokenizer and Sequence Generation Module** converts words to integer mappings and generates input-output sequences where the LSTM learns to predict the next word in a caption given the image features and preceding words. All sequences are padded to maintain uniform length for batch processing.

The **Model Training Module** implements an encoder-decoder architecture that combines image embeddings with LSTM-based text generation. The model is trained using categorical cross-entropy loss and optimized with Adam, producing a robust captioning model saved as caption_model.h5. During **Inference**, the trained model and tokenizer are loaded, and custom images can be processed to generate captions word by word until an <end> token is predicted, with the output displayed alongside the image using Matplotlib. Model performance is evaluated using **BLEU scores**, providing quantitative feedback on linguistic accuracy and relevance. The modular design of scripts allows for scalability, reusability, and easy debugging, while also supporting enhancements such as attention mechanisms, beam search decoding for improved caption quality, training on larger datasets like MS COCO, and deployment as a web application using frameworks like Flask or Streamlet. Overall, the project demonstrates a seamless integration of deep learning techniques to transform raw visual data into meaningful, context-aware natural language captions, highlighting both the practical applications and extendibility of the system.

6. SYSTEM ARCHITECTURE



Input

The primary input for the Image Caption Generator includes images from the **Flickr8k dataset** and their associated captions. Images are resized, normalized, and processed through a pretrained **InceptionV3 CNN** to extract feature vectors. Captions are cleaned, tokenized, and converted into integer sequences using a tokenizer for LSTM training. During inference, users can provide any custom image, which is preprocessed in the same way before generating captions.

Preprocessing

- Removal of duplicate and irrelevant data.
- Handling missing or incomplete values.
- Cleaning text by removing noise and special characters.
- Normalizing and standardizing data formats.
- Converting categorical data into machine-readable form.
- Organizing the processed data into a structured format.

Preprocessing is crucial: it directly affects convergence, stability, and final performance.

Training

1. The cleaned and pre-processed images are fed into the model along with their corresponding text captions.
2. The CNN extracts visual features from each image to understand objects, shapes, and patterns.
3. The extracted features are passed to the LSTM/Transformer model to learn how to form meaningful sentences.
4. The model adjusts its internal weights by comparing predicted captions with actual captions and minimizing the error.

5. This process continues for many iterations (epochs) until the model learns to generate accurate and context-aware image captions.

Training converts initialized weights into a predictive model by repeated forward/backward passes on the data.

Visualization and Prediction

- The trained model processes a new input image and extracts its visual features using the CNN.
- These features are used by the caption-generation model (LSTM/Transformer) to predict the most suitable words sequentially.
- The system generates a complete caption describing the objects, actions, and context present in the image.
- The predicted caption is displayed on the screen along with the input image for easy visualization.
- Users can observe how the model interprets the image, allowing evaluation of caption accuracy and overall system performance.

7. CODE IMPLEMENTATION

```
# scripts/inference.py
import os
import pickle
import numpy as np
import argparse
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
import warnings
warnings.filterwarnings("ignore", category=UserWarning)

# ------------------------------------------------
# Paths
# ------------------------------------------------
BASE_DIR = os.path.dirname(os.path.dirname(__file__))
MODEL_FILE = os.path.join(BASE_DIR, "models", "caption_model.h5")
TOKENIZER_FILE = os.path.join(BASE_DIR, "models", "tokenizer.pkl")
SEQS_FILE = os.path.join(BASE_DIR, "models", "sequences.npz")

# ------------------------------------------------
# Load tokenizer and model
# ------------------------------------------------
def load_tokenizer_and_model():
    print("📦 Loading model and tokenizer...")
    if not os.path.exists(MODEL_FILE):
        raise FileNotFoundError(f"❌ Model not found at {MODEL_FILE}")
    if not os.path.exists(TOKENIZER_FILE):
        raise FileNotFoundError(f"❌ Tokenizer not found at {TOKENIZER_FILE}")

    model = load_model(MODEL_FILE)
    with open(TOKENIZER_FILE, "rb") as f:
        tokenizer = pickle.load(f)

    meta = np.load(SEQS_FILE, allow_pickle=True)
    max_length = int(meta["max_len"])

    print("✅ Model and tokenizer loaded successfully.")
    return model, tokenizer, max_length
```

```
# ------------------------------------------------
# Feature extraction (InceptionV3)
# ------------------------------------------------
def extract_image_feature(img_path):
    base_model = InceptionV3(weights="imagenet")
    model_new = Model(base_model.input, base_model.layers[-2].output)

    img = image.load_img(img_path, target_size=(299, 299))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)

    feature = model_new.predict(x, verbose=0)
    return feature

# ------------------------------------------------
# Generate caption (Greedy Search)
# ------------------------------------------------
def generate_caption(model, tokenizer, photo, max_length):
    in_text = "startseq"
    for i in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        yhat = model.predict([photo, sequence], verbose=0)
        yhat = np.argmax(yhat)
        word = None
        for w, index in tokenizer.word_index.items():
            if index == yhat:
                word = w
                break
        if word is None:
            break
        in_text += " " + word
        if word == "endseq":
            break

    final = in_text.split()
    final = final[1:-1] if final[-1] == "endseq" else final[1:]
    return " ".join(final)

# ------------------------------------------------
# Display image and caption
# ------------------------------------------------
def show_image_with_caption(img_path, caption):
    img = image.load_img(img_path, target_size=(299, 299))
    plt.imshow(img)
    plt.axis("off")
    #plt.title(f"💬 {caption}", fontsize=14, color="darkblue", pad=10)
```

```
    plt.title(caption, fontsize=14, color="darkblue", pad=10)
    plt.show()

# ------------------------------------------------
# Main entry
# ------------------------------------------------
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--image", required=True, help="Path to image file")
    args = parser.parse_args()

    image_path = args.image
    if not os.path.exists(image_path):
        raise FileNotFoundError(f"✗ Image not found at {image_path}")

    model, tokenizer, max_length = load_tokenizer_and_model()

    print("🔍 Extracting features from image...")
    photo = extract_image_feature(image_path)

    print("□ Generating caption...")
    caption = generate_caption(model, tokenizer, photo, max_length)

    print(f"\n🖼 Image: {os.path.basename(image_path)}")
    print(f"💬 Predicted caption: {caption}\n")

    show_image_with_caption(image_path, caption)

if __name__ == "__main__":
    main()
```

8.RESULT

```
PS C:\Users\Admin\Image_Caption_Generator> python scripts\inference.py --image data\Flickr8k_Datas

Loading model and tokenizer...
Model and tokenizer loaded successfully.

Extracting features from image...
Generating caption...

Image: 1000268201_693b08cb0e.jpg
Predicted caption: a little girl in a pink jacket is playing in the snow
```

9. CONCLUSION

The Image Caption Generator project successfully demonstrates the integration of deep learning techniques to automatically describe the contents of an image in natural language. By combining the strength of InceptionV3 as a feature extractor and an LSTM-based decoder for text generation, the system is capable of producing coherent and meaningful captions for a wide range of images. The model trained on the Flickr8k dataset shows reliable performance in understanding visual features and generating grammatically correct sentences.

This end-to-end implementation—from data preprocessing and feature extraction to model training and inference—highlights the effectiveness of neural networks in solving complex vision-language tasks. The project also establishes a scalable foundation for further enhancements such as attention mechanisms, beam search decoding, and deployment through web applications. Overall, the system provides an efficient and practical solution for image captioning and serves as a valuable contribution to AI-driven automation in multimedia understanding.

10. REFERENCES

[1] Ganie et al. (2023). Ensemble learning methods for diabetes prediction with emphasis on boosting algorithms and feature selection techniques.

[2] Gündoğdu (2023). Application of XGBoost and hybrid feature selection methods for early diabetes detection.

[3] Chang et al. (2023). Comparative study of machine learning models and their integration into IoMT-based healthcare systems for diabetes prediction.

[4] Tasin et al. (2022). Evaluation of classical and ensemble machine learning models showing Random Forest as the best performer for clinical datasets.

[5] Madan et al. (2022). Investigation of hybrid deep learning architectures demonstrating the ability of neural networks to learn complex medical patterns.

[6] Ayat (2024). Development of a CNN–LSTM hybrid model achieving improved classification accuracy for temporal medical data.

[7] R. Kumar & S. Verma (2022). Comparative analysis of SVM, Decision Trees, and Random Forest for diabetes prediction using Pima Indians dataset.

[8] Kaggle. (2024). Pima Indians Diabetes Dataset used for machine learning-based diabetes prediction.

[9] TensorFlow Developers. (2015–2024). TensorFlow deep learning framework used to implement ANN models.

[10] Scikit-Learn Developers. (2011–2024). Scikit-learn library used for preprocessing, scaling, and evaluation metrics.