Image captioning System

For

AI Project(K24MCA18P)
Session (2024-25)

Submitted by

Rani Kumari (202410116100163)
Roopsi Srivastava (202410116100 173)
Rabita Yadav(202410116100156)

Submitted in partial fulfilment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of Ms.

Komal Salgotra Assistant

Professor

Submitted to

Department Of Computer Applications KIET

Group of Institutions, Ghaziabad Uttar

Pradesh-201206

(April- 2025)

📌 **Project Title:**
Automatic Image Captioning System using Deep Learning

Introduction and Project Details:
In the era of artificial intelligence and computer vision, the ability for machines to understand and describe the visual world is becoming increasingly important. **Image captioning** is the process of automatically generating a textual description of an image. It combines two major fields of AI:

- **Computer Vision**: to extract meaningful information from images
- **Natural Language Processing** : to generate coherent and meaningful descriptions in natural language

An image captioning system interprets the contents of an image and generates relevant textual captions, which can be helpful in various applications such as accessibility tools for visually impaired individuals,
automatic image annotation, and intelligent photo management.

🎯 **Objective:**
The goal of this project is to build an end-to-end image captioning system that can:

- Extract high-level visual features from an image using CNNs
- Decode those features into a sequence of words using RNNs (LSTMs)
- Generate accurate and human-like image descriptions

🛠️ **Tools and Technologies Used:**

- **Programming Language**: Python
- **Development Platform**: Google Colab / Jupyter Notebook
- **Libraries**:
  - TensorFlow / Keras
  - NumPy, Pandas
  - Pillow (PIL)
  - Matplotlib
- **Model Architectures**:
  - CNN (InceptionV3 or VGG16 for feature extraction)
  - LSTM (for caption generation)
  - Tokenizer for text preprocessing

💡 **Key Features:**

- Automatic, meaningful, and context-aware image descriptions
- Modular design — easily swappable image models or text decoders
- Trainable on custom datasets for domain-specific captioning (e.g., medical, retail, etc.)

📈 **Future Scope:**

- Use of **transformer models** (like Vision Transformer + GPT-style decoder)
- **Multilingual captioning**
- Integration with **real-time camera feed**

1.Install required libraries

```
!pip install tensorflow pillow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.1.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/p
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.0.1)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.13.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.15.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tenso
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.19
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflo
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorf
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.
```

Step 1: Import Libraries

```
import tensorflow as tf
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, LSTM, Embedding, Dropout, add
import numpy as np
import os
import pickle
from PIL import Image
```

Step 2: Load and Modify InceptionV3 (Feature Extractor)

```
model_incep = InceptionV3(weights='imagenet')
model_incep_new = Model(model_incep.input, model_incep.layers[-2].output)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_orderin
96112376/96112376 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

Step 3: Preprocess Image Function

```
def preprocess_image(img_path):
    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)
    return x
```

Step 4: Extract Features from Image

```
def encode_image(img_path):
    img = preprocess_image(img_path)
    fea_vec = model_incep_new.predict(img)
    fea_vec = np.reshape(fea_vec, fea_vec.shape[1])
    return fea_vec
```

Step 5: Load Captions Data (Dummy Example)

```
def encode_image(img_path):
    img = preprocess_image(img_path)
    fea_vec = model_incep_new.predict(img)
    fea_vec = np.reshape(fea_vec, fea_vec.shape[1])
    return fea_vec
```

Step 6: Tokenizer & Vocabulary

```
captions = {
    "dog.jpg": ["startseq a dog is running in the park endseq"]
}
images = {
    "dog.jpg": encode_image("/content/n.jfif")
}
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
```

Step 7: Define Model

```
from tensorflow.keras.preprocessing.text import Tokenizer

all_captions = []
for key in captions:
    all_captions.extend(captions[key])

tokenizer = Tokenizer()
tokenizer.fit_on_texts(all_captions)
vocab_size = len(tokenizer.word_index) + 1
max_length = max(len(c.split()) for c in all_captions)
```

Step 8: Caption Generation Function

```
embedding_dim = 200

inputs1 = Input(shape=(2048,))
fe1 = Dropout(0.5)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)

inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, embedding_dim, mask_zero=True)(inputs2)
se2 = Dropout(0.5)(se1)
se3 = LSTM(256)(se2)

decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.summary()
```

```
Model: "functional_1"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 9)	0	-
input_layer_1 (InputLayer)	(None, 2048)	0	-
embedding (Embedding)	(None, 9, 200)	2,000	input_layer_2[0]…
dropout (Dropout)	(None, 2048)	0	input_layer_1[0]…
dropout_1 (Dropout)	(None, 9, 200)	0	embedding[0][0]
not_equal (NotEqual)	(None, 9)	0	input_layer_2[0]…
dense (Dense)	(None, 256)	524,544	dropout[0][0]
lstm (LSTM)	(None, 256)	467,968	dropout_1[0][0], not_equal[0][0]
add (Add)	(None, 256)	0	dense[0][0], lstm[0][0]
dense_1 (Dense)	(None, 256)	65,792	add[0][0]
dense_2 (Dense)	(None, 10)	2,570	dense_1[0][0]

```
Total params: 1,062,874 (4.05 MB)
Trainable params: 1,062,874 (4.05 MB)
Non-trainable params: 0 (0.00 B)
```

```
def generate_caption(photo, tokenizer, max_length):
    in_text = 'startseq'
    for _ in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        yhat = model.predict([photo.reshape((1,2048)), sequence], verbose=0)
        yhat = np.argmax(yhat)
        word = tokenizer.index_word.get(yhat)
        if word is None:
            break
        in_text += ' ' + word
        if word == 'endseq':
            break
    return in_text
```

✓ Example Usage

```
photo = images['dog.jpg']
caption = generate_caption(photo, tokenizer, max_length)
print("Caption:", caption)
```

```
Caption: startseq park park park park park park park park park
```