# Image Caption Generator

## Overview

This project implements an Image Caption Generator using deep learning techniques. It consists of three main components:

1. Model Training: A neural network model that learns to generate captions for images.
2. Prediction: A module that uses the trained model to generate captions for new images.
3. Streamlit App: A user-friendly web interface for interacting with the model.

The system uses a combination of Convolutional Neural Networks (CNN) for image feature extraction and Recurrent Neural Networks (RNN) for sequence generation.

## Installation

### Prerequisites

* Python 3.7+
* TensorFlow 2.x
* Keras
* NumPy
* Matplotlib
* NLTK
* Streamlit
* Pillow

### Setup

1. Clone the repository:

* git clone https://github.com/shivam92211/image-caption-generator.git  
  cd image-caption-generator

1. Install the required packages:

* pip install -r requirements.txt

1. Download the Flickr8k dataset:

* import opendatasets as od  
  od.download("https://www.kaggle.com/datasets/adityajn105/flickr8k")

## Usage

### Training the Model

To train the model, run the training script:

python train\_model.py

This script will: 1. Load and preprocess the Flickr8k dataset 2. Extract image features using VGG16 3. Prepare the text data 4. Define and train the caption generation model 5. Save the trained model and necessary artifacts

### Making Predictions

To generate captions for new images, use the predict\_caption function:

from prediction import predict\_caption  
  
image\_path = "path/to/your/image.jpg"  
caption = predict\_caption(model, image\_path, tokenizer, max\_length)  
print(caption)

### Running the Streamlit App

To launch the Streamlit app, run:

1. streamlit run app.py

This will start a local server, and you can access the app through your web browser.

## Code Explanation

### Model Architecture

The model uses a combination of CNN and RNN:

1. VGG16 for image feature extraction (pre-trained on ImageNet)
2. LSTM layers for sequence generation
3. Dense layers for feature combination and output generation

# Encoder model  
inputs1 = Input(shape=(4096,), name="image")  
fe1 = Dropout(0.4)(inputs1)  
fe2 = Dense(256, activation='relu')(fe1)  
  
inputs2 = Input(shape=(max\_length,), name="text")  
se1 = Embedding(vocab\_size, 256, mask\_zero=True)(inputs2)  
se2 = Dropout(0.4)(se1)  
se3 = LSTM(256, use\_cudnn=False)(se2)  
  
# Decoder model  
decoder1 = add([fe2, se3])  
decoder2 = Dense(256, activation='relu')(decoder1)  
outputs = Dense(vocab\_size, activation='softmax')(decoder2)  
  
model = Model(inputs=[inputs1, inputs2], outputs=outputs)

### Data Preprocessing

The code includes several preprocessing steps:

1. Image preprocessing:
   * Resizing images to 224x224 pixels
   * Applying VGG16 preprocessing
   * Extracting features using VGG16
2. Text preprocessing:
   * Tokenization
   * Adding start and end tokens
   * Padding sequences

### Training Process

The training process uses a custom data generator to efficiently handle large datasets:

def data\_generator(data\_keys, mapping, features, tokenizer, max\_length, vocab\_size, batch\_size):  
 # ... (generator code)

The model is trained for a specified number of epochs using this generator.

### Prediction Process

The predict\_caption function generates captions for new images:

def predict\_caption(model, image, tokenizer, max\_length):  
 # ... (prediction code)

This function uses a beam search approach to generate the most likely caption.

## Streamlit App Functionality

The Streamlit app provides a user-friendly interface for the Image Caption Generator:

1. File Upload: Users can upload an image file (JPG format).
2. Image Display: The uploaded image is displayed in the app.
3. Caption Generation: The app uses the trained model to generate a caption for the uploaded image.
4. Result Display: The generated caption is shown below the image.

Key components of the Streamlit app:

st.title("Image Caption Generator")  
  
uploaded\_file = st.file\_uploader("Choose an image...", type="jpg")  
  
if uploaded\_file is not None:  
 # Process the image and generate caption  
 # ...  
 st.image(uploaded\_file, caption="Uploaded Image", use\_column\_width=True)  
 st.write("Generated Caption: ", caption)

## Performance Evaluation

The model’s performance is evaluated using the BLEU score, which measures the similarity between the generated captions and the ground truth captions:

print("BLEU-1: %f" % corpus\_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))  
print("BLEU-2: %f" % corpus\_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))

## Limitations and Future Improvements

* The model’s performance is limited by the size and diversity of the training dataset (Flickr8k).
* The current implementation uses a pre-trained VGG16 model. Experimenting with other architectures (e.g., ResNet, EfficientNet) might yield better results.
* Implementing attention mechanisms could potentially improve the quality of generated captions.
* The Streamlit app currently only supports JPG images. Adding support for other image formats would enhance usability.

## Conclusion

This Image Caption Generator demonstrates the application of deep learning techniques in natural language processing and computer vision. The project provides a complete pipeline from data preprocessing to model deployment, with a user-friendly interface for easy interaction.