# Image Caption Generation Assignment

Mentors-Ankit Maurya ,Sachin Awasthi

## Objective

The objective of this assignment is to design and implement a deep learning model for image captioning. The task involves using a pre-trained **VGG16** model to extract image features and a custom sequence model to generate meaningful captions. This exercise will help you understand the integration of computer vision and natural language processing (NLP).

## Project Steps

Follow these detailed steps to complete the assignment:

### Step 1: Setup Environment and Load Data

- Install the required Python libraries:

```
pip install tensorflow numpy pandas matplotlib nltk
```

- Download the **Flickr8k** dataset and organize it as follows:

```
BASE_DIR/
 Images/          # Folder containing all images
 captions.txt     # File containing image IDs and their captions
```

- Verify that the dataset is loaded correctly by printing a sample of image IDs and captions.

### Step 2: Extract Image Features

- Use the **VGG16** model pre-trained on ImageNet as the feature extractor.
- Preprocess each image:
  - Resize all images to `224x224x3`.
  - Normalize pixel values to the range [0, 1].
- Extract the 4096-dimensional feature vector from the `fc1` layer.

- Save these feature vectors to a `features.pkl` file using `pickle`.
- Example code:

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import pickle

# Load VGG16 model
model = VGG16(include_top=True, weights='imagenet')
model = Model(inputs=model.input, outputs=model.get_layer('fc1').output)

features = {}
for image_name in os.listdir('Images'):
    img_path = os.path.join('Images', image_name)
    image = load_img(img_path, target_size=(224, 224))
    image = img_to_array(image) / 255.0
    image = np.expand_dims(image, axis=0)
    features[image_name] = model.predict(image).flatten()

with open('features.pkl', 'wb') as f:
    pickle.dump(features, f)
```

## Step 3: Preprocess Captions

- Load and clean the captions:
  - Convert text to lowercase.
  - Remove special characters and numbers.
  - Add `<start>` and `<end>` tokens to each caption.
- Tokenize the captions using `Tokenizer` and calculate the vocabulary size and maximum caption length.
- Example code:

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

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 4: Split Data

- Divide the dataset into training (90%) and test (10%) sets based on image IDs.
- Ensure no overlap between training and test images.

## Step 5: Build the Model Architecture

- **Encoder**: Use the pre-extracted VGG16 features as input.
- **Decoder**:
  - Use an embedding layer to embed caption words into dense vectors.
  - Use an LSTM layer to capture the temporal dependencies of the captions.
- Combine the encoder and decoder outputs using an `add` layer and generate the final output using a dense layer with a softmax activation function.
- Example code:

```
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense, add
from tensorflow.keras.models import Model

# Image feature input
img_features = Input(shape=(4096,))
img_dense = Dense(256, activation='relu')(img_features)

# Caption input
cap_input = Input(shape=(max_length,))
cap_embed = Embedding(vocab_size, 256)(cap_input)
cap_lstm = LSTM(256)(cap_embed)

# Combine features and captions
decoder = add([img_dense, cap_lstm])
output = Dense(vocab_size, activation='softmax')(decoder)

model = Model(inputs=[img_features, cap_input], outputs=output)
model.compile(optimizer='adam', loss='categorical_crossentropy')
```

## Step 6: Train the Model

- Use a generator to yield batches of data:
  - Image features (from `features.pkl`).
  - Input captions (padded sequences).
  - Target captions (one-hot encoded words).
- Train the model for 10 epochs with a batch size of 32 and save the trained model as `best_model.h5`.

### Step 7: Evaluate and Generate Captions

- Implement a function to generate captions:
  - Start with the `<start>` token.
  - Predict one word at a time until `<end>` or the maximum length is reached.
- Evaluate the model using the BLEU score.
- Display the image and its generated caption using `Matplotlib`.

## Expected Deliverables

- A Python script or Jupyter Notebook implementing all steps.
- Saved model file (`best_model.h5`).
- A brief report including:
  - Training and validation loss plots.
  - Example captions with BLEU score.

## Additional Tips

- **Debugging**: Verify intermediate outputs, such as preprocessed captions and extracted features.
- **Optimization**: If memory issues arise, reduce the batch size or the dataset size during training.
- **Documentation**: Comment your code for clarity.