Funny Image Caption Generator

Introduction

Image captioning is a challenging task in artificial intelligence that involves generating textual descriptions for images automatically. This project combines deep learning techniques to achieve this goal, along with an integration of emoji prediction based on generated captions.

Brief Overview of VGG16 Model

Introduction

VGG16 (Visual Geometry Group 16) is a deep convolutional neural network architecture developed by the Visual Geometry Group at the University of Oxford. It is renowned for its simplicity and effectiveness in image classification tasks.

Architecture

- **Layers:** VGG16 consists of 13 convolutional layers followed by 3 fully connected layers.
- **Filter Sizes:** All convolutional layers use 3x3 filters, and max-pooling layers use 2x2 filters.
- **Activation:** ReLU (Rectified Linear Unit) activation functions are used throughout the network.

Features

- **Applications:** VGG16 excels in image classification tasks, achieving state-of-the-art performance on datasets like ImageNet.
- **Transfer Learning:** Its pre-trained weights are often used as a feature extractor in transfer learning, enhancing performance on domain-specific tasks.

Code Overview

The project is implemented in Python using several libraries and frameworks, including TensorFlow/Keras for deep learning, Streamlit for creating a user interface, and PIL (Pillow) for image handling.

Code Components

1. Model Loading and Initialization:

- Model Loading: The project loads a pre-trained image captioning model (`best_model.h5`) and a tokenizer (`tokenizer.pkl`) for text preprocessing.
- Feature Extraction: Image features extracted using a VGG16 model are pre-computed and stored in `features.pkl`.

2. Caption Generation Function:
- The script defines functions to predict captions for uploaded images. It uses the loaded model to generate a caption for each image.
- A function is implemented to clean and preprocess text data before feeding it into the captioning model.

3. Streamlit Interface:
- Utilizes Streamlit, a Python framework for building interactive web applications, to create a user-friendly interface.
- Users can upload images through the interface, which are then displayed along with the generated captions.
- A loading bar is implemented to indicate the progress of caption generation for better user experience.

4. Emoji Prediction Integration:
- The app integrates an emoji predictor model (`BLSTM.h5`) alongside the image captioning functionality.
- After generating a caption for an uploaded image, the app predicts and displays the most probable emoji associated with the caption using a trained deep learning model.

Usage

To use the application:
- Run the Streamlit application locally.
- Upload an image using the file uploader provided.
- Wait for the caption to be generated (indicated by a loading bar).
- View the generated caption and associated emoji prediction.

Challenges Faced

1. Mapping Captions with Flickr Dataset

Mapping captions with the Flickr dataset was a time-consuming task. The dataset's vast size and the need for accurate mapping resulted in the process taking more than a day to complete.

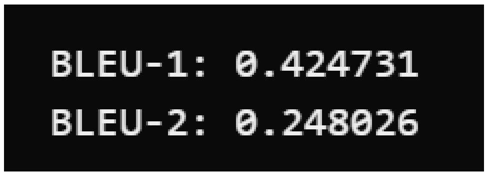
2. Generating Multiple Captions

Initially, generating more than one caption per image posed a challenge. This was essential for improving diversity and accuracy in caption generation. The solution involved modifying the code to iterate and generate multiple captions per image.

3. CPU Limitations

The project encountered limitations due to CPU constraints. This affected the training and prediction phases, impacting both performance and model scores. Optimization techniques and batch processing were employed to mitigate these limitations.

Metrics:



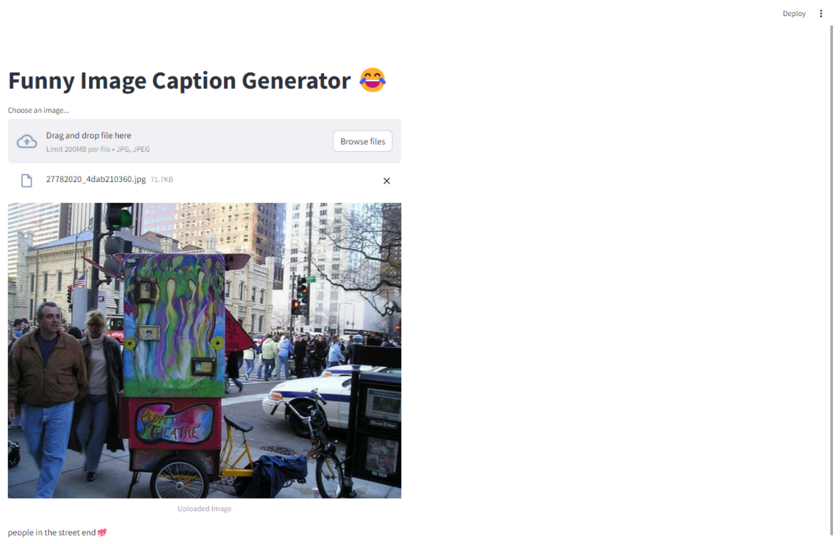
```
BLEU-1: 0.424731
BLEU-2: 0.248026
```

A BLEU-1 score of 0.424 indicates that 42.4% of the individual words in the machine-generated translations match the reference translations, suggesting a moderate level of word accuracy. Meanwhile, a BLEU-2 score of 0.24 shows that 24% of the bigrams (pairs of consecutive words) match the reference translations, highlighting the increased difficulty of capturing correct word sequences and context. Achieving these scores is challenging due to the need for understanding context and semantics, handling variability in language (such as synonyms and idiomatic expressions), ensuring high-quality and diverse training datasets, and optimizing the model architecture to generalize well to unseen data. These aspects make improving translation accuracy a complex task.

Additional info:

To open the streamlit app you will need to install streamlit via pip and use the command streamlit run app.py

Output:



Funny Image Caption Generator 😂

people in the street end 💯

Conclusion

The image captioning project demonstrates the integration of deep learning models with web-based user interfaces for real-time application. By addressing challenges such as dataset mapping, caption diversity, and resource constraints, the project showcases solutions that enhance the functionality and usability of AI-driven applications.

This document provides a comprehensive overview of the image captioning project, highlighting its technical implementation, challenges faced, and the solutions applied.

Done By
Adeib Arief