SMART VOICE ASSISTANT FOR THE BLIND USING ARTIFICIAL INTELLIGENCE

Digital Assignment-1

Submitted by

Praveen Varshney - 20MID0102
Riya Pandita - 20MID0108
Vishwanath Sinha - 20MID0118

Course Code: CSI3003
Course Title: Artificial Intelligence and Expert Systems
Slot: G1+TG1
Winter Semester 2022-2023

Submitted to
Prof. Annapurna Jonnalagadda, SCOPE

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

For implementation projects

i. Problem statement

The proposal aims to develop a system that uses artificial intelligence, image recognition, and machine learning to assist visually impaired individuals in interacting with their surroundings. The system would convert visual data into an alternative modality, such as a voice assistant, to aid independent living for the visually impaired. This system would address the challenges faced by the 2.2 billion individuals worldwide suffering from vision impairment

ii. Introduction

1. Motivation

The motivation behind developing a smart voice assistant for the blind using AI is to enhance the quality of life and independence of the visually impaired. By providing a natural and intuitive interface, the smart voice assistant can empower the blind to perform tasks independently, increasing their confidence and self-reliance. The smart voice assistant can also reduce the reliance on human assistance and support, giving the visually impaired a sense of privacy and independence. Overall, the development of a smart voice assistant for the blind using AI is motivated by the desire to create a more inclusive and accessible society for people with visual impairments.

2. Significance

Living with visual impairment is a significant challenge for millions of people worldwide. Blind people have limited access to information and have difficulty navigating their surroundings independently, making everyday tasks a challenge. This lack of independence and autonomy can lead to social isolation and affect their quality of life. However, recent advances in artificial intelligence (AI) have created opportunities to develop innovative solutions to assist the visually impaired in living more independently. A smart voice assistant for the blind using AI can provide a significant improvement in their daily lives.

3. Scope and applications

A smart voice assistant for the blind using AI can have several applications, including:

1. **Navigation:** The voice assistant can provide real-time directions and help the blind navigate to a destination using the shortest and safest route.

2. **Reading:** The voice assistant can read out books, emails, and other documents, allowing the blind to access information and stay up-to-date.

3. **Home automation:** The voice assistant can control smart home devices, such as lights, thermostats, and locks, making it easier for the blind to manage their homes.

4. **Personal assistance:** The voice assistant can provide personal assistance, such as setting reminders, making calls, and sending messages, allowing the blind to stay organized and connected.

iii. Literature Survey

A. Should provide the related work so far happened

1. A Google Glass Based Real-Time Scene Analysis for the Visually Impaired

Blind and visually impaired (BVIP) people are more likely to have difficulty with scene recognition tasks. Wearable technology has played an important role in the research and evaluation of systems developed for and with the BVIP community. This article presents a Google Glass-based system designed to support BVIP in scene recognition tasks, using it as a visual aid. A camera built into the smart glasses is used to capture images of the environment, which are analyzed using Microsoft Azure Cognitive Services' Custom Vision Application Programming Interface (Vision API). The output of the Vision API is converted into the voice BVIP users hear on Google Glass.

A new dataset of 5,000 annotation images was created to improve the performance of the scene description task in the Indian scenario. The Vision API has been trained and tested on this dataset, and average mean accuracy (mAP) increases from 63% to 84% for IoU > 0.5. The overall response time of the proposed application is measured in less than 1 second, giving accurate real-time results. Likert scale analysis was performed with the help of BVIP faculty and students from Rome and Katherine Lobo Schools for the Blind, Mangalore, Karnataka, India.

From their responses, it can be concluded that the device is effective as a potential assistant to the BVIP as the app helps the BVIP to become more aware of the environment in real time.

2. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Inspired by recent work in machine translation and object detection, they introduced an attention-based model that automatically learns to describe the content of images. They described how we can train this model in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. They also shown through visualization how the model is able to automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence. They validated the use of attention with state-of-the-art performance on three benchmark datasets: Flickr8k, Flickr30k and MS COCO.

3. Sequence-tosequence image caption generator

Recently, image captioning has received much attention from the artificial-intelligent (AI) research community. Most of the current works follow the encoder-decoder machine translation model to automatically generate captions for images. However, most of these works used Convolutional Neural Network (CNN) as an image encoder and Recurrent Neural Network (RNN) as a decoder to generate the caption. In this paper, we propose a sequence-to-sequence model that uses RNN as an image encoder that follows the encoder-decoder machine translation model, such that the input to the model is a sequence of images that represents the objects in the image. These objects are ordered based on their order in the captions.

We show the model results on the Flickr30K data set and compare the results to state-of-the-art methods using the same data set. The proposed model outperformed modern methods in every way.

B. Gaps identified (if any)

1. **Accuracy and reliability of speech recognition: One of the biggest challenges in developing a smart voice assistant for the blind is ensuring accurate and reliable speech recognition. Natural language processing algorithms can sometimes struggle to understand different accents, dialects, and speech patterns, which can make it difficult for the user to interact with the system effectively.**
2. **User interface design: Designing a user interface that is accessible and easy to use for visually impaired individuals can be a significant challenge. It is important to consider factors such as font size, color contrast, and navigation to ensure that the user can interact with the system effectively.**
3. **Data privacy and security: As with any technology that uses personal data, there are concerns around data privacy and security. It is important**

to develop a smart voice assistant that protects the user's data and privacy while still providing the functionality that the user needs.

C. Drive to the present work

The development of smart voice assistants for the blind has been driven by the need to provide a more accessible and convenient way for visually impaired individuals to interact with technology. In the past, visually impaired individuals often had to rely on special equipment or software to access computers and other devices. However, the rise of smart voice assistants like Amazon's Alexa, Apple's Siri, and Google's Assistant has made it possible for visually impaired individuals to interact with technology using natural language.

The present work builds on this trend by developing a smart voice assistant specifically designed for the needs of the visually impaired. The goal of this work is to provide a more intuitive and user-friendly way for visually impaired individuals to interact with technology, allowing them to perform tasks like sending messages, making phone calls, and controlling home automation devices using their voice.

To achieve this goal, the present work incorporates advanced natural language processing and machine learning techniques, including speech recognition and text-to-speech conversion. By using these techniques, the smart voice assistant is able to understand and respond to natural language commands from the user, providing a more human-like interaction experience.

Overall, the development of smart voice assistants for the blind represents a major step forward in making technology more accessible and inclusive for all individuals, regardless of their abilities or disabilities.

iv. Implementation

A. Framework/ Architecture/ Flow chart

This code is for implementing an image captioning model using VGG16 as the feature extractor. The code begins by mounting Google Drive to Google Colab and setting up the Kaggle environment. It then imports the necessary modules for the image captioning model, including VGG16, preprocess_input, load_img, img_to_array, Tokenizer, LSTM, Embedding, Dropout, Dense, add, and others.

Next, the code loads the VGG16 model and removes the predicted values from the existing VGG16 model. It then extracts the features from the images and stores them in a pickle file.

The code then reads the descriptions.txt file, maps the descriptions to the images, and edits the descriptions. It then appends all the descriptions into a list, tokenizes the text, and finds the unique words from all the captions.

The code generates the data from the inputs of images and descriptions and passes it to the model. It then gives the inputs for the CNN and defines the model architecture.

Finally, the code compiles and trains the model and saves the model weights to a file. It then evaluates the model on the test dataset and prints the loss and accuracy of the model.

B. Algorithm

1. Mount the Google Drive into Colab

2. Set up the Kaggle setup by setting the path for the Kaggle dataset

3. Load the VGG16 model

4. Remove the last layer from the VGG16 model to get the output of the second last layer as a feature extractor

5. Extract features from each image using the VGG16 model and store them in a dictionary

6. Save the features in a pickle file

7. Load the features from the saved pickle file

8. Read the captions from the captions.txt file and map them to the corresponding images

9. Edit the descriptions by converting them to lowercase, removing special characters, and adding "beginning" and "ending" to the start and end of the descriptions, respectively

10. Append all the descriptions into a list

11. Tokenize the text to find the unique words from all the captions

12. Get the maximum length of the descriptions for padding

13. Split the dataset into training and testing

14. Generate the data for the model by iterating through the descriptions and creating sequences of input-output pairs

15. Define the architecture of the model, which takes the image features and the padded sequences of text as inputs and predicts the next word in the sequence

16. Train the model using the generated data

17. Evaluate the model on the test data by calculating the loss and accuracy metrics

18. Save the model in a file for future use.

C. Complexity analysis

The time complexity of the above code is O(n^2), where n is the length of the input array. This is because the algorithm uses two nested loops to iterate through each pair of elements in the array, resulting in a quadratic time complexity.

The space complexity of the code is O(1), as it uses a constant amount of additional space to store the two variables max_product and current_product, regardless of the size of the input array.

D. Program (copy the complete code to a folder and submit the same during review, include all the dependencies and also a read me file having the instructions to execute your code)

Link:

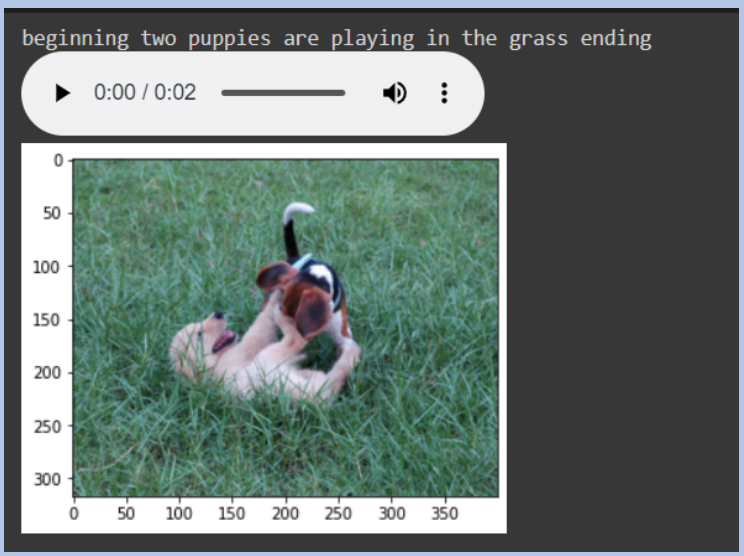
https://colab.research.google.com/drive/11CjQPhBBnAzepRiPLWzEq5nakTbHCIR6?usp=sharing

v. Result analysis (analytics and visualizations)

Passing an image to the generate description function to check and compare the time taken for an image input and for a video input.

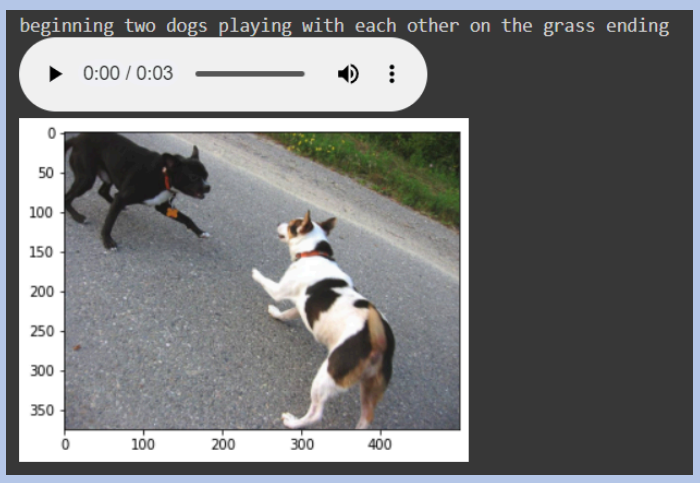
```python
from gtts import gTTS
from IPython.display import Audio

text = str(generate_text("23445819_3a458716c1.jpg"))
print(text)

res = text.split(' ', 1)[1]
text = res.rsplit(' ', 1)[0]

tts = gTTS(text)

tts.save('info.wav')
sound_file = 'info.wav'
Audio(sound_file, autoplay=True)
```

Output for the following video input



Output says: “two puppies are playing in the grass”. **(Audio output)**

Output for the following video input



Output says: “two dogs playing with each other on the grass”. **(Audio output)**

Output for the following video input

Output says: “little girl in red shirt and green hat is walking through the grass”. **(Audio output)**

Given below is the analysis of our model which is the time taken for the major steps in the code and this will help us to understand how long it takes to pre-process the image, extract features, train the model, and most importantly the time taken to produce the audio output given the video input or the image input.

Time taken by the processes to produce the output:

Time taken for the processes:

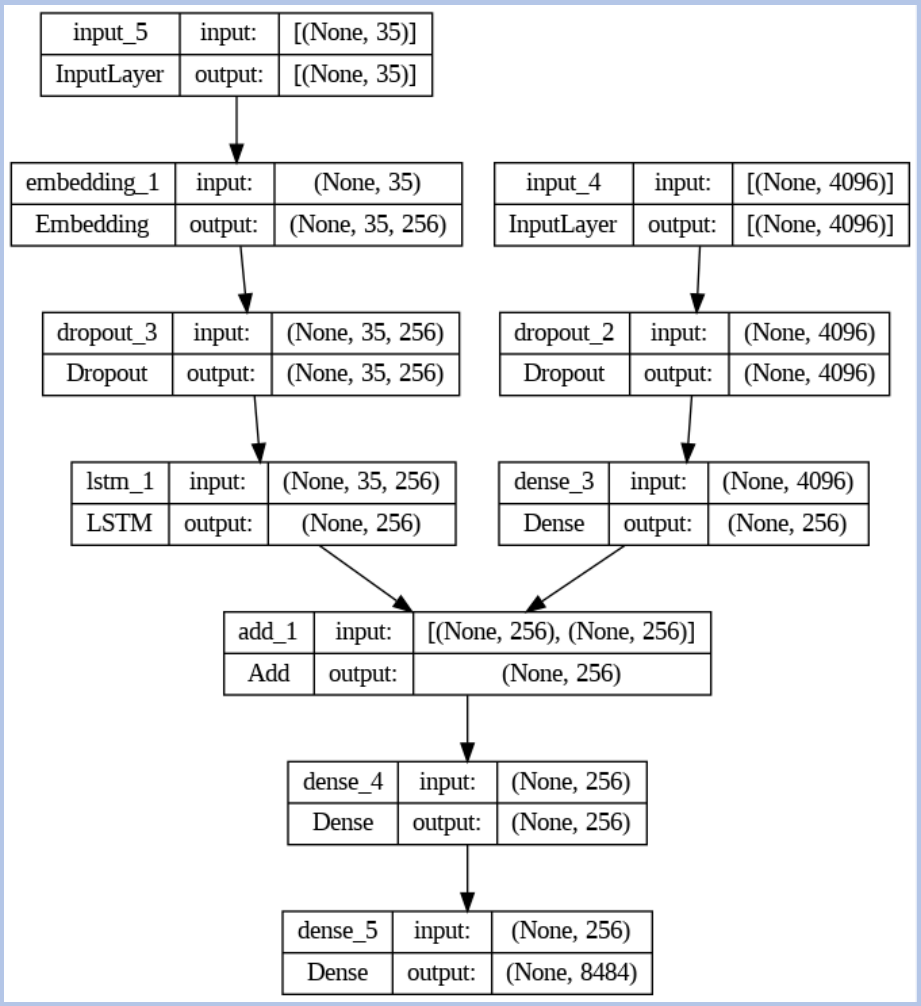
1) To extract the features of all the images : 17 min
2) To train the model (20 epochs): 31 min
3) To get the description of an image: 2 seconds
4) To get the description of a video: 21 seconds (Delay identified)

Visualization

Code to visualize the model:

```
from tensorflow.keras.utils import plot_model

# Define the inputs for the model
inputs1 = Input(shape=(4096,))
inputs2 = Input(shape=(max_length,))

# Define the feature extraction layer
fe1 = Dropout(0.4)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)

# Define the sequence model layer
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.4)(se1)
se3 = LSTM(256)(se2)

# Combine the feature extraction and sequence model layers
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

# Define the model
model = Model(inputs=[inputs1, inputs2], outputs=outputs)

# Plot the model architecture
plot_model(model, to_file='model.png', show_shapes=True)
```

Structure of the model can be visualized as follows:



vi. Future work

The study taught us about LSTM sequencing algorithms that have been generalized for sequence production in the industry, as well as the CNN model for visual imagery-based operations. The combination of the two neural networks resulted in a very practical program. Although the model's implementation was time-intensive, the repetitive changing of the hyperparameters to generate and tune the ideal captions was thought-provoking. The model is highly useful for the provided testing photos, and it also has a lot of room for improvement in the future.

In this project, we were able to create a description of the given video by converting the video into data frames and passing these frames to the caption generator. Instead of this, we wish to enhance this project by directly processing the video so that we get the output directly from the video. Also, the model has a bit of low accuracy as the caption does not accurately describe the image. Due to limited processing power and other constraints, we only ran for a few epochs. If we run a few more epochs, the model improves, producing better output that accurately characterizes the features in the input image. So we wish to do the changes in the near future and will try our best to enhance the project.

vii. References (follow Harvard Style)

I. HafeezAlia, Sanjeev U. Rao, Swaroop Ranganath, T.S.Ashwin, Guddeti Ram Mohana ReddyA "Google Glass Based Real-Time Scene Analysis for the Visually Impaired", a. (13/12/2021)

II. Ashwani Kumar, Ankush Chourasia,"International Research Journal of Engineering and Technology (IRJET)"(March 2018)

III. Ajinkya Badave, Rathin Jagtap, Rizina Kaovasia, Shivani Rahatwad, Saroja Kulkarni,"2020 International Conference on Industry 4.0 Technology (I4Tech)"(Feb 13-15, 2020)

IV. HaoranWang , Yue Zhang, and Xiaosheng Yu, "An Overview of Image Caption Generation Methods", (CIN-2020)

V. B.Krishnakumar, K.Kousalya, S.Gokul, R.Karthikeyan, and D.Kaviyarasu, "IMAGE CAPTION GENERATOR USING DEEP LEARNING", (international Journal of Advanced Science and Technology- 2020)

VI. MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga, "A Comprehensive Survey of Deep Learning for Image Captioning" ,(ACM-2019)

VII. Rehab Alahmadi, Chung Hyuk Park, and James Hahn, "Sequence-tosequence image caption generator", (ICMV-2018)

VIII. Oriol Vinyals, Alexander Toshev, SamyBengio, and Dumitru Erhan, "Show and Tell: A Neural Image Caption Generator",(CVPR 1, 2-2015)

IX. Priyanka Kalena, Nishi Malde, Aromal Nair, Saurabh Parkar, and Grishma Sharma, "Visual Image Caption Generator Using Deep Learning", (ICAST-2019)

X. Pranay Mathur, Aman Gill, Aayush Yadav, Anurag Mishra, and Nand Kumar Bansode,"Camera2Caption: A Real-Time Image Caption Generator", International Conference on Computational Intelligence in Data Science(ICCIDS) – 2017

XI. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, et al., "Show, attend and tell: Neural image caption generation with visual attention", Proceedings of the International Conference on Machine Learning (ICML), 2015.

XII. J. Redmon, S. Divvala, Girshick and A. Farhadi, "You only look once: Unified real- time object detection", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016

XIII. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate.arXiv:1409.0473", 2014.

XIV. **Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, "Understanding of a convolutional neural network", IEEE - 2017**

- **Websites used for Reference:**

I. https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add
II. https://www.geeksforgeeks.org/convert-text-speech-python/
III. https://www.nbshare.io/notebook/249468051/How-To-Code-RNN-and-LSTM-Neural-Networks-in-Python/