

Image Captioning System to Assist the Blind – Project Report

Ammar Ahmed (023-19-0107), Filza Akhlaq (023-19-0080), Adnan Ali
(143-19-0019)

Instructor: Sir Sher M. Daudpota

Introduction:

The world is undeniably getting digital at a fast pace. New technologies from social media and GPS systems to artificial intelligence – make the planet unrecognizable from even a decade ago. We have gotten accustomed to using various devices to accomplish daily tasks such as working, reading, playing games, communicating, and interacting with one another. While the world is changing for the better, it can be challenging for the visually impaired to participate in such a world, just like it was a decade ago. The visually impaired face challenges when it comes to visualizing their milieu. Anyone who owns a digital device having a camera captures and can see what they captured, except the visually impaired ones. To assist the visually impaired, the area of assistive technologies has seen rapid growth in the past few years. This project aims to develop an assistive app with the help of state-of-the-art deep learning techniques such as image captioning, image recognition, and text generation.

Project Idea:

The aim of this project is to develop a system that helps blind people cheaply get answers in their everyday lives. The task here will be to describe an image taken by people who are blind. The system can predict a suitable caption for the given image and speak it out. Once deployed, it would enable blind people to see what they can not see by visualizing the captions that they hear, and develop a better understanding of their surroundings.

Dataset Discussion:

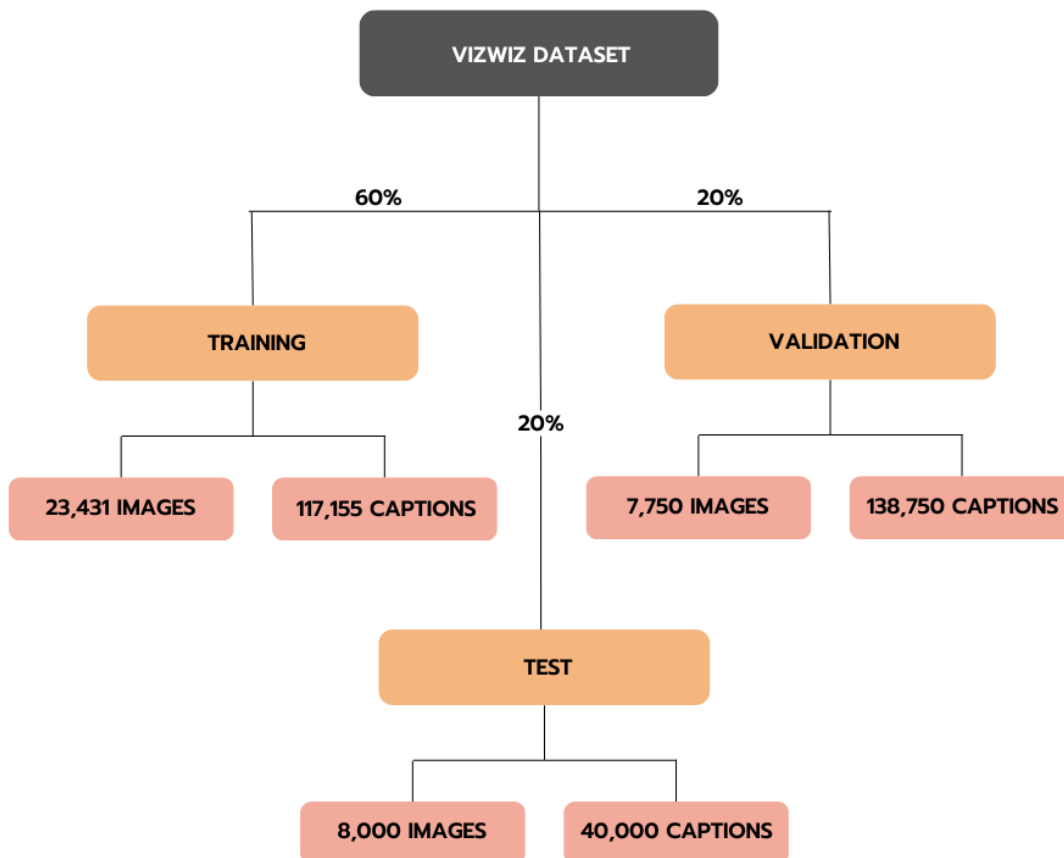
This project uses a publicly available dataset called VizWiz. The dataset was constructed from a natural visual question-answering setting where blind people each took a photo and recorded a spoken question about it, together with 10 crowdsourced answers per photo [2]. The dataset includes 23,431 training images, 117,155 training captions, 7,750 validation images, 38,750 validation captions, 8,000 test images, and 40,000 test captions. The annotations are written inside of a JSON file. There also is a “text_detected” flag which is set to true for the image if it is set to true for at least three of the five crowdsourced results and false otherwise. Details about the image are in the following format.

```

images = [image]
image = {
    "file_name": "VizWiz_train_00023410.jpg",
    "id": 23410
    "text_detected": true
}
annotations = [annotation]
annotation = {
    "image_id": 23410,
    "id": 117050,
    "caption": "A plastic rewards card lying face down on the
floor."
    "is_rejected": false,
    "is_precanned": false,
    "text_detected": true
}

```

Following is the dataset format.



Example of an image along with some of its captions.

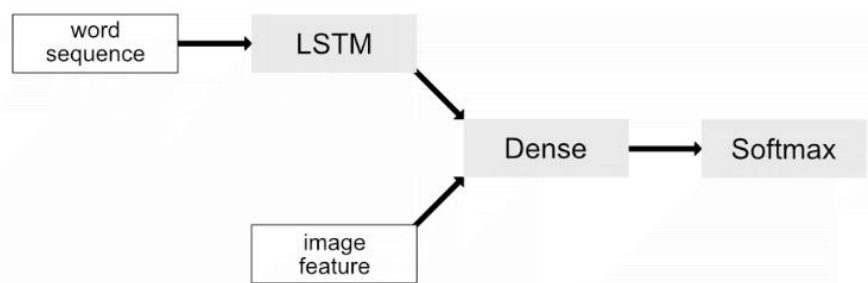


Image captions:

1. A large plastic tub of Kirkland Mixed Nuts.
2. A mostly empty jar of mixed nuts sitting on a desk.
3. A mostly empty, clear plastic jar of Kirkland mixed nuts, with a blue plastic lid, standing on a closed laptop on an office desk with some books and stuffed penguins in the background.
4. a package of peanuts and other assorted nuts on a desk
5. imagine how you would describe this image on the phone to a friend.

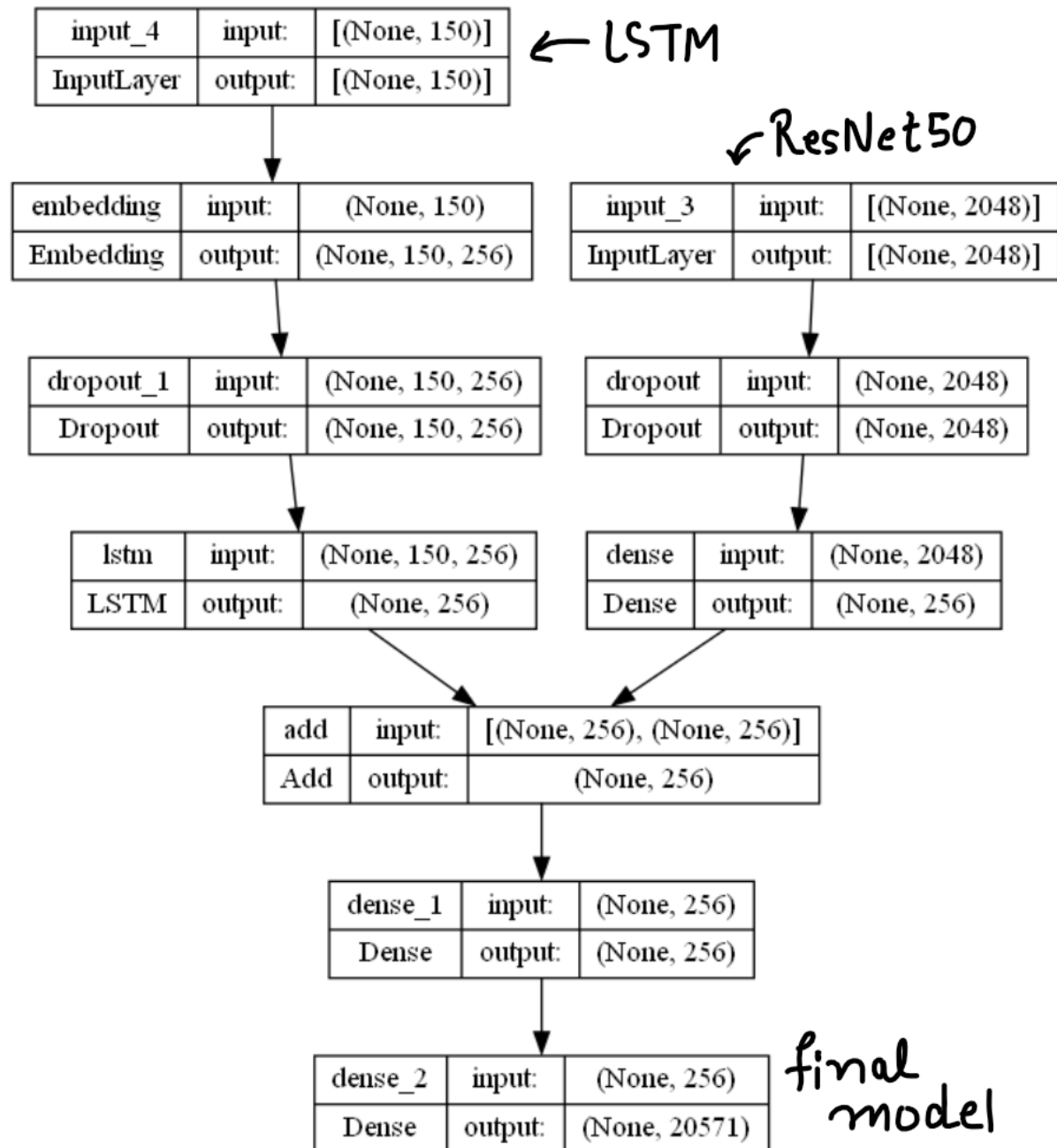
Project Methodology:

The system was built using deep learning techniques. The project specifically incorporates computer vision and natural language processing techniques. Since this is an image captioning task, the system was developed using a CNN model along with an NLP model. These models were then concatenated to create a single image captioning system that takes image features as input and produces a text sequence that can describe the image. To achieve this, an **add layer** was used provided by Keras API. Which allows for adding a list of inputs. It takes as input a list of tensors, all the same shape, and returns a single tensor also of the same shape. The **add layer** allowed us to give our model image features as well as associated captions as input and gives text sequence as output. The following figure shows the basic architecture of 'add' mechanism.



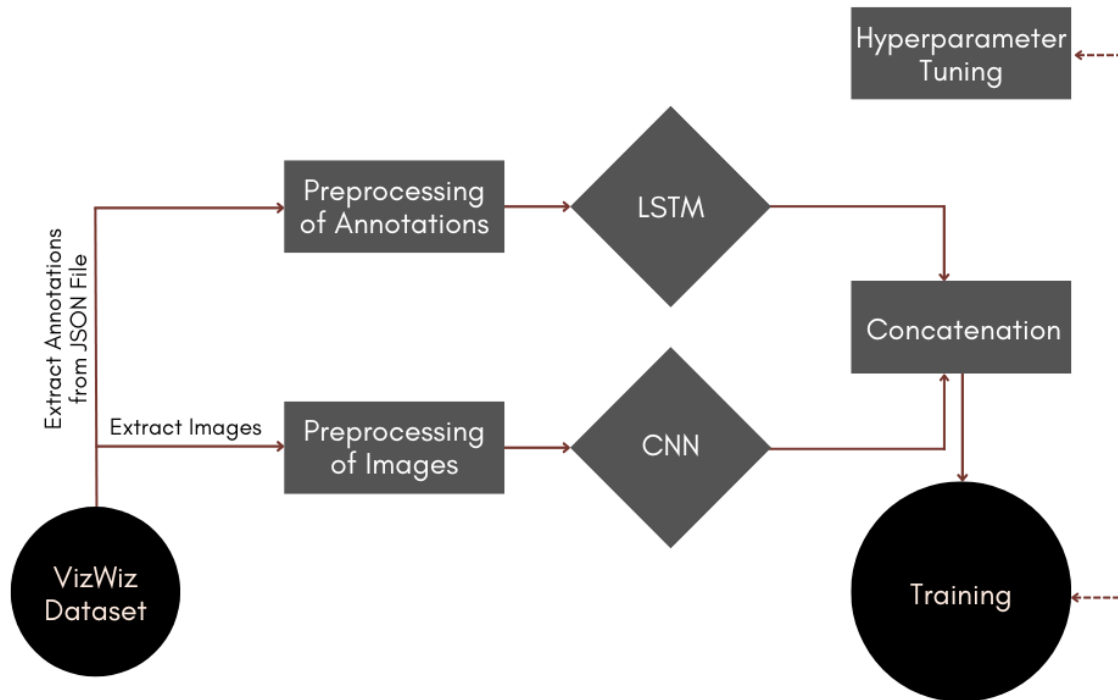
The following figure shows how a convolutional neural network was combined with an NLP network to create an image captioning network. In this specific figure, a ResNet50

model is concatenated with LSTM. This was one of the best-performing models in our task of image captioning.



First, all the images were extracted from the dataset along with its name, then that name was used to extract the image id provided within the JSON file included in the dataset. Finally, that image id was then used to extract all the annotations associated with that

image from the JSON file. Given the images and their associated captions, the next step was to perform preprocessing steps on both the captions as well as on the images. The following figure illustrates the workflow of the entire project.



For image feature extraction, we used state-of-the-art pre-trained models such as ResNet50, VGG16, and VGG19. For text, we used LSTM and Bidirectional LSTM in combination with the image feature extraction models discussed. Various experiments were carried out with the combination of these text + image feature extraction models while keeping track of the validation loss during training and evaluating these models using the metric called BLEU score after the training. After the model selection based on the highest BLEU score, the model was deployed using Python's web framework, Flask. The web interface was designed using the web technologies such as CSS3, HTML5, and JavaScript. Finally, a Google API called gTTS (Google Text-to-Speech) was tested to speak the caption out loud, but since we found it to be too slow, we decided to use the alternative Text-to-Speech library called pyttsx3 (Python Text-to-Speech 3) which was then integrated into the app to allow a user to listen to a predicted caption by the model.

Results and Discussion:

Various image feature extraction models along with text models were tested and assessed based on the BLEU score. The two best-performing models out of many models were chosen based on their BLEU score and were selected and deployed in the end. The following table shows the BLEU score for all the models we experimented with. The BLEU-1 and BLEU-2 indicate a score achieved for 1-gram and 2-gram tokens.

Model	BLEU-1	BLEU-2
VGG16 + LSTM	0.49	0.33
VGG19 + LSTM	0.35	0.23
ResNet50 + LSTM	0.59	0.40
ResNet50 + Bi-LSTM	0.61	0.42

As seen in the table, the two models that were chosen are **ResNet50 + LSTM** and **ResNet50 + Bidirectional LSTM**. In the deployed web app, the user will have the option to choose one of these models to generate a caption. The reason is that some images are described better by ResNet50 with LSTM while others are described better with ResNet50 with Bidirectional LSTM.

Following is the comparison of the captions generated by the top three best-performing models **VGG16 + LSTM**, **ResNet50 + LSTM**, and **ResNet50 + Bi-LSTM** for the image shown below. This specific example shows that the model **ResNet50 + LSTM** seems to describe the image better compared to **ResNet50 + Bi-LSTM** even though the latter has a higher BLEU score.



VGG16 + LSTM:

Generated caption: white and white dog with white and white and white stripes

ResNet50 + LSTM:

Generated caption: person is holding big white dog

ResNet50 + Bidirectional-LSTM:

Generated caption: dog is sniffing outside of the room

Following is an example where Bidirectional LSTM seems to perform better than LSTM:



VGG16 + LSTM:

Generated caption: quality issues are too severe to recognize visual content

ResNet50 + LSTM:

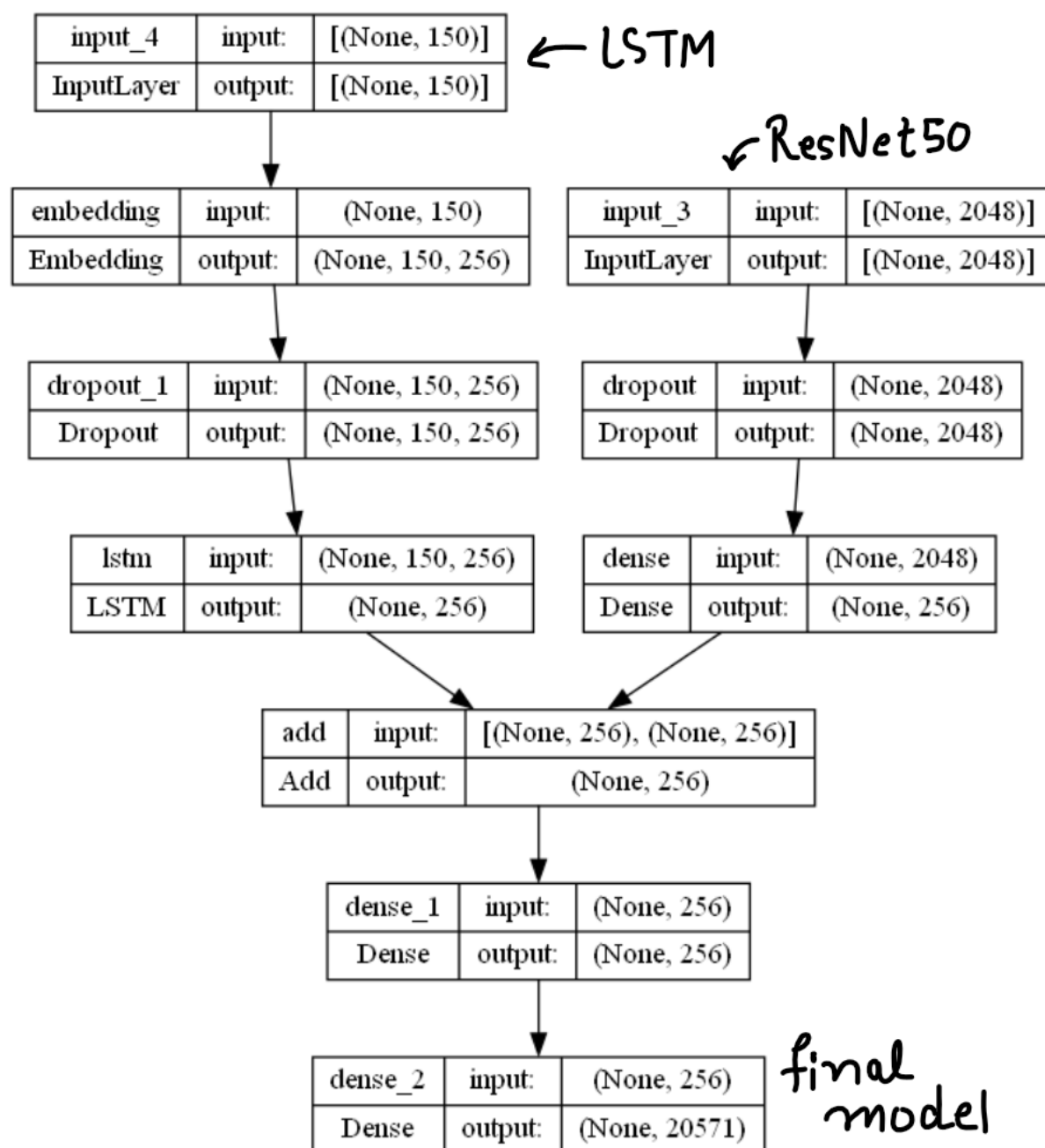
Generated caption: the back of singapore currency with the word twenty dollar bill

ResNet50 + Bidirectional-LSTM:

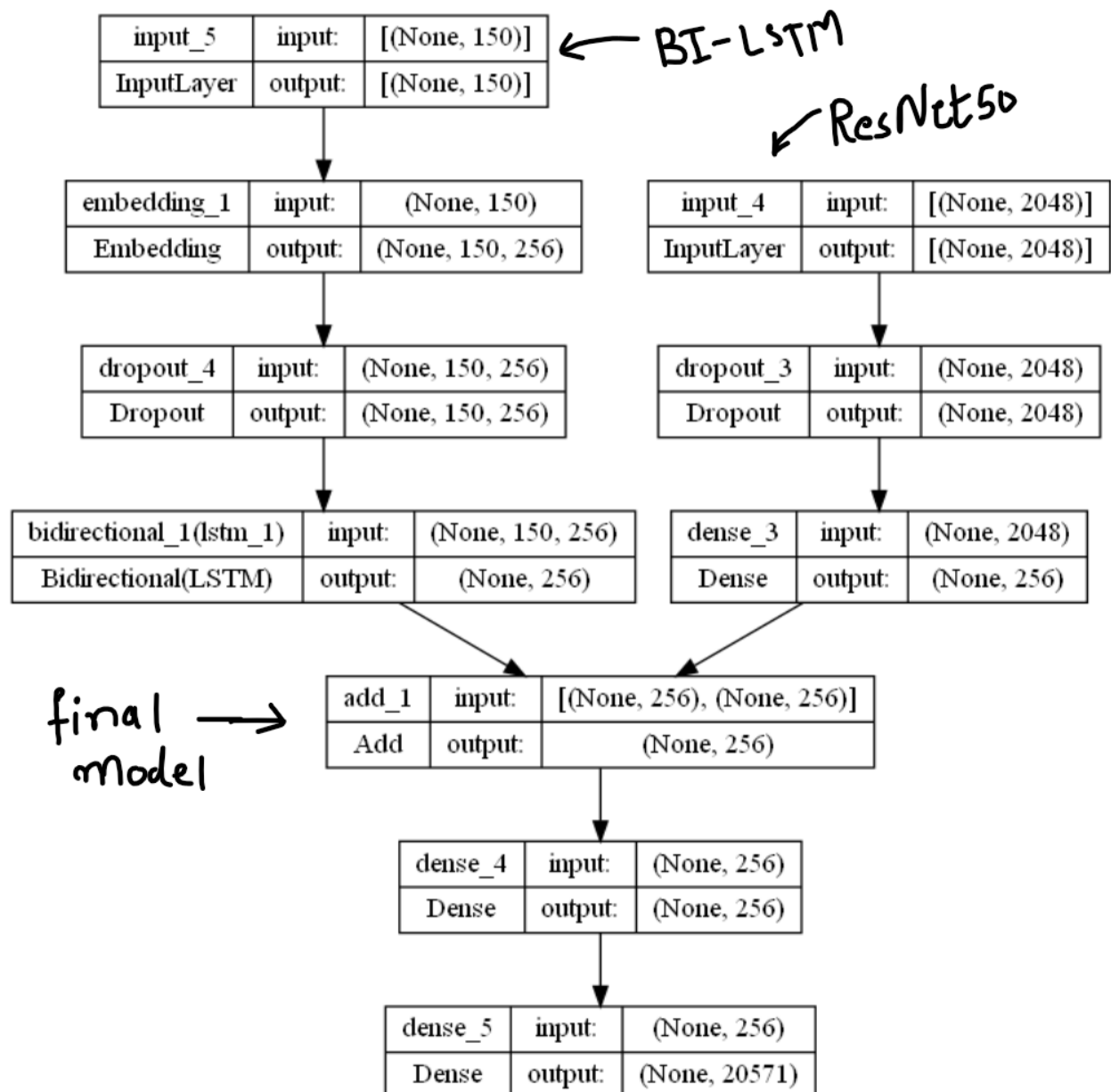
Generated caption: foreign currency is being held in someone's hand

As discussed in the preceding paragraphs, the **ResNet50 + LSTM** and **ResNet50 + Bidirectional LSTM** models were chosen in the end as they gave a more or less similar performance in terms of the BLEU score. Although the model ResNet50 + Bidirectional LSTM has the highest BLEU score, some images were described better by the 1st model while some are described better by the 2nd. Following are the summaries of both models.

ResNet50 + LSTM:



ResNet50 + Bidirectional LSTM:



Project Implementation:


Following is the technology stack and all the tools that have been used in developing the entire project.

- Keras & Tensorflow
- Matplotlib and Plotly
- Numpy
- NLTK
- Flask
- HTML5 + CSS3
- Bootstrap
- Javascript
- gTTS (Google's Text-To-Speech)
- pyttsx3 (Python's Text-to-Speech 3)

What follows is the demonstration of the entire developed app with some examples of captions generated by the chosen model.

Home Interface:

Image Caption Generator


Drag & Drop to Upload Image
or

Choose Model:

Users can choose one of the two models to generate captions from:

Choose Model:

Choose Model:

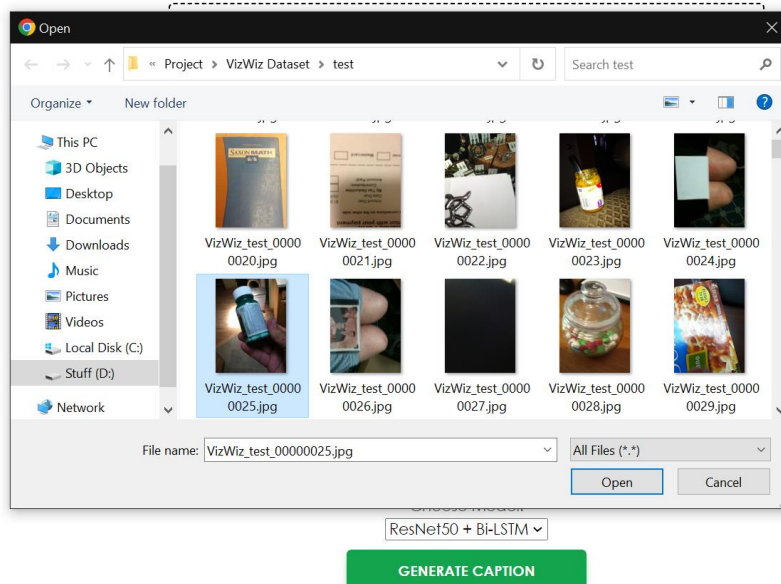
ResNet50 + Bi-LSTM

ResNet50 + Bi-LSTM

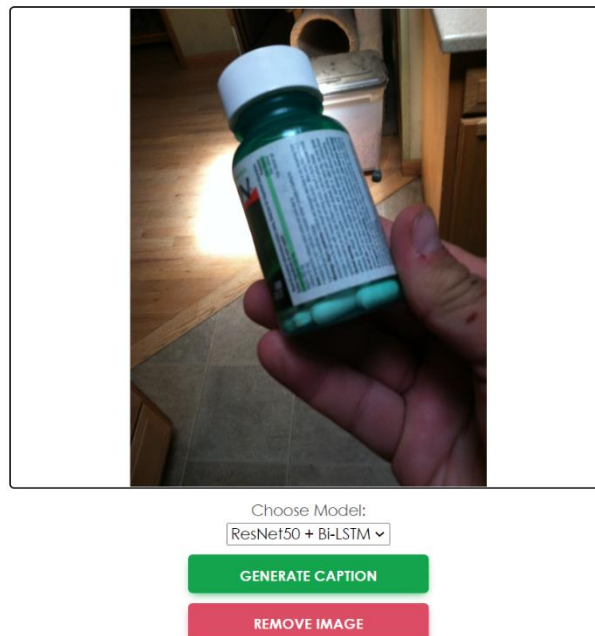
ResNet50 + LSTM

Browse an image:

Image Caption Generator




After image selection, the user has two options to either remove the image or generate a caption:



Generating a caption:

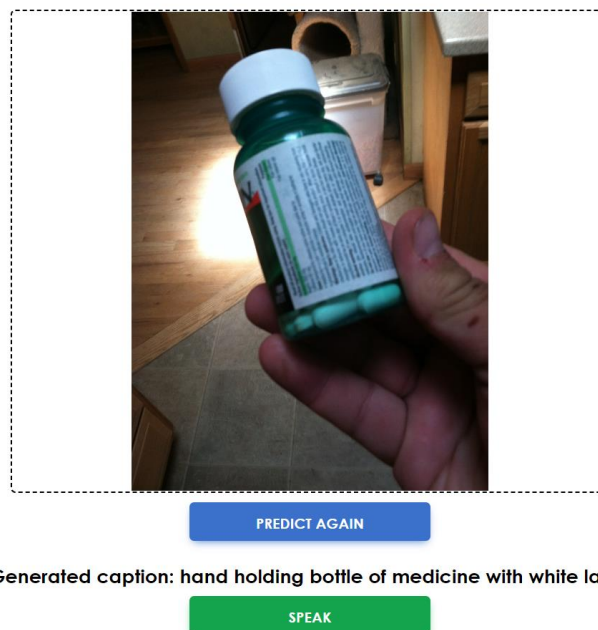
Choose Model:
ResNet50 + Bi-LSTM ▾



GENERATE CAPTION

REMOVE IMAGE

Generated caption:



The user can choose to hear the caption out loud if needed by clicking on the **‘speak’** button. The user can generate another caption by clicking on the ‘predict again’ button.

Major Outcomes:

1. Providing an automated place for visually impaired people to see the world by hearing the images that they have captured.
2. Increasing the ability of visually impaired people to understand their surroundings.
3. The analysis of the performance of different models on the dataset would reveal which model architecture seems to perform best on this specific dataset.

Learning Outcomes:

We Learned:

1. How to work with image data using CNN.
2. How to work with text data using NLP models.
3. How image captioning systems work.
4. How the 'add' module works that was used to concatenate NLP and CNN models together.
5. Preprocessing techniques for both image and text data.
6. The implementation of data generators to load the data in batches.
7. How to integrate ML with the web using python's web framework called Flask.
8. Python API pytsx3

Conclusion(s):

In this rapidly growing modern world with an ever-increasing population, there is a significant portion that is especially abled. One of the kinds from which is the visually impaired, who face hardships in understanding that which is in right front of them. This project which has combined the wonders of NLP and Computer Vision algorithms will assist the blind in perceiving their environment in a cheap and quick manner.

Reference(s):

- [1] Gurari, Danna, Yinan Zhao, Meng Zhang, and Nilavra Bhattacharya. 2020. "Captioning Images Taken By People Who Are Blind". Arxiv.Org. <https://arxiv.org/abs/2002.08565>.
- [2] "Papers With Code - Vizwiz Dataset". 2022. Paperswithcode.Com.
<https://paperswithcode.com/dataset/vizwiz#:~:text=The%20VizWiz%20DVQA%20dataset%20originates,cr%20sourced%20answers%20per%20visual%20question>