**CSCE 5290: Natural Language Processing**

**Final Project**

**Image Captioning and Speech Recognition using NLP**

Group Members

Ali Shah - 11518121

Nithish Kumar Boggula -11559328 Aditya Kapilavai - 11594174

GitHub Repository Link:- <https://github.com/Nithishkumar-11/NLP_project>

GOALS AND OBJECTIVES:

# Motivation:

The project automatically produces a Natural language description of an image when given as input and also To get the image as an output when an input text or voice is given. It has numerous potential uses, including helping people with visual impairments to comprehend images, enhancing the effectiveness of image search engines, and automatically generating descriptions for images in social media and news articles. This technology has the capacity to transform the way we engage with visual content, making it more widely accessible.

This strategy could lower crime and/or accidents since it simulates the human ability to describe visuals using computer language. Because every image might be transformed into a caption before being searched, automatic captioning could also contribute to the improvement of Google Image Search. Speech recognition is used for converting speech into text, searching for keywords, and finding appropriate pictures using NLP tools. The goal is to create technologies that allow machines to comprehend and produce human language, which could assist in enhancing the interaction between humans and machines.

**Objective**:

The primary objective of the project "Image Captioning and speech recognition using Natural language processing" is to develop a system that can automatically generate a textual description of an image using natural language processing techniques and by using speech recognition which converts the input voice of a person into a text and then based on the text find the images best suited for that particular description. The system takes an image as input and produces a description that accurately describes the content of the image and vice-versa.

Some specific objectives of this project include:

1. Developing an image recognition system that can identify the objects, people, and other elements present in the image.
2. Develop a speech recognition system to take the voice of a person as input and recognize the input and convert it into text.
3. Creating a natural language processing model that can generate coherent and descriptive captions based on the image features.
4. Integrating the speech recognition module with the text classifier and searching for the keywords.
5. Based on that if an image exactly matches the description then the output is displayed.
6. Integrating image recognition and natural language processing models to create a unified system that can generate captions for any given image.
7. Evaluating the performance of the system through various metrics such as accuracy, fluency, and relevance of the generated captions.
8. Exploring ways to improve the system's performance, such as incorporating attention mechanisms, using more advanced natural language processing techniques, or incorporating user feedback to refine the captions.

### Significance:

This project has several significant implications and potential benefits. Some of the key significances of the project are:

➔ Accessibility: This project's primary advantage is its ability to improve the accessibility of visual content for individuals with visual impairments. The automatic generation of text descriptions for images and speech-to-text conversion can significantly enhance the inclusivity and accessibility of visual content.

➔ Content Discovery: One potential application of this technology is to facilitate the organization and navigation of extensive image collections, thereby simplifying the process of locating and browsing visual content. This could prove particularly advantageous in fields such as art, fashion, and photography, which rely heavily on images as a means of communication.

➔ Automation: This method might also be used to automatically generate captions for a huge number of photographs, which would save a significant amount of time and money. This would be especially helpful in fields like e-commerce and advertising, where a lot of photos are used to promote and highlight different goods and services.

➔ Personalization:Using image captioning, you can create captions that are customized to each user's preferences and areas of interest. This can increase engagement with visual material and help customise user experiences.

➔ Research: Image captioning can be used to advance research in computer vision, natural language processing, and machine learning. The development of accurate and efficient image captioning systems can help advance the state-of-the-art in these fields and lead to new breakthroughs and innovations.

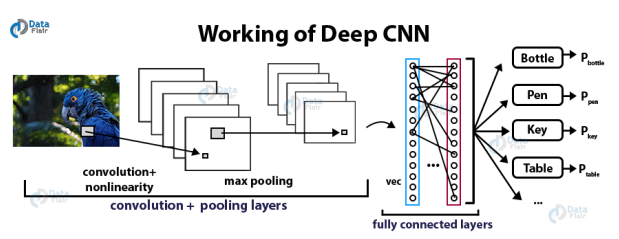
Overall, the project has the potential to make significant contributions to various fields and improve the accessibility, usability, and relevance of visual content.

**Related Work**: The field of artificial intelligence and image captioning has seen rapid progress, generating widespread interest. Image captioning involves creating a description of an image using a combination of deep learning, natural language processing, machine learning, and computer vision. The goal of this work is to demonstrate how a model can generate a brief description of an image using deep learning and natural language processing techniques. This technology can be used to assist visually impaired individuals and automate caption generation on various websites, making it easier for users to understand the content.

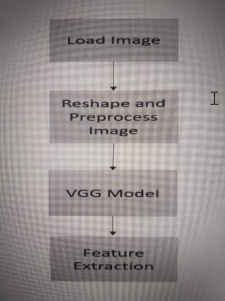
* Researchers have suggested many methods for image captioning using deep learning. These techniques require a lot of images with captions for training and testing the models. However, collecting such data can be expensive and time-consuming. In this paper, we present a new method for image captioning that uses both real and synthetic data for training and testing the model.

Steps Involved in our Project:-

* Importing the dataset
* Finding the Most & Least frequently occurring words.
* image preparation for VGG16 model.
* Linking the text & Image data
* Tokenizing
* Splitting the data
* Training the data
* Test the data

Working:- 

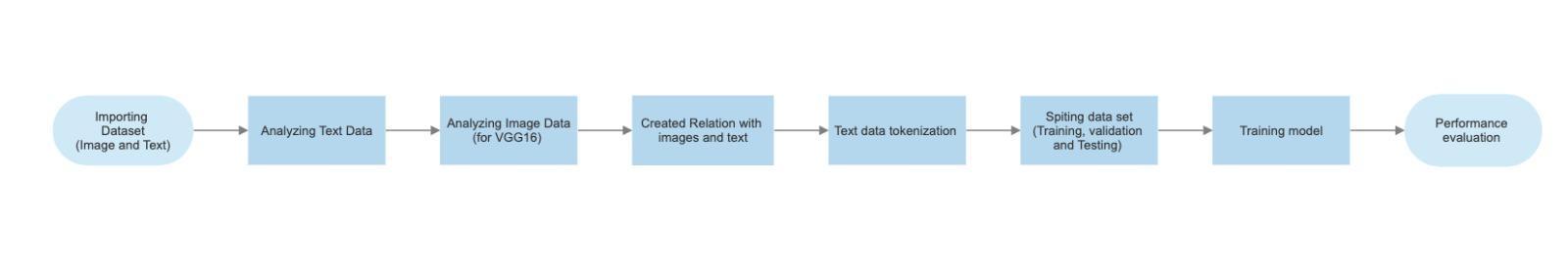
How it works:-



### Dataset: <https://drive.google.com/drive/folders/1tcTHIFSpJStpYpQq_R1OVUiVfNO7R0vh?usp=sharing>

<https://drive.google.com/drive/folders/123xXSEMBHpqGXcUUINa_Q_Eu2eKJWPDc?usp=sharing>

**FlowChart**:



### Features:

The features can vary depending on the specific implementation and approach used, but here are some common features that are often included in such projects:

Image recognition: To identify the various components within an image, the system employs an image recognition model. This process commonly involves utilizing convolutional neural networks (CNNs) to extract distinctive characteristics from the image and then implementing machine learning approaches to categorize the objects, individuals, and other relevant elements that appear within the image.

Speech recognition for converting speech to text: Speech recognition uses machine learning algorithms and neural networks to recognize and transcribe spoken words into text. This technology has several applications, including voice assistants, and speech-to-text transcription services. Some popular speech recognition APIs include Google Cloud Speech-to-Text, Amazon Transcribe, and Microsoft Azure Speech Services.

Natural language processing: The system uses natural language processing techniques to generate a textual description of the image. This typically involves using recurrent neural networks (RNNs) or transformer models to generate a sequence of words that describe the objects in the image.

Searching for keywords in text: You may use natural language processing (NLP) techniques to look for keywords after audio to text conversion. The Python Natural Language Toolkit (NLTK) is a well-liked NLP tool used for keyword extraction. A variety of algorithms for processing natural language text are included in the NLTK, including ones for named entity recognition, part-of-speech tagging, and tokenization. To find and extract crucial terms from the text, utilize these tools.

Attention mechanisms: The system may utilize attention techniques that concentrate on different sections of the picture when creating different parts of the caption in order to make sure that the generated caption appropriately represents the contents of the image.

Fine-tuning: Transfer learning approaches, which entail pre-training on large datasets and then fine-tuning on a smaller dataset to improve performance on a particular task, may be used to adjust the system.

Evaluation metrics: To evaluate the performance of the system, various metrics are used, such as the BLEU score, METEOR score, ROUGE score, and CIDEr score, which measure the similarity between the generated caption and a reference caption.

User interface: The system may feature a user interface that enables users to upload photographs and examine the automatically generated captions, depending on the desired use case. Users may be able to give comments on the produced captions' quality through the user interface, which can help the system get better.

Finding appropriate pictures: We may utilize image search APIs like Google Image Search or Bing Image Search to identify relevant images based on the keywords. These APIs provide you the ability to programmatically look for photographs using particular keywords or phrases. In order to analyze photos and find pertinent details, such as colors, shapes, and objects, we may also leverage computer vision APIs from companies like Microsoft Azure Cognitive Services or Google Cloud Vision.

In order to provide precise and evocative captions for photos, the project integrates a number of computer vision and natural language processing techniques. The specific implementation and needs of the use case will determine the system's characteristics.

Implementation

Many steps are involved in implementation. Here is a high-level breakdown of the procedures:

Data gathering: Compile a sizable database of pictures with captions that go along with them as well as a database of speech recordings with related transcriptions.

Preparing the data for analysis by preparing the audio and picture data. This could entail scaling pictures, adjusting pixel values, and employing speech-to-text software to transform audio files to text.

Extracting features: Use a pre-trained image recognition model to extract features from the picture data, and a pre-trained speech recognition model to extract features from the voice data.

Model training: Develop a deep learning model that can both transcribe voice and provide picture captions. Recurrent neural networks (RNNs) or transformer models like BERT or GPT will probably be used for this.

Model evaluation: Assess the effectiveness of the voice recognition and image captioning models using measures like the word error rate (WER) for speech recognition and the BLEU score for picture captions.

Deployment: Use a user-friendly interface, such as a web or mobile application, to embed the trained model in a system that enables users to input photos or record audio and retrieve captions or transcriptions.

You must have a thorough grasp of NLP and deep learning principles, as well as practical expertise with deep learning frameworks like TensorFlow or PyTorch. You may also need access to a GPU for training the deep-learning models.

Evaluation Methods:

Evaluation of Image Captioning:

* BLEU Score: The BLEU (Bilingual Evaluation Understudy) score is a statistic for assessing the quality of text produced by a computer in comparison to a collection of reference texts that were produced by humans. The degree to which the produced captions resemble the reference captions is measured.
* METEOR Score: The METEOR score (Metric for Evaluation of Translation with Explicit ORdering) measures how well-machine-generated text compares to a collection of reference texts that were created by humans. By calculating the harmonic mean of unigram accuracy and recall and accounting for word order, it assesses the value of a produced caption.
* ROUGE Score: The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score is a set of metrics intended to assess the quality of text produced by machines in comparison to a collection of reference texts created by humans. Calculating the overlap between a generated caption and the reference captions, it evaluates the caption's quality.
* CIDEr Score: The CIDEr score (Consensus-Based Image Description Evaluation) measures how well machine-generated text compares to a set of reference texts that were created by humans. By calculating the consensus between the generated caption and the reference captions, it assesses the caliber of the created caption.

Evaluation of Speech Recognition:

* Word Error Rate (WER): WER is a common evaluation metric for speech recognition systems. It measures the percentage of words that are incorrectly recognized by the system.
* Character Error Rate (CER): CER is similar to WER, but measures the percentage of characters that are incorrectly recognized by the system.
* Sentence Error Rate (SER): SER measures the percentage of sentences that are incorrectly recognized by the system.
* F1 Score: F1 score is the harmonic mean of precision and recall. It is a useful metric for evaluating speech recognition systems when the number of false positives and false negatives is important.
* Perplexity: Perplexity is a measure of how well a language model can predict the next word in a sequence. It is a useful metric for evaluating the quality of the language model used in a speech recognition system

Workflow:- For this project, we have used Agile Methodology to implement our Tasks. As we know In Agile we split the tasks into smaller sprints and implement them as planned. Initially, we gathered all the information and then designed task phases where we need to complete the specific task in that time period. The following are the phases we have completed and are yet to complete.

| Project | Status | Related file | Notes |
| --- | --- | --- | --- |
| Phase 1 | completed | https://colab.research.google.com/drive/1CZeDqsurMuTSgSl3oT10dD\_zldc2db\_q?usp=sharing | Image to Text |
| Phase 2 | Completed | https://colab.research.google.com/drive/13wCSTwnN791YreMYmtdl4UrvueOta18r?usp=sharing | Text to Image |

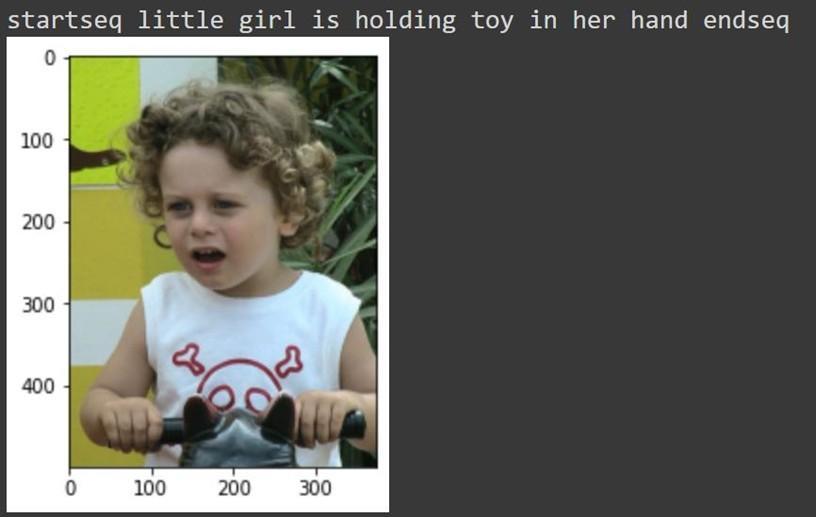
Contributions:-

Nithish Kumar Boggula - 33% ( Documentation, and Coding)

Aditya kapilavai - 33% (Coding, Documentation, dataset collection)

Ali Shah - 33% ( Coding and Documentation)

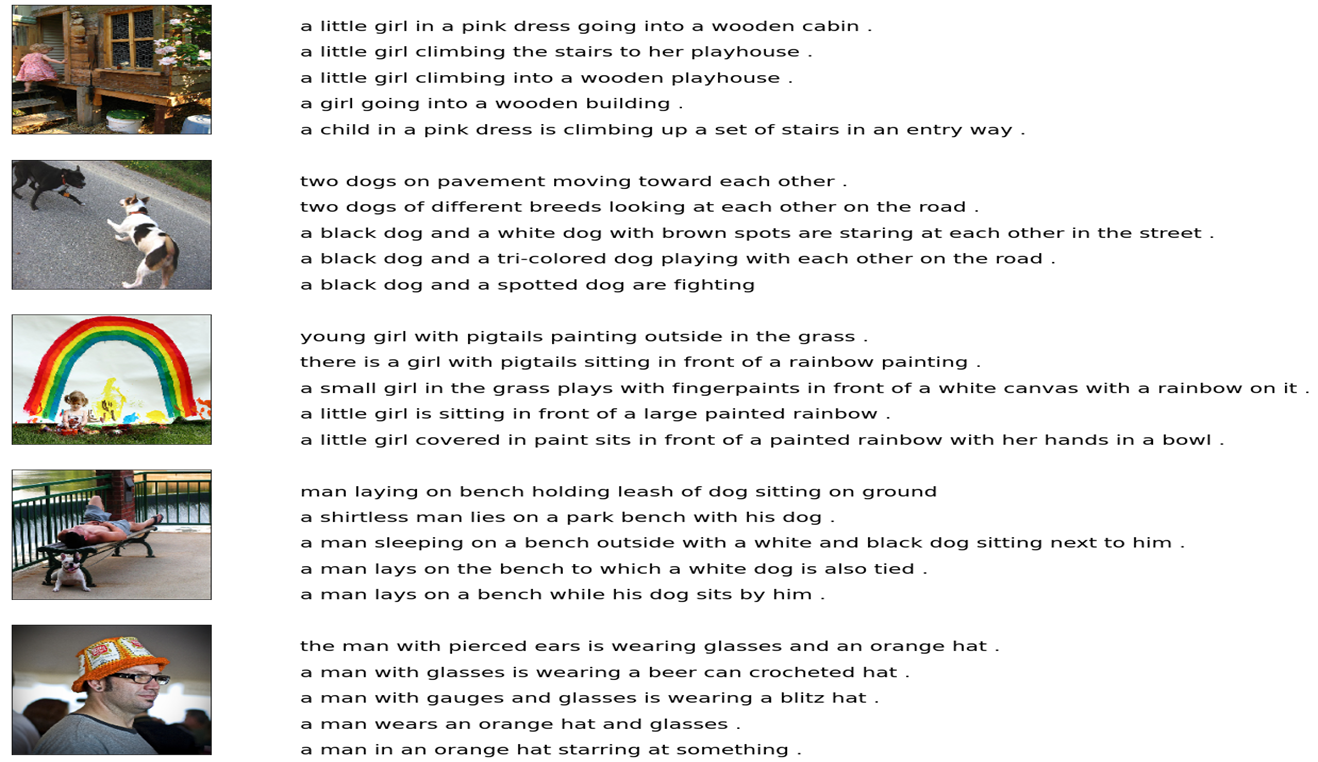
Analysis and Preliminary Results



In the above fig, If the image input is given, Caption will be generated.



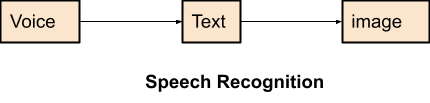
If the input words/sentences are given, images/related images will be the output.



The above fig demonstrates the dataset of input and training Captions.

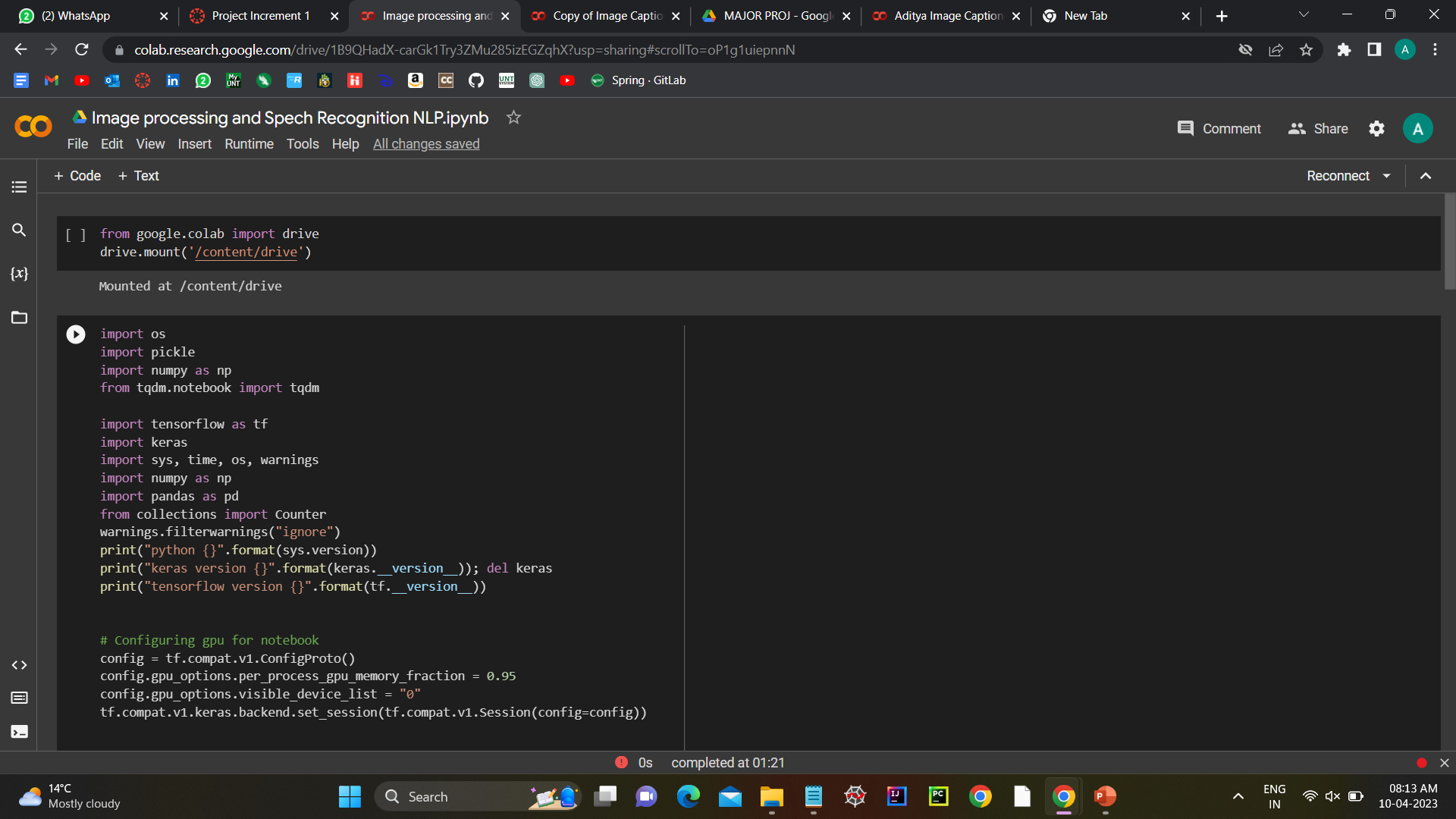
This can be applied to both Texts to speech and speech-to-text respectively. Once the speech or audio has been given as text then the text is tokenized and we search for keywords in the text. Based on the keywords we can match the text with the image.

In order to make it work properly the model needs to be trained and must recognize the keywords and speech must be exactly identified. There can be many features added to this project in the coming future.



**Results**

The below screenshots show the code snippets



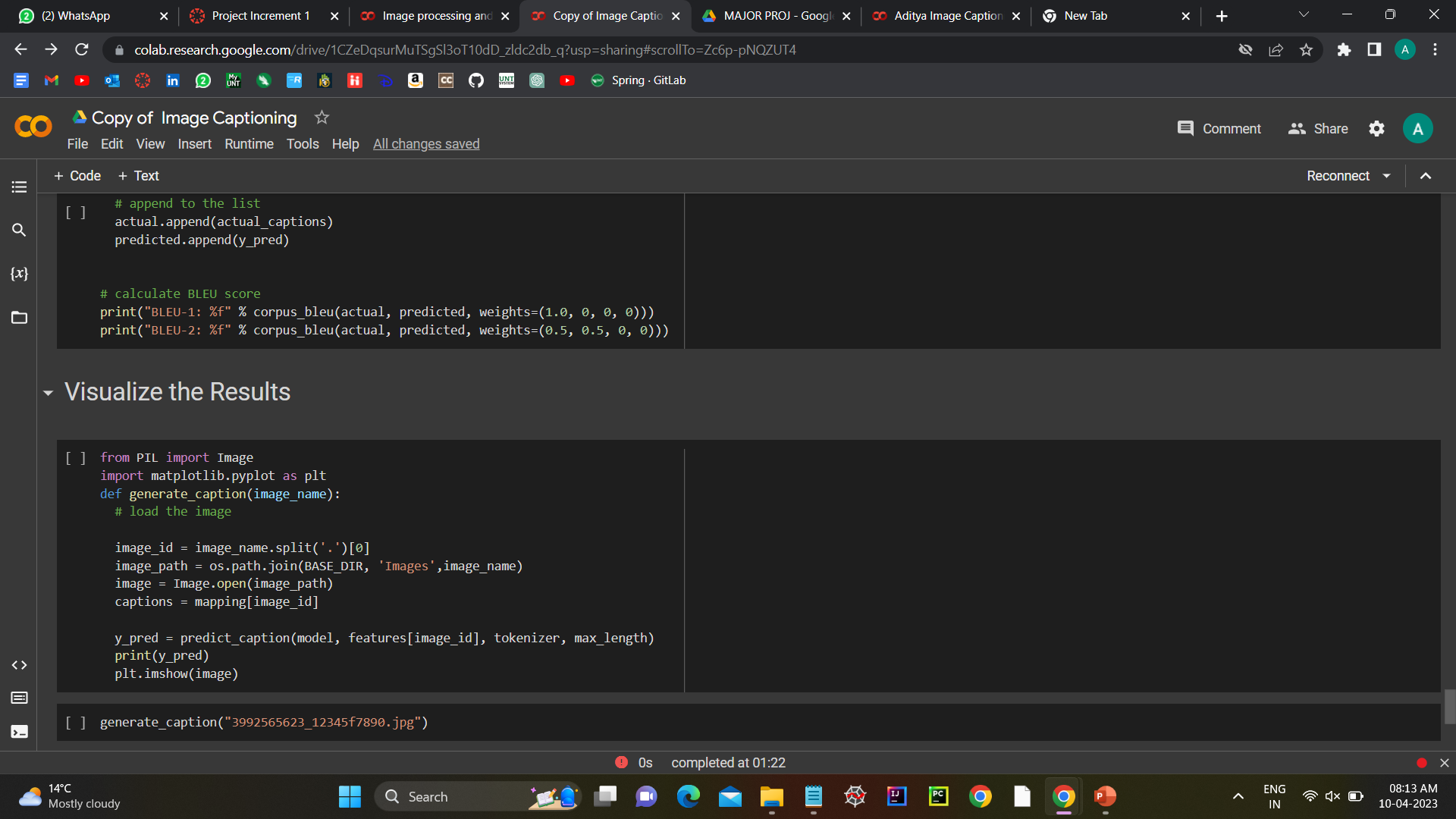
Example: This code snippet creates the necessary conditions for a machine learning model to execute using the Keras and TensorFlow libraries.

'os', 'pickle', 'numpy', 'tqdm', 'tensorflow', and 'keras' are among the first libraries to be imported.

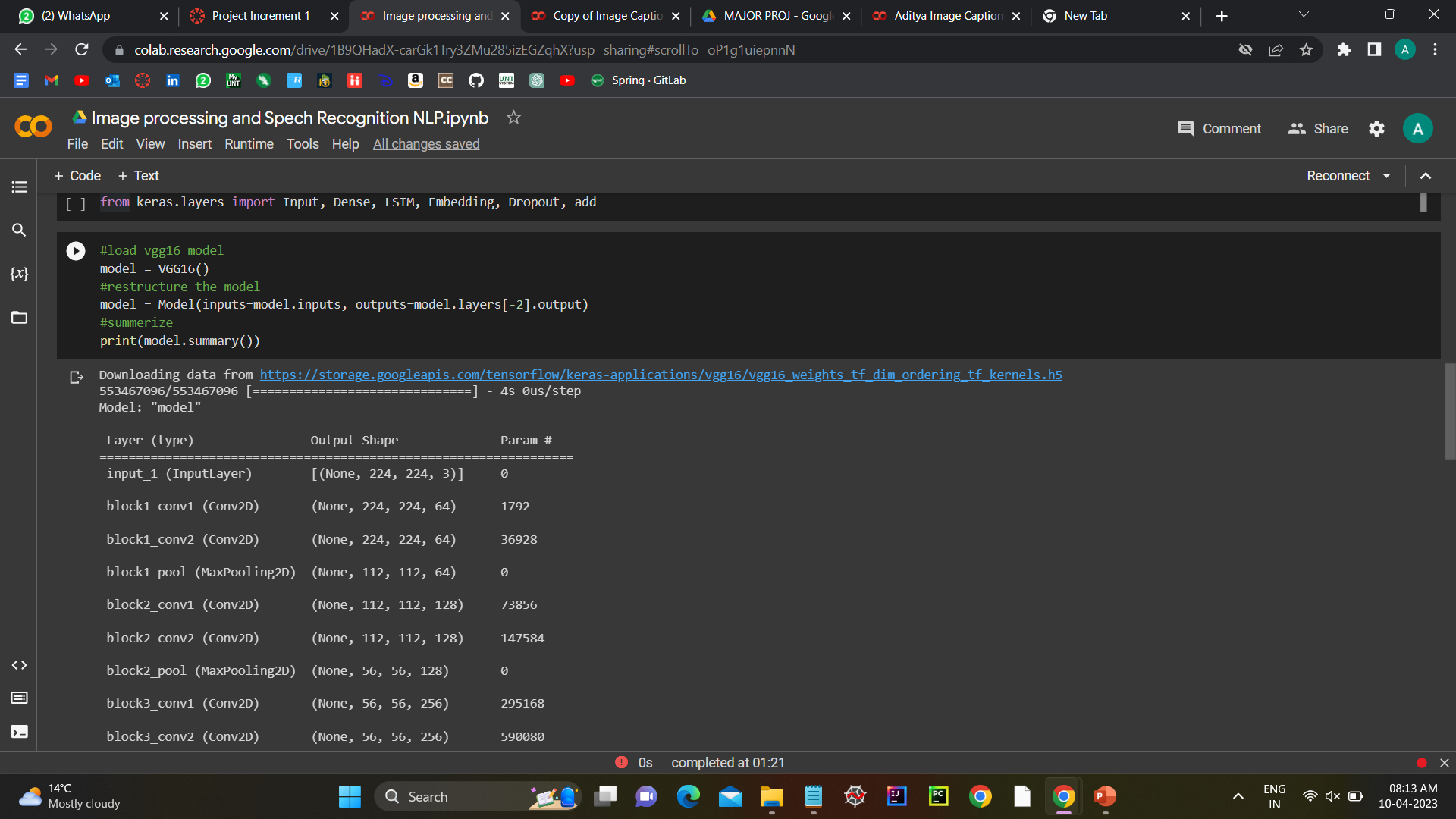
The 'print()' method is then used to output the version information for Python, Keras, and TensorFlow to the console.

Then, a TensorFlow session is set up with GPU computation enabled. The 'gpu\_options' property is configured to allocate 95% of the GPU RAM for calculation when the 'config' variable is generated using the 'tf.compat.v1.ConfigProto()' function. To utilize just one GPU device, the 'visible\_device\_list' property is set to "0". The 'tf.compat.v1.keras.backend.set\_session()' function is then used to set the session.

The method "set\_seed()" is defined lastly. To guarantee that the findings can be replicated, this method sets the random seed for the TensorFlow, NumPy, and core Python random number generators. The random seed for NumPy is set using the'seed()' method, the random seed for TensorFlow is set using the'set\_random\_seed()' method, and the random seed for Python's built-in 'random' module is set using the 'rn.seed()' method.

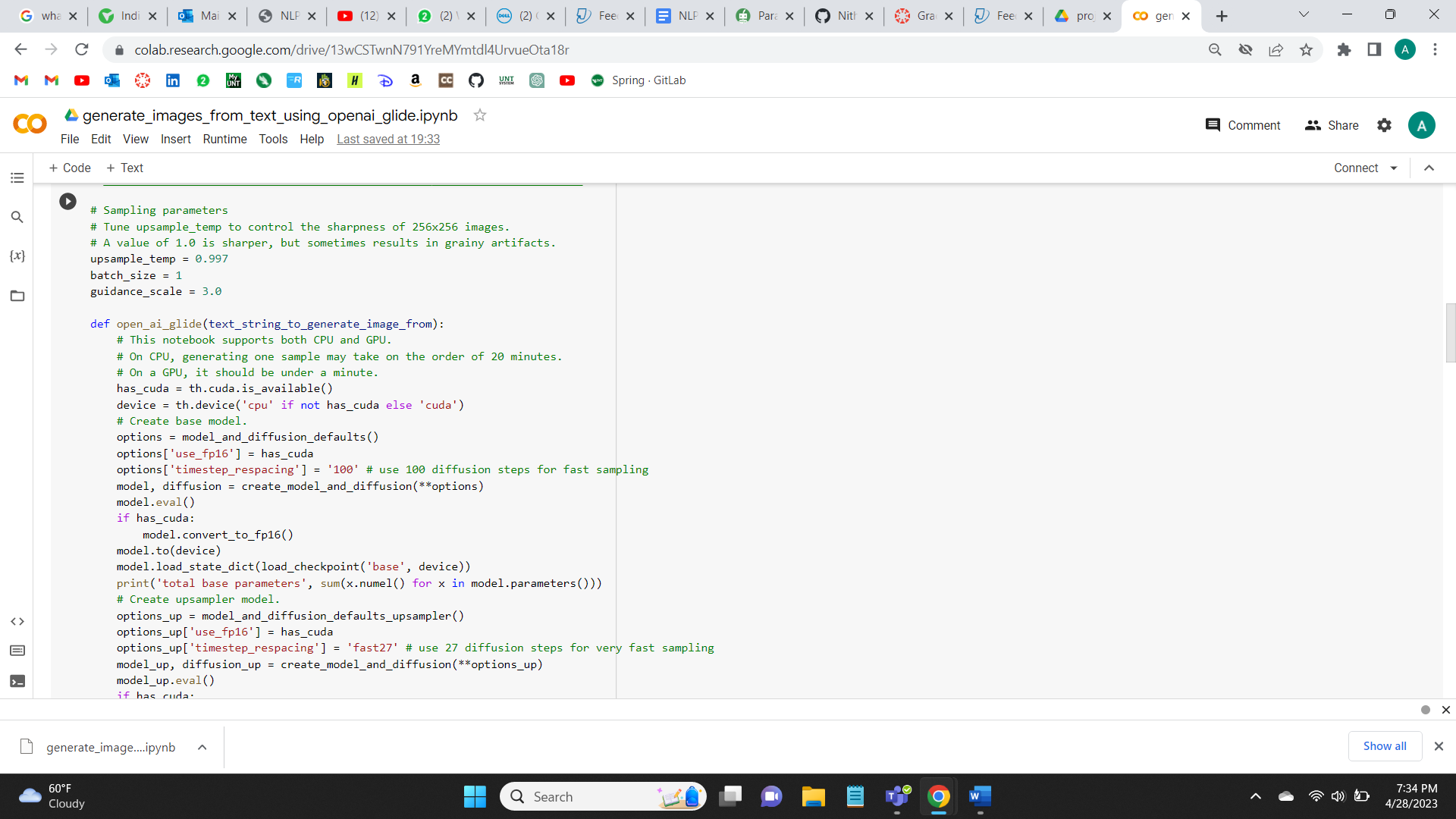


* **Explanation**: This code seems to be processing a dictionary "mapping" that has lists of captions as values and picture file names as keys. The code is broken down as follows:
* The first line 'clean(mapping)' seems to be a function call that cleans the text in each caption in the dictionary. In order to improve the text's suitability for use in a natural language processing (NLP) job, it is probable that this function eliminates any extraneous characters, stopwords, or performs other text preparation operations. It is challenging to describe the 'clean()' function's actual function, though, without understanding how it is implemented.
* The following few lines build the empty list "all\_captions," iterate over all of the dictionary's keys, and then return. It repeatedly cycles through the



* **Explanation**: By the end of this loop, "all\_captions" will contain a flattened list of each caption in the dictionary.
* The final line, "len(all\_captions)", displays the number of captions in the list "all\_captions".
* The last line, "all\_captions[:20]," shows the first 20 captions in the list.

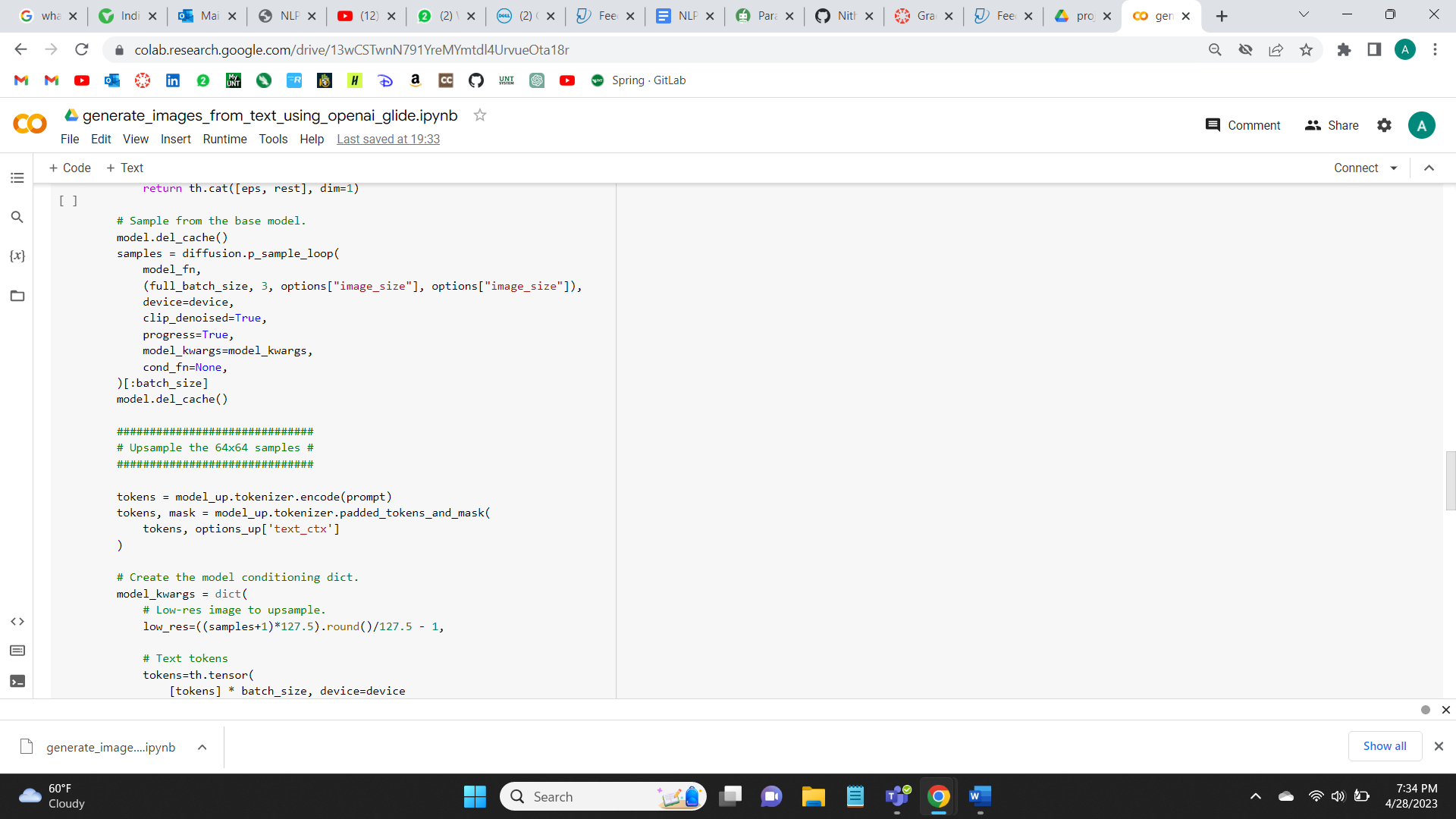
A list of cleaned captions that may be used for a variety of NLP tasks, such as text categorization, language modeling, or text synthesis, appears to be created by flattening and preprocessing a dictionary of photo captions.



Explanation: The method 'open\_ai\_glide' is defined in this code and it accepts a 'text\_string\_to\_generate\_image\_from' input. The program loads a pre-trained model and uses diffusion models to create a picture based on the supplied text.

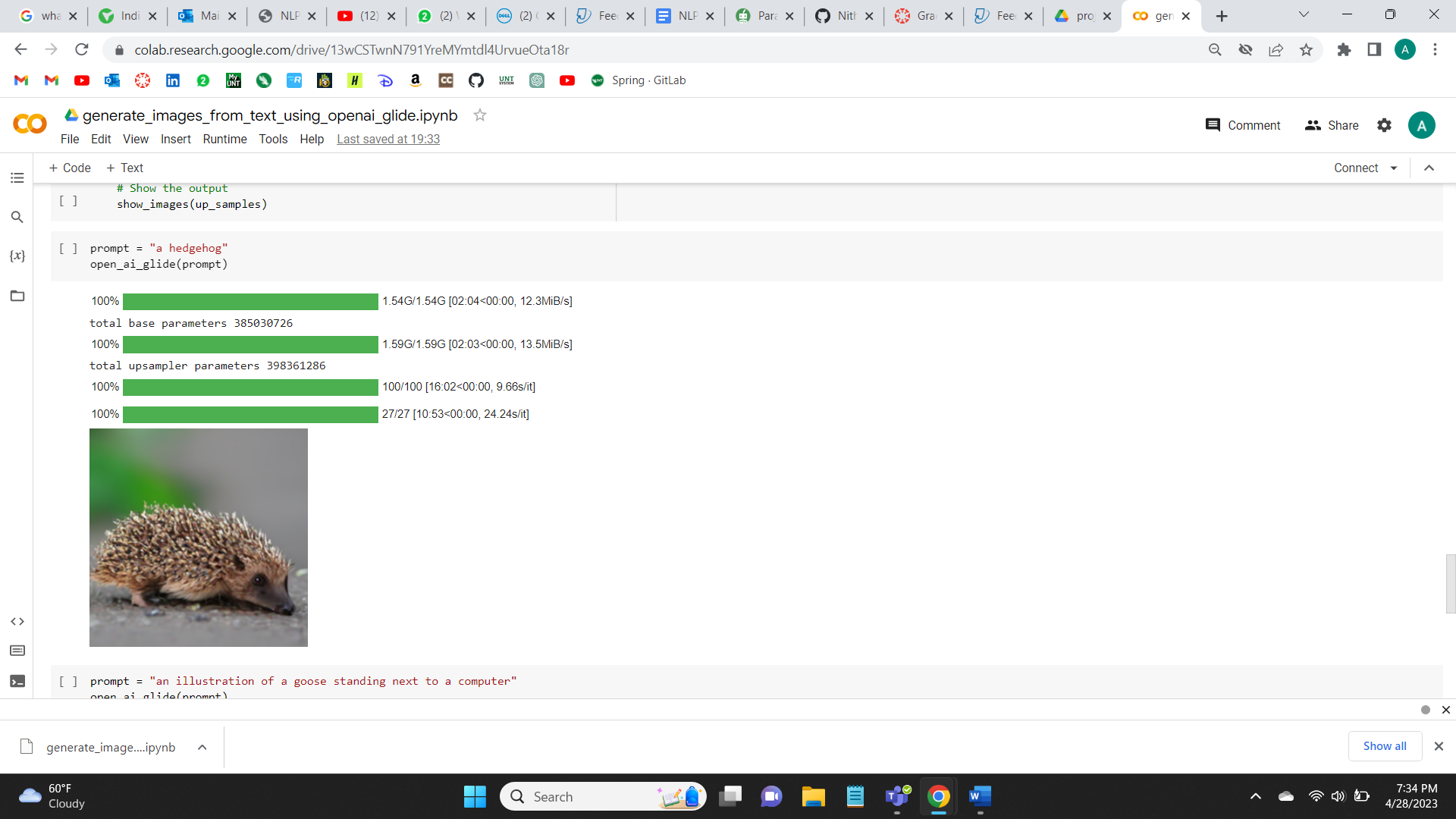
When a GPU is found, the function determines how to configure the device. It then uses predefined settings to generate a base model and an upsampler model, switches them into evaluation mode, and loads their previously learned weights. The total number of parameters for each model is printed by the function.

The function also has a "show\_images" inner function that accepts a PyTorch tensor as input and outputs any images it contains.



Explanation: - The model computes the graphs are deleted in the first line i.e. model.del\_cache(). This function will help us to free up memory and utilize it.

The diffusion model is used to create picture samples in the next line using the 'diffusion\_sample\_loop()' function. The "model\_fn" function accepts the noise inputs and outputs a batch of pictures. The batch size, picture dimensions, and the device to utilize such as a GPU are all specified in the parameters to the 'p\_sample\_loop()' function. If the unnoised images must be cropped between 0, 1 as it is specified by the 'clip\_denoised' option. Whether to show a progress bar when sampling is determined by the 'progress' parameter. The dictionary of keyword arguments for "model\_fn" called "model\_kwargs" is the last one. The model's cached computation graphs are once again deleted in the next line, "model.del\_cache()." The conditional upsampling model is used in the following portion of the code to upsample the 64x64 samples produced in the previous section to a higher resolution. The upsampling model appears to go by the moniker "model\_up." The model\_up.tokenizer.encode()' function is used in the code to first tokenize a prompt. It uses the "model\_up.tokenizer.padded\_tokens\_and\_mask()" function to pad the tokens and produce a mask. Finally, it builds a dictionary of keyword parameters, containing the text tokens, mask, and low-resolution pictures to upsample, to provide to the upsampling model. Overall, it appears that this code combines a diffusion model with a conditional upsampling model to produce high-quality pictures from a text prompt.



Explanation: This is how an image is generated when a text is given as input. Here in the code we have used OpenAI Glide Model inorder to get an image when we are giving a text. It takes a string as an input and it initializes the model which gives us the image based on the prompt. The image is in the form of a tensor. The show\_images function is used to display the images in line. It takes a batch of images as tensors and it adjusts the pixels from 0-255 and reshapes the tensor to an image.

Here we are using PIL, IPython.display and torch to complete this project. GLIDE which is abbreviated as Generative Language and image diffusion engine model. The most important libraries for generating and displaying images are PyTorch and PIL which is abbreviated as python Imaging Library.

## References/Bibliography:

Enhancing Descriptive Image Captioning with Natural Language Inference

<https://aclanthology.org/2021.acl-short.36/>

Generating Image Captions based on Deep Learning and Natural Language Processing

<https://ieeexplore.ieee.org/abstract/document/9596486>

Text to Image Synthesis for Improved Image Captioning

<https://ieeexplore.ieee.org/abstract/document/9416431>

An Efficient Text-Based Image Retrieval Using Natural Language Processing (NLP) Techniques

<https://link.springer.com/chapter/10.1007/978-981-15-5400-1_52>

D.V.T and V.R, "Retrieval Of Complex Images Using Visual Saliency Guided Cognitive Classification", J. Innov. Image Process, vol. 2, no. 2, pp. 102-109, Jun. 2020.

[chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://irojournals.com/iroiip/V2/I2/05.pdf](https://irojournals.com/iroiip/V2/I2/05.pdf)