CHAPTER 1

INTRODUCTION

CHAPTER 1 INTRODUCTION

In recent years, with the rapid development of artificial intelligence, image caption has gradually attracted the attention of many researchers in the field of artificial intelligence and has become an interesting and arduous task. Automatically generating captions for a given image shows the amount of understanding a computer model hash about the image, this is somewhat a fundamental task of intelligence.

Visuals and imagery continue to dominate social and professional interactions globally. With a growing scale, manual efforts are falling short on tracking, identifying, and annotating the prodigious amounts of visual data. With the advent of artificial intelligence, multimedia businesses can accelerate the process of image captioning while generating significant value. AI-powered image caption generator employs various artificial intelligence services and technologies like deep neural networks to automate image captioning processes.

Image Captioning in basic terms means generating a natural language description of a given image, i.e., for any given image giving a description describing the objects, subjects and action taking place in the image. This in computers can be done using some deep learning algorithms, knowledge of computer vision and Natural Language Processing (NLP), which is a branch of computer science which deals with making a computer perform tasks to understand texts and spoken words same way a human does.

1.1. SCOPE

The project extends study mainly into the area of Deep Learning Neural Networks concerning digital image processing and natural language sentence formation using CNN (Convolutional Neural Networks) and RNN (Recurrent Neural Networks) – LSTM (Long Short-Term Memory).

1.2. NOVELTY OF IDEA

1.2.1. Image Captioning:

1) Recommendations in Editing Applications

The image captioning model automates and accelerates the close captioning process for digital content production, editing, delivery, and archival. Well-trained models replace manual efforts for generating quality captions for images as well as videos.

2) Assistance for Visually Impaired

The advent of machine learning solutions like image captioning is a boon for visually impaired people who are unable to comprehend visuals. With AI-powered image caption generator, image descriptions can be read out to visually impaired, enabling them to get a better sense of their surroundings.

3) Media and Publishing Houses

The media and public relations industry circulate tens of thousands of visual data across borders in the form of newsletters, emails, etc. The image captioning model accelerates subtitle creation and enables executives to focus on more important tasks.

4) Social Media Posts

For social media, artificial intelligence is moving from discussion rooms to underlying mechanisms for identifying and describing terabytes of media files. It enables community administrators to monitor interactions and analysts to formulate business strategies.

1.2.2. Emotion Detection:

- Mental health: Emotion detection can be used to diagnose and monitor mental health conditions such as depression, anxiety, and bipolar disorder.

- Customer service: Emotion detection can be used to improve customer service by detecting customer emotions and sentiments in real-time, which can help customer service representatives tailor their responses to provide better customer support.

- Marketing and advertising: Emotion detection can be used to measure the emotional response of consumers to marketing and advertising campaigns, which can help companies optimize their marketing strategies.

- Education: Emotion detection can be used to assess the emotional state of students in the classroom and provide personalized support and feedback.

- Human-robot interaction: Emotion detection can be used to create more human-like interactions between humans and robots by allowing robots to understand and respond to human emotions.

- Gaming: Emotion detection can be used to create more immersive gaming experiences by detecting and responding to the emotions of the player.

- Law enforcement: Emotion detection can be used to assess the emotional state of suspects during interrogations and investigations.

1.2.3. 3D model of 2D images:

Virtual reality and augmented reality: 3D models can be used to create immersive virtual reality, and augmented reality experiences, where users can interact with virtual objects and environments.

Gaming and animation: 3D models can be used to create realistic game characters and environments, as well as animated movies and TV shows.

Architecture and construction: 3D models can be used to create detailed architectural plans and visualizations, as well as simulating construction projects.

Industrial design and manufacturing: 3D models can be used to design and test products before they are manufactured, as well as to create prototypes using 3D printing.

Medical imaging: 3D models can be used to visualize and study complex anatomical structures and assist with surgical planning.

Education: 3D models can be used to create interactive educational content, such as virtual science labs and historical reenactments.

Art and cultural heritage: 3D models can be used to create digital representations of art and cultural artifacts, allowing them to be studied and preserved for future generations.

CHAPTER 2

PROBLEM DEFINITION

CHAPTER 2 PROBLEM DEFINITION

2.1. Image Captioning:

The use of Image Captioning to describe the content of an image using natural language can be very difficult task, but can have an impact such as, helping a visually challenged person to understand the content of the images in the Web or even in the real-world scenario while using their phone cameras.

In the following work, we have tried to make a proverbial bridge between vision and natural language, using some of the Deep Learning Models and techniques. Under deep learning we are using convolutional neural networks and recurrent neural networks for the following project. For the dataset required, we are currently using Flickr8K dataset.

Automatically generating captions to an image shows the understanding of the image by computers, which is a fundamental task of intelligence. For a caption model it not only need to find which objects are contained in the image and need to be able to be expressing their relationships in a natural language such as English. Recent works show that using Convolutional Neural Networks and Recurrent Neural Networks can be used for generating captions for a given image. Using models such as ResNet50 based on CNN (Convolutional Neural Network) for image processing and RNN (Recurrent Neural Network) for caption generation for respective images we can create a prototype of the Image Caption Generator. This will be a tool to utilize the massive unformatted image data, which dominates the whole data in the world.

2.2. Emotion Detection:

Emotion detection can be defined as developing a machine learning model that can accurately identify the emotional state of a person based on their facial expressions, speech patterns, and other physiological signals. The objective is to create a system that can detect emotions such as happiness, sadness, anger, fear, surprise, and disgust, and use this information to improve communication, customer service, and other related applications. The primary challenge is to design a model that can accurately identify subtle emotional cues and variations, as well as handle diverse cultural and linguistic contexts.

2.3. 3D Representation of a 2D Image:

The goal of creating a 3D model from a 2D image is to create a computer vision system that can precisely recreate a scene's or object's 3D structure from a 2D image. The goal is to produce a 3D model that can be used in a variety of contexts, including gaming, computer graphics, augmented reality, and virtual reality. The main challenge is to handle occlusions, fluctuations in illumination, and perspective distortions while accurately estimating the depth and position of the numerous objects in the picture. The system should be capable of handling a variety of image formats, such as photographs, sketches, and paintings, and it should be able to create accurate 3D models that accurately reflect the features and subtleties of the original scene or object.

CHAPTER 3

LITERATURE REVIEW

CHAPTER 3 LITERATURE REVIEW

3.1. Image Captioning:

The paper explains how in artificial intelligence (AI), computer vision and natural language processing are used to automatically create an image's contents (Natural Language Processing). The regenerative neuronal model is developed. It is dependent on machine translation and computer vision. Using this technique, natural phrases are produced that finally explain the image. Convolutional neural networks (CNN) and recurrent neural networks are also components of this architecture (RNN). The CNN is used to extract features from images, and the RNN is used to generate sentences. The model has been taught to produce captions that, when given an input image, almost exactly describe the image. On various datasets, the model's precision and the fluency or command of the language it learns from visual descriptions are examined. Most of the earlier attempts have recommended combining all of the current answers to the aforementioned challenge in order to generate description from an image. Whereas we will create a single model that uses an image as input and is trained to produce a string of words, each of which is a term from the dictionary and appropriately describes the image. The relationship between descriptions and visual importance moves on to the issue of natural language text summary (NLP) processing. The fundamental objective of text summarizing deciding on or producing an abstract for a document. In image captioning issue, we would like to caption any image create a caption that highlights the different aspects of that image. In this study, a paradigm for creating original descriptions from photos is proposed. We used the Flickr 8k dataset for this assignment, which consists of 8000 photos and five descriptors per image. Both CNN and RNN are used in this work. For the job of picture categorization, a pre-trained Convolutional Neural Network (CNN) is utilized. This network serves as an encoder for images. The final hidden layer serves as the Recurrent Neural Network's input (RNN). This network serves as a decoder and sentence generator. Sometimes the generated sentence appears to get off course or forecast the incorrect sentence compared to the substance of the original image.

Author's Name and Paper Title	**Conference/Journal Name and year**	**Technology/ Design**	**Results shared by author**	**What you infer**
N. Komal Kumar, D. Vigneswari, A. Mohan, K. Laxman, J. Yuvaraj / Detection and Recognition of Objects in Image Caption Generator System: A Deep Learning Approach	IEEE 2019	Object detection, feature extraction, Convolution Neural Network (CNN) for feature extraction and for scene classification, Recurrent Neural Network (RNN) for human and objects attributes, RNN encoder and a fixed length RNN decoder system.	The proposed deep learning methodology generated captions with more descriptive meaning than the existing image caption generation generators.	Using RNN and CNN together for image captioning allows us to produce a more accurate data and train the model better.
Varsha Kesavan, Vaidehi Muley, Megha Kolhekar / Deep Learning based Automatic Image Caption Generation	IEEE 2019	Attention model used to view the image from the perceptive of captioner to decide what is important and what is not important when it comes to captioning an image.	The results achieved were near accurate.	Using attention model to have a better perspective for captioning achieves good results.

Chetan Amritkar, Vaishali Jabade / Image Caption Generation using Deep Learning Technique	IEEE 2018	In this work, neural framework is proposed for generating captions from images which are basically derived from probability theory. By using a powerful mathematical model, it is possible to achieve better results, which maximizes the probability of the correct translation for both inference and training.	The model is trained to produce the sentence or description from given image.	Uses mathematical model which helps in better prediction and training of the model.
SiZhen Li, Linlin Huang / Context-based Image Caption using Deep Learning	IEEE 2021	Fully extract the image features of the picture, redesign the network structure, propose a context-based image caption, and combine reinforcement learning	Experimental results show that the model algorithm designed in this paper can produce good description effects. The model proposed in this paper improves the performance of image description	Uses redesigning and reinforcement learning which makes the model to make better descriptions.

3.2. Emotion Detection:

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image.

The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples and the public test set consists of 3,589 examples.

Author's Name And Papers Title	Conference / journal Name and Year	Technology / Design	Results Shared by Author
Emotion Recognition Based On CNN Guolu Caoˈ Yuliang Ma*ˈ Xiaofei Mengˈ Yunyuan Gaoˈ Ming Meng	IEEE 2019	Emotion recognition is an important research topic in the fields of neuroscience, psychology, cognitive science, computer science and artificial intelligence. Neural network is a statistical learning model inspired by biological neural networks. Emotions are very important to human decision-making, interaction and cognitive processes. With the development of technology and better understanding of emotional states, automatic emotion recognition systems are widely used.	The primary classification model used in the study is the CNN, which effectively classifies the pre-processed EEG data presented as a 2D array. The study reports a considerable classification accuracy that is better than previous research. The study concludes that the neural network can be used as an effective classifier for EEG signals and is superior to traditional learning techniques.

Facial Emotion Detection Using Deep Learning	IEEE - 2020	It is true that artificial intelligence (AI) has been applied to emotion recognition, particularly in the field of facial expression analysis. This technology has practical applications in fields such as security and crowd management, as well as in the development of emotionally sensitive human-computer interfaces (HCI) that can enhance user experience and efficiency.	The authors compared the performance of their proposed model with existing models in the literature and found that their model produced better results in terms of emotion detection. They also conducted experiments using both trained and test sample images and found that their proposed model produced state-of-the-art results on both datasets.

CHAPTER 4

PROJECT DESCRIPTION

CHAPTER 4 PROJECT DESCRIPTION

Image Captioning:

The project aims to detect, recognize, and generate captions using deep learning. Image Caption Generator deals with generating captions for a given image. And this can be useful in many real-life applications and scenarios.

Humans have a natural tendency to describe an image with a wealth of information by giving it just a fast glance. Artificial intelligence and machine learning researchers have long sought to build computer systems that can mimic human talents. Research has advanced in a number of areas, including the identification of objects in an image, attribute classification, picture classification, and the categorization of human actions. Making an image caption generator system which is a computer programme that can identify a picture and generate a description using natural language processing which is a difficult task.

The Images required for this project are sourced from a dataset like Flickr8k. The dataset consists of several images and predefined captions for training of such models.

The program takes images and process them in the Convolutional Neural Network and produces a vector form of the image, which basically takes every aspect of the image into consideration for example, the colors present, the objects etc. The vector form describes the objects and colors present in the image. The vector form is then sent to RNN for caption generation. Using LSTM, the captions are generated in a grammatically and syntactically accurate sentence.

4.1. System Design:

The design of the model can be split into three main parts:

• Dataset Preprocessing

• Image Feature Extraction

• Caption generation

4.1.1. Dataset Preprocessing

The dataset currently planned for use is Flickr8k, which is a dataset which provides stock images with a set of captions which can be used to describe each individual image provided by the respective dataset. There are many other datasets which can be used for the training of the model such as Flickr30k, MSCOCO etc., but for the initial stage of the project we plan on using Flickr8k for the prototype of the model for easier and faster training. In this part of the project, we look at the file name and caption text document name and we split the required part of the name removing the root path of the file names.

4.1.2. Image Feature Extraction

Convolutional Neural Network is mainly used for feature extraction of the images, along with RelU Non-linear Activation. It is a class of Artificial Neural Network which is applied to visual imagery. It is also known as Shift Invariant or Shift Invariant Artificial Neural Networks and is based on shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps.

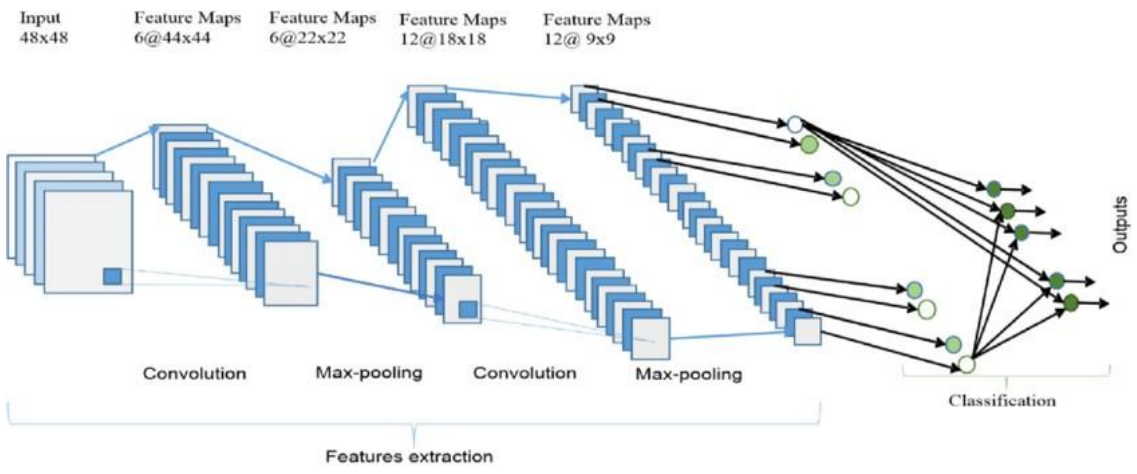


Fig. 4.1.2(a) CNN representation of Image feature extraction

For the Image Feature Extraction, we are using ResNet50, which is a Convolutional Neural Network used for the models based on Computer Vision. ResNet50 is a 50 layer deep Convolutional Neural Network, which can load a pretrained version of the network trained on more than a million images from the ImageNet Database. The images from Flickr8k database are fed to the ResNet50 model for feature extraction.

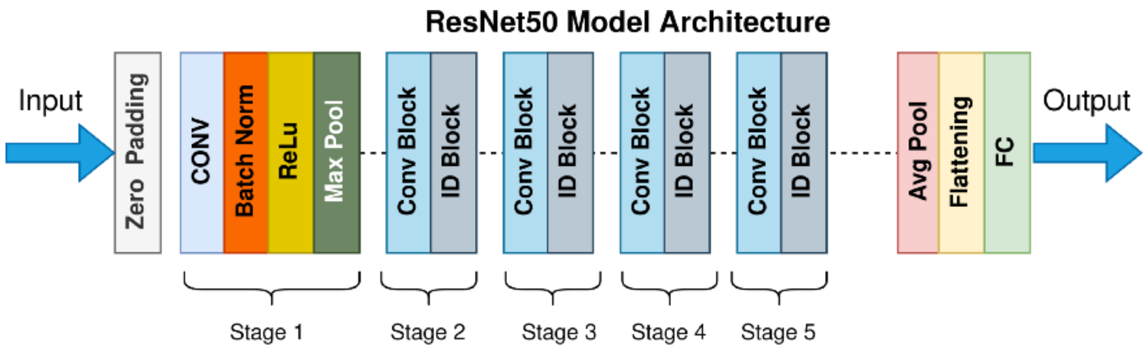
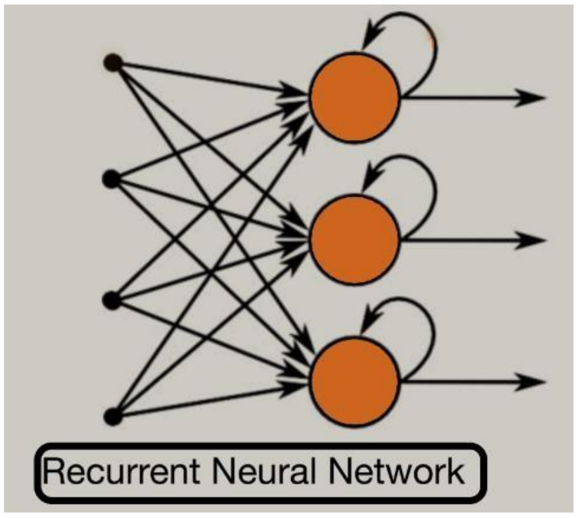


Fig. 4.1.2(b) ResNet50 Model Architecture

4.1.3 Caption Generation

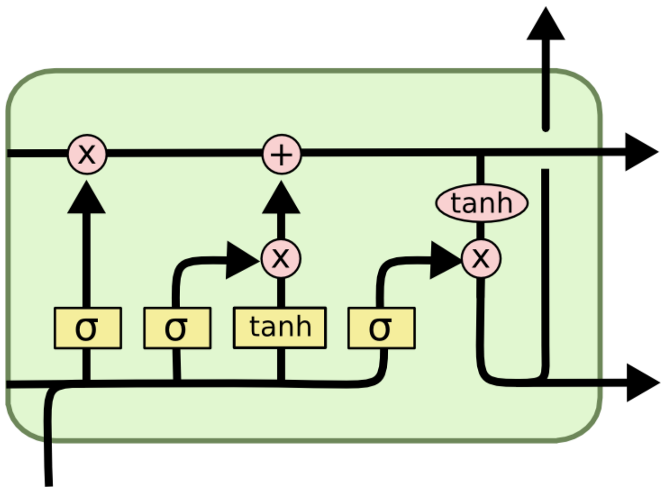
For the caption generation, we use RNN (Recurrent Neural Networks), which is used for text and audio processing. RNN is mainly used in the processing of natural language. RNN where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. Here we use a specific type of Recurrent Neural Network known as LSTM which stands for Long Short-Term Memory.

Fig. 4.1.3(a) Simple representation of RNN



The above figure shows the basic representation of Recurrent Neural Network. We have used LSTM which slightly differs from the above. The LSTM architecture consists of a “tanh” function in addition to the sigmoid function.

Fig. 4.1.3(b) LSTM Architecture



Deep Learning Model:

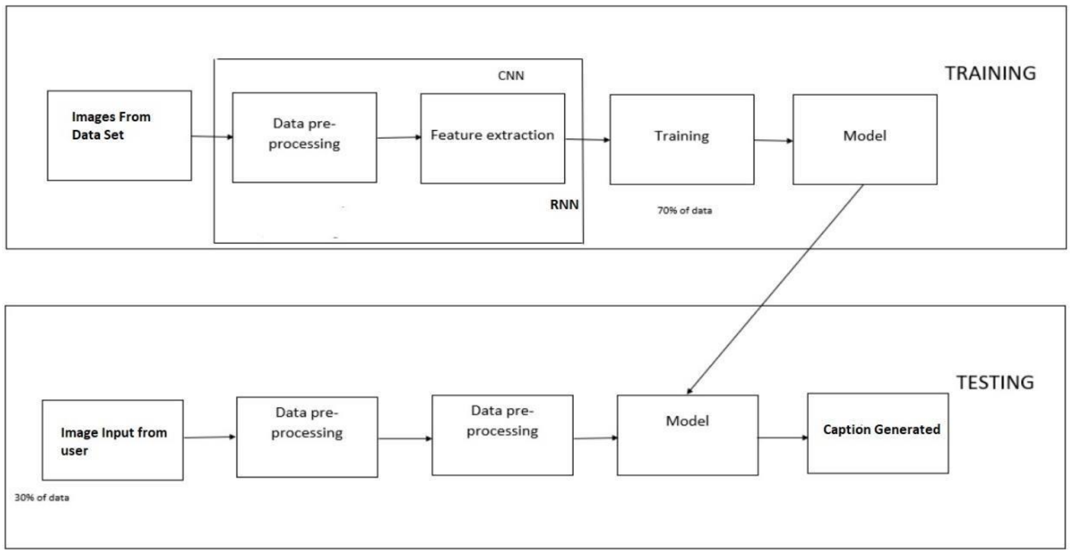


Fig. 4.1.4(a) Model for Training and Testing

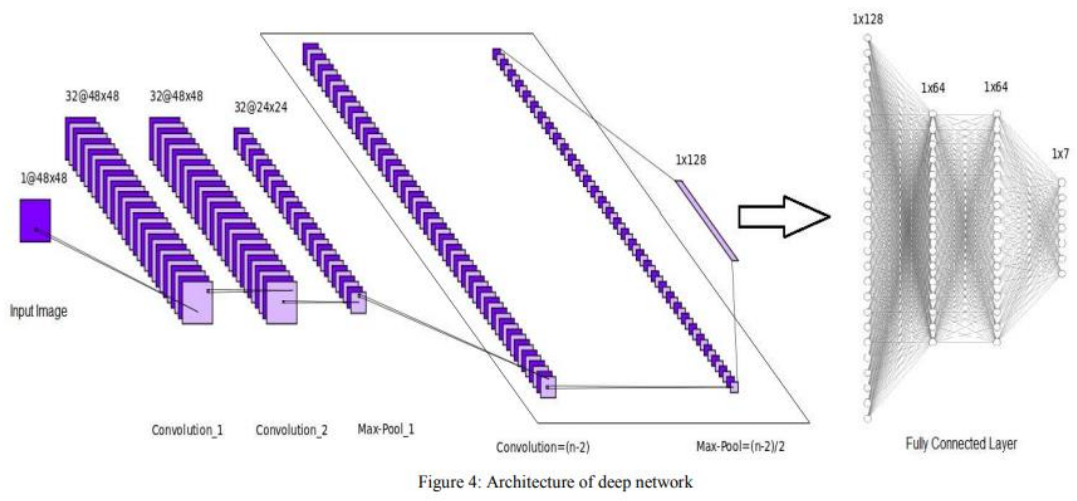


Figure 4.2.1: Architecture used for Emotion Detection

3D Model of 2D Images:

This program uses numpy-stl as the main library for functioning. When an image is given to the program, we use CV library to convert the given image to a greyscale image. When the greyscale image is produced it uses CV to analyze the intensity of the dark areas to stimulate the sense of depth.

Then using numpy-stl library the values are assigned in such a way that the dark spots have higher values which creates depth in the model making a 3D model of the given image. The 3D model is then saved in .stl format file which can be viewed using a 3D model viewer such as blender or Microsoft 3D Viewer.

CHAPTER 5 REQUIREMENTS

CHAPTER 5 REQUIREMENTS

The required Functional and Non-Functional Requirements are:

5.1. Functional Requirements:

1. The program must be able to take any images as inputs and generate captions.

2. The model must be able to train with any given dataset and provide accurate results while testing.

5.2. Non-Functional Requirements:

1. The program must not suffer any down time and be available when required by the user.

2. The program must be reliable and produce accurate results.

3. The program must easily maintainable and updatable at any given time.

4. The program must function as accurate as possible, irrespective of the size of the dataset used.

5. The program must be compatible with most of the systems, which are or might be used by the users.

5.3. Hardware Requirements:

The Minimum Hardware required for training and testing of the source code of the program are:

1. Intel i5 Core processor 2.4Ghz.

2. Nvidia GTX 960 4GB

3. 256 GB Solid State Drive

4. 8GB RAM

5.4. Software Requirements:

The software required for the program are:

1. Windows / Linux / Apple MAC OSX Operating System.

2. Python3 – preferably with Jupyter notebook.

3. TensorFlow Libraries.

4. ResNet50 Model library.

5. RNN/LSTM model Library

6. Numpy-Stl library

7. Blender or any other 3D Viewer.

CHAPTER 6
METHODOLOGY

CHAPTER 6 METHODOLOGY

Image Captioning:

Image captioning with ResNet50 involves a combination of computer vision and natural language processing techniques. ResNet50 is a deep convolutional neural network architecture that is trained on large amounts of image data and is capable of extracting features from an input image. The features extracted by ResNet50 are then fed into a second model, LSTM, that generates a natural language description or caption of the image.

The general procedure of the model is as follows:

1. Input image: The image captioning system takes an image as input.

2. Image feature extraction: The ResNet50 network is used to extract features from the input image. This network has already been trained on a large image dataset, so it can identify important features in an image and extract them for use in the next step.

3. Encoding of image features: The extracted image features are then fed into an encoding layer, where they are processed and transformed into a dense vector representation.

4. Decoding: The dense vector is then fed into a decoder, typically an RNN or LSTM, which generates a sequence of words that form the caption for the image. The decoder uses the encoded image features to generate a sequence of words that are most likely to describe the content of the image.

5. Output: The output of the decoder is a sequence of words that form a coherent and descriptive caption for the image

Existing data

We have used Flickr8k dataset which consists of eight thousand images with preloaded captions which makes it easier to train. We can use any other dataset of our choice for the model for training and testing.

Code Snippets:

Importing ResNet50:

```
from tensorflow.python.keras.applications.resnet import ResNet50
incept_model = ResNet50(include_top=True)
```

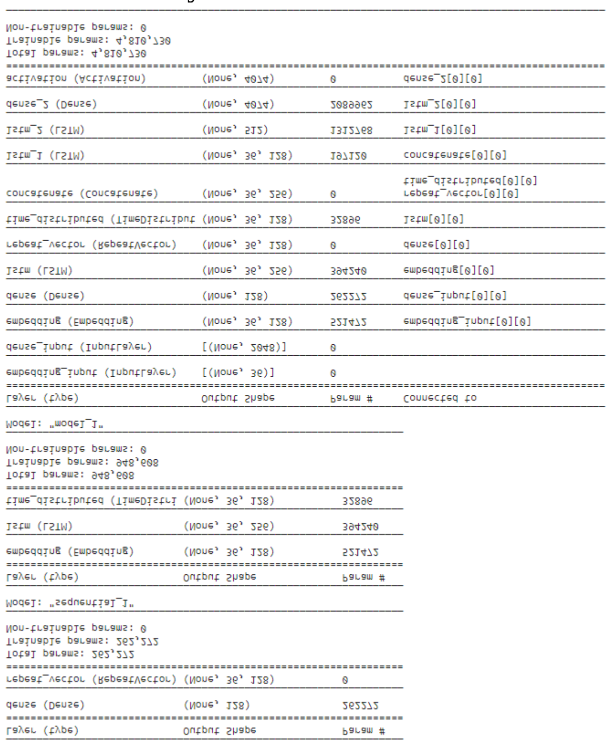
Reading features and resizing the images:

```
images_features = {}
count = 0
for i in images:
    img = cv2.imread(i)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224,224))

    img = img.reshape(1,224,224,3)
    pred = modele.predict(img).reshape(2048,)

    img_name = i.split('/')[-1]

    images_features[img_name] = pred

    count += 1

    if count > 1499:
        break

    elif count % 50 == 0:
        print(count)
```

Creating a caption dictionary:

```
captions_dict = {}
for i in captions:
    try:
        img_name = i.split('\t')[0][:-2]
        caption = i.split('\t')[1]
        if img_name in images_features:
            if img_name not in captions_dict:
                captions_dict[img_name] = [caption]

            else:
                captions_dict[img_name].append(caption)

    except:
        pass
```

Model Summary:



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               262272
_________________________________________________________________
repeat_vector (RepeatVector) (None, 36, 128)           0
=================================================================
Total params: 262,272
Trainable params: 262,272
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 36, 128)           521472
_________________________________________________________________
lstm (LSTM)                  (None, 36, 256)           394240
_________________________________________________________________
time_distributed (TimeDistri (None, 36, 128)           32896
=================================================================
Total params: 948,608
Trainable params: 948,608
Non-trainable params: 0
_________________________________________________________________
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
embedding_input (InputLayer)    [(None, 36)]         0
__________________________________________________________________________________________________
dense_input (InputLayer)        [(None, 2048)]       0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 36, 128)      521472      embedding_input[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          262272      dense_input[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 36, 256)      394240      embedding[0][0]
__________________________________________________________________________________________________
repeat_vector (RepeatVector)    (None, 36, 128)      0           dense[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, 36, 128)      32896       lstm[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 36, 256)      0           repeat_vector[0][0]
                                                                 time_distributed[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 36, 128)      197120      concatenate[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 512)          1312768     lstm_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 4074)         2089962     lstm_2[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 4074)         0           dense_2[0][0]
==================================================================================================
Total params: 4,810,730
Trainable params: 4,810,730
Non-trainable params: 0
__________________________________________________________________________________________________
```

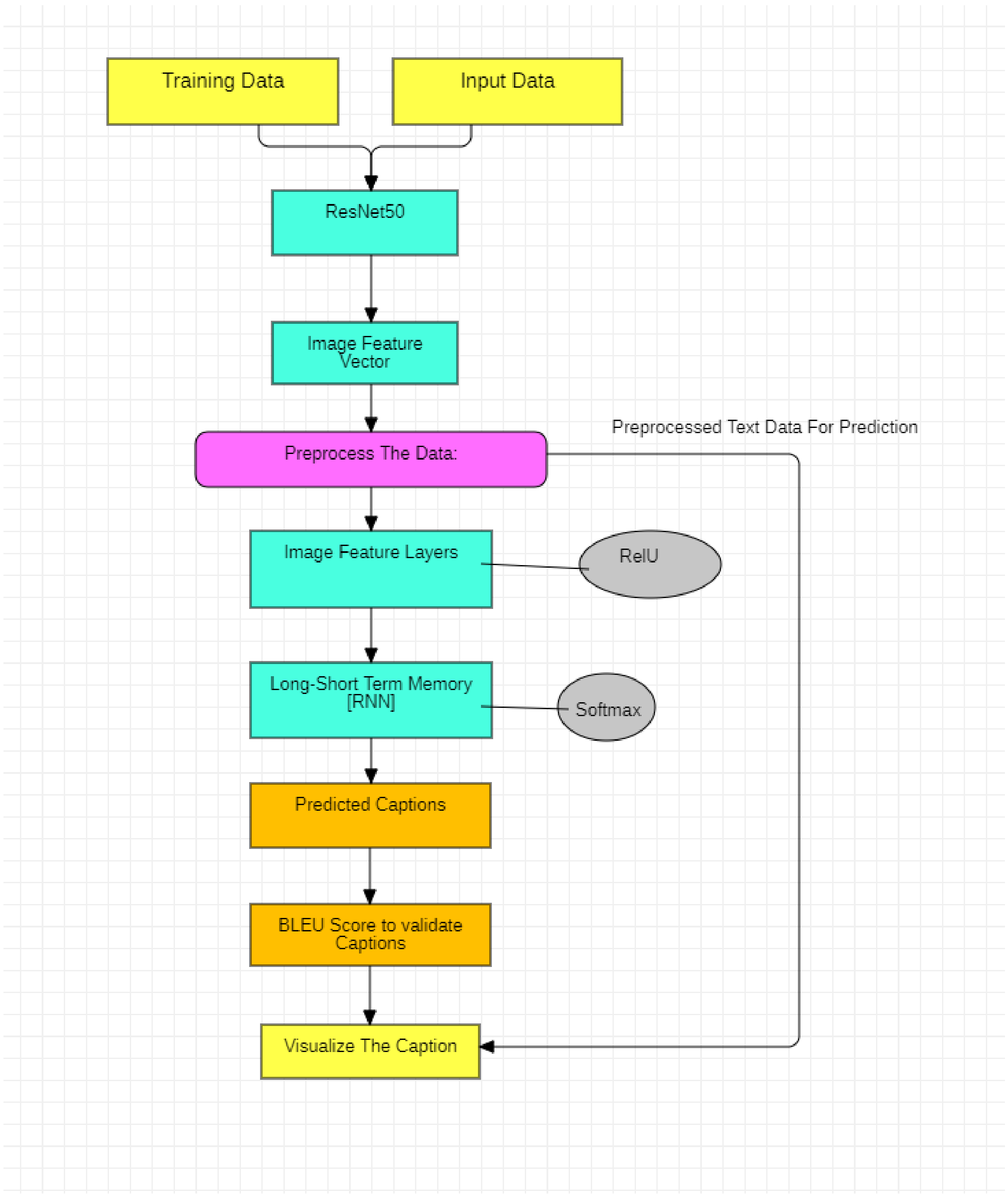


Fig 6.1: Data Flow of The Model

Emotion Detection:

Emotion detection and characterization using facial features is a technique that uses computer vision and machine learning algorithms to analyze facial expressions and identify emotions. The methodology involves several steps:

1. Face detection: In this step, the algorithm detects the faces in an image or video frame. This can be done using various techniques, such as Haar cascades, Viola-Jones algorithm, or deep learning-based methods.

2. Facial feature extraction: Once the face is detected, the algorithm extracts various facial features, such as the position of the eyes, eyebrows, mouth, nose, and the shape of the face. This can be done using landmark detection algorithms, which identify specific points on the face.

3. Feature representation: In this step, the extracted features are represented in a form that can be used by a machine learning algorithm. This can be done by encoding the features as numerical values or using feature descriptors, such as Local Binary Patterns (LBP) or Histogram of Oriented Gradients (HOG).

4. Emotion classification: In this step, the machine learning algorithm is trained on a dataset of labelled facial expressions to recognize different emotions, such as happiness, sadness, anger, fear, surprise, and disgust. The algorithm can be trained using various techniques, such as Support Vector Machines (SVM), Random Forests, or Convolutional Neural Networks (CNN).

5. Emotion characterization: In this step, the algorithm characterizes the detected emotion by analysing the intensity, duration, and context of the facial expression. This can provide additional information about the emotional state of the person, such as whether they are experiencing a mild or intense emotion, or whether the emotion is spontaneous or deliberate.

Human emotion recognition using Convolutional Neural Network (CNN) in real-time is a popular research area in computer vision and deep learning. The methodology for this task involves the following steps:

1. Data Collection: The first step is to collect a dataset of facial expressions. This can be done by capturing images or videos of people displaying different emotions such as happiness, sadness, anger, surprise, disgust, and fear.
2. Data Pre-processing: The collected data needs to be pre-processed before feeding it to the CNN model. This involves tasks such as cropping the face region from the images, resizing the images, and normalizing the pixel values.
3. CNN Model Architecture: The CNN model architecture for emotion recognition typically consists of multiple convolutional layers followed by a few fully connected layers. The convolutional layers extract features from the input images, while the fully connected layers classify the emotions.
4. Training the Model: The pre-processed data is split into training and testing sets. The training set is used to train the CNN model by optimizing its weights using backpropagation and gradient descent. The testing set is used to evaluate the performance of the trained model.
5. Real-Time Emotion Recognition: Once the CNN model is trained, it can be used for real-time emotion recognition. This involves capturing a live video stream from a camera and using the trained CNN model to classify the emotions displayed by the person in the video.

3D Model of 2D Images:

This program uses numpy-stl as the main library for functioning. When an image is given to the program, we use CV library to convert the given image to a greyscale image. When the greyscale image is produced it uses CV to analyze the intensity of the dark areas to stimulate the sense of depth.

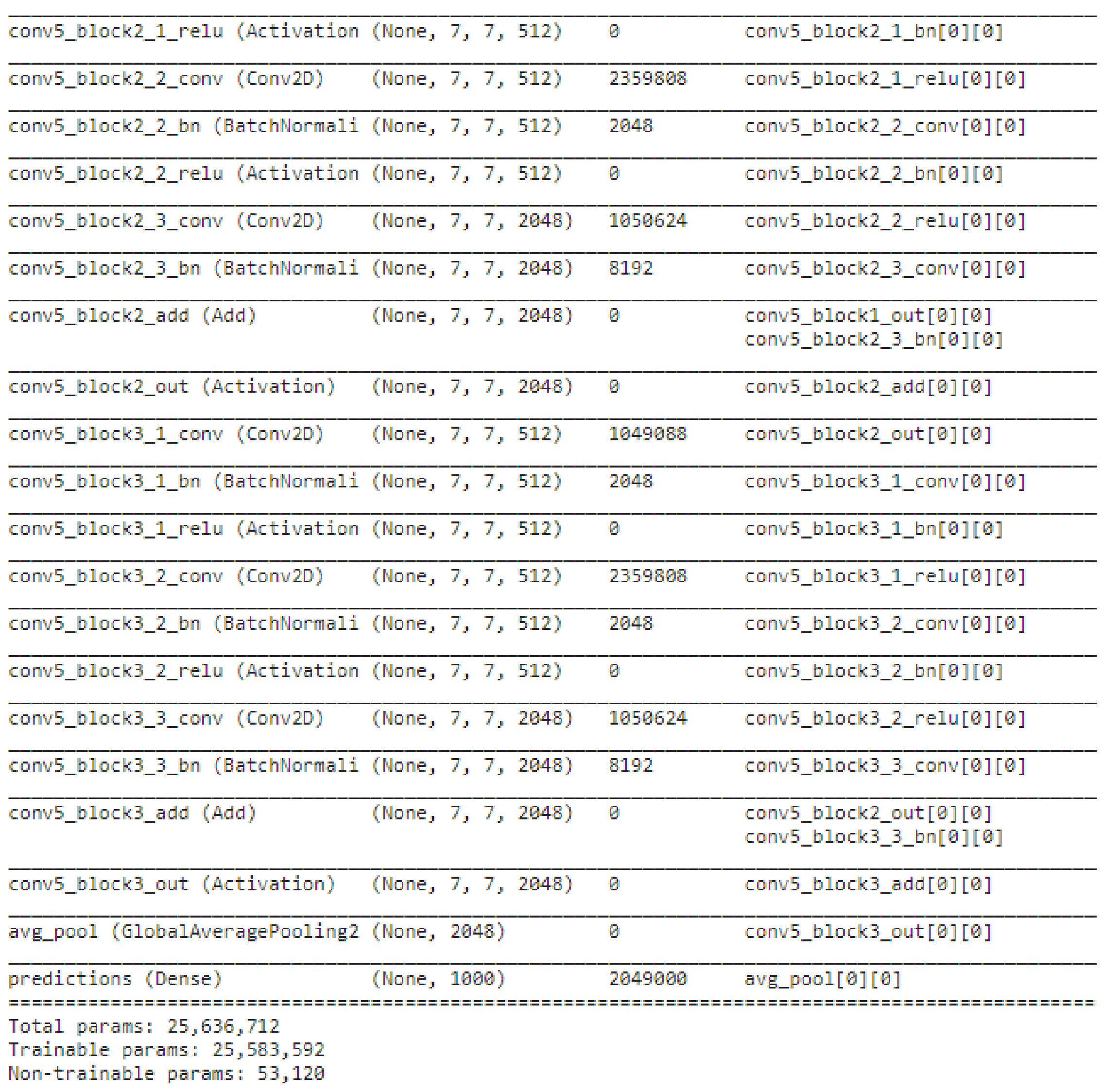
Then using numpy-stl library the values are assigned in such a way that the dark spots have higher values which creates depth in the model making a 3D model of the given image. The 3D model is then saved in .stl format file which can be viewed using a 3D model viewer such as blender or Microsoft 3D Viewer.

CHAPTER 7
EXPERIMENTATION

CHAPTER 7 EXPERIMENTATION

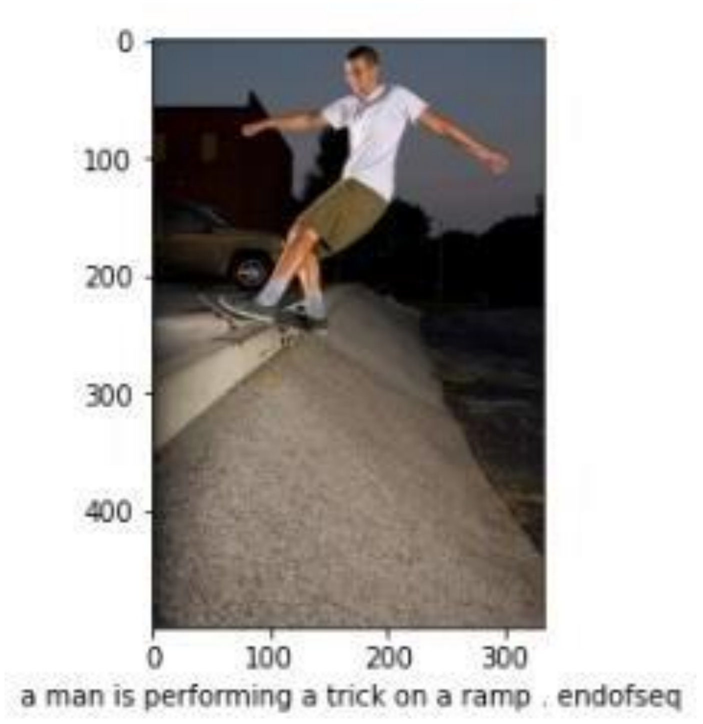
Snippet of ResNet50 model Summary:

fig 7.1: Model Summary



```
__________________________________________________________________________________________________
conv5_block2_1_relu (Activation (None, 7, 7, 512)    0           conv5_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_2_relu (Activation (None, 7, 7, 512)    0           conv5_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_add (Add)          (None, 7, 7, 2048)   0           conv5_block1_out[0][0]
                                                                 conv5_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_out (Activation)   (None, 7, 7, 2048)   0           conv5_block2_add[0][0]
__________________________________________________________________________________________________
conv5_block3_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_block2_out[0][0]
__________________________________________________________________________________________________
conv5_block3_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_1_relu (Activation (None, 7, 7, 512)    0           conv5_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 7, 7, 512)    0           conv5_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           conv5_block3_out[0][0]
__________________________________________________________________________________________________
predictions (Dense)             (None, 1000)         2049000     avg_pool[0][0]
==================================================================================================
Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120
```

PREDICTION:



```
a man is performing a trick on a ramp . endofseq
```

FIG 7.2: PREDICTED RESULT

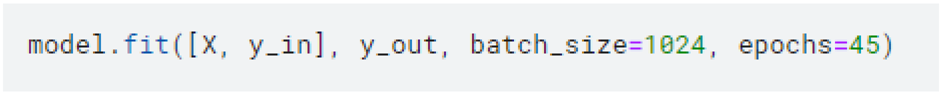
CHAPTER 8

TESTING AND RESULTS

CHAPTER 8 TESTING AND RESULTS

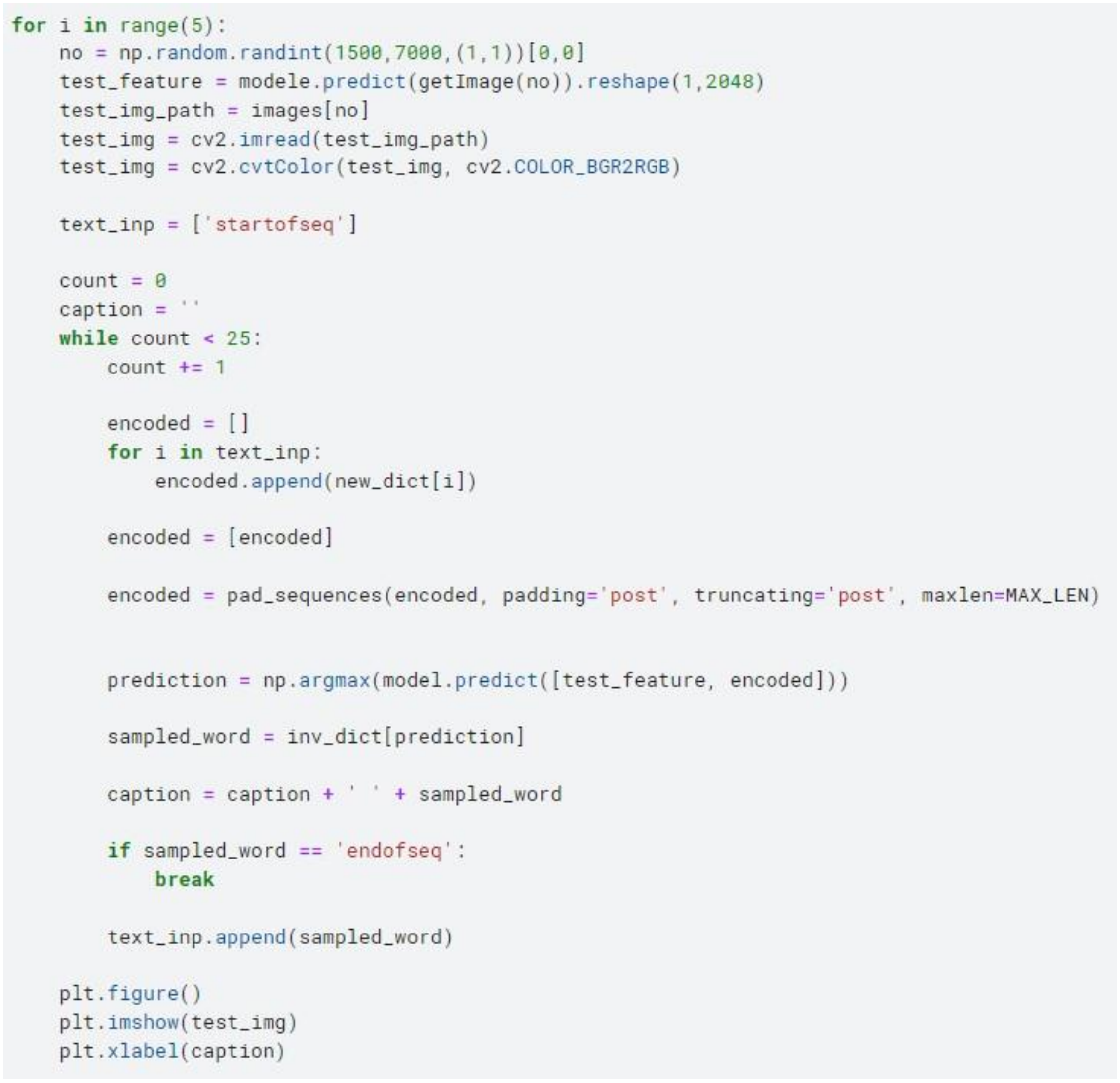
We have tested the model with the following conditions of 45 epochs and 1024 batch size to get the following results. The epoch test provided us with an accuracy of 70%, we further are planning to train and test the model with different datasets and different iterations of test conditions.

Epoch testing:

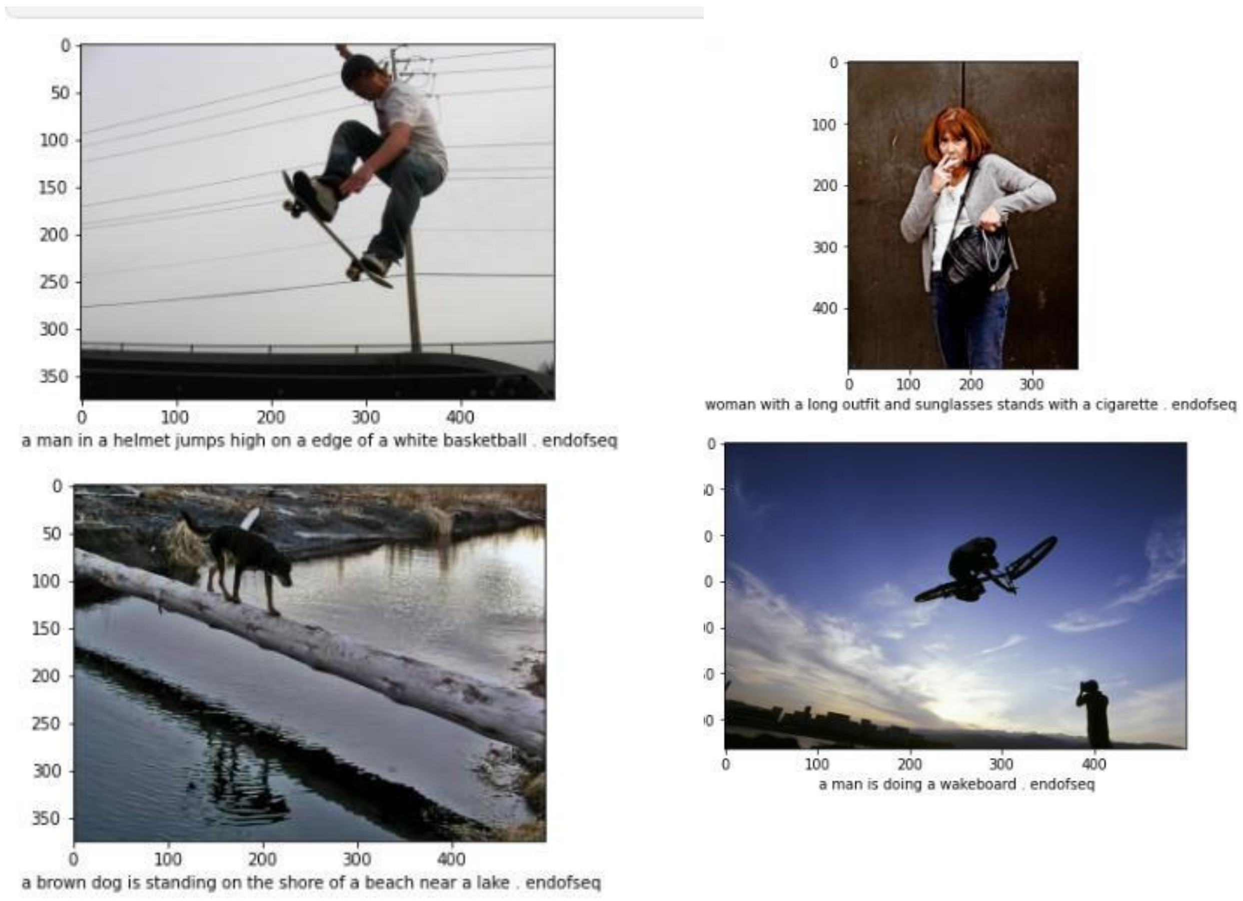


```
model.fit([X, y_in], y_out, batch_size=1024, epochs=45)
```

Prediction of random images from data set:



```
for i in range(5):
    no = np.random.randint(1500,7000,(1,1))[0,0]
    test_feature = modele.predict(getImage(no)).reshape(1,2048)
    test_img_path = images[no]
    test_img = cv2.imread(test_img_path)
    test_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)

    text_inp = ['startofseq']

    count = 0
    caption = ''
    while count < 25:
        count += 1

        encoded = []
        for i in text_inp:
            encoded.append(new_dict[i])

        encoded = [encoded]

        encoded = pad_sequences(encoded, padding='post', truncating='post', maxlen=MAX_LEN)


        prediction = np.argmax(model.predict([test_feature, encoded]))

        sampled_word = inv_dict[prediction]

        caption = caption + ' ' + sampled_word

        if sampled_word == 'endofseq':
            break

        text_inp.append(sampled_word)

    plt.figure()
    plt.imshow(test_img)
    plt.xlabel(caption)
```



a man in a helmet jumps high on a edge of a white basketball . endofseq

woman with a long outfit and sunglasses stands with a cigarette . endofseq

a man is doing a wakeboard . endofseq

a brown dog is standing on the shore of a beach near a lake . endofseq

Fig 8.1: Resulted Prediction

Emotion Detection:

We have tested the emotion detection model with varying numbers of epochs and batch sizes we have used about 20 epochs along with the batch size of 1024 to obtain 80% accuracy which works fairly well when tested against a video involving human emotions.

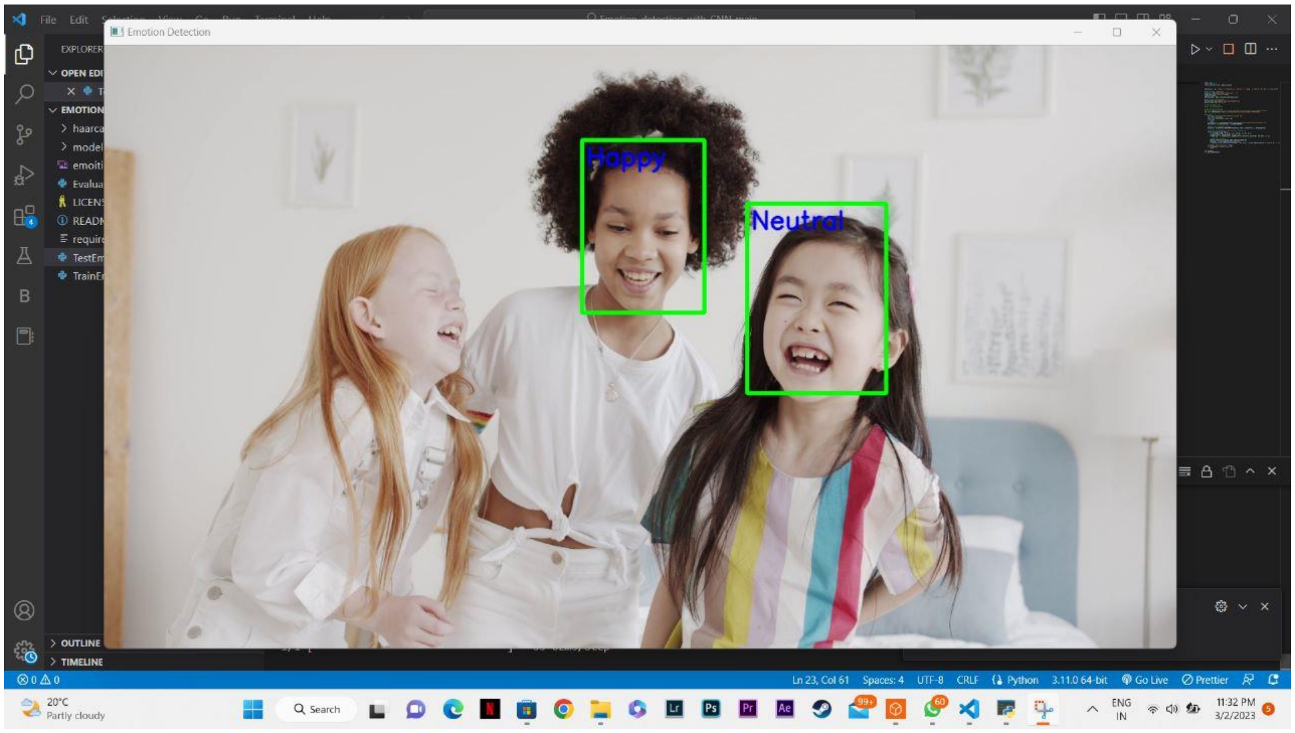


Fig 8.2: Result predicted for Emotion Detection

3D model of 2D Images:

When tested the program against multiple images we had satisfactory results produced which were 3d model of the images in an .stl file.

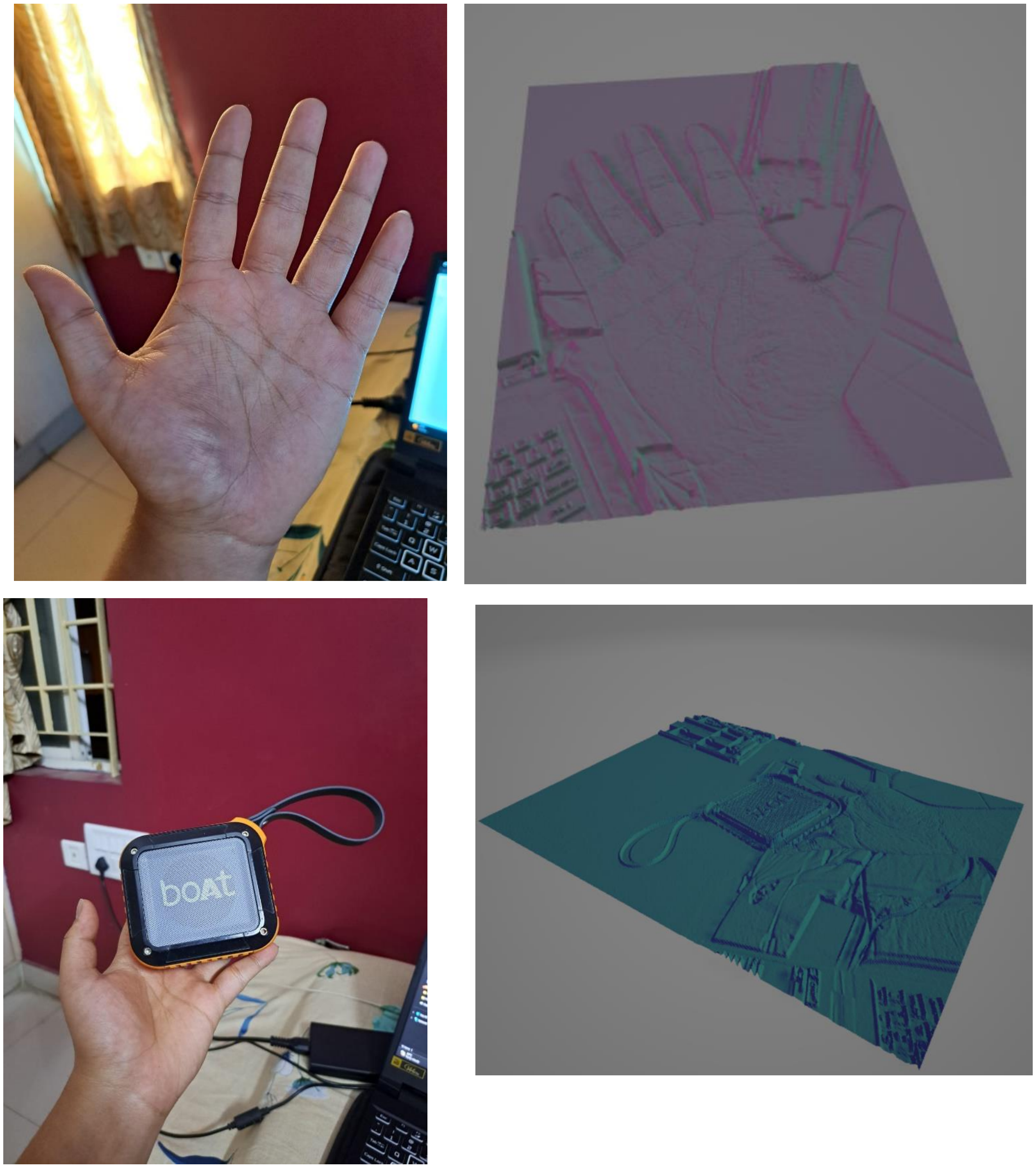


Fig 8.2: Conversion images of 2D images to 3D models.

CHAPTER 9 CONCLUSION AND FUTURE WORK

Conclusion:

From the conducted work involving three Image Based Applications, two of which are deep learning models and one is a computational program, we have acquired near satisfactory results of accuracy and predictions. Image Captioning model provides an accuracy of 70%, but the predictions are still not accurate for all given images and emotion detection provides 80% accuracy with satisfactory results. The 3D model of 2D image conversion program provides with a satisfactory depth of the model which can be useful for certain applications.

Future Work:

We would like to carry-out the following work before the final review:

- Make a user interface using a library like Streamlit, for the user to upload and use their own images.
- Make any changes that might increase the accuracy of the above Image Based applications.

CHAPTER 10 REFERENCES

REFERENCES

[1] J Sudhakar; Viswesh V Iyer; Sree T Sharmila (2022). " Image Caption Generation using Deep Neural Networks," in IEEE Journal; 2022 International Conference for Advancement in Technology (ICONAT) , 2022.

[2] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan “Show and Tell: A Neural Image Caption Generator” in IEEE Journal, Computer Vision Foundation, 2021.

[3] N. Komal Kumar; D. Vigneswari; A. Mohan; K. Laxman; J. Yuvaraj. Detection and Recognition of Objects in Image Caption Generator System: A Deep Learning Approach,IEEE Journal 2019.

[4] Chetan Amritkar; Vaishali Jabade, Image Caption Generation Using Deep Learning Technique, IEEE Journal 2018.

[5] Wang, Ziwei, Zi Huang, and Yadan Luo. "Human consensus-oriented image captioning." Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. 2021.

[6] Vaishnavi Agrawal, Shariva Dhekane, Neha Tuniya, Vibha Vyas “ ImageCaption generator using Attention mechanism” in IEEE Journal, ICCCNT, 2021.

[7] Prachi Waghmare, Swati Shinde, Jayshree Katti, “Image Captioning usingNeural Network Model” in IEEE Journal, ICSTSN, 2022.

[8] Sai Vikram Patnaik; Rohi Mukka; Roy Devpreyo; Ankita

Wadhawan “ ImageCaption Generator using EfficientNet”, in IEEE Journal, 2022.

Code Snippets:

Image Captioning Model Summary:

```
embedding_size = 128
max_len = MAX_LEN
vocab_size = len(new_dict)

image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))

image_model.summary()

language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size, input_length=max_len))
language_model.add(LSTM(256, return_sequences=True))
language_model.add(TimeDistributed(Dense(embedding_size)))

language_model.summary()
conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)
model.compile(loss='categorical_crossentropy', optimizer='RMSprop', metrics=['accuracy'])
model.summary()
```

Emotion Detection Training:

```
# import required packages
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
from tensorflow.keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator

# Initialize image data generator with rescaling
train_data_gen = ImageDataGenerator(rescale=1./255)
validation_data_gen = ImageDataGenerator(rescale=1./255)

# Preprocess all test images
train_generator = train_data_gen.flow_from_directory(
        'train',
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

# Preprocess all train images
```

```
validation_generator = validation_data_gen.flow_from_directory(
        'test',
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

# create model structure
emotion_model = Sequential()

emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48, 48, 1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))

cv2.ocl.setUseOpenCL(False)

emotion_model.compile(loss='categorical_crossentropy',
optimizer=Adam(lr=0.0001, decay=1e-6), metrics=['accuracy'])

# Train the neural network/model
emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=50,
        validation_data=validation_generator,
        validation_steps=7178 // 64)

# save model structure in jason file
model_json = emotion_model.to_json()
with open("emotion_model.json", "w") as json_file:
    json_file.write(model_json)
```

```
# save trained model weight in .h5 file
emotion_model.save_weights('emotion_model.h5')
```

Emotion Detection Testing:

```
import cv2
import numpy as np
from keras.models import model_from_json


emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4:
"Neutral", 5: "Sad", 6: "Surprised"}

# load json and create model
json_file = open('model/emotion_model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
emotion_model = model_from_json(loaded_model_json)

# load weights into new model
emotion_model.load_weights("model/emotion_model.h5")
print("Loaded model from disk")

# start the webcam feed
cap = cv2.VideoCapture(0)

# pass here your video path
# you may download one from here : https://www.pexels.com/video/three-
girls-laughing-5273028/
#cap = cv2.VideoCapture("pexels-artem-podrez-6952256.mp4")

while True:
    # Find haar cascade to draw bounding box around face
    ret, frame = cap.read()
    frame = cv2.resize(frame, (1280, 720))
    if not ret:
        break
    face_detector =
cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_default.xml')
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces available on camera
    num_faces = face_detector.detectMultiScale(gray_frame,
scaleFactor=1.3, minNeighbors=5)
```

```
    # take each face available on the camera and Preprocess it
    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (0, 255, 0), 4)
        roi_gray_frame = gray_frame[y:y + h, x:x + w]
        cropped_img = 
np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -1), 
0)

        # predict the emotions
        emotion_prediction = emotion_model.predict(cropped_img)
        maxindex = int(np.argmax(emotion_prediction))
        cv2.putText(frame, emotion_dict[maxindex], (x+5, y-20), 
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)

    cv2.imshow('Emotion Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

3D Model of 2D images Conversion:

```
grey_img = Image.open('C:\\Users\\hellm\\OneDrive\\Desktop\\Final 
Proj\\face.jpeg').convert('L')

max_size=(500,500)
max_height=10
min_height=0

#height=0 for minPix
#height=maxHeight for maxPIx

grey_img.thumbnail(max_size)
imageNp = np.array(grey_img)
maxPix=imageNp.max()
minPix=imageNp.min()



print(imageNp)
(ncols,nrows)=grey_img.size

vertices=np.zeros((nrows,ncols,3))
```

```
for x in range(0, ncols):
  for y in range(0, nrows):
    pixelIntensity = imageNp[y][x]
    z = (pixelIntensity * max_height) / maxPix
    #print(imageNp[y][x])
    vertices[y][x]=(x, y, z)

faces=[]

for x in range(0, ncols - 1):
  for y in range(0, nrows - 1):
    # create face 1
    vertice1 = vertices[y][x]
    vertice2 = vertices[y+1][x]
    vertice3 = vertices[y+1][x+1]
    face1 = np.array([vertice1,vertice2,vertice3])

    # create face 2
    vertice1 = vertices[y][x]
    vertice2 = vertices[y][x+1]
    vertice3 = vertices[y+1][x+1]

    face2 = np.array([vertice1,vertice2,vertice3])

    faces.append(face1)
    faces.append(face2)

print(f"number of faces: {len(faces)}")
facesNp = np.array(faces)
# Create the mesh
surface = mesh.Mesh(np.zeros(facesNp.shape[0], dtype=mesh.Mesh.dtype))
for i, f in enumerate(faces):
    for j in range(3):
        surface.vectors[i][j] = facesNp[i][j]
# Write the mesh to file "cube.stl"
surface.save('C:\\Users\\hellm\\OneDrive\\Desktop\\Final Proj\\face.stl')
print(surface)
```