SPRINT -2

|  |  |
| --- | --- |
| DATE | 21 October 2022 |
| TEAM ID | PNT2022TMID30496 |
| PROJECT NAME | Real time communication system powered by AI specially abled |

**Image Preprocessing**

In this, we will pre-process the images which will be used for building the model. Image pre-processing includes zooming, shearing, flipping to increase the robustness of the model after it is built. We will be using the Keras package for pre-processing images.

**Import ImageDataGenerator Library And Configure It**

Import ImageDataGenerator and create an instance for which include shearing, rescale, zooming, etc to make the model robust with different types of images.

**Apply ImageDataGenerator Functionality To Train And Test Set**

Specify the path of both the folders in the flow\_from\_directory method.

Flow from directory loads the images from a given directory and can bring all the images to the target size. We will be loading all the images of the train and test using the flow from directory method.

**Model Building**

**In this milestone, we start building our model by:**

1.Initializing the model

2. Adding Convolution layers

3. Adding Pooling layers

4. Flatten layer

5. Full connection layers which include hidden layers

At last, we compile the model with layers we added to complete the neural network structure

**Import The Required Model Building Libraries**

Import the libraries that are required to initialize the neural network layer, create and add different layers to the neural network model.

**Initialize The Model**

Initialize the neural network layer by creating a reference/object to the Sequential class.

**Add The Convolution Layer**

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes the number of feature detectors, feature detector size, expected input shape of the image, activation function as arguments. This layer applies feature detectors on the input image and returns a feature map (features from the image).

**Add The Pooling Layer**

After the convolution layer, usually, the pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. The efficient size of the pooling matrix is (2,2). It returns the pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can be added)

**Add The Flatten Layer**

The flatten layer is used to convert the n-dimensional array to a 1-dimensional array. This 1D array will be given as input to ANN layers.

**Adding The Dense Layers**

Three dense layers are added which usually takes the number of units/neurons. Specifying the activation function, kind of weight initialization is optional.

**Compile The Model**

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer, and metrics for evaluation can be passed as arguments.

**Fit And Save The Model**

Fit the neural network model with the train and test set, number of epochs, and validation steps.

The weights are to be saved for future use. The weights are saved in as .h5 file using save().

**Test The Model**

Now we test the model by passing an image to get predictions. While test the model we should make sure that the test image should meet the target size of the model, dimensions need to meet, and should undergo rescaling before giving it to the model.

**Import The Packages And Load The Saved Model**

As a first step to start prediction we import packages that are used for loading the model and used to expand the dimension of the image. We use the Keras package to load the model which was saved when we built the model.

**Load The Test Image, Pre-Process It And Predict**

Pre-processing the image includes converting the image to the array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

**Application Building**

Now we will be building a Flask application that is used for building our UI which in backend can be interfaced to the model to get predictions. Flask application requires an HTML page for Frontend and a Python file for the backend which takes care of the interface with the model.

**Build A Flask Application**

**Step 1:**Load the required packages

**Step 2:** Initialize graph, load the model, initialize the flask app and load the video

Graph element is required to work with tensorflow. So, graph element is created explicitly.

**Step 3:**Configure the home page

. 

**Building Flask Application -Part 3**

Each frame is taken from the camera and processed and sent to the model for prediction. As discussed image undergoes different processing steps to meet model requirements to get predictions.

This below in the snippet for calling video feed from the HTML page.

**Build The HTML Page**

Build an HTML page to display the processed video on the screen, so that the person can show signs which can be detected.

Run the application by going to the location of your program and run

python webstreaming.py

When the python file is executed the localhost is activated on 5000 port and can be accessed through it.

**Output**

**Step 2:**Open the browser and navigate to localhost:5000 to check your application