REPORT

Submitted by: Ashutosh Soni

Email Id: ashutoshsoni0699@gmail.com

Introduction:

We have to make an application using pretrained model of resnet-18 and having an UI where we can upload image as well as we can take image from camera and accordingly, we have to predict the output using the above pretrained model as well as we have to dockerize the app so that we can run it without installing all the dependencies.

Technologies Used:

1. Python
2. HTML
3. CSS
4. Boot-strap

Implementation and Design:

1. I used python **torch-vision** for importing the pre-trained model of resnet-18 trained under image-net dataset.
2. Then I applied some sort of **image-processing** like resizing the image and transforming the image into tensor and some others transforms so that the testing image will fit into our model and so that model can predict the image.
3. Then I make a test file containing the output classes (1000 classes) of the model on which dataset it is trained on and I save it for further use.
4. Then I made a **predict_label** function for predicting the output of the image inputted from the UI.
5. I use cv2 module for taking the image from the camera and then I save it in a directory and then predict the output label form predict_label function made earlier. This all I club under **take_picture** function.
6. I also used **flask** web framework which provides libraries to build lightweight web applications in python using flask I connected the render page i.e., HTML page.
7. Then after I designed a **HTML** page for giving user the Interface from which it can perform the above task of prediction. I also used **CSS** for styling and **Boot-strap** for making the website responsive.
8. After that from the main file I run this HTML file and our application is ready.
9. As of now I used localhost in order to render the page.
10. For saving the image I use local storage for saving the image and then performing our desired task. I save the image as I Think we may use these images for later use.

Specification required:

1. Admin prospective: We have to host this application on cloud like amazon cloud or some else and we have sufficient amount of RAM as well as GPU if not then CPU having sufficient capacity in order to run our application and then our application is ready to use.
2. User prospective: User have browser installed in his/her device in order to use this application.

Result:

I used localhost to run this application and got the desired result. I was able to predict the image by both taking image as input from user as well as taking image from the web camera of the laptop. In both the cases I got the desired result that I have to get. As of now as I had single user and single system so there is no need to install the dependencies and requirements every time It is enough to install and download all at once. I was not able to dockerize the app as I have only theoretical knowledge of the same.

Future Work:

As I have only theoretical knowledge of docker so I was not able to dockerize this application as we can further optimize the code as we used python here for rendering and doing our task, we can use some other framework which can take less time to do the same work. As well as in UI prospective we can use some better framework in order to make our UI cleaner and more attractive. We can I also use backend for saving the images in case our model is trained on them and improve our model as of now I don't think it is of use.

Conclusion:

Our application is ready to be used as if user and admin have above required set of hardware and software. Our application is ready to predict the image by both way taking the image from the user as well as taking the picture of the object which we have to determine.

We can successfully get the desired result of the object we are predicting and I was able to make a user interface form where the user the achieve the desired task. In our UI we have given the user significant information about the page so that he or she can perform the task it needed.

References:

1. https://pytorch.org/vision/stable/models.html