A Mini Project on

"HUMAN ACTIVITY RECOGNITION"

Submitted as part of Image Processing course taken for partial fulfillment of B.Tech. (Computer Engineering)

By:

Sarvesh Agrawal	B006
Anmol Batra	B013
Naman Bhansali	B015
Yatharth Bijolia	B016

Under the Guidance of
Dr. Dhirendra S Mishra
Professor, Computer Engineering

SVKM's NMIMS (Deemed- to be University)
Mukesh Patel School of Technology Management & Engineering
Vile Parle West, Mumbai-56

SVKM's NMIMS Deemed –to be University
Mukesh Patel School of Technology Management and Engineering

Declaration of Originality

We, undersigning students of the MINI Project group hereby declare that we have worked on the mini project titled **" HUMAN ACTIVITY RECOGNITION "** under **Image Processing course** taken in semester V of our undergraduate B.Tech. Computer Engineering Program during **Academic year 2020-21**.

All contributions in this project are our own original contributions. To the best of our knowledge and belief this project report contains no material previously published or written by any another person, except where due acknowledgement has been made in the text.

Group Member 1	**Group Member 2**	**Group Member 3**	**Group Member 4**
Signature: [illegible]	Signature: [illegible]	Signature: Naman	Signature: Yatharth
Name: Sarvesh Agrawal	Name: Anmol Batra	Name: Naman Bhansali	Name: Yatharth Bijolia
SAP ID: 70021018004	SAP ID: 70021018009	SAP ID: 70021018011	SAP ID: 70021018013

SVKM's NMIMS Deemed –to be University
Mukesh Patel School of Technology Management and Engineering

Certificate of Completion

This is to declare that **Sarvesh Agrawal, Anmol Batra, Naman Bhansali and Yatharth Bijolia** have worked together in a team to complete a **Mini Project** as the part of **Image processing course** in **semester V (Division B) of their B. Tech Computer Engineering Program** during **Academic year 2020-21.**

Their performance in the project has been found to be excellent.

Signature of Course in-charge faculty
Dr. Dhirendra S. Mishra

Signature of Examiner
(   )

Head of the Department
Dr. Pravin Shrinath

INDEX

Chapter	Title	Page Number
1	Abstract	5
2	Introduction	6
3	Problem Definition	7
4	Literature Survey	8
5	Proposed Solution	9
5.1	Idea with overall block diagram of the system	9
5.2	Detailed explanation of algorithmic steps involved in each step	10
5.3	Database Designed	20
5.4	Metrics Performance	22
6	Implementation	25
6.1	Contribution of each group member	25
6.2	Graphical User interface screen shots / all possible input images and its corresponding output images obtained at various stages of processing.	27
6.3	Working code	28
7	Conclusion	29
8	Limitation and Future Scope	30
9	List of References [in IEEE Format]	31

ABSTRACT

Human activity recognition (HAR) is a widely studied computer vision problem. Human Activity recognition aims to recognize the actions and goals of a person from a series of observations on the persons actions and the environmental conditions. Applications of HAR include video surveillance, health care, and human-computer interaction. As the imaging technique advances and the camera device upgrades, novel approaches for HAR constantly emerge. This project aims to provide a comprehensive introduction to the video-based human activity recognition, giving an overview of various approaches as well as their evolutions by covering both the representative classical literatures and the state-of-the-art approaches.

In recent years, an extensive research has been conducted in this field. This project picks this problem and recognises what a human is doing in a video. This data is very crucial now in this age of digital medium. It contributes significantly to various domains and its scope goes on increasing. From healthcare and security to the science of understanding human behaviour, whether using it to transcript a movie by just providing the machine with a video or use it as the first step in the huge mechanism of building a robot and its application in AIML, we try to resolve and make the best possible and efficient model for detection of accurate human activities. Though it is just the beginning, it is an integral step towards the forementioned fields and more. Our objective is thus clear- to make an efficient trainable model which predicts accurate human activity in the given video. Performance comparisons between the proposed system and existing similar systems were also shown.

We have thoroughly explained our approach and each step of our making this model a success by getting a high accuracy on our test data and predicting the videos correctly. A major stepping stone we feel proud to accomplish is the deployment of our model online to make it reach to everyone. This have just been possible because of Neural Networks. We have used a block diagram to represent our thinking process and defined each stage of development. The dataset we used and how that dataset was distributed in our program is also displayed. The limitation to our hardware needs and also the payable difference for deployment in servers has restricted us to take only six activities for now. The performance metrics is measured by the accuracy of our model.

For references and understanding purposes only, we have attached the link to our code. This being only a part of a much bigger and parallelly run algorithms of the same kind integrated to predict multiple activities and run continuously, our model sure has its own limitations and thus giving us our future scope. We will try to continue it in future, but for now, given the hardware restrictions and time bound completion among others, we have completed this project successfully.

Keywords:- human activity recognition, AIML, deployment, predicting, model, Neural Network, metrics

INTRODUCTION

Human Activity recognition aims to recognize the actions and goals of a person from a series of observations on the persons actions and the environmental conditions. Recognizing human activities from video sequences or still images is a challenging task due to problems, such as background clutter, partial occlusion, changes in scale, viewpoint, lighting, and appearance. Many applications, including video surveillance systems and Robots, require a multiple activity recognition system.

In the broader spectrum, this application can also be used in robots for detecting human activities which are taking place in their field of vision. Everything the capture can be processed which will allow them to recognize various activities and interact with human more coherently. This will also help in increasing the efficiency of the robot and its knowledge about the human activities. This can be considered as Application of Artificial Intelligence.

Video surveillance systems are a system of one or more video cameras on a network that send the captured video and audio information to a certain place. The images are not available to the public like television. They are live monitored or transmitted to a central location for recording and storage. Human activity recognition plays a significant role in human-to-human interaction and interpersonal relations. Because it provides information about the identity of a person, their personality, and psychological state, it is difficult to extract. The human ability to recognize another person’s activities is one of the main subjects of study of the scientific areas of computer vision and machine learning.

Computer vision's goal is not only to see, but also process and provide useful results based on the observation. For example, a computer could create a 3D image from a 2D image, such as those in cars, and provide important data to the car and/or driver. For example, cars could be fitted with computer vision which would be able to identify and distinguish objects on and around the road such as traffic lights, pedestrians, traffic signs and so on, and act accordingly. When a human who is driving a car sees someone suddenly move into the path of the car, the driver must react instantly. In a split second, human vision has completed a complex task, that of identifying the object, processing data and deciding what to do. Computer vision's aim is to enable computers to perform the same kind of tasks as humans with the same efficiency. A camera By providing many video samples our system can identify if a driver is using his phone while driving for more than 3 seconds and send him a warning about upcoming four ways ,diversions or red lights .

PROBLEM STATEMENT

The project is an integral cog in Computer Vision and Artificial Intelligence and Machine learning. It aims to determine the activity of a human from a video provided to the machine. It is a step forward in solving various problems like surveillance, fall detection for elderly or sick people, robotics and computer interaction, security among many others. We want a higher accuracy in doing so with respect to the videos used for training the machine.

LITERATURE SURVEY

We compared our project the recently implemented project of Human Activity Recognition by Ms. Shikha, Rohan Kumar, Shivam Aggarwal and Shrey Jain which was implemented in May 2020.

Following are the key-points which we observed in their project:

- They used the RESNET-34 CNN Model which is the predecessor of what we have used in our project that is RESNET50v2 model.

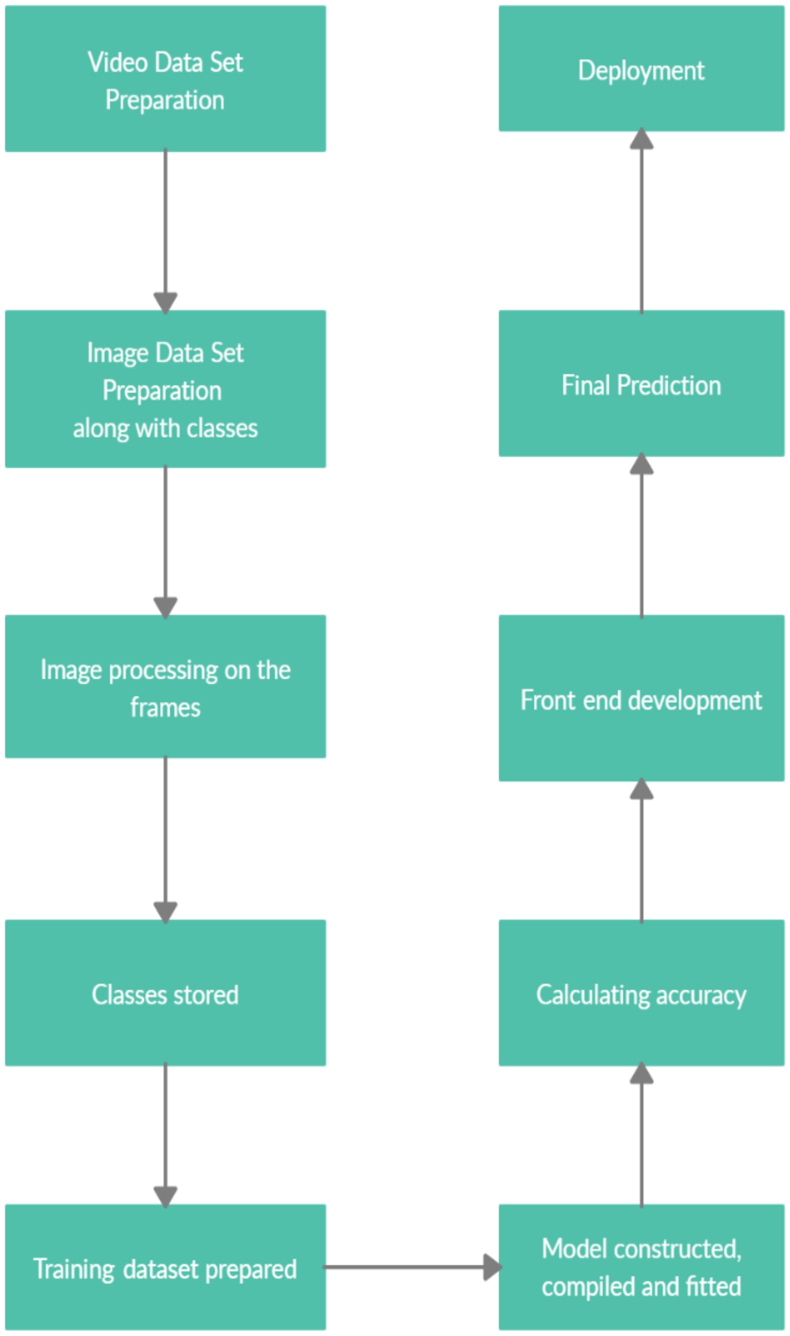
- They were getting an accuracy of 79% on the kinetic dataset, on the other hand our project provides an accuracy of 97% using the RESNET50v2 model.

- In their project, the dataset used to train the model was the Kinetics human action video dataset. This dataset contains 400 classes of human activities, with 400 and more films for each and every action. Each film lasts around tenth of a second. We used the UCF-101 dataset. UCF101 is an action recognition data set of realistic action videos, collected from YouTube, having 101 action categories. This data set is an extension of UCF50 data set which has 50 action categories.

Reference:
http://www.ijitee.org/wp-content/uploads/papers/v9i7/G5225059720.pdf

Proposed Solution

Idea with Overall block diagram of the system



Video Data Set Preparation
Image Data Set Preparation along with classes
Image processing on the frames
Classes stored
Training dataset prepared
Model constructed, compiled and fitted
Calculating accuracy
Front end development
Final Prediction
Deployment

Detailed explanation of Algorithmic steps involved for every block

I. Datasets, Pre-processing and Classes-

We originally decided to take 10 activities and started using those only, but because it increases weights a lot and uploading it on server while deploying takes up more storage, it charged us more. So, we now limit using to 6 activities. These 6 classes we used of UCF-101 Dataset are–

- Baby Crawling
- Typing
- Knitting
- Walking a dog
- Surfing
- Playing Violin

1. Video Data Set Preparation-

We have stored video in UCF-101 file in their corresponding classes. There is a test list and train list prepared to separate the videos into their respective classes. First, we created tags of each video to know which label it belonged to.

2. Image Data Set Preparation along with classes-

Now from every, video we take out frames at a particular rate.

3. Image processing on the frames-

We have tried many image processing filters and applied these three.
Now we retrieve each frame and apply Gamma correction to increase the brightness of the image and then we have sharpened 2 times to remove the blurriness of the image and then applied median filter to remove the noise and then save it to another folder. We have used PIL for image sharpening and median filter and CV2 for gamma correction. So, we have converted the image from CV2 to PIL format by changing the colour by BGR2RGB.

We have tried many image processing filters and used three filters – Median filter, Gamma Correction filter and Sharpening the image. These filters helped us in improving the quality of the frames that we have got from the videos. The improvement in the quality of images helped us to increase the overall efficiency of the program as well as the accuracy also got increased. We have applied these techniques in both the training dataset and testing dataset.

The process involved in improving the efficiency of the image is as follows-

- We retrieve each frame from the train_4 folder one by one and apply Gamma correction. Gamma Correction is a filter that controls the overall brightness of an image. Images which are not properly corrected can look either bleached out, or too dark. Varying the amount of gamma correction changes not only the brightness, but also the ratios of red to green to blue. We have applied this to increase the brightness of the image that have dark background and to enhance the foreground to extract more features for the model.

- After that we take each frame that has been gamma corrected and sharpen the image. Sharpening is an image-manipulation technique for making the outlines of a digital image look more distinct. Sharpening increases the contrast between edge pixels and emphasizes the transition between dark and light areas. Sharpening increases local contrast and brings out fine detail. We have applied this two times to increase the overall contrast between edge pixels. This also helps in extracting different movements of the activities performed that are difficult to capture for the model to capture otherwise.

- After sharpening the image 2 times we have applied the median filter. The median filter is normally used to reduce noise in an image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image. Such noise reduction is a typical pre-processing step to improve the results of later processing such as to extract features for the models.

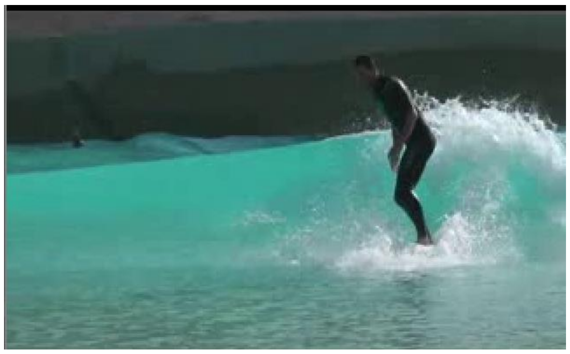
These all techniques improve the overall quality of the images and to extract more features for the model to get trained better and to increase the accuracy of the model.
After applying all these filters, we save these frames back to train_6 folder. We have used these techniques three times in our code first for the training dataset for which we have taken the frames from train_4 and saved back to train_6, accuracy calculation for which we have taken the frames form temp_4 and saved back to temp_5. Similarly, we have used these techniques for predicting a given activity.

We have used both PIL and CV2 modules for image processing techniques. Image sharpening is done by PIL module and same for the median filter. The gamma correction is done using CV2 module. For converting an image to be used in CV2 module to PIL module we have a function that changes the colour from BGR2RGB.

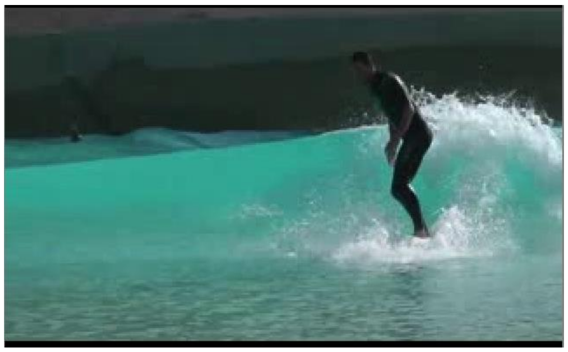
A limitation we face while applying image processing is that because we have so many classes of image as opposed to having only 2 classes, we had to choose only those processing techniques which on an average enhances all the images. Like Baby crawling is still a bit noisy, but typing is way better and surfing is also sharpened.

SURFING

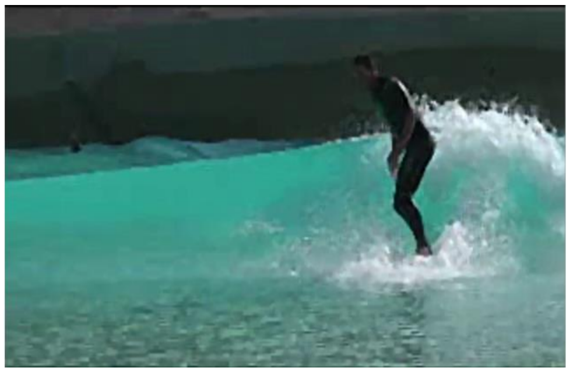
Original



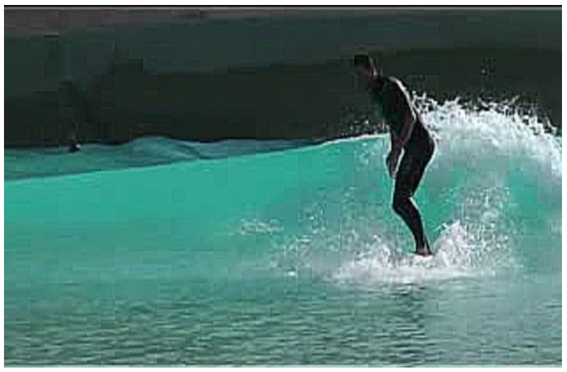
Gamma Corrected



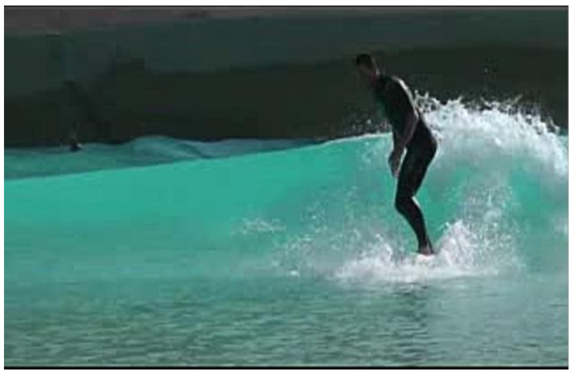
Sharpened - 1



Sharpened - 2

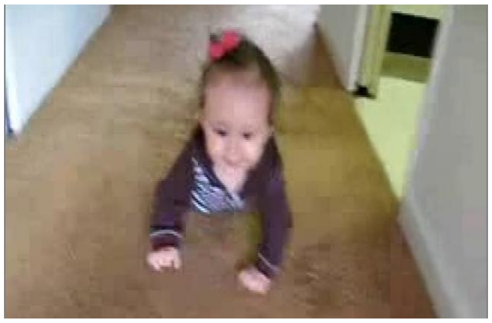


Final Image

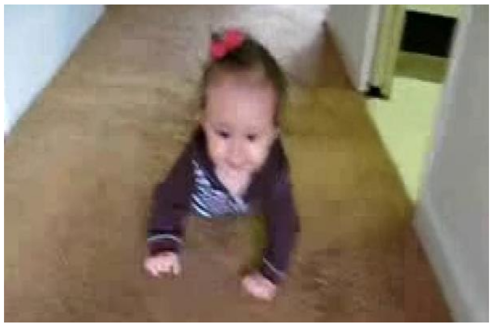


BABY CRAWLING

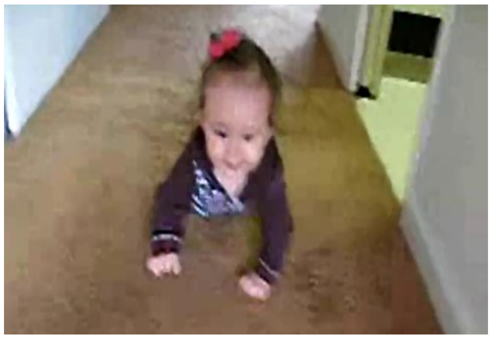
Original



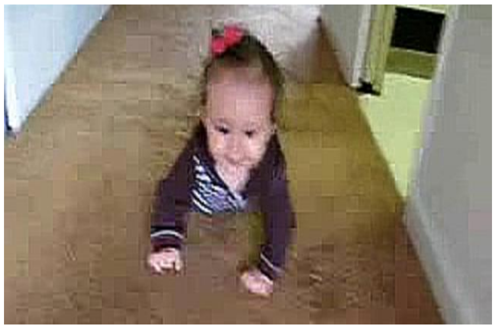
Gamma Corrected



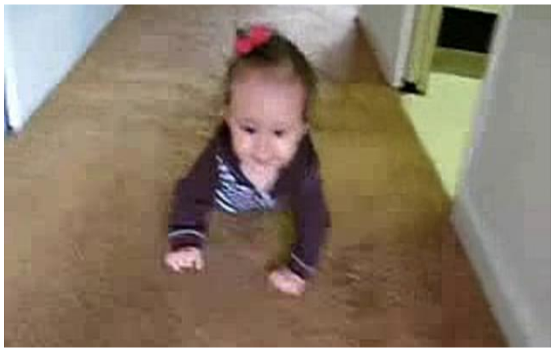
Sharpened - 1



Sharpened - 2

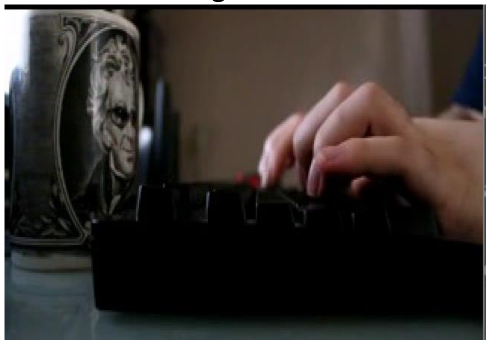


Final Image

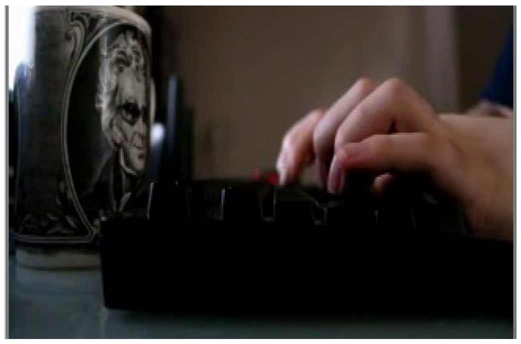


TYPING

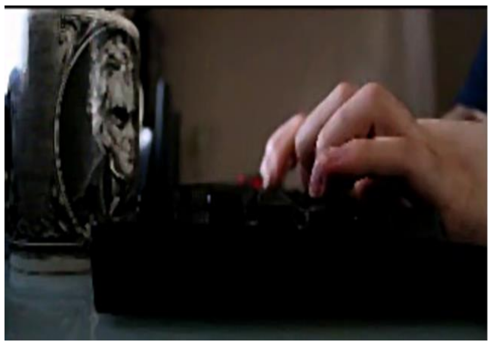
Original



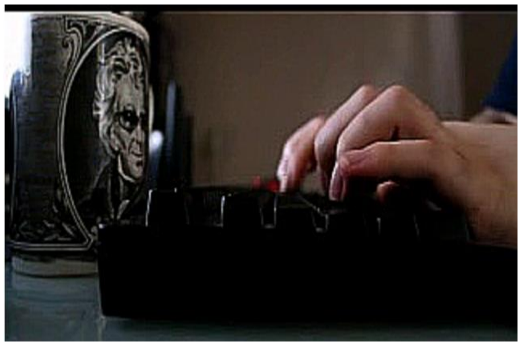
Gamma Corrected



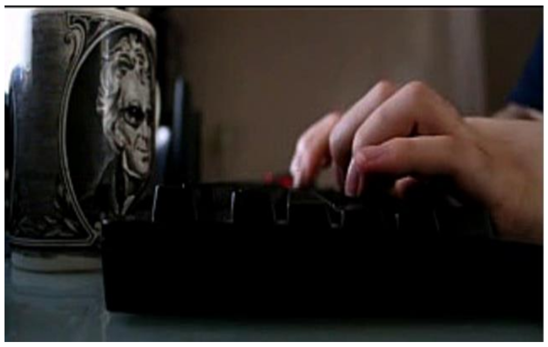
Sharpened - 1



Sharpened - 2



Final Image



4. **Labels stored-**

   After applying image processing, we are storing the name of every image along with its label in a csv folder.

5. **Training dataset prepared-**

   Now, we first load all the images using glob library and then we reshape them to 224*224 size. After that we are changing the image in a NumPy array.
   Then we are splitting our whole training dataset into two parts: train and validation sets. The distribution is random but normalised using stratify. Thus, we end up with 3430 images for training and 858 images for validation which is 20% of the original whole training dataset which adds up to 4288 images.

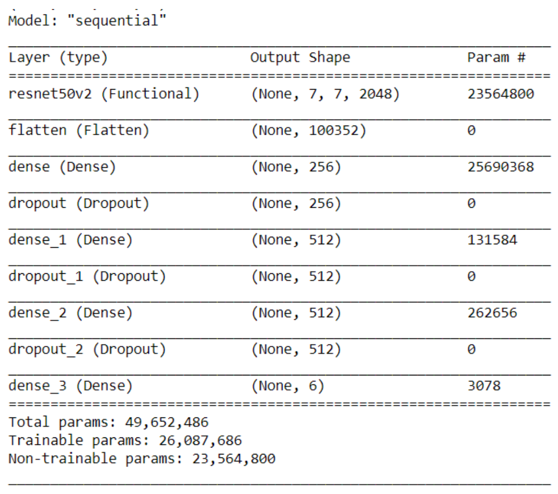
II. Model creation and training-

In the second part, we are creating our model and training it using the train and validation data.

1. **Model Construction-**

   Now we define our base model which is Resnet50v2 and give it our input shape of image and 3 signifying RGB. We initially chose VGG-16, but reverted our decision because of this being more recent and accurate model. This will be our first layer in the model We are developing a sequential mode where layers are added in series one after another. This concept of model creation is called Transfer Learning, in which we use a model trained on millions of images whose classification accuracy is found to be very high for a large number of objects. Then we add a series of dense and dropout layers. Dropout Layers is a simple ay to avoid overfitting in the model. This creates our Convolution Neural Network.
   This model was selected mainly because

   - ResNets are easy to optimize, but the “plain” networks (that simply stack layers) shows higher training error when the depth increases.
   - ResNets can easily gain accuracy from greatly increased depth, producing results which are better than previous networks.



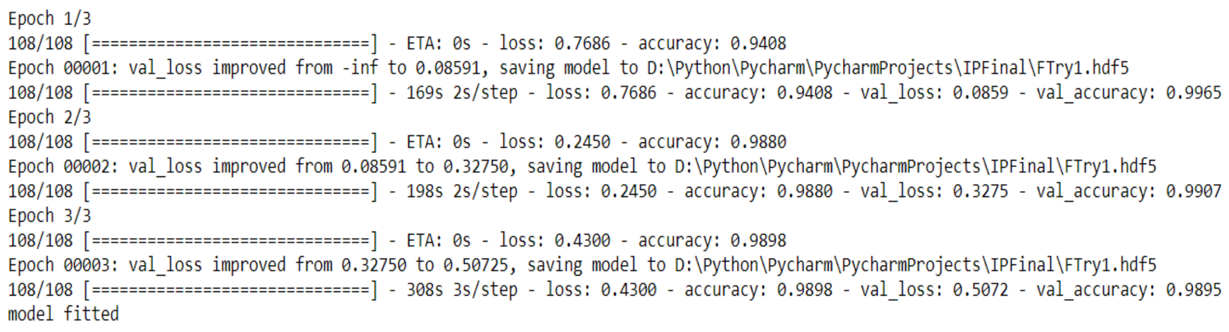
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50v2 (Functional)      (None, 7, 7, 2048)        23564800
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
dense (Dense)                (None, 256)               25690368
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 3078
=================================================================
Total params: 49,652,486
Trainable params: 26,087,686
Non-trainable params: 23,564,800
_________________________________________________________________
```

2. Model Compiling-

We are using loss function as categorical cross entropy as we have multiple classes. Optimizer is set to Adam. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. The metrics we measure is accuracy.

3. Model Fitting-

Now we fit the model. For this purpose, we set batch size to 32 which means that now3 2 images will make a batch and images will process in these batches We set epoch as 3 because it sufficiently increases val accuracy without increasing val loss. So whole data will run thrice. This creates model weights which stores all the learned data in numerical format.



```
Epoch 1/3
108/108 [==============================] - ETA: 0s - loss: 0.7686 - accuracy: 0.9408
Epoch 00001: val_loss improved from -inf to 0.08591, saving model to D:\Python\Pycharm\PycharmProjects\IPFinal\FTry1.hdf5
108/108 [==============================] - 169s 2s/step - loss: 0.7686 - accuracy: 0.9408 - val_loss: 0.0859 - val_accuracy: 0.9965
Epoch 2/3
108/108 [==============================] - ETA: 0s - loss: 0.2450 - accuracy: 0.9880
Epoch 00002: val_loss improved from 0.08591 to 0.32750, saving model to D:\Python\Pycharm\PycharmProjects\IPFinal\FTry1.hdf5
108/108 [==============================] - 198s 2s/step - loss: 0.2450 - accuracy: 0.9880 - val_loss: 0.3275 - val_accuracy: 0.9907
Epoch 3/3
108/108 [==============================] - ETA: 0s - loss: 0.4300 - accuracy: 0.9898
Epoch 00003: val_loss improved from 0.32750 to 0.50725, saving model to D:\Python\Pycharm\PycharmProjects\IPFinal\FTry1.hdf5
108/108 [==============================] - 308s 3s/step - loss: 0.4300 - accuracy: 0.9898 - val_loss: 0.5072 - val_accuracy: 0.9895
model fitted
```

III. Testing-

After fitting the model and saving it, we now prepare our test videos. Each video is converted into frames and activity in the frame is predicted in all the frames. The mode all the predictions is taken and video is classified by that. We do this for all the test videos. Then we calculate the accuracy which is around 97%.

IV. Front-end development-

We did simple front-end development in HTML file with very little CSS.

- The submit page is created to upload the video.
- Result page to show the prediction.

V. Video prediction -

This part has same code as testing file. The only difference is that we were testing for a lot of videos to measure the accuracy of the model developed, but here we only predict for a single video given by the user. We made it in integration with Flask to deploy it on local host.

VI. Deploy-

Local:

We deploy our model with help of flask on local host. The first function opens the HTML page to upload a video. The submit button calls uploader function which requests the video and stores it on the system. Processing is applied on the caught frames and prediction is made. This prediction is sent to result page and displayed.

Online:

We explored various platforms for online deployment. Digital Ocean, Heroku, Google and AWS. We explored git and Heroku CLI. Finally, we deployed our model using AWS EC2 instances. Below we mention the approach we took and steps we used and our experience on each of the platforms.

a) Deployment via Heroku:

There were many Stages for Deployment via Heroku.

- We created PROCFILE to specify open the first file for runtime and its properties.
- We created APTFILE to specify the buildpacks and external libraries for download when runtime.
- We created requirements.txt file which had all the modules we are using in our project and their version so that they are downloaded in the server while running the code.

We encountered following major problems/errors:

- First there were errors due to various File semantics, version compatibility among many others. We searched web to solve them.
- Weights file being a large file of 192 MB cannot be uploaded on github manually. We installed git and GIT LFS on our system and uploaded using it.

- OS error while building the model which was not recognizing the weights file. We had to add buildpack in Heroku and its key which was generated in Github app access token.

The final problem which forced us to shift to another platform was that the slug size in Heroku is limited to 500mb while we our whole file on server was a compressed 618MB file which included all python files and weights.
Below is mentioned link to see the files.

https://github.com/proacc2022/HAR

b) Deployment on Docker and Digital Ocean

Although Docker was creating a virtual environment in my system, using Digital ocean was infeasible due to its fully paid mode. So even trying it was a bit heavy on pockets.

c) Deployment using AWS

AWS is very feasible and mostly free platform. We used EC2 instances to deploy our Model. Below are the steps we went through for our final deployment.

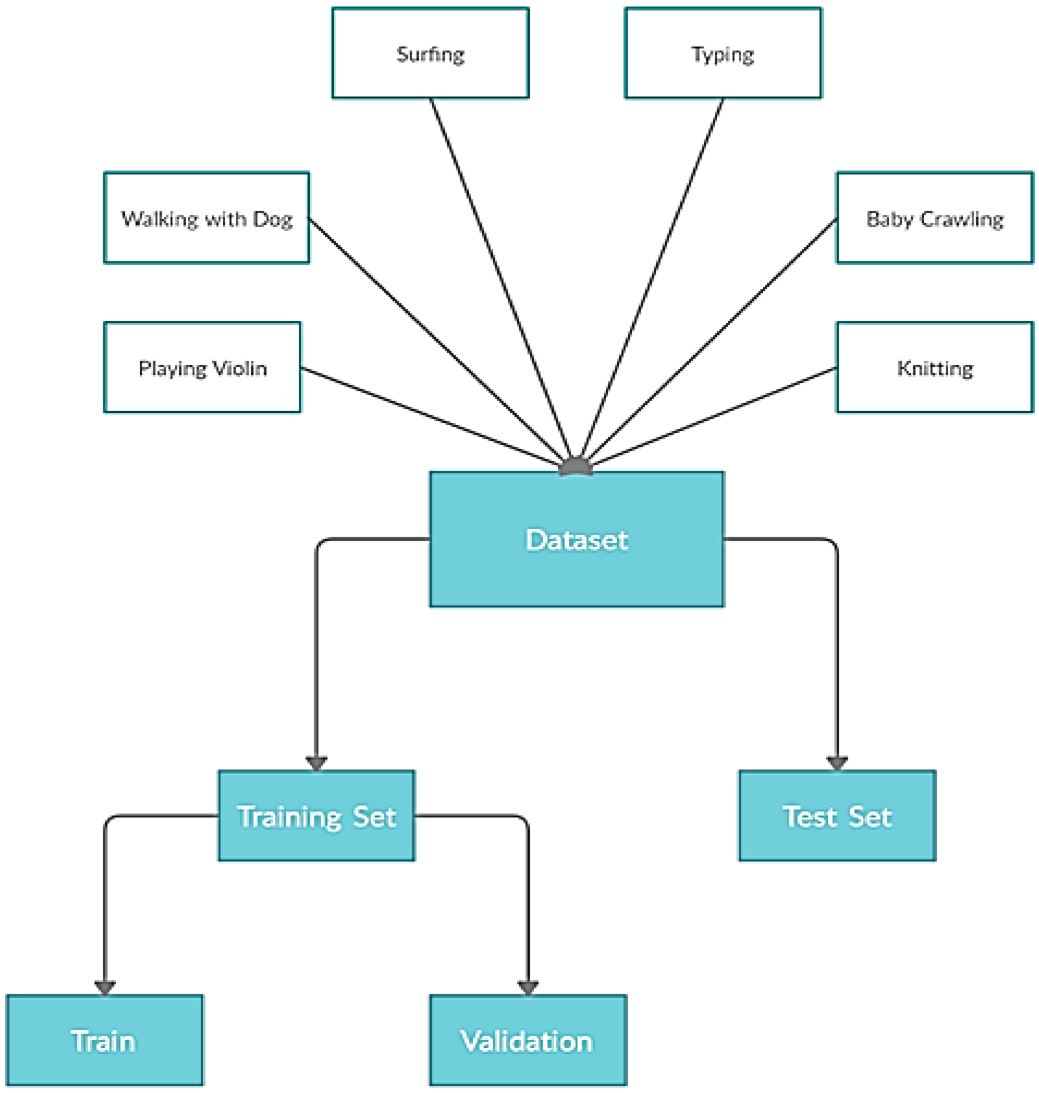
- We created a free instance and chose our AMI to be ubuntu.
- We opened the site from all traffic in security groups
- We created an SSH key pair for instance.
- Then we downloaded 3 software: Putty, Putty Generator and WinSCP
- Using Putty generated, we saved a ppk file from pem file which was provided as key. This will be used to access our server and network.
- Then we uploaded our project files using WinSCP.
- Using Putty I logged in Ubuntu and downloaded pi3 on server and the using it, also downloaded the python modules mentioned in requirements.txt file.
- But because this was a free instance, its memory on server was limited. So we repeated this for a paid server. This had just a little more memory than the previous one and was light on our pockets also.
- After all modules were installed, I turned my server on by running my main app file.
- The project is now fully online.
- The Public DNS mentioned in AWS instance and host given together gives the link of site.
- Every time, I restart my server, the DNS changes. So, the link also changes.

Database Designed

This dataset was made by Khurram Soomro, Amir Roshan Zamir and Mubarak Shah, **UCF101**: A **Dataset** of 101 Human Action Classes from Videos in The Wild., CRCV-TR-12-01, November, 2012.

This dataset has 101 action categories given below:
Apply Eye Makeup, Apply Lipstick, Archery, Baby Crawling, Balance Beam, Band Marching, Baseball Pitch, Basketball Shooting, Basketball Dunk, Bench Press, Biking, Billiards Shot, Blow Dry Hair, Blowing Candles, Body Weight Squats, Bowling, Boxing Punching Bag, Boxing Speed Bag, Breaststroke, Brushing Teeth, Clean and Jerk, Cliff Diving, Cricket Bowling, Cricket Shot, Cutting In Kitchen, Diving, Drumming, Fencing, Field Hockey Penalty, Floor Gymnastics, Frisbee Catch, Front Crawl, Golf Swing, Haircut, Hammer Throw, Hammering, Handstand Push-ups, Handstand Walking, Head Massage, High Jump, Horse Race, Horse Riding, Hula Hoop, Ice Dancing, Javelin Throw, Juggling Balls, Jump Rope, Jumping Jack, Kayaking, Knitting, Long Jump, Lunges, Military Parade, Mixing Batter, Mopping Floor, Nun chucks, Parallel Bars, Pizza Tossing, Playing Guitar, Playing Piano, Playing Tabla, Playing Violin, Playing Cello, Playing Daf, Playing Dhol, Playing Flute, Playing Sitar, Pole Vault, Pommel Horse, Pull Ups, Punch, Push Ups, Rafting, Rock Climbing Indoor, Rope Climbing, Rowing, Salsa Spins, Shaving Beard, Shotput, Skate Boarding, Skiing, Skijet, Sky Diving, Soccer Juggling, Soccer Penalty, Still Rings, Sumo Wrestling, Surfing, Swing, Table Tennis Shot, Tai Chi, Tennis Swing, Throw Discus, Trampoline Jumping, Typing, Uneven Bars, Volleyball Spiking, Walking with a dog, Wall Push-ups, Writing On Board, Yo-Yo.

We are using 6 of these classes in our dataset due to hardware restrictions shown below. The dataset is first divided in training dataset and testing dataset. Training Dataset is further divided in train and validation dataset. All these distributions are equally done, that is the train, validation and testing set have equal number of videos of each class so to make an unbiased model and test it efficiently.


Surfing
Typing
Walking with Dog
Baby Crawling
Playing Violin
Knitting
Dataset
Training Set
Test Set
Train
Validation

Performance Metrics

Machine learning model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

1. Precision-In pattern recognition, information retrieval and classification (machine learning), precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances.

   Precision = TruePositives / (TruePositives + FalsePositives)

2. Recall-In pattern recognition, information retrieval and classification (machine learning), recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved.

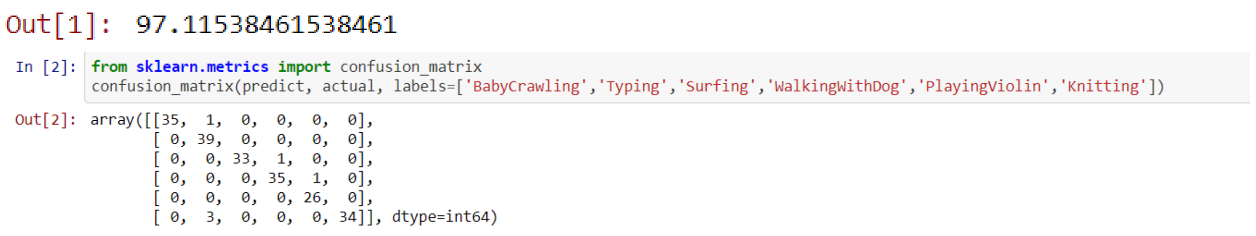
   Recall = TruePositives / (TruePositives + FalseNegatives)

3. The F_1 score (also F-score or F-measure) is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive.

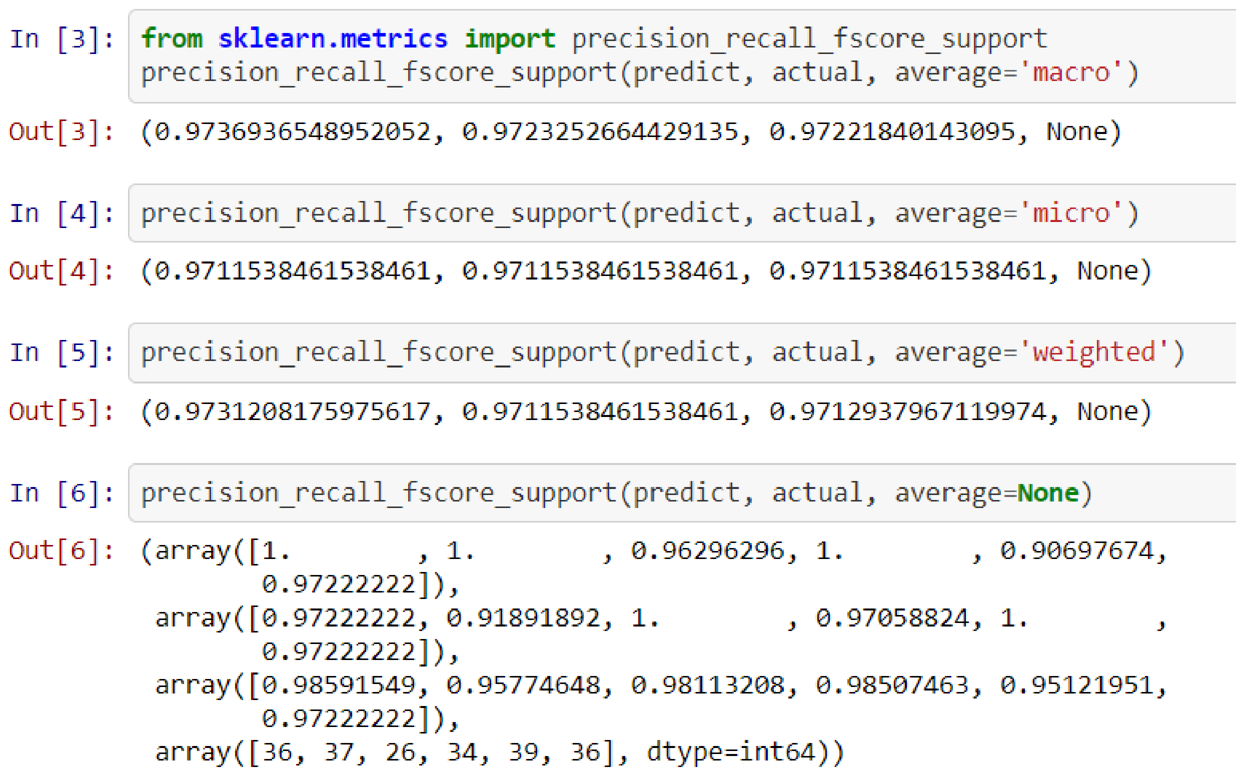
   F-Measure = (2 * Precision * Recall) / (Precision + Recall)

4. Support is the number of actual occurrences of the class in the specified dataset.

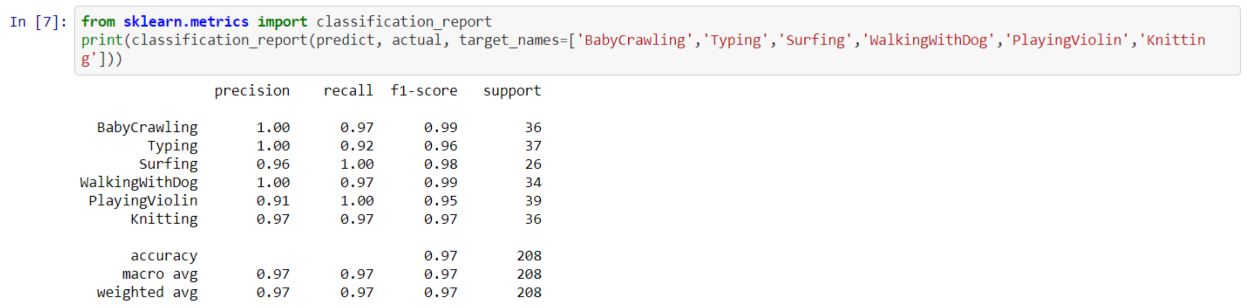
A confusion matrix is nothing but a table with two dimensions viz. “Actual” and “Predicted” and furthermore, both the dimensions have “True Positives (TP)”, “True Negatives (TN)”, “False Positives (FP)”, “False Negatives (FN)”



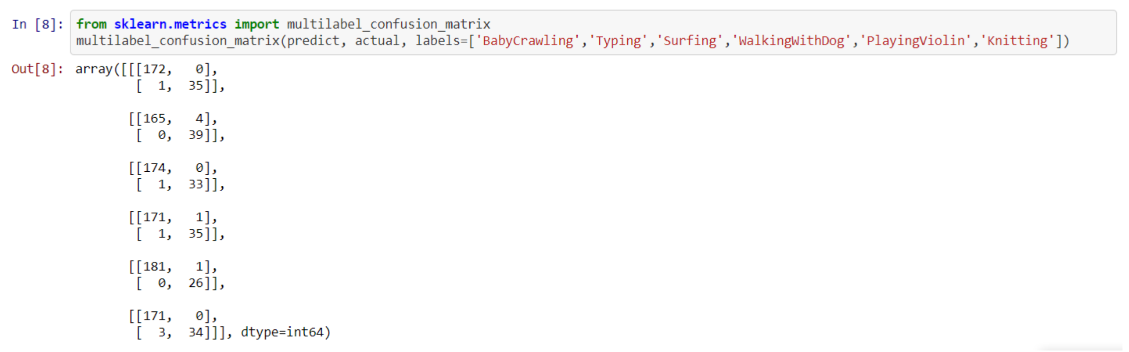
```
Out[1]:  97.11538461538461

In [2]: from sklearn.metrics import confusion_matrix
        confusion_matrix(predict, actual, labels=['BabyCrawling','Typing','Surfing','WalkingWithDog','PlayingViolin','Knitting'])

Out[2]: array([[35,  1,  0,  0,  0,  0],
               [ 0, 39,  0,  0,  0,  0],
               [ 0,  0, 33,  1,  0,  0],
               [ 0,  0,  0, 35,  1,  0],
               [ 0,  0,  0,  0, 26,  0],
               [ 0,  3,  0,  0,  0, 34]], dtype=int64)
```



```
In [3]: from sklearn.metrics import precision_recall_fscore_support
        precision_recall_fscore_support(predict, actual, average='macro')

Out[3]: (0.9736936548952052, 0.9723252664429135, 0.97221840143095, None)

In [4]: precision_recall_fscore_support(predict, actual, average='micro')

Out[4]: (0.9711538461538461, 0.9711538461538461, 0.9711538461538461, None)

In [5]: precision_recall_fscore_support(predict, actual, average='weighted')

Out[5]: (0.9731208175975617, 0.9711538461538461, 0.9712937967119974, None)

In [6]: precision_recall_fscore_support(predict, actual, average=None)

Out[6]: (array([1.        , 1.        , 0.96296296, 1.        , 0.90697674,
                0.97222222]),
         array([0.97222222, 0.91891892, 1.        , 0.97058824, 1.        ,
                0.97222222]),
         array([0.98591549, 0.95774648, 0.98113208, 0.98507463, 0.95121951,
                0.97222222]),
         array([36, 37, 26, 34, 39, 36], dtype=int64))
```



```
In [7]: from sklearn.metrics import classification_report
        print(classification_report(predict, actual, target_names=['BabyCrawling','Typing','Surfing','WalkingWithDog','PlayingViolin','Knittin
        g']))

                precision    recall  f1-score   support

  BabyCrawling       1.00      0.97      0.99        36
        Typing       1.00      0.92      0.96        37
       Surfing       0.96      1.00      0.98        26
WalkingWithDog       1.00      0.97      0.99        34
 PlayingViolin       0.91      1.00      0.95        39
      Knitting       0.97      0.97      0.97        36

      accuracy                           0.97       208
     macro avg       0.97      0.97      0.97       208
  weighted avg       0.97      0.97      0.97       208
```



```
In [8]: from sklearn.metrics import multilabel_confusion_matrix
        multilabel_confusion_matrix(predict, actual, labels=['BabyCrawling','Typing','Surfing','WalkingWithDog','PlayingViolin','Knitting'])

Out[8]: array([[[172,   0],
                [  1,  35]],

               [[165,   4],
                [  0,  39]],

               [[174,   0],
                [  1,  33]],

               [[171,   1],
                [  1,  35]],

               [[181,   1],
                [  0,  26]],

               [[171,   0],
                [  3,  34]]], dtype=int64)
```

Implementation

Contribution of each Group member

Sarvesh Agrawal
B006

- Researched on the topic and laid down the structure and flow of working.
- Prepared the test and train list to separate the videos so as to call them when they are required.
- Did most of the coding as my laptop was apt for it.
- Defined model structure and how to save and load the model and weights.
- Found out how model was compiled and fitted and what parameters are required for our project.
- Deployed the model on flask
- Deployed model online

Anmol Batra
B013

- Worked to find the dataset for our project.
- Helped Naman to implement the techniques chosen in both test and train data
- Found how to create the csv file for all the frames.
- Researched and tested various CNN models.
- Made html file to show the result

Naman Bhansali
B015

- Searched various IP techniques and applied them on frames of various classes.
- Applied the techniques chosen in both test and train data
- Found how to get frames from a video.
- Found out how model was compiled and fitted and what parameters are required for our project.
- Did coding with Sarvesh
- Deployed model on flask

Yatharth Bijolia
B016

- Prepared the test and train list to separate the videos so as to call them when they are required.
- Researched and tested various CNN models.
- Searched various IP techniques and applied them on frames of various classes.
- Helped Sarvesh to do model creation.
- Made html file to submit the video.

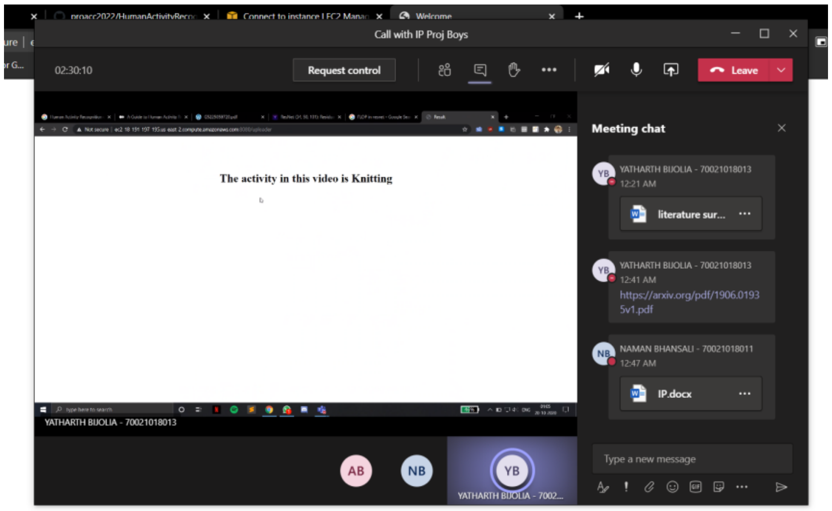
All of us together made the report and presentation.

Graphical User interface screen shots / all possible input images and its corresponding output images obtained at various stages of processing.

Online hosting

http://ec2-18-191-197-195.us-east-2.compute.amazonaws.com:8080/

(This is just the link used in the images. Every time I restart the server, I will have to provide a new link. This is because server costs money.)

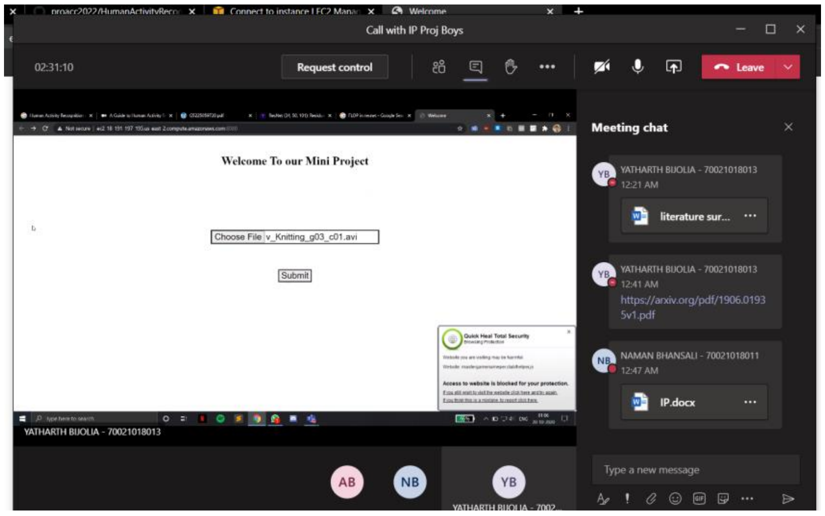


The activity in this video is Knitting

YATHARTH BIJOLIA - 70021018013 12:21 AM
literature sur...

YATHARTH BIJOLIA - 70021018013 12:41 AM
https://arxiv.org/pdf/1906.01935v1.pdf

NAMAN BHANSALI - 70021018011 12:47 AM
IP.docx



Welcome To our Mini Project

v_Knitting_g03_c01.avi

YATHARTH BIJOLIA - 70021018013 12:21 AM
literature sur...

YATHARTH BIJOLIA - 70021018013 12:41 AM
https://arxiv.org/pdf/1906.01935v1.pdf

NAMAN BHANSALI - 70021018011 12:47 AM
IP.docx

Working Code

The following files are uploaded on the GitHub link provided below-

- Codes
- Model
- HTML files
- Deploying file
- Csv file
- Train list
- Test list

The link is-

https://github.com/proacc2022/HumanActivityRecognition

Conclusion

In this project of Human Activity Recognition System, we proposed a model trained using Convolutional neural network (CNN) on UCF-101 data set to recognize almost 6 human activities with a great accuracy level.
Convolutional Neural Network models, or CNNs for short, are a type of deep neural network that were developed for use with image data, e.g. such as handwriting recognition.
They have proven very effective on challenging computer vision problems when trained at scale for tasks such as identifying and localizing objects in images and automatically describing the content of images.
The model we used was RESNET50v2 which has 50 convolutional layers and around 3.8 billion FLOPs (Floating Point Operations per Second).
Various models were researched before selecting this particular model like VGG-16, RESNET-34.
Challenges were also faced like enhancing videos of every class using the same Image Processing techniques while implementation of the same but they were tackled as we all researched more to nullify the issue.
A HTML page was created to upload a video and get the on-going activity in the video as the output on the next page. This also acts as the front end of our project.
We have made an efficient trainable model in this project which gives accurate human activity in the given video. There is a shortage of videos because of the limitation of the hardware but although there are only 6 videos taken, we get an excellent accuracy of 97% which is not easy to get.
The code is deployed online too which is a great confidence boost for all of us.
This project is highly significant as it has numerous applications in the world like Healthcare monitoring, Security and surveillance, Tele-Immersion, human-computer interaction and can also be used in robots to detect the human activities in their field of vision. Some places are for the authorized personnel only, if someone else enters that area, an alarm can be triggered or siren can we wailed to process security. Thus, in accordance with future scopes and the current day scenario, we selected this topic for our project because it was best suited in terms of applications and learning AIML.

The future holds more advanced training models as well as hardware technologies which will together run to give a super smooth functionality in the Image processing Domain. With further modifications in this project it can also contribute in the future where Human Activity recognition is a problem and a solution is needed.

LIMITATIONS AND FUTURE SCOPE

Complex and Various Backgrounds

Sports event broadcast is a typical case of dynamic recording. In fact, with the popularity of smart devices such as smart glasses and smartphones, people tend to record videos with embedded cameras from wearable devices anytime. Most of these real-world videos have complex dynamic backgrounds. First, those videos, as well as the broadcasts, are recorded in various and changing backgrounds. Second, realistic videos abound with occlusions, illumination variance, and viewpoint changes, which make it harder to recognize activities in such complex and various conditions.

Multisubject Interactions and Group Activities

Earlier research concentrated on low-level human activities such as jumping, running, and waving hands. One typical characteristic of these activities is having a single subject without any human-human or human-object interactions. However, in the real world, people tend to perform interactive activities with one or more persons and objects. An American football game is a good example of interaction and group activity where multiple players (i.e., human-human interaction) in a team protect the football (i.e., human-object interaction) jointly and compete with players in the other team. It is a challenging task to locate and track multiple subjects synchronously or recognize the whole human group activities as “playing football” instead of “running.”

Long-Distance

Long-distance and low-quality videos with severe occlusions exist in many scenarios of video surveillance. Large and crowded places like the metro and passenger terminal of the airport are representative occasions where occlusions happen frequently. Besides, surveillance cameras installed in high places cannot provide high-quality videos like present datasets in which the target person is clear and obvious. Though we do not expect to track everyone in these cases, some abnormal or crime-related behaviors should be recognized by the HAR system. Another typical long-distance case is the football broadcast. Due to the long distance of cameras, the subject is rather small which makes it difficult to analyze activities of the torso, and the relatively low quality of those long-distance videos further increases the difficulty.

Multiple Action Recognition

If a person is performing more than one activity in a single video, our model will predict only that activity whose frames are predicated more in that class. For example, A person is Jumping and then starts running, then if jumping has more frames captured and the prediction of the frames is also jumping then the activity predicted would be jumping only even though the person was running also

REFERENCES

- R. Zhu *et al.*, "Efficient Human Activity Recognition Solving the Confusing Activities Via Deep Ensemble Learning," in *IEEE Access*, vol. 7, pp. 75490-75499, 2019, doi: 10.1109/ACCESS.2019.2922104.
- E. Kim, S. Helal and D. Cook, "Human Activity Recognition and Pattern Discovery," in *IEEE Pervasive Computing*, vol. 9, no. 1, pp. 48-53, Jan.-March 2010, doi: 10.1109/MPRV.2010.7.
- Ke, S.-R.; Thuc, H.L.U.; Lee, Y.-J.; Hwang, J.-N.; Yoo, J.-H.; Choi, K.-H. A Review on Video-Based Human Activity Recognition. *Computers* **2013**, *2*, 88-131.
- Weiyao Lin, Ming-Ting Sun, R. Poovandran and Zhengyou Zhang, "Human activity recognition for video surveillance," *2008 IEEE International Symposium on Circuits and Systems*, Seattle, WA, 2008, pp. 2737-2740, doi: 10.1109/ISCAS.2008.4542023.
- T. Subetha and S. Chitrakala, "A survey on human activity recognition from videos," *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, Chennai, 2016, pp. 1-7, doi: 10.1109/ICICES.2016.7518920.
- Ribeiro, Pedro & Santos-Victor, José. (2005). Human Activity Recognition from Video: modeling, feature selection and classification architecture. Proceedings of International Workshop on Human Activity Recognition and Modeling, HAREM'05.
- *Tutorial 4- Deployment Of ML Models In AWS EC2 Instance*. (2019, October 30). [Video]. YouTube. https://www.youtube.com/watch?v=oOqqwYI60FI&t=620s
- Brownlee, J. (2020, August 27). *How to Save and Load Your Keras Deep Learning Model*. Machine Learning Mastery. https://machinelearningmastery.com/save-load-keras-deep-learning-models/
- Kurama, V. (2019, December 16). *Deploying Deep Learning Models on the Web With Flask*. Paperspace Blog. https://blog.paperspace.com/deploying-deep-learning-models-flask-web-python/
- *Git large*. (n.d.). Git Large File Storage. https://git-lfs.github.com/
- *Cuda driver errors on the machine without GPU while loading model*. (2020, March 16). Stack Overflow. https://stackoverflow.com/questions/60706122/cuda-driver-errors-on-the-machine-without-gpu-while-loading-model
- *Heroku Buildpacks | Heroku*. (n.d.). Heroku. Retrieved from https://www.heroku.com/elements/buildpacks

- *DevTeam*. (n.d.). DevTeam.Space. Retrieved October 10, 2020, from https://www.devteam.space/blog/how-to-deploy-a-web-app-with-docker-containers/

- Khurram Soomro, K. S. (2012, November 5). *UCF Dataset*. CRCV. https://www.crcv.ucf.edu/data/UCF101.php

- *Sharpening an Image*. (n.d.). Sharpening an Image. https://northstar-www.dartmouth.edu/doc/idl/html_6.2/Sharpening_an_Image.html

- *High Pass Filter for image processing in python by using scipy/numpy*. (2011, May 23). Stack Overflow. https://stackoverflow.com/questions/6094957/high-pass-filter-for-image-processing-in-python-by-using-scipy-numpy

- *TGS Salt Identification Challenge | Kaggle*. (n.d.). Kaggle. https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/62261

- *What is the difference between the terms accuracy and validation accuracy*. (2018, July 15). Stack Overflow. https://stackoverflow.com/questions/51344839/what-is-the-difference-between-the-terms-accuracy-and-validation-accuracy

- Xinding Sun, Ching-Wei Chen and B. S. Manjunath, "Probabilistic motion parameter models for human activity recognition," Object recognition supported by user interaction for service robots, Quebec City, Quebec, Canada, 2002, pp. 443-446 vol.1, doi: 10.1109/ICPR.2002.1044751.

- O. C. Ann and L. B. Theng, "Human activity recognition: A review," 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014), Batu Ferringhi, 2014, pp. 389-393, doi: 10.1109/ICCSCE.2014.7072750.