HUMAN ACTION RECOGNITION USING DEEP LEARNING

Prof. Rajesh.U.Shekokar, Ms. Shubhada Karajkhede, Mr. Prathamesh Joshi, Mr. Kunal Chandolkar,

Department of Electronics and Telecommunication, RMDSSOE Pune, India

ABSTRACT

We propose in this paper a fully automated deep learning model, which learns to classify human actions. Human action recognition is an important application domain in computer vision. Its primary aim is to accurately describe human actions and their interactions from a previously unseen data sequence acquired by sensors. The ability to recognize, understand and predict complex human actions enables the construction of many important applications such as intelligent surveillance systems, human-computer interfaces, health care, security and military applications. Human activity recognition (HAR) is the outcome of a similar motive. Recent times have seen the theory of Human Activity Recognition (HAR) catering to multiple challenging applications built on the increase in ubiquitous, persuasive computing.

Key words - *Neural Networks, Deep Learning, Crime dataset, CNN, Action recognition, Confusion Matrix.*

INTRODUCTION

Human Action recognition has gained much importance in the past decade due to its numerous applications in human centric applications as in the field of medical, security and also military. Important goal of the HAR in the current scenario is to identify the actions of the user in order to assist them with their tasks with the help of computing systems. Deep learning, a subfield of machine learning is the study of these neural networks, which over time have introduced several variations of these networks for various problems. For Video Recognition, this approach utilizes deep learning - in the context of a number of labeled images, a model is built so that it can generate a prediction label for a new video. Steps have been taken for execution are downloading, extracting and pre- processing a video dataset then dividing the dataset into training and testing data then creation of a neural network and train it on the training data finally testing the model on the test data [1].

METHODOLOGY

Data Preprocessing

The Visual Dataset involves 13 human Crime situations some of them are fighting, road accident, burglary, robbery, etc. Regardless of these circumstances, the layout will be developed. The videos were captured at a frame rate of 25 fps and the resolution of 240 pixels was sampled in each frame. The collection comprises too many images–100 videos.

The video's spatial dimension (width x height) can be found to be 160 x 120 pixels. Also, on loading a single video into a NumPy array in python, the shape of the array obtained was (1, 515, 120, 160, 3), this indicates that the video has 6000 frames with the spatial dimension of the video is 160 x 120 (width x height) pixels and each frame has 3 channels Red(R), Green(G) and Blue(B).

Fig.1 Single frame of a sample video of Fighting.

Dataset preprocessing operations

i. Reading in the video frame-by-frame.

ii. The videos were captured at a frame rate of 25fps. This means that for each second of the video, there willbe 25 frames. We know that within a second, a human body does not perform very significant movement. This implies that most of the frames (per second) in our video will be redundant. Therefore, only a subset of all the frames in a video needs to be extracted. This will also reduce the size of the input data which will in turn help the model train faster and can also prevent over-fitting.

Different strategies would be used for frame extraction like:

- Extracting a **fixed number of frames from the total frames** in the video - say only thefirst 250 frames
- Extracting a **fixed number of frames each second** from the video – say we need only 5 frames per second from a video whose duration is of 10 seconds. This would return a totalof 50 frames from the video. This approach is better in the sense that we are extracting theframes sparsely and uniformly from the entire video.

iii. Each frame needs to have the same spatial dimensions (height and width). Hence each

frame in a videowill have to be resized to the required size.

iv. In order to simplify the computations, the frames are converted to grayscale.

v. Normalization - The pixel values range from 0 to 255. These values would have to be normalized in orderto help our model converge faster and get a better performance.

IMPLEMENTATION

The video data set was loaded and the necessary preprocessing measures were one of the most critical parts of the project. Therefore, we created a function named Preprocess and processed the images on terms of different parameters. It was very difficult to create this because we worked on generalizing this feature for any type of video (not unique to that project). We used NumPy (wherever) to store and process the videos (with a additional functionality much faster than the built-in python lists). We extracted frames from videos of crime actions. Then we used transfer learning for model training. Figure 3 reveals that a number of video clips of events (Fighting scene), including the whole guy, can be included in the film. In addition, the majority of frames would be redundant if the man moved very slowly. The model would face such a problem as a major challenge [3]. But after a better training of model with plenty of more images the model works in a fluent and more smooth way.

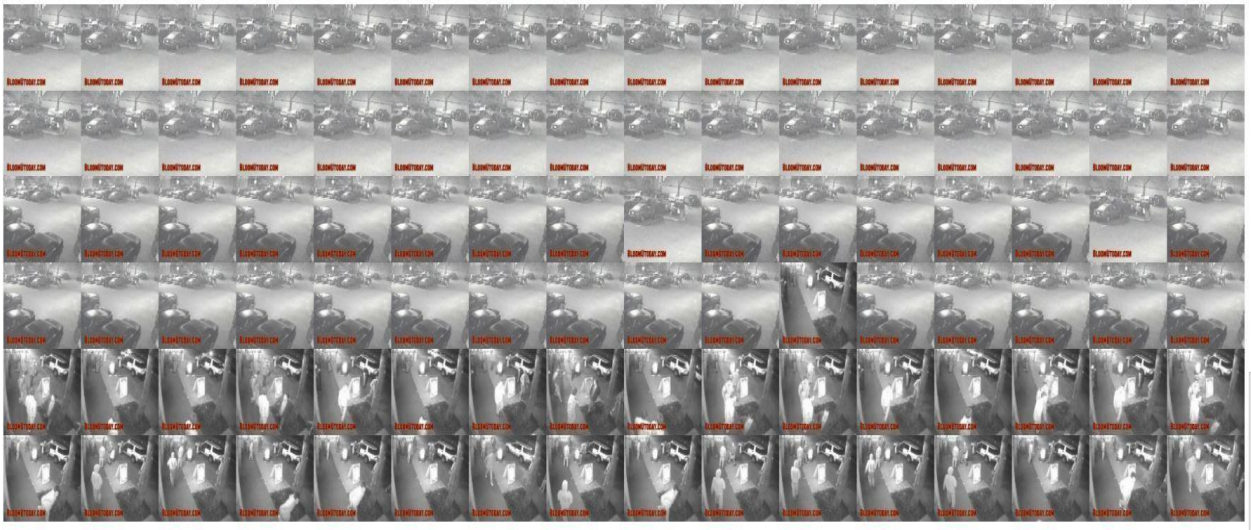

Fig.2 Consecutive frames of a sample video of ‘Fighting’ We must configure the parameters given below for each convolutionary layer.

i. Filters: The number of characteristic maps required for this convolutionary layer output.

ii. Kernel size: window size that is to be converted into a single feature map along all axes of theinput data.

iii. Strides: the number of pixels that should shift through the convolutionary window.

iv. Padding: to determine what happens on the borders–either the input is cropped (valid) or the inputis padded to the same dimension (same), with zeros.

v. Activations: The enabling function for that layer to be used. (ReLU works best with deep neuralnetworks due to its non-linearity and the ability to prevent the disappearance of gradients) [1].

This combination of alternating convolution and pooling layers follows a global pooling layer. If the spatial data are no longer left in the input (the input cannot be further reduced by pooling layers in the spatial dimension), the Global layer converts the input to a 1-d vector (with the same depth). This 1-dimensional vector is then used as the input for a dense neural network that is fully connected. There are several hidden layers and an output layer in the fully connected network. The activation function like softmax can be used for the output layer, which gives the chance that the input belongs to each class. Finally, this input is assigned to the class label with the highest probability [4].

```
from keras.applications import ResNet50
from keras.layers import Input
from keras.layers.pooling import AveragePooling2D
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.layers.core import Dropout
from keras.models import Model
```

```
baseModel = ResNet50(weights="imagenet", include_top=False, input_tensor=Input(shape=(244,244,3)))
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7,7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(512, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(len(lb.classes_), activation="softmax")(headModel)
model = Model(inputs=baseModel.input, outputs=headModel)

for basemodelLayers in baseModel.layers:
  basemodelLayers.trainable = False
```

Fig.3 Screenshot of Convolutional Model built using Keras library in Python

The proposed model was trained on the training data for 20 epochs. The weights of the model which gave the best performance on the validation data were loaded. The model was then tested on the test data. The model gave an accuracy of 86.21% as compared to 64.5% of base paper [5] on the test data. This model gave a higher accuracy than the previous models, using Image AI API for training. In this model, a pair of convolutional and max pooling layer was added.

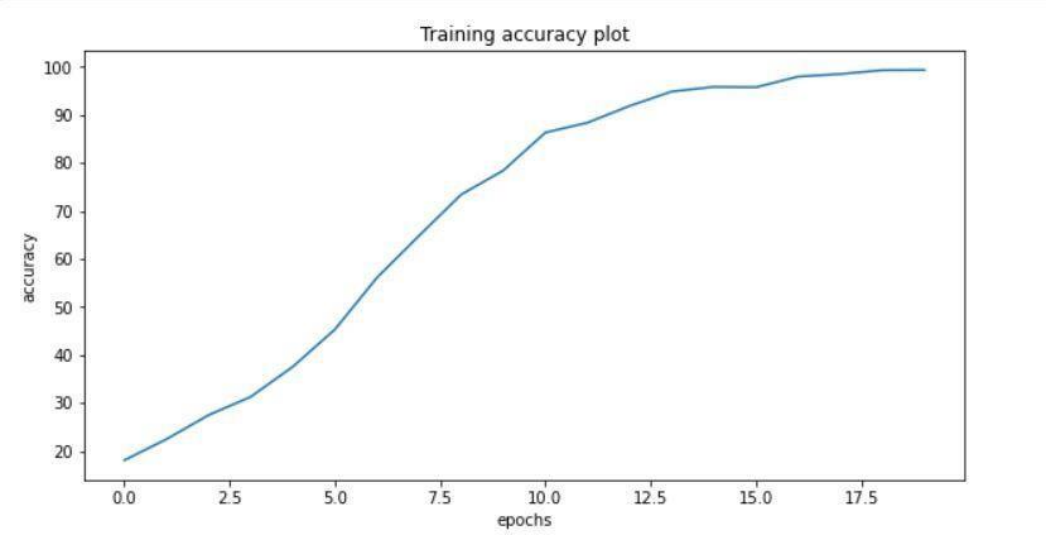

Fig.4 Screenshot of Accuracy plot produced by Model built using Keras library in Python.

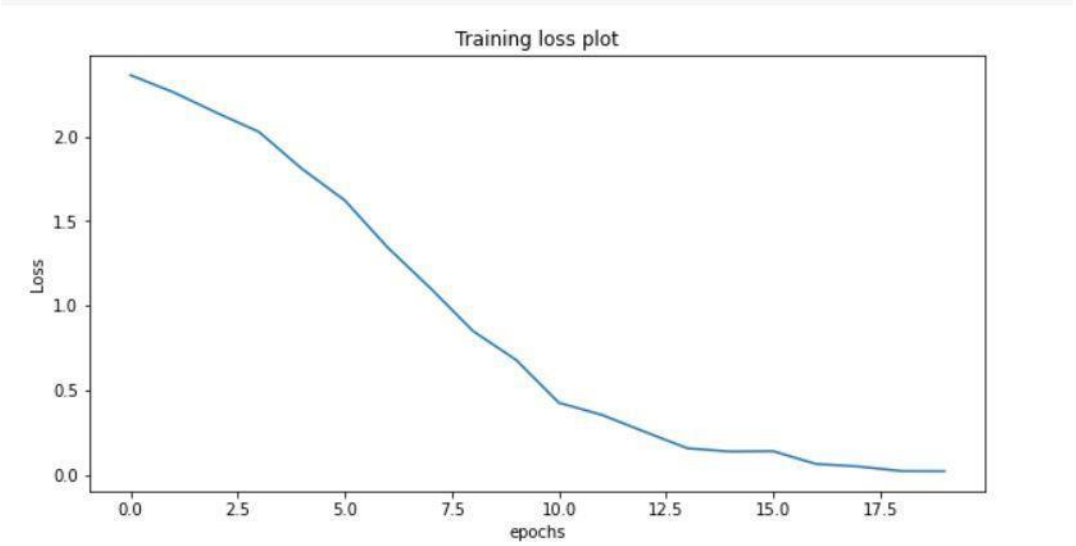

Fig.5 Screenshot of Error plot

CONCLUSION

High speed processors available for computing are not sufficient for processing deep learning models as the tensors of very large size created after preprocessing datasets. Already datasets used for deep learning video processing are normally in big size so advance processing demands GPU processing with large memory. Many standard datasets are available for video analysis to validate designed model's accuracy. In recent times it is practically possible to build a good model using very high capacity and complex libraries like Keras, and Torch [6] on Python platform to make machines intelligent, the proposed model achieved nearly 75% Accuracy.

REFERENCES

[1] Learn Computer Vision Using OpenCV - With Deep Learning CNNs and RNNs | Sunila Gollapudi | Apress. .

[2] "Video Dataset Overview.": https://www.di.ens.fr/~miech/datasetviz/.

[3] W. Sultani, C. Chen, and M. Shah, "Real-world Anomaly Detection in Surveillance Videos," ArXiv180104264 Cs, Feb. 2019.

[4] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8, doi: 10.1109/CVPR.2008.4587756.

[5] M. Jain, MrinalJain17/Human-Activity-Recognition. 2019.

[6] "Keras vs TensorFlow vs PyTorch | Deep Learning Frameworks," Edureka, 05-Dec-2018. https://www.edureka.co/blog/keras-vs- tensorflow-vs-pytorch/.

[7] International Conference on Robotics and Smart Manufacturing (RoSMa2018) Human Action Recognition using 3D Convolutional Neural Networks with 3D Motion Cuboids in Surveillance Videos.