International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 8, Issue 3, March-2023, http://ijmec.com/, ISSN: 2456-4265

DEEP LEARNING FOR HUMAN ACTION RECOGNITION

Amer Khan[1], Faisal Baig[2], Mohammed Athar[3], Mr.Somanatha Rao4
B.E Student, Department of IT, Lords Institute of Engineering and Technology, Hyderabad
Professor & HoD of IT, Lords Institute of Engineering and Technology, Hyderabad
ksrao@lords.ac.in

Abstract- The abstract describes work being done to create a model that can identify and categorize human motions including running, jogging, walking, clapping, hand-waving, and boxing from a collection of movies.The project's main goal is to develop a system that can analyze video footage and classify human behavior. Activities like running, jogging, and others fall under this category.The project makes use of a dataset including footage of people carrying out certain tasks. Each video has an attached label that describes the task performed in the clip. The primary goal of the model is to understand how the behaviors seen in the movies map to their respective labels. Without depending on written explanations, it must be able to interpret and identify the activities seen in the video. Once the model has learnt this association, it should be able to predict the action (label) for an unexplored input video. This indicates that the model can learn to detect new activities outside of its training data.In spite of having descriptions of the activities, the abstract admits that the model still has to learn to discern between different human behaviors based on the visual input alone. This implies that the model should be able to represent the complexity and subtlety of various activities based purely on picture analysis. The abstract also hints that the project's approaches might be used for more than simply human action recognition. Possible applications include learning patterns in human movement to direct different tasks and active object tracking (identifying things or individuals in CCTV video). In conclusion, this abstract provides an overview of work in progress toward developing a machine learning model that can extract semantic information from video data to identify human activities. Possible uses for the concept include directing human activities based on previously observed patterns of behavior and monitoring objects in the environment.

Key words- *SVM, KTH, CNN, Action recognition, Confusion Matrix ,Neural Networks, Deep Learning,*

I. INTRODUCTION

Advances in computer vision allow machines to solve previously intractable issues (like interpreting a picture). Given a picture, these models may make predictions about the image's content or determine whether or not a target item is there. Inspired by the structure and operation of the human brain, these models are referred to as neural networks (or rtificial neural networks). The study of these neural networks is a subject of machine learning known as "deep learning," and throughout time other variants of these networks have been produced to address a wide range of challenges. This method employs deep learning for video recognition by training a model to predict a label for an unlabeled video based on the context of a set of previously identified pictures. A video dataset has been downloaded, extracted, and preprocessed, and then split into training and testing data, a neural network has been created and trained on the training data, and lastly the model has been tested on the test data [1].

on the solving of very challenging issues (like picture recognition). Given a picture, these models may make predictions about the image's content or determine whether or not a target item is there. Neural networks (or artificial neural networks) are a kind of model that mimics the structure and behavior of the human brain. The study of these neural networks is a subject of machine learning known as "deep learning," and throughout time other variants of these networks have been produced to address a wide

range of challenges. This method employs deep learning for video recognition by training a model to predict a label for an unlabeled video based on the context of a set of previously identified pictures. A video dataset has been downloaded, extracted, and preprocessed, and then split into training and testing data, a neural network has been created and trained on the training data, and lastly the model has been tested on the test data [1].

II. METHODOLOGY

A. Dataset preprocessing

The Visual Dataset includes a large number of repetitions of six human actions (boxing, palpability, hand waving, running, leaping, and walking) across four settings (outdoor* s1*, outdoors with [s2*] size, outdoors with specific* s3* garments, and indoor* s4*). The plan will be made regardless of the conditions. The movies were recorded at a rate of 25 frames per second, with each frame sampling a resolution of 160 by 120 pixels. There are a total of 599 pictures here, with 100 films representing each of the six categories (excluding Handclapping, which only has 99). Boxing, handpicking, handshaking, jogging, and walking are the other six types [2]. Six of them, to be exact. These identifiers were transformed to integers for use in the data load process in the ways described below.

TABLE I. ACTION LABELS CODING

Action	**Coding**
Boxing	0
Handclapping	1
Hand waving	2
Jogging	3
Running	4
Walking	5

The physical size (width x height) of the movie is 160 by 120. Additionally, the shape of the array obtained when loading a single video into a NumPy array in python was (1, 515, 120, 160, 3), which means the video has 515 frames, 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).

B. 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

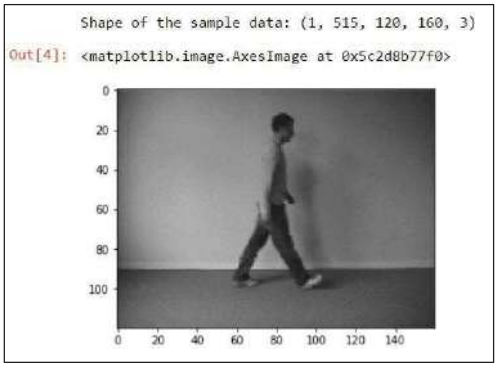

Fig. 1. Single frame of a sample video of walking of KTH dataset with shape

frames, or 25. It's well knowledge that the human body can't do anything of note in less than a second. Because of this, the vast majority of our video's frames will be unnecessary filler. This means that just a small fraction of a video's frames need to be taken out. This will help the model train more quickly and also lower the quantity of the input data, which may assist avoid 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 the first 200 frames (i.e., first 8 seconds of the video).
- 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 total of 50 frames from the video. This approach is better in the sense that we are extracting the frames *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 video will 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 ranges from 0 to 255. These values would have to be normalized in order to help our model converge faster and get a better performance. Different normalization techniques can be applied such as:

- Min-max Normalization – Get the values of the pixels in a given range (say 0 to 1)
- Z-score Normalization – This basically determines the number of standard deviations from the mean a data point is.

We would finally get a 5-dimensional tensor of shape –

(<number of videos>, <number of frames>, <width>, <height>, <channels>)

- 'channels' can have the value 1 (grayscale) or 3 (RGB)
- 'number of frames' - the extracted frames (will have to be the same for each video)

Also, the categorical labels should be encoded using a technique called **One-hot Encoding.** One-hot Encoding converts the categorical labels into a format that works better with both classification and regression models.

C. Implementation

Loading the video data collection and doing any required preparation steps was an integral element of the whole endeavor. To play and analyze photos, we thus developed a class first called Videos but later renamed read videos. Since we wanted to make this a universal function (not only for that one project), it was challenging to implement. In order to keep track of and quickly analyze the movies, we relied on NumPy (anywhere) instead of the standard Python lists. As may be seen in Figure 3, the whole person can be included among a variety of other video clips of happenings (no one doing anything). Furthermore, if the guy walked extremely slowly, most frames would be unnecessary. Such a hurdle would provide a significant obstacle for the model [3]. Using an artificial intelligence (AI) image processing system to identify and categorize images for the presence of a human body provides a workable answer to this challenge.

Class Label	Mapped Integer
Boxing	0
Handclapping	1
Handwaving	2
Jogging	3
Running	4
Walking	5

Before One-hot Encoding

Class Label	0	1	2	3	4	5
Boxing	1	0	0	0	0	0
Handclapping	0	1	0	0	0	0
Handwaving	0	0	1	0	0	0
Jogging	0	0	0	1	0	0
Running	0	0	0	0	1	0
Walking	0	0	0	0	0	1

After One-hot Encoding

Fig. 2. One hot encoding of actions

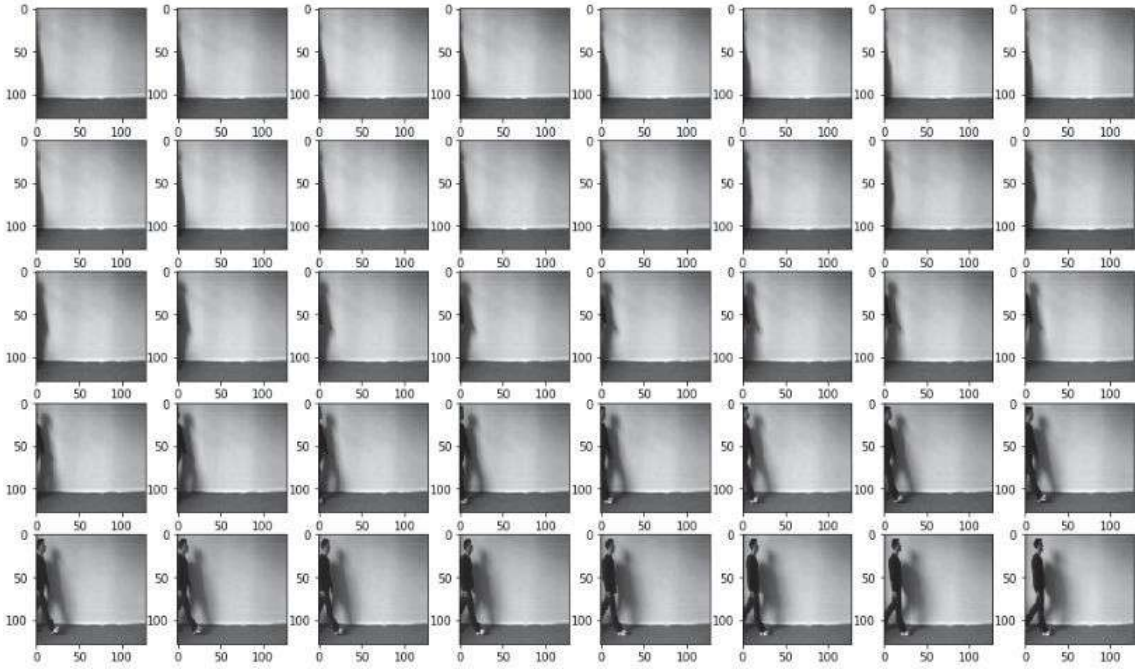

Fig 3: Consecutive frames of a sample video of 'Walking' of KTH dataset

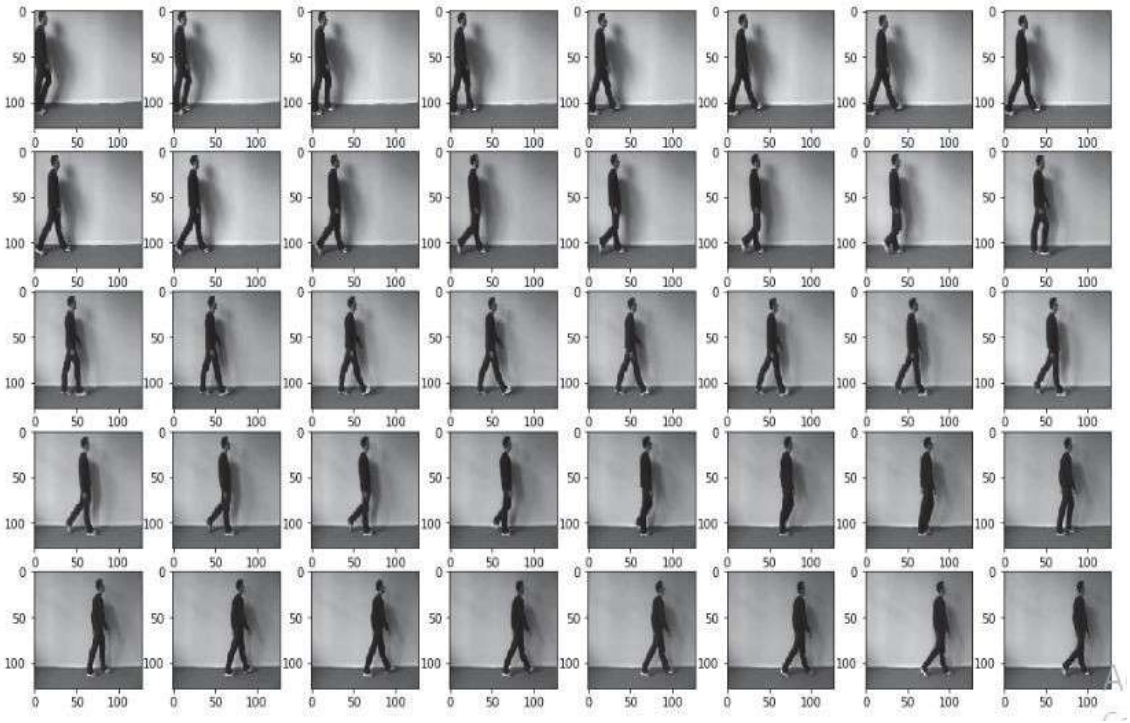

Fig. 4. The frames of a sample video of 'Walking' of KTH dataset with Image AI

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 the input 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 input is padded to the same dimension (same), with zeros.
v. *Activations:* The enabling function for that layer to be used. (ReLU works best with deep neural networks 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

In cases when there is no information no Global layer reduces the input to a 1-d vector (with the same depth) (the input can no longer be further lowered by pooling layers in the spatial dimension). A dense, fully connected neural network is fed this 1-dimensional vector as an input. The completely linked network has numerous intermediate levels, as well as an output layer. For the output layer, you may use an activation function like softmax, which estimates the probability that the input falls into each class. Finally, the most probable class label is used to categorize this data [4].

Forty epochs were used to train the suggested model on the training data. The model's weights were loaded

that produced the best results on the validation data. The model was put through its paces on test data. On the validation set, the model outperformed the reference article [5] with an accuracy of 86.21 percent. With the help of the Image AI API, this model outperformed its predecessors in terms of accuracy. A convolutional layer and a max pooling layer were included into this model.

```
# Using the Sequential Model
model = Sequential()

# Adding Alternate convolutional and pooling layers
model.add(Conv3D(filters=16, kernel_size=(10, 3, 3), strides=(5, 1, 1), padding='same', activation='relu',

                 input_shape=X_train.shape[1:]))
model.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2), padding='same'))

model.add(Conv3D(filters=64, kernel_size=(5, 3, 3), strides=(3, 1, 1), padding='valid', activation='rel
u'))
model.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2), padding='same'))

model.add(Conv3D(filters=256, kernel_size=(5, 3, 3), strides=(3, 1, 1), padding='valid', activation='rel
u'))
model.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2), padding='same'))

# A global average pooling layer to get a 1-d vector
# The vector will have a depth (same as number of elements in the vector) of 256
model.add(GlobalAveragePooling3D())

# The Global average pooling layer is followed by a fully-connected neural network, with one hidden and on
e output layer

# Hidden Layer
model.add(Dense(32, activation='relu'))

# Output layer
model.add(Dense(6, activation='softmax'))

model.summary()
```

Fig. 5. Screenshot of Convolutional Model built using Keras library in Python

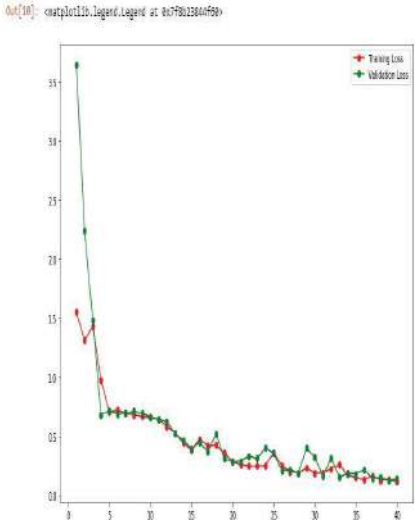

Fig. 6. Screenshot of results produced by Model built using Keras library in Python

III. 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, Theano and Torch [6] on Python platform to make machines intelligent, the proposed model achieved nearly 20% more accuracy by preprocessing the dataset as compared to the base model[5].

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/.