IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025

DOI: 10.17148/IJARCCE.2025.14104

A New Image Classification Method at A High-Speed Using a Deep Convolutional (Conv2D) and Long Short-Term Memory (LSTM) Algorithm

Fatin A. Hamadain[1], Abdalla A. Osman[2], Ahmed Abdelrahman Mohamed Hamed[3]

Graduate College University of Bahri, Khartoum, Sudan[1]

Department of Computer Sciences, University of Elahlia, Wad Madani, Sudan[2]

Department of Computer Sciences and Mathematics, University of Bahri, Khartoum, Sudan[3]

Abstract: Recently, the object classification in digital images and videos has been addressed in various research works. Convolutional neural networks (CNN) are effective to process image data, while long-short term memory (LSTM) networks are effective to process sequence data. However, when these two technologies are combined, the result is a solution to challenging computer vision problems, such as video classification. The process of image classification involves passing an image to a classifier, which can be either a trained CNN or a classical classifier and obtaining class predictions. An LSTM is engineered specifically to operate with a data sequence, as it considers all the previous inputs when producing an output. LSTMs are a form of neural network known as a Recurrent Neural Network (RNN). In general, RNNs are not known to be effective in addressing the long-term dependencies in the input sequence due to a problem known as the vanishing gradient problem. LSTMs are created to circumvent the vanishing gradient, thereby enabling an LSTM cell to retain context for lengthy input sequences. This paper is intended to rapidly design a novel image classification method by using Conv2D and LSTM algorithms. The number of filters, the size of the filters, the activation function, and the padding mode are among the numerous parameters that the Conv2D function accepts. The HiSVidClassifer method applied time-distributed Conv2D layers, followed by MaxPooling2D and Dropout layers. The Flatten layer is used to flatten the feature extracted from the Conv2D layers, which are then transmitted to an LSTM layer for analysis to classify and detect the object. Our HiSVidClassifer method is trained and evaluated on UCF50 dataset. It achieved outstanding results with low loss equals to 0.1935, and good accuracy equals to 95.08%, compared to ConvLSTM method, which obtained loss equal to 0.3773 and the accuracy equal to 87.70%.

Keywords: Image classification method, video classification, object detection, computer vision, deep learning.

I. INTRODUCTION

Machine learning algorithms that can understand the semantic content of videos have greatly improved as it has become easier to share and watch videos. There are many uses for video classification, including computer vision, search engine optimization, and summarization. Videos are distinct from images in that they also have a time dimension. Image classification is a good place to start because they are essentially a collection of individual images [1]. In the context of video, it is reasonable to presume that the semantic contents of subsequent frames are interconnected. The most straightforward approach to classifying videos is to run each frame from a video file through a CNN, classify each frame separately and independently of the others, select the label with the highest corresponding probability, label the frame, and assign the video the label that has been assigned to the most images, as a video is merely a collection of frames [2]. It is a very challenging task to train neural networks on video due to the enormous amount of data needed. Common techniques involve using the entire training dataset of videos to train an image-based network.

Utilizing basic image classification for video categorization will result in rapid alterations of the film's label as various scenes are identified. The phrase "prediction flickering" denotes these phenomena. Subsequent frames in a video are presumed to possess identical semantic meaning. If the assumption that prior frames are analogous to the current frame holds true, we can leverage the temporal characteristics of videos. Consequently, by refining the predictions, we may improve the video classifier through averaging. An effective video classifier not only delivers precise frame labels but also comprehensively characterizes the entire video based on the features and annotations of the individual frames. In a

video, a tree may be present in one frame, yet the label may serve as the primary focus. The required specificity of the labels for describing the frames and the video will vary based on the task [3].
Convolutional Neural Networks (CNNs) are commonly employed to tackle problems related to spatial data, including photographs. The dimensions of the input and output are both constant. A CNN processes images of a predetermined size and produces them at the appropriate resolution. Researchers have utilized deep convolutional networks (CNNs) to attain superior performance in picture classification. This design has proven to be an effective model class for comprehending content associated with image identification, segmentation, detection, and retrieval. Convolutional Neural Networks necessitate extensive training durations to effectively optimize the millions of parameters that define the model [24]. The network faces increased difficulty due to the necessity of processing several video frames simultaneously, in addition to a single image, as the architecture's connectivity expands over time [4]. We demonstrate that enhancing the architecture by incorporating two distinct processing streams—a context stream that extracts features from low-resolution frames and a high-resolution fovea stream that functions solely on the central portion of the frame—can markedly enhance the runtime performance of CNNs.

An additional challenge in training video classifiers is figuring out how to feed the videos to a network, as CNNs require inputs of a standardised length. We could simply extract the frames from a video and place them in a 3D tensor because a video is an ordered sequence of frames. However, as each video may have a different number of frames, we are unable to stack them into batches. Alternatively, we could store video frames until a certain number of frames is reached at a predetermined interval. Determining or forecasting the course of an action carried out by one or more individuals is known as "human activity recognition." It is a particular kind of time series classification problem in which the correct classification of the action being performed requires data from a sequence of timestamps [5]. For human action recognition, as opposed to object detection, a sequence of data points is required for the machine-learning algorithm to precisely forecast the activity, which is derived from video files. Three primary categories can be used to categorise human activity recognition, including:

- Simple Activity Recognition - This category includes short videos featuring a single person clearly carrying out a task; the video's duration solely consists of the activity like running, playing football, and opening doors [3].
- Temporal Activity Recognition/Localization - This also applies to lengthier videos that have multiple activities. It is necessary to use models for temporal activity localization whose architecture is able to both classify and localise individual actions into temporal proposals, such as video clips [6].
- Spatio-Temporal Detection - In this kind of video, there are multiple humans in the same frame, possibly executing distinct tasks, as opposed to just one person performing numerous actions. The system must recognize and find each individual in the video and categorize the jobs they are undertaking to accomplish this objective. Similar to temporal activity recognition, it must also document the duration of each action [5].

II. IMAGE FEATURE EXTRACTING

A concept in image processing and computer vision is feature extraction. In order to ascertain whether the points in each image are part of an image feature extraction, it describes the process of using a computer to extract information from images. Feature extraction seeks to partition the image's points into distinct subsets, often comprising single points, continuous curves, or regions. Numerous features can generally characterize an image. Depending on how the features are represented in the image data, these features can be categorised using a variety of criteria, including point, line, and regional characteristics. Features can be categorised into two groups based on their region size: global features and local features [7].
Once the feature vectors from the image are recovered, the image can be represented as a fixed-length vector, necessitating a classifier to categorize the feature vectors. From input to output, a typical convolution neural network typically consists of the following layers: input, convolution, activation, pool, full connection, and final output. The convolutional neural network layer transfers layer by layer, the input information; this also establishes connections among diverse computational neural nodes. Simultaneously, the feature signals of the original data are decoded, inferred, converged, and mapped to the hidden layer feature space through the continuous convolution-pool architecture. The retrieved characteristics are utilized to classify and produce data in the ensuing fully connected layer. Here are some techniques for extracting features from an image:

A. Convolutional Neural Networks(CNN): When it comes to feature extraction from images, CNNs are typically the better option because they are made expressly for processing colour images and can handle more complicated tasks such as object detection, segmentation, and image classification. These tasks enable CNNs to extract intricate features from photos, irrespective of variations in illumination, scale, and other factors. This is the most efficient technique for processing photos with a high level of precision. [8].

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025

DOI: 10.17148/IJARCCE.2025.14104

B. Gray-scale features: It is helpful when the task at hand does not require or warrant the use of colour information. A few instances include: Image similarity: Comparing two images' similarity, OCR (Text Recognition): Extracting text from images, Edge detection: Recognising borders or edges in a picture, Medical Imaging: To detect tumours or blood vessels in the image, analyse grayscale medical images from CT, MRI, and X-rays, and Facial recognition: Is the process of obtaining details about a person's face texture in order to identify them [9].

C. Mean pixel values of channels: Calculating the mean intensity of the red, green, and blue color channels in a picture is a crucial step in the feature extraction process. When achieving basic image representation or performing simple image processing tasks like thumbnail creation, image compression, and resizing is the aim, this method may prove helpful. However, for more complicated tasks like segmentation, object detection, or image classification, mean pixel value of channels is usually not a good approach. The reason for this is that the mean pixel value of a channel ignores both the correlation between various channels and the spatial relationship between the pixels in the image [10].

D. Edge Features: Various image processing tasks, including object detection, motion detection, picture segmentation, medical imaging, and optical character recognition (OCR), can derive advantages from its application. These tasks entail identifying and evaluating edges or boundaries in an image. Edge features can be used to spot abrupt changes in image region or pixel intensity. When performing basic image processing tasks like edge detection or boundary extraction, edge features can be useful. When the edges are clearly defined and the image data is relatively simple, edge features can also be helpful [11].

E. Autoencoders: When the amount of labelled data is limited or the CNN architecture is too computationally expensive, autoencoders can be used for feature extraction. In this application, features that can be utilised in subsequent machine learning models are extracted by training the autoencoder on an image dataset. When learning a compact representation of the input data is the aim of unsupervised learning tasks like anomaly detection, dimensionality reduction, image denoising and compression, and the creation of new data points, autoencoders are typically utilised. They can assist in removing noise and extracting pertinent features, which makes them especially helpful when the input data is noisy and high dimensional. CNNs are a better option if the task involves supervised learning, like object detection or image classification. In some circumstances, autoencoders and CNNs can be combined to extract features. In transfer learning, for instance, features from an image, dataset can be extracted using an autoencoder and a CNN can be used to refine the features using a smaller labelled dataset. The CNN can start enhancing its performance by using the features that the autoencoder has learned [12].

F. Histogram of Oriented Gradients (HOG): HOG is a feature descriptor that computes histograms of oriented gradients in specific areas of an image to extract gradient information. Computer vision tasks like text classification, object detection, face detection, pedestrian detection, and OCR frequently use HOG. HOG might be a better option for feature extraction in some circumstances, even though CNNs have demonstrated superior performance in a variety of image-related tasks. An example of this would be if the dataset were small and there is insufficient data or computational power to support the training of a deep learning model. Under certain circumstances, HOG can offer a straightforward and practical feature extraction method that is easily integrated with a common machine-learning algorithm, like Support Vector Machines (SVM). Another situation in which HOG could be helpful is if the images are extremely pixelated or have a low resolution. HOG is a good option for feature extraction in these situations since the gradients might offer more reliable features than the raw pixel values [13].

G. Scale-Invariant Feature Transform (SIFT): Based on key point detection and local feature descriptor extraction, SIFT extracts features. Often utilised in computer vision applications like those that object recognition, image stitching, and three-dimensional reconstruction, SIFT features are invariant to scale, orientation, and affine transformations. When extracting features from images, SIFT or CNN should be used depending on a number of factors, including the problem at hand, the computational capacity, and the size of the dataset (which may be limited) [14].

H. Local Binary Patterns (LBP): Analysis of texture information in images is commonly done using Local Binary Patterns (LBP), a feature extraction technique. By comparing the grey values of a pixel to those of its neighbours, LBP encodes the result into a binary pattern that represents the local structure of an image. In situations where there is a lack of data, LBP features are frequently employed in machine learning algorithms for a variety of computer vision tasks, including object recognition, texture analysis, and face recognition. In real-time applications where processing speed is critical, like tracking and surveillance systems, LBP can function with computational efficiency [15].

I. Frequency-based features: The frequency content of an image can be extracted using a feature extraction technique called frequency-based features, such as Fourier descriptors. Applications in computer vision, including object recognition, texture analysis, data compression, and image retrieval, frequently make use of these characteristics.

When there is a lack of data and a need for both rotational and spatial invariance, frequency-based features are employed [16].

J. Color-based features: Image retrieval and colour distribution and statistics are captured by color-based features, a kind of feature extraction technique. Computer vision applications including content-based image retrieval, object recognition, and image segmentation frequently use these features. Used in scenarios with limited data, it is more computationally efficient than CNNs [10].

III. VIDEO CLASSIFICATION

Computer vision and machine learning are rapidly advancing topics in video classification. Video classification operates similarly to image classification, aiming to create algorithms capable of identifying and categorizing the content of videos with near-human precision, subsequently organizing them into specific categories such as actions (dancing, walking, running, etc.) or emotional behaviours (happy, sad, surprised, etc.) [17].

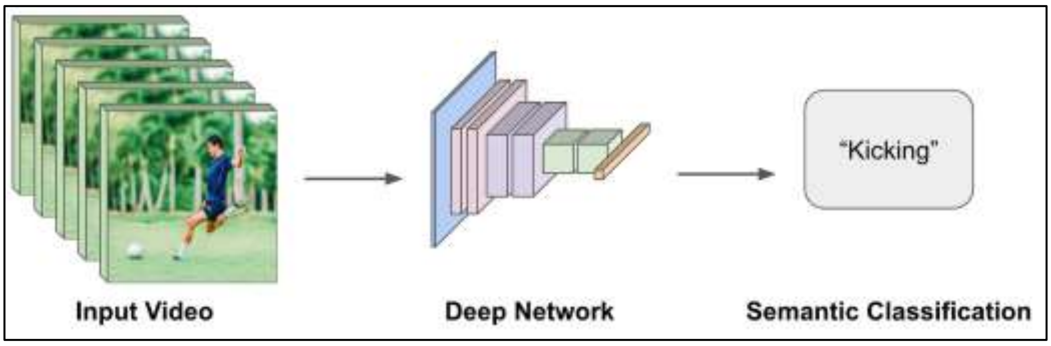


Figure 1: video classification

Videos must be classified according to both their spatial and temporal content, whereas photos are classified only based on their spatial content (e.g., an image of a person versus an image of a dog). This occurs because two movies may depict the same individual (same spatial information) executing distinct behaviours (variations in temporal content). The substantial volume of data requiring processing becomes a primary challenge in video categorization. A typical video comprises several frames, each containing a wealth of information. Films can be captured at diverse frame rates, from many perspectives, and under different lighting conditions, which adds complexity. The most prevalent video classification issues are as follows [3]:

- Scalability: This video data keeps getting bigger, making it harder and longer to handle and sort it all in a reasonable amount of time. These changes have a big impact on deep learning methods that need a lot of data and computer power to train.
- Generalisation: There are many algorithms for classifying movies that are designed to work well with a certain dataset or task. However, they may not work well with other datasets or tasks.
- Video annotation: Labelling extensive video data requires significant time and effort. Consequently, acquiring extensive, high-quality datasets for the training and evaluation of video classification algorithms, especially supervised learning models, may prove difficult.
- Security and privacy: The use of video data presents a number of privacy and security issues, especially in relation to face recognition and surveillance. As with Deep Fake videos, there are worries regarding possible misuse of video data.
- Video quality: A video classifier's performance may be negatively impacted by the different quality and format variations that can be challenging to manage.

IV. VIDEO CLASSIFICATION METHOD

A variety of deep learning-based video classification methods has been developed. The majority utilize supervised learning, a paradigm that instructs neural networks (either convolutional or recurrent) with videos and their corresponding labels. As labelled data are costly and time-consuming to gather, current approaches concentrate on minimising their dependence. While methods with low supervision may seem ideal in theory, accuracy is often sacrificed in practise. It therefore depends on the use case to determine whether to employ supervised or low supervision techniques. Accuracy is prioritised over computational load in sensitive applications, such as surveillance and medical. Conversely, models with relatively lower accuracy can be used for general-purpose action recognition, like sports, or entertainment.

Nevertheless, because these domains frequently contain substantial volumes of readily available unlabelled data, they still require effective storage options. Supervised and unsupervised methods for video classification include the following [18]:

A. Supervised: In supervised learning, a model is trained to produce predictions utilizing labelled data. To evaluate a model, a collection of labelled data is partitioned into training, validation, and testing subsets. Subsequently, unlabelled data is employed to qualitatively assess the performance of the trained model. Supervised learning can be utilized for video categorization to train a model to recognize specific objects, behaviours, or sequences inside a video. [19][24].

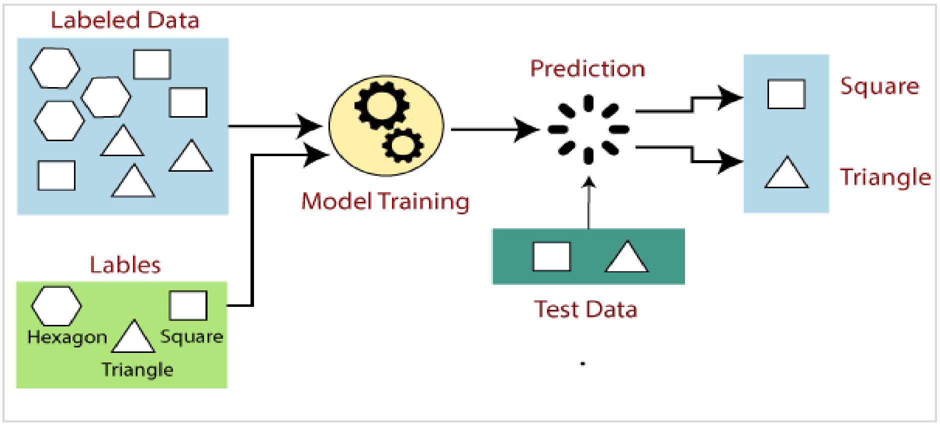


Figure 2: Supervise machine learning

B. Semi-Supervised: Semi-supervised learning is a machine learning method that entails training a model using both labelled and unlabelled data. In these scenarios, the ratio of labelled data within the dataset is generally minimal (5%–10%) and utilized seldom, whereas a substantial volume of unstructured data is readily available [20][24].

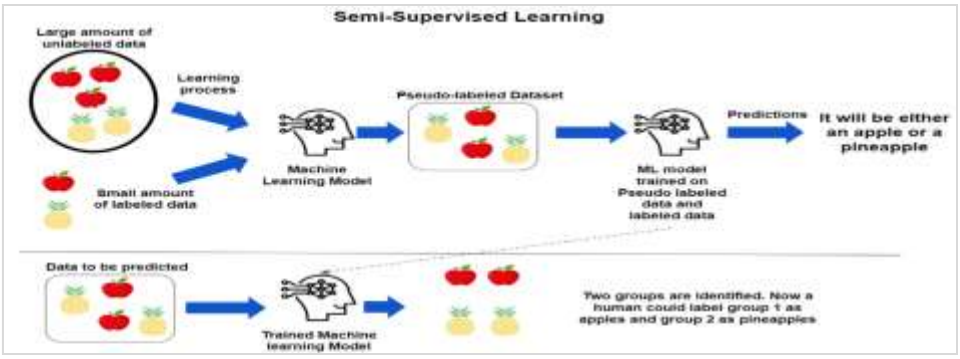

Semi-Supervised Learning
Large amount of unlabeled data
Learning process
Small amount of labeled data
Machine Learning Model
Pseudo-labeled Dataset
ML model trained on Pseudo labeled data and labeled data
Predictions
It will be either an apple or a pineapple
Data to be predicted
Trained Machine learning Model
Two groups are identified. Now a human could label group 1 as apples and group 2 as pineapples

Figure 3: Semi-Supervised

C. Weakly-Supervised: A specific type of machine learning problem is weakly-supervised video classification, which involves training a model to categorize videos using weak labels, meaning that explicit classification labels are absent, although potentially valuable information is present. Conversely, a substantial quantity of labelled data, encompassing the class labels for each frame or segment of the video, is utilized to train the model in fully supervised video classification [19][24].

DOI: 10.17148/IJARCCE.2025.14104

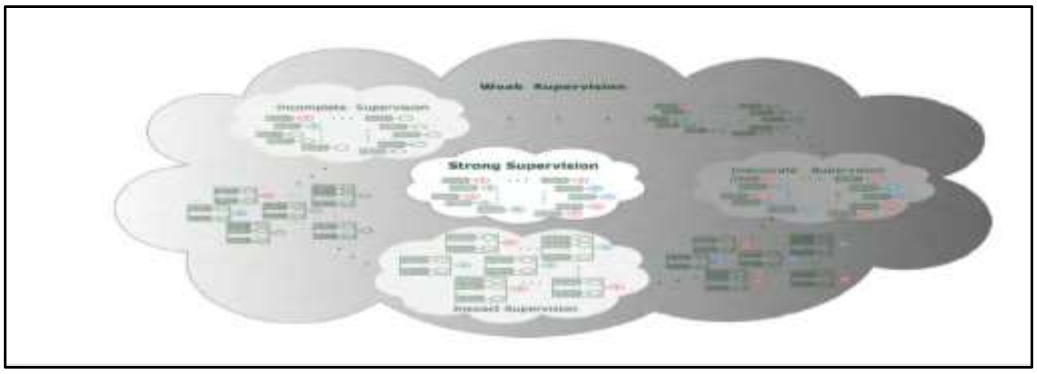


Figure 4: Weakly-Supervised

D. Self-supervised: Self-supervised learning is a special kind of unsupervised machine learning in which a model creates (pseudo) labels based on training data samples. The highly confidently predicted pseudo-labels from this set serve as the ground truth for the subsequent training algorithm iteration. In this manner, the training dataset is fully labelled, and the network is suitably trained before being assessed on a test set [20][24].

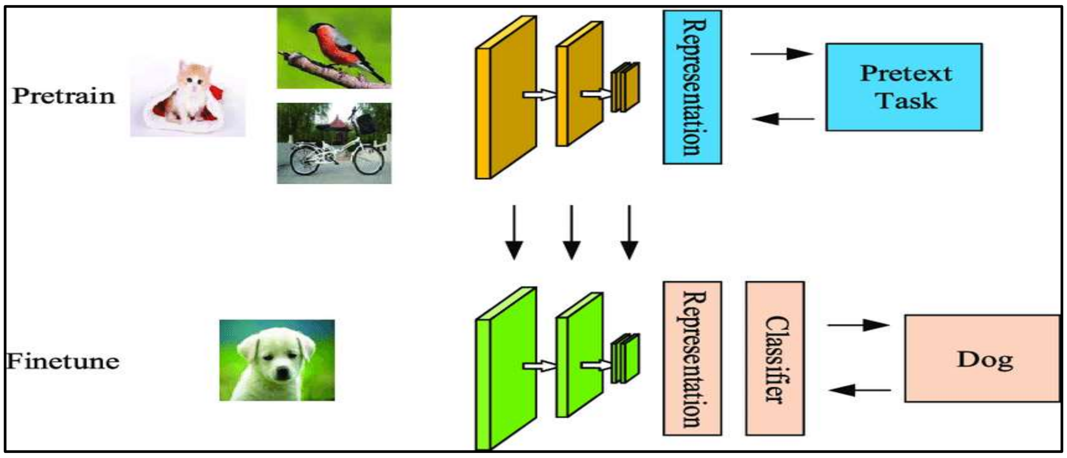


Figure 5: Self-supervised

V. THE PROPOSED METHOD

The approach of our proposed classification method at a high-speed using a deep convolutional (Conv2D) and Long Short-Term Memory (LSTM) algorithm, namely HiSVidClassifer), consists of consecutive phases as shown in Figure 1, as follows [8]:

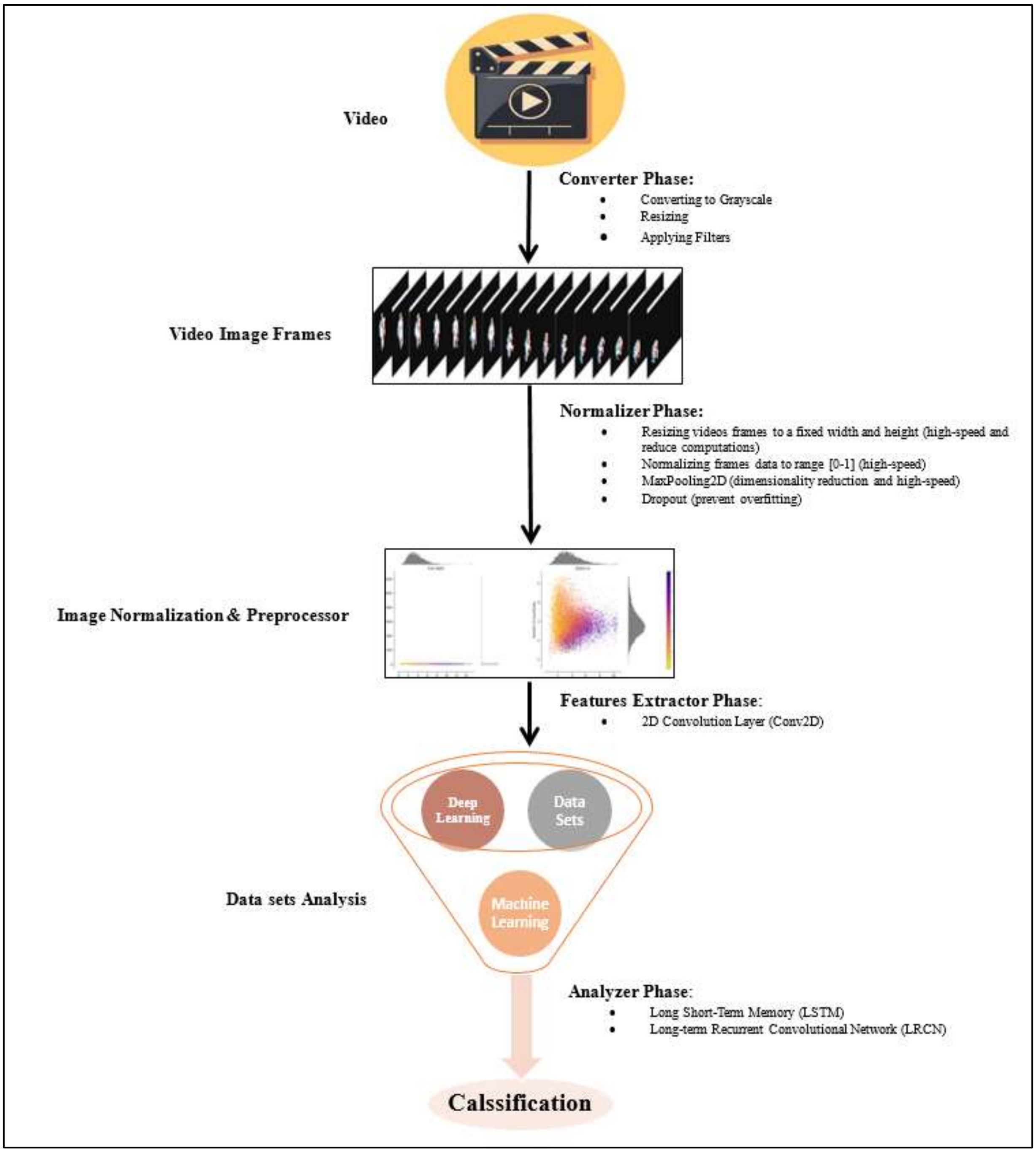

Video
Converter Phase:
• Converting to Grayscale
• Resizing
• Applying Filters
Video Image Frames
Normalizer Phase:
• Resizing videos frames to a fixed width and height (high-speed and reduce computations)
• Normalizing frames data to range [0-1] (high-speed)
• MaxPooling2D (dimensionality reduction and high-speed)
• Dropout (prevent overfitting)
Image Normalization & Preprocessor
Features Extractor Phase:
• 2D Convolution Layer (Conv2D)
Deep Learning
Data Sets
Data sets Analysis
Machine Learning
Analyzer Phase:
• Long Short-Term Memory (LSTM)
• Long-term Recurrent Convolutional Network (LRCN)
Calssification

Figure 6: A proposed classification method (HiSVidClassifer) at a high-speed using a deep convolutional (Conv2D) and Long Short-Term Memory (LSTM) algorithm.

A. Converter phase: This phase aims to convert video-to-video images frames for recognition purposes. A video is made up of a series of images playing in sequence. This phase defines how you can convert a video to image frames using OpenCV tool, where frames are read from the video in order to benefit from them in the classification phase. OpenCV is a library that is used to provide tools for reading image frames from video files. OpenCV, an acronym for Open Source Computer Vision Library, is a robust library of programming functions primarily designed for real-time computer vision applications. This library is designed to impart essential methods for reading video files, taking video from cameras, processing video frames, and writing these frames to a new video file. Reading videos is the first critical step in video processing with OpenCV. In this step defines the essential function cv2.VideoCapture (), a versatile method used to capture video from video files or directly from a camera [21].

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025

DOI: 10.17148/IJARCCE.2025.14104

Once video frames are red, the next step is to process them, which involves various operations such as converting to grayscale, resizing, or applying filters. Here is the some common frame processing techniques:

- Converting to Grayscale: This simplifies the frame to a single channel, making it easier for further processing or analysis.
- Resizing: This changes the frame dimensions, which can be useful for reducing computation or fitting specific display requirements.
- Applying Filters: This is like blurring or edge detection enhances or reduces features within frames.

After processing video frames, the results are saved into a new video file. OpenCV provides a straightforward way to accomplish this through the cv2.VideoWriter class.

B. Normalizer phase: This step seeks to pre-process video image frames based on specific criteria, wherein several images are standardized to a common statistical distribution regarding size and pixel values. Furthermore, it seeks to normalize the pixel values before supplying the photos as input to a deep learning neural network model during the model's training or evaluation.
Initially, this research acquired and visualized the data along with labels to understand the subject matter at hand. The UCF50 – Action Recognition Dataset utilized realistic footage sourced from YouTube, distinguishing it from most other action recognition datasets, which are typically staged and performed by actors. The UCF50 dataset has 50 action categories, with 25 video groups per category, an average of 133 videos per category, 199 average frames per video, 320 average frame width per video, 240 average frame height per video, and 26 average frames per second per video.
In this research, 20 random categories are selected from UCF50 dataset, and is split into 75% for train dataset and 25% for test dataset.
Subsequently, it conducted pre-processing on the dataset, including shrinking the video frames to a uniform width and height to minimize computational demands. Next, it standardized the data to range [0-1] by dividing the pixel values with 255, which makes convergence faster during training the network [22].
Afterward, a MaxPooling2D, a prevalent strategy employed in convolutional neural networks for feature extraction and dimensionality reduction, is utilized in this step to diminish the dimensions of the frames and circumvent superfluous computations. The process entails partitioning the input feature map into a grid of non-overlapping rectangular areas, and for each area, identifying the maximum value to represent it in the output feature map. The primary advantages of MaxPooling2D are:

- Downsampling: By selecting the greatest value in each zone, Max Pooling 2D decreases the spatial dimensions of the feature map, thus Downsampling the input. This mitigates the computational complexity of the network and regulates overfitting.
- Feature Extraction: Max Pooling 2D retrieves the most significant features from each region by picking the maximum value, hence eliminating less critical information. This enables the network to concentrate on the most prominent aspects relevant to the task.

Eventually, the normalizer phase utilized the Utilize dropout layers to mitigate model overfitting on the dataset.

C. Features extractor phase: During this step, essential video attributes are retrieved for recognition objectives. When the input data for an algorithm is excessively large and deemed redundant, it might be transformed into a condensed set of features. The chosen features are anticipated to encapsulate the essential information from the input data, enabling the accomplishment of the intended job through this condensed representation rather than the entire original dataset. A function named create_dataset() is developed to extract features by iterating through all classes defined in the CLASSES_LIST constant. It invokes the frame_extraction() function for each video file corresponding to the selected classes, returning the frames (features), class indices (labels), and video file paths (video_files_paths). The previously defined method create_dataset() is employed to extract data from the selected classes and generate the necessary dataset [23].
Conv2D serves as a fundamental component for constructing convolutional neural networks, executing a two-dimensional convolution operation on input pictures. Conv2D is engineered to extract features or patterns from an input image by utilizing a collection of learnable filters. The filters are acquired during the training phase, enabling the model to identify the most pertinent features from the input image for a certain job. The Conv2D function accepts multiple parameters, such as the quantity of filters, the dimensions of the filters, the activation function, and the padding mode. The model employed time-distributed Conv2D layers, succeeded by MaxPooling2D and subsequently Dropout layers. The features taken from the Conv2D layers are subsequently flattened using the Flatten layer and input into an LSTM layer for the purpose of classification and object detection.

D. Classifier phase: The aim of this phase is to analyze the extracted video features from the previous phase to detect and classify objects into predefined categories using an appropriate classification technique that compares image

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁙ Peer-reviewed & Refereed journal ⁙ Vol. 14, Issue 1, January 2025

DOI: 10.17148/IJARCCE.2025.14104

patterns with target patterns through high-speed performance. The object's action was predicted by employing a Dense layer with Softmax activation applied to the output of the LSTM layer.

Long Short-Term Memory (LSTM) is an enhanced variant of recurrent neural networks. A conventional RNN possesses a singular hidden state that propagates through time, complicating the network's ability to acquire long-term dependencies. LSTM models tackle this issue by incorporating a memory cell, which serves as a reservoir for retaining information over prolonged durations. LSTM architectures can learn long-term dependencies in sequential data, making them ideal for tasks like language translation, speech recognition, and time series forecasting.

This study employed the Long-term Recurrent Convolutional Network (LRCN) methodology, integrating CNN and LSTM layers inside a unified model. The LRCN method is selected because to its superior accuracy, enhancing overall performance. The Convolutional layers facilitate spatial feature extraction from the frames, which are subsequently input into LSTM layer(s) at each time step for temporal sequence modelling. Consequently, the network acquired spatiotemporal properties through end-to-end training, yielding a robust model. The LRCN technique integrates Convolutional and LSTM layers inside a unified model. A comparable method involves training a CNN model and an LSTM model independently. The CNN model is employed to extract spatial characteristics from video frames, utilizing a pre-trained model that may be fine-tuned for the specific problem. Furthermore, the LSTM model utilizes the information retrieved by the CNN to anticipate the action occurring in the video [8].

VI. ANALYSIS AND DISCUSSION

In this research, HiSVidClassifer method recognizes the classes of randomly chosen video files from retrieved UCF50 dataset via reading the first frame of the video file and writes the class name on the video frame, and then displays the frame, as shown in Figure 7.

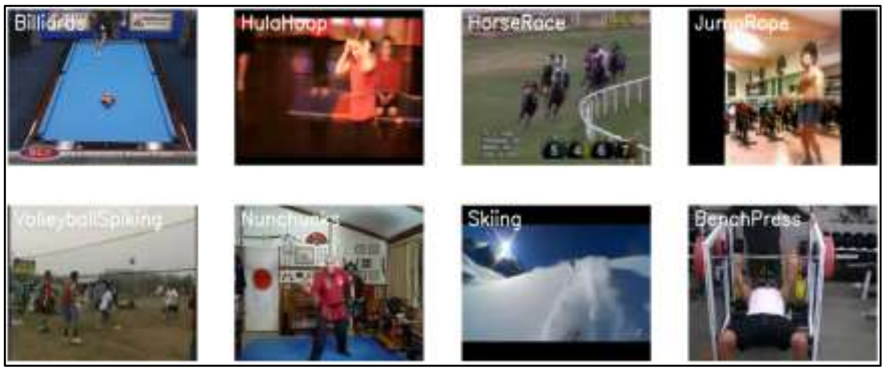


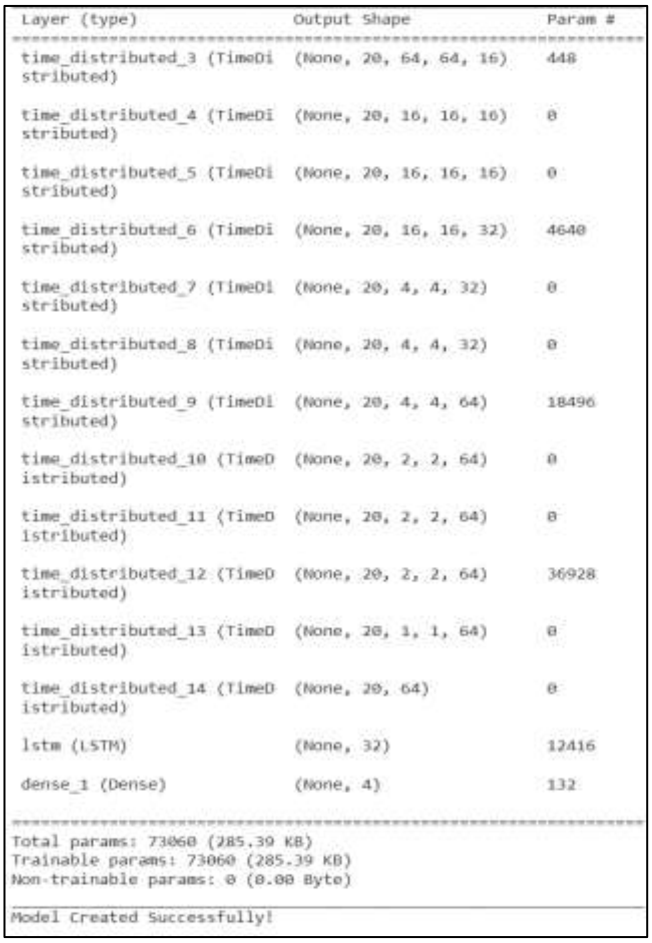
Figure 7: Randomly selected video file from UCF50

The HiSVidClassifier method initially extracts data from the selected classes and constructs the requisite dataset, as illustrated below:

- features: a compilation of the extracted frames from the movies,
- labels: a compilation of the indices corresponding to the classes linked with the videos, and
- video_files_paths: a compilation of the file paths of the films stored on the disk.

Subsequently, HiSVidClassifier traverses all classes in the classes list and all files in the files list, extracting frames from the video files. It verifies whether the extracted frames equal the SEQUENCE_LENGTH, thereby disregarding videos that contain fewer frames than the SEQUENCE_LENGTH value.
Then, HiSVidClassifer method employes LRCN model, and utilizes a sequential model for model construction as shown in the Figure 8. The total parameters of HiSVidClassifer method are 73060, trainable model parameters are 73060, and non-trainable model parameters are 0.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025
DOI: 10.17148/IJARCCE.2025.14104



```
Layer (type)                Output Shape              Param #
=================================================================
time_distributed_3 (TimeDi  (None, 20, 64, 64, 16)    448
stributed)

time_distributed_4 (TimeDi  (None, 20, 16, 16, 16)    0
stributed)

time_distributed_5 (TimeDi  (None, 20, 16, 16, 16)    0
stributed)

time_distributed_6 (TimeDi  (None, 20, 16, 16, 32)    4640
stributed)

time_distributed_7 (TimeDi  (None, 20, 4, 4, 32)      0
stributed)

time_distributed_8 (TimeDi  (None, 20, 4, 4, 32)      0
stributed)

time_distributed_9 (TimeDi  (None, 20, 4, 4, 64)      18496
stributed)

time_distributed_10 (TimeD  (None, 20, 2, 2, 64)      0
istributed)

time_distributed_11 (TimeD  (None, 20, 2, 2, 64)      0
istributed)

time_distributed_12 (TimeD  (None, 20, 2, 2, 64)      36928
istributed)

time_distributed_13 (TimeD  (None, 20, 1, 1, 64)      0
istributed)

time_distributed_14 (TimeD  (None, 20, 64)            0
istributed)

lstm (LSTM)                 (None, 32)                12416

dense_1 (Dense)             (None, 4)                 132

=================================================================
Total params: 73060 (285.39 KB)
Trainable params: 73060 (285.39 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Model Created Successfully!
```

Figure 8: Parameters Of Hisvidclassifer Method.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025
DOI: 10.17148/IJARCCE.2025.14104

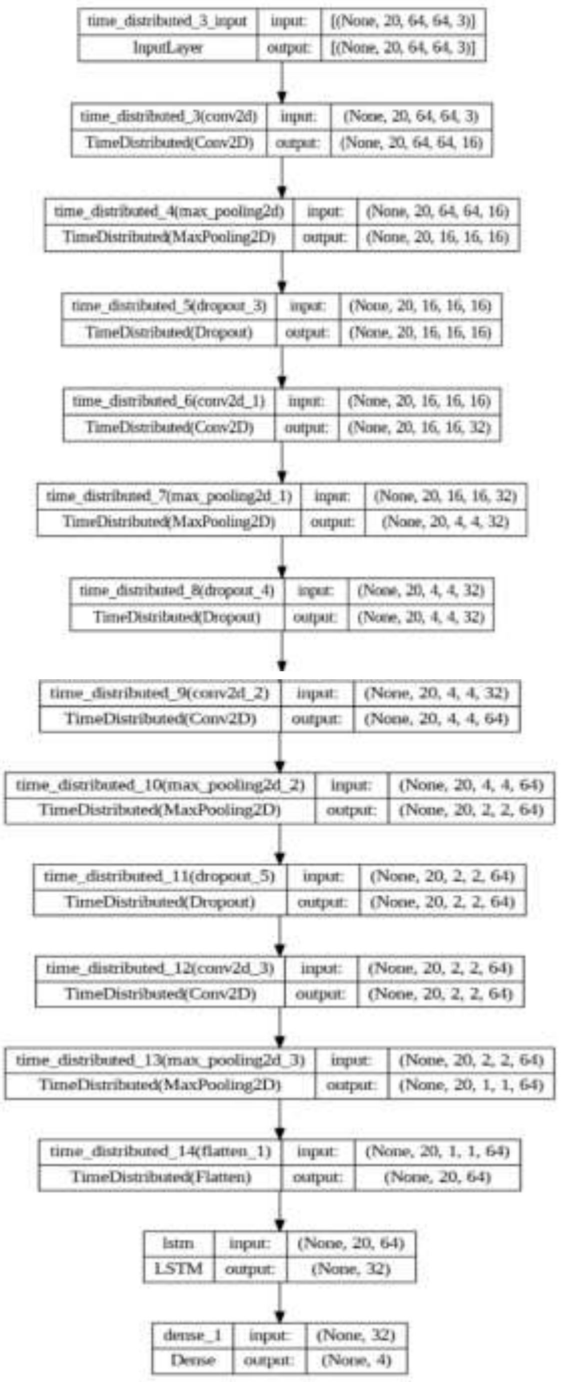

time_distributed_3_input | input: | [(None, 20, 64, 64, 3)]
InputLayer | output: | [(None, 20, 64, 64, 3)]
time_distributed_3(conv2d) | input: | (None, 20, 64, 64, 3)
TimeDistributed(Conv2D) | output: | (None, 20, 64, 64, 16)
time_distributed_4(max_pooling2d) | input: | (None, 20, 64, 64, 16)
TimeDistributed(MaxPooling2D) | output: | (None, 20, 16, 16, 16)
time_distributed_5(dropout_3) | input: | (None, 20, 16, 16, 16)
TimeDistributed(Dropout) | output: | (None, 20, 16, 16, 16)
time_distributed_6(conv2d_1) | input: | (None, 20, 16, 16, 16)
TimeDistributed(Conv2D) | output: | (None, 20, 16, 16, 32)
time_distributed_7(max_pooling2d_1) | input: | (None, 20, 16, 16, 32)
TimeDistributed(MaxPooling2D) | output: | (None, 20, 4, 4, 32)
time_distributed_8(dropout_4) | input: | (None, 20, 4, 4, 32)
TimeDistributed(Dropout) | output: | (None, 20, 4, 4, 32)
time_distributed_9(conv2d_2) | input: | (None, 20, 4, 4, 32)
TimeDistributed(Conv2D) | output: | (None, 20, 4, 4, 64)
time_distributed_10(max_pooling2d_2) | input: | (None, 20, 4, 4, 64)
TimeDistributed(MaxPooling2D) | output: | (None, 20, 2, 2, 64)
time_distributed_11(dropout_5) | input: | (None, 20, 2, 2, 64)
TimeDistributed(Dropout) | output: | (None, 20, 2, 2, 64)
time_distributed_12(conv2d_3) | input: | (None, 20, 2, 2, 64)
TimeDistributed(Conv2D) | output: | (None, 20, 2, 2, 64)
time_distributed_13(max_pooling2d_3) | input: | (None, 20, 2, 2, 64)
TimeDistributed(MaxPooling2D) | output: | (None, 20, 1, 1, 64)
time_distributed_14(flatten_1) | input: | (None, 20, 1, 1, 64)
TimeDistributed(Flatten) | output: | (None, 20, 64)
lstm | input: | (None, 20, 64)
LSTM | output: | (None, 32)
dense_1 | input: | (None, 32)
Dense | output: | (None, 4)

Figure 9: The structure of HiSVidClassifer Method.

In this research, the original ConvLSTM model is trained and tested based on the UCF50 dataset. As shown in Figure 10, the results depicted that the loss of ConvLSTM model equals to 0.3773, the accuracy equals to 0.8770, and run time equals to 205ms per step. In a comparison with our HiSVidClassifer method, which achieved lower loss equals to 0.1935, higher accuracy equals to 0.9508, and run time equals to 40ms per step as presented in Figure 11, Figure 12 and Figure 13, respectively. From there, our HiSVidClassifer method outperforms ConvLSTM model.

The loss function, or error function, is calculated by using the following equation (1):

$$L(y, f(x)) = -[y * \log (f(x)) + (1 - y) * \log (1 - f(x))] \quad (1)$$

The accuracy is calculated based on the sum of true positives and true negatives, divided by the sum of true positives, true negatives, false positives, and false negatives, using the next equation (2):

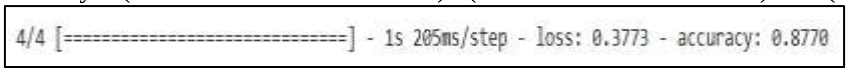
ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁙ Peer-reviewed & Refereed journal ⁙ Vol. 14, Issue 1, January 2025

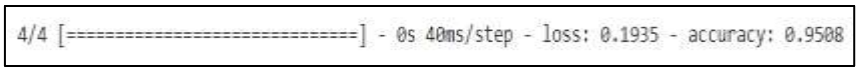
DOI: 10.17148/IJARCCE.2025.14104

Accuracy = (Count of Correct Predictions) / (Total Count of Predictions) (2)



```
4/4 [==============================] - 1s 205ms/step - loss: 0.3773 - accuracy: 0.8770
```

Figure 10: The Loss and Accuracy of ConvLSTM model.



```
4/4 [==============================] - 0s 40ms/step - loss: 0.1935 - accuracy: 0.9508
```

figure 11: The Loss and Accuracy Of HisvidClassifer Method.

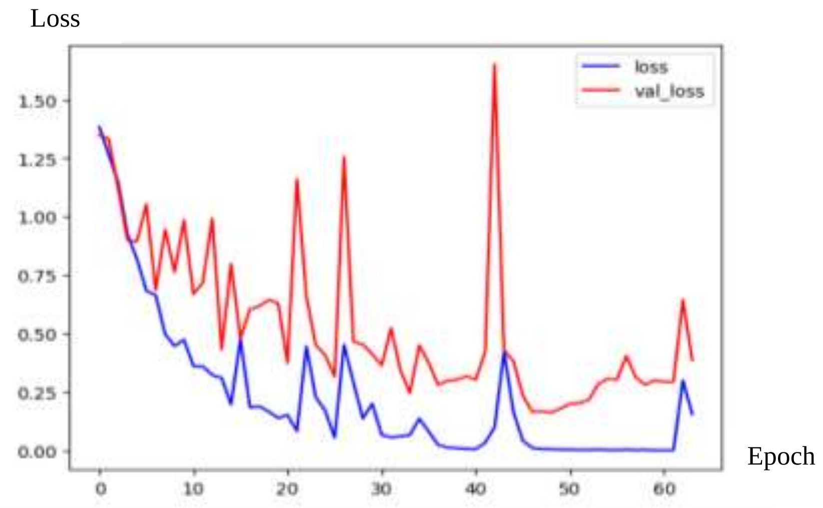


Figure 12: Total Loss vs Total Validation Loss.

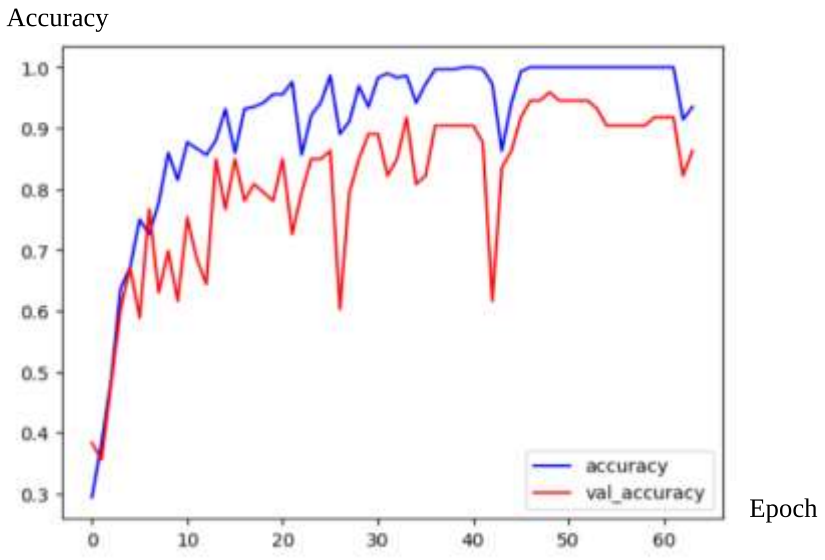


Figure 13: Total Accuracy vs Total Validation Accuracy.

VII. CONCLUSION AND FUTURE DIRECTIONS

This research analyzed various deep convolutional neural network (DCNN) models to examine the existing conditions and advancements in image recognition utilizing deep convolutional neural networks. As a result, we explored various methodologies for video classification, recognized the significance of the temporal dimension of data for enhanced accuracy, and executed two Conv2D + LSTM architectures on videos, leveraging both temporal and spatial information. Additionally, conduct pre-processing on videos with the OpenCV library to generate an image dataset; we also explored acquiring YouTube videos just through their URLs with the assistance of the Pafy package for model testing.

The model uses time-distributed Conv2D layers, which followed by MaxPooling2D, and finally Dropout layers. The features extracted from the Conv2D layers are subsequently flattened using the Flatten layer and input into an LSTM layer for analysis to categorize and detect the object. LSTM models tackle this issue by incorporating a memory cell, which serves as a repository for retaining information over prolonged durations. LSTM architectures may learn long-term dependencies in sequential data, making them suitable for tasks like language translation, speech recognition, and time series forecasting[8].

HiSVidClassifer method recognizes the classes of randomly chosen video files from retrieved UCF50 dataset via reading the first frame of the video file and writes the class name on the video frame, and then displays the frame. The total parameters of HiSVidClassifer method are 73060, trainable model parameters are 73060, and non-trainable model parameters are 0. HiSVidClassifer method is trained and evaluated on UCF50 dataset. It achieved outstanding result with low loss equals to 0.1935, and good accuracy equals to 0.9508, which outperforms the preceding videos recognizers.
In future work, this research recommends reviewing LSTM algorithm to image recognition. In addition, it recommends apply it to several different data sets to measure and improve performance to provide greater accuracy.

REFERENCES

[1]. I. H. Sarker, “Machine Learning: Algorithms, Real-World Applications and Research Directions,” SN Comput. Sci., vol. 2, no. 3, pp. 1–21, 2021, doi: 10.1007/s42979-021-00592-x.
[2]. R. Chauhan, K. K. Ghanshala, and R. C. Joshi, “Convolutional Neural Network (CNN) for Image Detection and Recognition,” 2018 First Int. Conf. Secur. Cyber Comput. Commun., no. February, pp. 278–282, 2021, doi: 10.1109/ICSCCC.2018.8703316.
[3]. M. S. Islam, M. S. Sultana, U. K. Roy, and J. Al Mahmud, “A review on Video Classification with Methods, Findings, Performance, Challenges, Limitations and Future Work,” J. Ilm. Tek. Elektro Komput. dan Inform., vol. 6, no. 2, p. 47, 2021, doi: 10.26555/jiteki.v6i2.18978.
[4]. L. Alzubaidi et al., Review of deep learning : concepts , CNN architectures , challenges , applications , future directions. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.
[5]. M. Bock, M. Moeller, and K. Van Laerhoven, “Temporal Action Localization for Inertial-based Human Activity Recognition,” 2023, [Online]. Available: http://arxiv.org/abs/2311.15831
[6]. G. Singh, V. Choutas, S. Saha, F. Yu, and L. Van Gool, “Spatio-Temporal Action Detection Under Large Motion,” Proc. - 2023 IEEE Winter Conf. Appl. Comput. Vision, WACV 2023, pp. 5998–6007, 2023, doi: 10.1109/WACV56688.2023.00595.
[7]. P. Corke, “Image Feature Extraction,” Springer Tracts Adv. Robot., vol. 142, no. February, pp. 157–202, 2022, doi: 10.1007/978-3-030-79175-9_5.
[8]. M. A. Uddin et al., “Deep learning-based human activity recognition using CNN, ConvLSTM, and LRCN,” Int. J. Cogn. Comput. Eng., vol. 5, no. June, pp. 259–268, 2024, doi: 10.1016/j.ijcce.2024.06.004.
[9]. D. W. Anggara et al., “Grayscale image enhancement for enhancing features detection in marker-less augmented reality technology,” J. Theor. Appl. Inf. Technol., vol. 98, no. 13, pp. 2671–2683, 2020.
[10]. Y. Chang and N. Mukai, “Color Feature Based Dominant Color Extraction,” IEEE Access, vol. 10, no. Dcd, pp. 93055–93061, 2022, doi: 10.1109/ACCESS.2022.3202632.
[11]. “Feature Extraction : Edges Images and Information”.
[12]. Q. Meng, D. Catchpoole, D. Skillicom, and P. J. Kennedy, “Relational autoencoder for feature extraction,” Proc. Int. Jt. Conf. Neural Networks, vol. 2017-May, pp. 364–371, 2017, doi: 10.1109/IJCNN.2017.7965877.
[13]. W. Zhou, S. Gao, L. Zhang, and X. Lou, “Histogram of Oriented Gradients Feature Extraction from Raw Bayer Pattern Images,” IEEE Trans. Circuits Syst. II Express Briefs, vol. 67, no. 5, pp. 946–950, 2020, doi: 10.1109/TCSII.2020.2980557.
[14]. W. Setiawan, A. Wahyudin, and G. R. Widianto, “The use of scale invariant feature transform (SIFT) algorithms to identification garbage images based on product label,” Proceeding - 2017 3rd Int. Conf. Sci. Inf. Technol. Theory Appl. IT Educ. Ind. Soc. Big Data Era, ICSITech 2017, vol. 2018-Janua, no. April, pp. 336–341, 2017, doi: 10.1109/ICSITech.2017.8257135.
[15]. Z. Sedaghatjoo, H. Hosseinzadeh, and B. S. Bigham, “Local Binary Pattern(LBP) Optimization for Feature Extraction,” pp. 1–21, 2024, [Online]. Available: http://arxiv.org/abs/2407.18665
[16]. D. Aiordachioaie, U. Dunarea, D. J. Galati, P. Theodor, U. Dunarea, and D. J. Galati, “A Method of Feature Extraction from Time-Frequency Images of Vibration Signals in Faulty Bearings for Classification Purposes A Method of Feature Extraction from Time-Frequency Images of Vibration Signals in Faulty Bearings for Classification Purposes,” no. July, 2020.

DOI: 10.17148/IJARCCE.2025.14104

[17]. F. Hu, G. S. Xia, J. Hu, and L. Zhang, “Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery,” Remote Sens., vol. 7, no. 11, pp. 14680–14707, 2015, doi: 10.3390/rs71114680.

[18]. A. R. Choudhuri et al., “MNIST Image Classification Using Convolutional Neural Networks,” Smart Innov. Syst. Technol., vol. 292, no. January, pp. 255–266, 2022, doi: 10.1007/978-981-19-0836-1_19.

[19]. W. Tan, Q. Yao, and J. Liu, “Overlooked Video Classification in Weakly Supervised Video Anomaly Detection,” Proc. - 2024 IEEE Winter Conf. Appl. Comput. Vis. Work. WACVW 2024, pp. 212–220, 2024, doi: 10.1109/WACVW60836.2024.00029.

[20]. L. Jing, T. Parag, Z. Wu, Y. Tian, and H. Wang, “VideoSSL: Semi-supervised learning for video classification,” Proc. - 2021 IEEE Winter Conf. Appl. Comput. Vision, WACV 2021, pp. 1109–1118, 2021, doi: 10.1109/WACV48630.2021.00115.

[21]. A. M. Conway, I. N. Durbach, A. McInnes, and R. N. Harris, “Frame-by-frame annotation of video recordings using deep neural networks,” Ecosphere, vol. 12, no. 3, 2021, doi: 10.1002/ecs2.3384.

[22]. D. Cai, A. Yao, and Y. Chen, “Dynamic Normalization and Relay for Video Action Recognition,” Adv. Neural Inf. Process. Syst., vol. 14, no. NeurIPS, pp. 11026–11040, 2021.

[23]. J. D. Bodapati and N. Veeranjaneyulu, “Feature extraction and classification using Deep convolutional Neural Networks,” J. Cyber Secur. Mobil., vol. 8, no. 2, pp. 261–276, 2019, doi: 10.13052/jcsm2245-1439.825.

[24]. F. A. Hamadain, A. A. Osman, and A. H. M. Abdelrahman, “Deep Convolutional Neural Network (DCNN) Models for Image Recognition : A Review,” vol. 8, no. 8, pp. 505–514, 2023.