Human Action Recognition Using Hybrid Deep Evolving Neural Networks

Pavan Dasari
Department of Computer Science
Royal Holloway, University of London
Surrey, TW20 0EX, UK
Pavan.Dasari.2021@live.rhul.ac.uk

Li Zhang
Department of Computer Science
Royal Holloway, University of London
Surrey, TW20 0EX, UK
li.zhang@rhul.ac.uk

Yonghong Yu
Nanjing University of Posts and Telecommunications
Nanjing, China
yuyh@njupt.edu.cn

Haoqian Huang
College of Energy and Electrical Engineering, Hohai University
Nanjing, China
hqhuang@hhu.edu.cn

Rong Gao
School of Computer Science
Hubei University of Technology
Wuhan, China
gaorong@hbut.edu.cn

***Abstract*—Human action recognition can be applied in a multitude of fully diversified domains such as active large-scale surveillance, threat detection, personal safety in hazardous environments, human assistance, health monitoring, and intelligent robotics. Owing to its high demands in real-world applications, it has drawn significant attention. In this research, we propose hybrid deep neural networks, i.e. Convolutional Long Short-Term Memory (ConvLSTM) Networks, Long-term Recurrent Convolutional Networks (LRCN), for tackling video action classification. In particular, for the LRCN model, different CNN encoder architectures such as VGG16, ResNet50, DenseNet121 and MobileNet, as well as several Long Short-Term Memory (LSTM) variant decoder architectures, such as LSTM, bidirectional LSTM (BiLSTM) and Gated Recurrent Unit (GRU), are used for spatial-temporal feature extraction to test model performance. We adopt diverse experimental settings including using different numbers of frames per video and learning configurations to optimize performance. The empirical results indicate the superiority of MobileNet in combination with a BiLSTM network over other hybrid network settings for the action classification using the UCF50 dataset. Owing to the lightweight MobileNet encoder, this LRCN model also achieves a better trade-off between performance and training and inference computational costs, while outperforming existing state-of-the-art methods.**

***Index Terms*—Human Action Recognition, Video Analytics, Convolutional Neural Network, Long Short-Term Memory (LSTM), Gated Recurrent Unit, Long-term Recurrent Convolutional Network, and ConvLSTM.**

I. Introduction

Human action recognition is one of the open problems that has been in the research community for the past two decades. This is mostly because of the complexity involved in accurately predicting the action from the video sequences. Humans have substantial contextual data and advanced sensory inputs that help in effectively recognizing the action being performed in real-time. That is not the case with machines. To correctly classify the action, a machine needs to consider which data points in the image represent the object of interest. It also needs to observe the changes between the frames that represent the motion, trajectory, interaction between different objects. Moreover, in comparison with image-based action recognition, Video analysis requires more discriminative features spanning over spatial and temporal dimensions [1]–[4] to capture motion, human subjects and background variations for classifying different action classes.

Therefore, besides the capturing of spatial features, video action recognition tasks introduce the need for an extra dimension that encodes temporal dependencies. The effective capturing of the spatial-temporal cues plays vital roles in action classification. In recent years, many new approaches have been used for extracting spatial-temporal features. The most popular methods among them are deep learning techniques, which conduct layer-wise feature learning and are able to capture spatial and temporal variations between image frames. Such deep neural networks thus show impressive performances for solving object detection, pose estimation [5], human body segmentation [6] pertaining to action recognition tasks.

Although there has been rapid development in deep learning algorithms, in the case of video analytics, deep learning algorithms require huge computation resources and complex pre-processing procedures for optical flow extraction to aid action classification [7]. In order to tackle action recognition in a large scale, the deployment of such systems to resource constrained environments poses significant challenges. In order to reduce the computational cost, much of the effort is focused on reducing the number of frames [8]–[10], which has proven to be promising. But, in this research, we focus on exploring further to find the best trade-off between the performance and the extraction of different numbers of frames for action recognition using diverse hybrid deep networks.

Specifically, in this research, we propose multiple hybrid deep networks, i.e. Convolutional Long Short-Term Memory (ConvLSTM) networks and Long-term Recurrent Convolutional Networks (LRCN), with different backbone and hyper-

parameter settings as well as different numbers of frames, for tackling action recognition. Diverse encoder-decoder architectures are also implemented in these hybrid networks to test their performances. Precisely, several Convolutional Neural Networks (CNNs), such as VGG-16 [11], ResNet [12], [13], MobileNet [14] and DenseNet [15], are used as the encoder, while Long Short-Term Memory (LSTM), bidirectional LSTM (BiLSTM) and Gated recurrent unit (GRU) are used as the decoder in the above two types of hybrid networks. We subsequently compare the intricate performance details of these approaches along with the insights gained from experimentation using a large-scale human action data set.

II. Related Work

In the earlier times, before the rise of deep learning, there were different strategies used for performing action recognition. Some of these approaches use feature descriptors such as Local Binary Patterns (LBP), Scale Invariant Feature Transformation (SIFT), Histogram of Oriented Gradients (HOG) [16], and Histogram of Optical Flow (HOF). Optical flow (i.e. the patterns of moving objects in a scene) as well as trajectory based methods such as Motion Boundary Histograms (MBH) [17] and Dense trajectories [18] are used in other studies. Such extracted spatial and temporal features are then used as inputs to classifiers, e.g. Support Vector Machines (SVM) [19] to perform the predictions for the image frame. But, after the successful demonstration of deep learning methods for image classification, deep networks have been quickly adopted in the field of video analysis, where videos are divided into a sequence of images. We will now discuss diverse deep networks that have been proposed in recent years for action classification.

As an example, [20] introduced multiple variants of information fusion methods across a frame sequence of a video for action classification. They extracted temporal features purely using CNNs based on single frame under the settings of early, late and slow fusions, for action prediction. The work also proposed multi-resolution networks that focused on the aspect of computation required to build such fusion networks [21], [22]. Since CNNs are resources demanding and require significant amount of time and computation power for model training. Increasing the number of times we apply convolution operations or increasing the depth of the network may result in extreme slow processing and that is not acceptable in case of video analysis as the network needs to work with high frame rates. There are multiple ways to reduce the computational cost. One potential strategy is to reduce the number of convolutional layers, but this approach reduces the performance of the network significantly. Therefore, instead of reducing CNN layers, their work reduced the image resolution of the input stream and used two networks, one with low-resolution context and the other with high-resolution foveate. This strategy has speeded up the training by 4 times. For predicting a video label, a sample set of 20 frames was augmented using different cropping and flipping operations. These augmented datasets passed four times through the network for action classification. Then the predictions of each frame are averaged at the end. This reduced the amount of complexity involved and the inference time required to perform video action recognition.

In the hybrid networks such as a CNN combined with a Recurrent Neural Network (RNN), i.e. CNN-RNN, we observe that the temporal features are local and information present in the sequence is not represented efficiently. Hence there is a need for a global representation which can be realized using temporal aggregation like temporal pooling. In the work of [23], the authors present various methods to perform temporal pooling along with the usage of LSTM Networks for learning from the sequential information generated from CNNs. These methods outperformed the previous models in terms of both computational efficiency and model performance, with an accuracy rate above 80% when evaluated using public action datasets. Their LSTM variant with 30 frames of optical flow images combined with the original images produced an accuracy rate of 88.6%. Motivated by this research, two variant networks, namely LRCN and ConvLSTM, with different network architectures and encoder-decoder backbones, are employed for action classification.

Three-dimensional CNNs (3D ConvNets) introduced in [24] are essentially similar to CNN architectures employed for 2D image classification, except that there is an extra dimension that is added while performing convolution and pooling. In traditional 2D CNN, the input shape will be W x H x C, where W, H, and C represent width, height, and depth/channels of the image being parsed, whereas, in case of 3D CNN, the input shape is F x W x H x C, where F represents the number of frames being parsed into the network. In 3D CNN, the network looks at the entire sequence of frames as a single image and extracts spatial-temporal features in single pass. But the amount of computation required to perform 3D-convolutions is extremely high as the increase in the input dimensions increases the number of multiply and accumulate (MAC) operations. But, their work argued that the performances of the networks out-weigh the computational costs as the 3D CNNs such as the C3D network achieve better performance in comparison with those of the fusion methods. Multiple variants [25]–[27] of this architecture have been adopted for solving tasks of video recognition and time series forecasting in other existing studies.

III. The Proposed Hybrid Deep Networks for Action Classification

In this research, we propose two hybrid deep networks, i.e. ConvLSTM and LRCN Networks, with different backbone and hyper-parameter settings as well as different numbers of frames, for tackling action recognition. Diverse encoder-decoder architectures are also implemented in these hybrid networks to test model performances. Specifically, several Convolutional Neural Networks (CNNs), such as VGG-16, ResNet, MobileNet and DenseNet, are used as the encoder, while LSTM and BiLSTM are used as the decoder in the above two types of hybrid networks. Comprehensive evaluations have

also been conducted to test model efficiency using a large-scale action recognition data set. We introduce each proposed hybrid encoder-decoder network in the following subsections.

A. The Proposed ConvLSTM Network for Action Recognition

The *Convolutional LSTM* (ConvLSTM) network was first introduced in the work of [28]. In a fully connected LSTM Network, flattening the image into a 1D space will not reserve any spatial information, thus creating the need for a CNN feature extractor to extract spatial features and transform them into a 1D vector space. Therefore, the ConvLSTM network was proposed for video classification tasks [29], which consumes 2D convolutions as inputs. It is able to directly work with a sequence of images and perform convolutional operations on the input images for spatial feature extraction, whereas the LSTM layers are able to extract temporal dynamics between the frames. Thus the ConvLSTM network is able to essentially capture both spatial and temporal cues, which can not be performed using a fully connected LSTM.

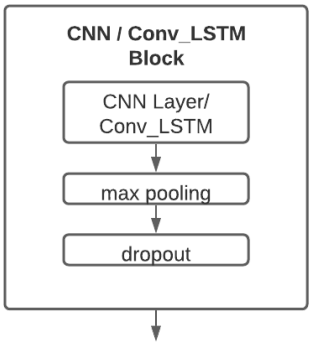

Fig. 1. An example ConvLSTM block

In this research, we employ the ConvLSTM network for Video classification. A lightweight ConvLSTM network is proposed, which consists of four ConvLSTM2D layers and a final dense layer, to improve computational efficiency while achieving promising performance. The detailed network architecture is shown in Figures 1 and 2. Specifically, we use tensorflow's implementation of ConvLSTM2D written in Python to build the action recognition model. We first create an instance of class Sequential() that has pre-built methods to construct a model structure with a validation functionality that checks if the structure provided is feasible.

A ConvLSTM2D layer is first added, which uses the tanh activation function and has a kernel size of 3 X 3 and the numbers of filters ranging from 8 to 64. Moreover, the return_sequences configuration is set to true in order to make the LSTM layer work in a synced many-to-many mode, such that multiple outputs can be passed to the next hidden ConvLSTM layer. We also use a recurrent dropout rate of 0.2, which means that 20% of the neurons outputs will be dropped while performing the linear transformation in the recurrent layer. In the end, we specify the input shape which is a 4D tensor with the number of frames, image height,

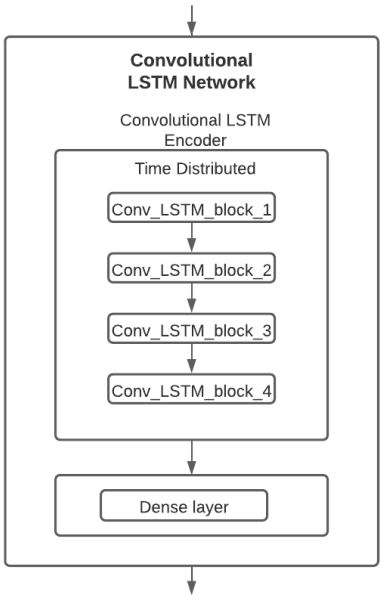

Fig. 2. An example ConvLSTM network with four ConvLSTM blocks and one dense layer

image width, and the number of channels. A pooling layer is then added after each ConvLSTM2D layer, in order to reduce the resolution 2-fold and make the network shift invariant. A time distributed dropout layer is subsequently added after each pooling operation, with the attempt to skip 20% of the output frames generated to prevent over-fitting. A set of aforementioned three layers, i.e. ConvLSTM2D, pooling and time distributed dropout layers, constitutes a ConvLSTM block as shown in Figure 1. Multiple such blocks are stacked together to make a ConvLSTM model, as illustrated in Figure 2. In the end, we use a flatten layer to convert the 2D vector to 1D vector and then connect it to a dense layer through a softmax activation. The model makes a prediction of the input video by selecting the highest probabilities among all the classes.

The proposed ConvLSTM model is then used to extract spatial-temporal dependencies to test its performance using a large human action dataset. Different numbers of frames are also extracted from the input video to test model efficiency. The identified best setting of the number of frame will be used in subsequent experiments to test model performance.

B. Model size and Parameters of ConvLSTM

The proposed ConvLSTM model with the input frames of 20 as the sequence length, 64 X 64 X 3 as the image resolution, and 12 as the target number of output classes, has a total of 317,292 trainable parameters and the split of parameters in each layer is shown in the Figure 3. We can observe that there are no trainable parameters associated with maxpooling and dropout layers as they are just transformations.

Besides the ConvLSTM model, we also propose another hybrid network, i.e. LRCN, for tackling action classification.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv_lst_m2d (ConvLSTM2D)    (None, 20, 62, 62, 64)    154624
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 20, 31, 31, 64)    0
_________________________________________________________________
time_distributed (TimeDistri (None, 20, 31, 31, 64)    0
_________________________________________________________________
conv_lst_m2d_1 (ConvLSTM2D)  (None, 20, 29, 29, 32)    110720
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 20, 15, 15, 32)    0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 20, 15, 15, 32)    0
_________________________________________________________________
conv_lst_m2d_2 (ConvLSTM2D)  (None, 20, 13, 13, 16)    27712
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 20, 7, 7, 16)      0
_________________________________________________________________
time_distributed_2 (TimeDist (None, 20, 7, 7, 16)      0
_________________________________________________________________
conv_lst_m2d_3 (ConvLSTM2D)  (None, 20, 5, 5, 8)       6944
_________________________________________________________________
max_pooling3d_3 (MaxPooling3 (None, 20, 3, 3, 8)       0
_________________________________________________________________
flatten (Flatten)            (None, 1440)              0
_________________________________________________________________
dense (Dense)                (None, 12)                17292
=================================================================
Total params: 317,292
Trainable params: 317,292
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3. ConvLSTM model parameters

C. The Proposed Long-term Recurrent Convolutional Network for Action Recognition

Long-term Recurrent Convolutional Networks (LRCN) is a deep learning architecture that is capable of extracting both visual and sequential features from a sequence of images. It was originally proposed by [30] to showcase that a LRCN can be used for action recognition, image captioning, and video description generation. This is achieved by employing different variations of RNNs. LRCN has two components, i.e. a CNN and an RNN. The CNN encoder extracts the spatial features and transforms them into a 1D vector. This 1D vector yielded by CNN is then passed into the RNN decoder to extract the temporal dynamics. With the use of shared weights, multiple frames can be passed into the network as inputs in a single shot. Such encoder-decoder architecture gives sufficient freedom in terms of the architecture selections for CNN and RNN, as both components are independent of each other and can be seen as two separate entities. This is possible because of the fact that a recurrent neuron can take a varying length vector as input. So, any CNN architecture whose last layer is flattened can be passed as input to the next stage irrespective of the dimensions of the flattened output of CNN.

To test model efficiency, we employ a number of CNNs such as a 4-layer CNN block, VGG16, ResNet50, DenseNet121 and MobileNet as the encoder, as well as different types of RNNs such as LSTM, BiLSTM and Gated Recurrent Unit (GRU) as the decoder for action classification. We subsequently compare the model performances of different model architectures. The model architecture of the CNNs with four convoluational layers in combination with three LSTM layers is shown in Figure 4.

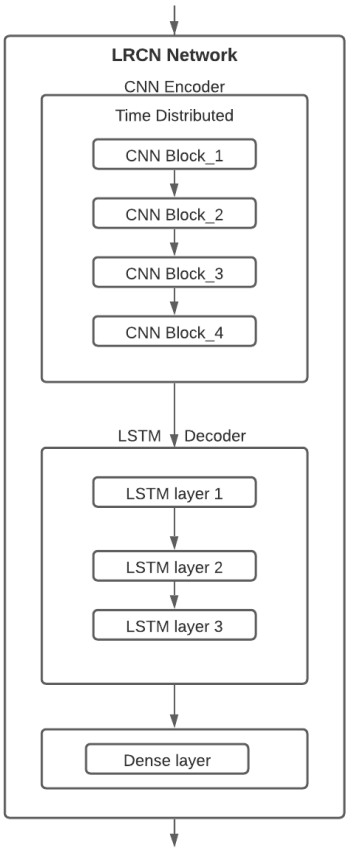

Fig. 4. An example LRCN model with four CNN layers and three LSTM layers

D. Model size and Parameters of LRCN

We will now compare the model sizes for all the variants of the proposed architectures. Table I presents the split of trainable and non-trainable parameters in all the variant models. Among them, MobileNet with GRU has the least number of parameters in total, owing to the adoption of the depthwise separated convolution filters instead of the traditional convolution filters. We can see that total number of parameters in Table I seem relatively large when compared with the ConvLSTM method in section III-B, this is because of the large CNN parameters and their output dimensions, as the number of parameters in an RNN depends on the size of the input. Since we just use CNNs as the feature extractor, all the parameters in CNN blocks are frozen so no training takes place at the CNN encoder. This reduces the training time significantly. But, at the same time, the network increases the inference time, as the model size increases the number of MAC operations to be performed while predicting, thus increasing the time required for prediction.

IV. Experiments

To test model efficiency, a comprehensive evaluation has been conducted. Specifically, multiple experiments with different numbers of extracted frames per video, and different hyper-

TABLE I
TRAINABLE PARAMETERS IN PRE-TRAINED LRCN MODELS

Model Type	CNN(Pre-trained)	RNN	Total
VGG16+LSTM	14,714,688	2,397,708	17,112,396
VGG16+BILSTM	14,714,688	4,795,020	19,509,708
VGG16+GRU	14,714,688	1,799,244	16,513,932
ResNet50+LSTM	23,564,800	8,689,164	32,253,964
ResNet50+BILSTM	23,564,800	17,377,932	40,942,732
ResNet50+GRU	23,564,800	6,517,836	30,082,636
DenseNet121+LSTM	7,037,504	4,494,860	11,532,364
DenseNet121+BILSTM	7,037,504	8,989,324	16,026,828
DenseNet121+GRU	7,037,504	3,372,108	10,409,612
MobileNet+LSTM	3,228,864	4,494,860	7,723,724
MobileNet+BILSTM	3,228,864	8,989,324	12,218,188
MobileNet+GRU	3,228,864	3,372,108	6,600,972

parameter (e.g. the learning rate, model depth, and dropout rate) settings, have been conducted. A large action data set, i.e. UCF50 [31], is used for model evaluation.

The UCF datasets [31]–[35] are a family of video analysis and classification datasets published by researchers at University of central Florida. They have a large variety of datasets for video classification. Among them, we will use UCF50, a dataset with 50 actions collected from YouTube videos. It is an extension of UCF11 dataset consisting of 11 actions. UCF50 is one of the most popular bench-marking datasets widely used for action recognition.

Moreover, we further split UCF50 by creating 4 datasets of incremental sizes, i.e. 12, 25, 38, and 50 classes, as experimenting on entire dataset will take a long time and require enormous amounts of computation. Hence we will first experiment with the dataset of 12 classes, then based on the results we will choose the best models for tackling other comparatively larger experimental settings.

A. Number of Frames

In order to identify the best frame settings, we first conduct the experiments using the lightweight ConvLSTM network as well as the LRCN model using the dataset of 12 action classes. We set the number of frames per video ranging from 20 to 150.

TABLE II
RESULTS OF EXPERIMENTS USING THE CONVLSTM MODEL WITH DIFFERENT NUMBERS OF EXTRACTED FRAMES PER VIDEO.

Number of frames	Accuracy	Loss	Training time(sec.)	Inference time(sec.)
20	0.75	1.11	165	5
30	0.7	1.02	240	7
50	0.72	0.93	381	14
70	0.73	1.25	509	18
100	0.65	0.98	563	26
150	0.68	1.01	590	40

Table II showcases the results of experiments conducted with the ConvLSTM model with respect to different numbers of extracted frames. After substantial experimentation with multiple hyper-parameter settings, we have employed the parameters that have performed the best in subsequent experiments to establish a relationship between the number of extracted frames (evenly distributed in time) and the time required to train and test the model. Please note that the training time reported in Table II is the time elapsed per epoch, while the inference time, loss and accuracy rates are for the entire test set. For fair comparison, we have kept the size of the test set consistent along with all other hyper-parameters, except for the number of frames, i.e. the controlled parameter, to identify the best frame setting. We observe that there is a linear increase in the processing time with the increase in the number of frames but the accuracy rate does not improve. Instead, it decreases as the number of frames increases. This is probably caused by the redundant features extracted between the frames. The model using 20 frames per video achieves the best performances, i.e. an accuracy rate of 75%. Therefore we fix the number of frames as 20 for subsequent experiments.

We repeat the same experiment using the LRCN model. For this experiment, we have used an LSTM layer with 32 units in the RNN block and a 4-layer CNN block with 3 x 3 filters as the visual feature extractor.

TABLE III
RESULTS OF EXPERIMENTS USING THE LRCN MODEL AND DIFFERENT NUMBERS OF FRAMES.

Number of frames	Accuracy	Loss	Training time(sec.) per epoch	Inference time(sec.)
20	0.86	0.57	8	0.8
30	0.87	1.51	11	1.26
50	0.82	0.61	20	2.06
70	0.85	0.81	29	2.72
100	0.81	0.67	43.8	3.36
150	0.72	1.13	61.9	7.92

Again we have fixed all the hyper-parameters except for the number of the frames and used the same subset of 12 classes from UCF50 in the experiments. Table III presents the results for different frame settings. We observe that the performances in this case are slightly better with the best accuracy rate of 87% achieved using 30 frames per video, followed by the results obtained using 20 frames per video. This is also because of the number of trainable parameters in LRCN as it has 5 times less size than that of ConvLSTM to enable the network to be fully trained using the subset of UCF50. We have also stacked up additional two hidden layers with LSTM cells and BiLSTM cells in our subsequent experiments, to further test LRCN model efficiency.

B. Number of Hidden Layers for RNN Block

Table IV presents the results of this further investigation using 20 frames per video and the dataset of 12 classes as the best trade-off between performance and computational cost. Hyper-parameter fine-tuning has also been conducted using the training and validation sets after running multiple random configurations of the learning rate, learning scheduler, dropout rate, and loss functions. We set the identified most optimal

TABLE IV
RESULTS OF EXPERIMENTS WITH THE RECURRENT BLOCK OF LRCN MODELS

Model Type	Accuracy	Loss	Training time(sec.)	Inference time(sec.)
LSTM	0.81	0.71	19	1
3 Layer LSTM	0.54	1.39	31	2
BILSTM	0.87	0.71	20	2
3 Layer BILSTM	0.81	0.75	32	3.72

hyper-parameters to test model performance using the test set. The empirical results indicate that increasing the layer depth in the RNN block does not improve the performance significantly, as the three-layer LSTM or BiLSTM architecture performs worse in the same experimental settings than the single layer LSTM or BiLSTM architecture in a setup of 20 frames per video. Therefore, a single layer RNN block is adopted in the LRCN model for further studies.

C. Base CNN Architectures

We have performed experiments with a lightweight LRCN model (i.e. a 4-layer CNN block) in the previous section and have also learnt that a single layer in the RNN block performs significantly better.

We subsequently make attempts to increase the CNN layer depth and modify the CNN block (i.e. the visual feature extractor) in the LRCN model to further enhance performances. We fix the number of frames as 20 and the RNN block with a single layer as recommended by the previous experiments. We use pre-trained CNN architectures with layer depths ranging from 16 to 121 leading to a steady increase in the numbers of parameters as compared with those of the LRCN model used in previous section. Table V presents the results to determine the relation between the number of epochs in training and the performance metrics, essentially looking at the learning curves to investigate any over-fitting issues and determine if early stopping should be used or not. We observe that there is not much difference between the performance metrics of 100 and 25 training epochs enabled with early stopping. All the results presented in Table V are obtained using the 12-class action dataset. From the above experiment, we notice that over-fitting occurs when using 100 epochs as the test set loss is higher than that obtained using the 25 training epochs.

We now extend the experiments based on the best findings of our previous investigations and scale up the number of classes from 12 to 25, 38, and 50 (i.e. the complete UCF50 dataset). We use 20 frames per video and a single layer RNN block with a learning rate of 10^{-3}, and compare the results of different CNN encoders in the LRCN model. Specifically, we employ VGG16, ResNet50, DenseNet121 and MobileNet as the encoder in the LRCN model. Tables VI, VII, and VIII present the results of the experiments on the 25, 38, and 50-class datasets, respectively. We observe that, in all the three datasets, MobileNet in combination with BiLSTM performs better with an accuracy rate of 87% and 19 seconds of inference time using the full UCF50 dataset (with 50 classes). Although VGG16 with GRU also has obtained an accuracy rate of 87%, it takes 149 seconds to compute the predictions for the same test set, which is roughly 7.5 times of the time required for MobileNet (as indicated in Table VIII). Hence the MobileNet model is the lightest and best choice as a visual feature extractor in an LRCN model based on our experiments.

TABLE V
COMPARISON BETWEEN THE METRICS FOR DIFFERENT TRAINING PERIODS

Model Type	100 Epochs			25 Epochs (with early stopping)		
	Accuracy	F1-score	Loss	Accuracy	F1-score	Loss
VGG 16 + LSTM	0.92	0.92	0.38	0.93	0.93	0.27
VGG 16 + BiLSTM	0.80	0.83	0.61	0.91	0.91	0.33
VGG 16 + GRU	0.94	0.93	0.71	0.93	0.93	0.23
ResNet 50 + LSTM	0.90	0.89	0.58	0.92	0.92	0.29
ResNet 50 + BiLSTM	0.88	0.88	0.68	0.91	0.92	0.26
ResNet 50 + GRU	0.91	0.91	0.52	0.92	0.91	0.32
DenseNet 121 + LSTM	0.93	0.93	0.37	0.93	0.92	0.30
DenseNet 121 + BiLSTM	0.92	0.92	0.33	0.91	0.91	0.32
DenseNet 121 + GRU	0.92	0.92	0.56	0.85	0.85	0.48
MobileNet + LSTM	0.91	0.91	0.45	0.91	0.90	0.35
MobileNet + BiLSTM	0.92	0.92	0.38	0.93	0.93	0.32
MobileNet + GRU	0.93	0.92	0.46	0.90	0.90	0.31

To summarize, we have first performed experiments with the a comparatively smaller 12-class dataset to identify the best hyper-parameters and model architectures. We then used those findings to build larger and more complex hybrid models that can perform well on larger datasets. Therefore we performed the experiments using a constructive design process thus reducing the amount of time and effort needed to obtain promising performances.

V. CONCLUSION AND FUTURE WORK

In this research, we have proposed several hybrid deep learning models for undertaking human action recognition tasks using videos or image sequences. Specifically the ConvLSTM and LRCN hybrid networks with different backbone and frame settings have been employed to tackle action classification using the UCF50 dataset. In particular, with respect to

TABLE VI
RESULTS OF EXPERIMENTATION WITH 25 CLASSES EXTRACTED FROM UCF50. (NOTE THAT ACCURACY, LOSS RESULTS ARE CALCULATED ON THE TEST SET FOR ALL THE MODELS. TRAINING TIME IS CALCULATED PER EPOCH, WHILE INFERENCE TIME IS CALCULATED FOR THE ENTIRE TEST SET.)

Model Type	Accuracy	Loss	Training time(sec.)	Inference time(sec.)
VGG16 LSTM	0.87	0.49	336	80
VGG16 BILSTM	0.9	0.42	398	83
VGG16 GRU	0.9	0.37	514	112
ResNet50 LSTM	0.8	0.75	447	75
ResNet50 BILSTM	0.82	0.64	597	88
ResNet50 GRU	0.8	0.71	376	86
DenseNet121 LSTM	0.78	0.86	458	109
DenseNet121 BILSTM	0.8	0.7	441	147
DenseNet121 GRU	0.76	0.87	372	100
MobileNet LSTM	0.88	0.44	164	26
MobileNet BILSTM	0.91	0.34	204	29
MobileNet GRU	0.87	0.43	115	31

TABLE VII
RESULTS OF EXPERIMENTATION WITH 38 CLASSES EXTRACTED FROM UCF50. (NOTE THAT ACCURACY, LOSS RESULTS ARE CALCULATED ON THE TEST SET FOR ALL THE MODELS. TRAINING TIME IS CALCULATED PER EPOCH, WHILE INFERENCE TIME IS CALCULATED FOR THE ENTIRE TEST SET.)

Model Type	Accuracy	Loss	Training time(sec.)	Inference time(sec.)
VGG16 LSTM	0.84	0.61	565	119
VGG16 BILSTM	0.84	0.65	923	204
VGG16 GRU	0.87	0.56	767	206
ResNet50 LSTM	0.81	0.76	545	129
ResNet50 BILSTM	0.84	0.78	659	94
ResNet50 GRU	0.76	0.94	588	118
DenseNet121 LSTM	0.76	0.87	300	78
DenseNet121 BILSTM	0.79	0.84	322	84
DenseNet121 GRU	0.69	1.12	256	62
MobileNet LSTM	0.85	0.56	155	29
MobileNet BILSTM	0.89	0.51	216	33
MobileNet GRU	0.84	0.67	122	22

TABLE VIII
RESULTS OF EXPERIMENTATION WITH 50 CLASSES FROM UCF50. (NOTE THAT ACCURACY, LOSS RESULTS ARE CALCULATED ON THE TEST SET FOR ALL THE MODELS. TRAINING TIME IS CALCULATED PER EPOCH, WHILE INFERENCE TIME IS CALCULATED FOR THE ENTIRE TEST SET.)

Model Type	Accuracy	Loss	Training time(sec.)	Inference time(sec.)
VGG16 LSTM	0.84	0.72	774	163
VGG16 BILSTM	0.86	0.67	1249	237
VGG16 GRU	0.87	0.6	727	149
ResNet50 LSTM	0.76	0.98	519	96
ResNet50 BILSTM	0.78	0.94	643	77
ResNet50 GRU	0.68	1.19	388	68
DenseNet121 LSTM	0.63	1.37	339	97
DenseNet121 BILSTM	0.74	1.06	454	91
DenseNet121 GRU	0.63	1.31	277	65
MobileNet LSTM	0.83	0.76	136	20
MobileNet BILSTM	0.87	0.6	193	19
MobileNet GRU	0.79	0.82	117	19

the LRCN model, different CNNs, such as VGG16, ResNet50, DenseNet121 and MobileNet, are used as the encoder, while LSTM variants, such as LSTM, BiLSTM and GRU with distinctive hidden neuron settings, are used as the decoder. Diverse experiments using the different numbers of frames ranging from 20 to 150 are conducted to test model efficiency. The empirical results indicate that the LRCN model with MobileNet as the encoder and a BiLSTM network as the decoder achieves the best accuracy rate, i.e. 87%, for the classification of 50 action classes in UCF50. For future directions, we aim to extend our network by taking into account pose estimation and human object interaction as additional feature extractors, along with the current visual feature extractor, to better describe the subtle variations between different actions. We also aim to incorporate with evolutionary algorithms to further fine-tune model hyper-parameters and architectures to further enhance performance [36]–[53].

REFERENCES

[1] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In IEEE conference on computer vision and pattern recognition, June 2018.

[2] L. Sevilla-Lara, S. Zha, Z. Yan, V. Goswami, M. Feiszli, and L. Torresani. Only time can tell: Discovering temporal data for temporal modeling. arXiv:1907.08340 [cs], October 2019.

[3] L. Zhang, C.P. Lim, and Y. Yu. Intelligent human action recognition using an ensemble model of evolving deep networks with swarm-based optimization. Knowledge-based systems, 220:106918, 2021.

[4] L. Wang, Y. Xu, J. Cheng, H. Xia, J. Yin, and J. Wu. Human action recognition by learning spatio-temporal features with deep neural networks. IEEE access, 6:17913–17922, 2018.

[5] I. Lee, D. Kim, S. Kang, and S. Lee. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In IEEE international conference on computer vision (ICCV), Oct 2017.

[6] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L. Guibas, and L. Fei-Fei. Human action recognition by learning bases of action attributes and parts. In 2011 International conference on computer vision, pages 1331–1338, 2011.

[7] K. Guo, P. Ishwar, and J. Konrad. Action recognition using sparse representation on covariance manifolds of optical flow. In 2010 7th IEEE international conference on advanced video and signal based surveillance, pages 188–195, 2010.

[8] S.N. Gowda, M. Rohrbach, and L. Sevilla-Lara. SMART frame selection for action recognition. arXiv:2012.10671 [cs], Decem- ber 2020.

[9] L. Zhu, L.Sevilla-Lara, D. Tran, M. Feiszli, Y. Yang, and H. Wang. Faster recurrent networks for efficient video classification. arXiv:1906.04226 [cs], September 2019. arXiv: 1906.04226.

[10] Y.D. Zheng, Z. Liu, T. Lu, and L. Wang. Dynamic sampling networks for efficient action recognition in videos. IEEE transactions on image processing, 29:7970–7983, 2020.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 [cs], April 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv:1512.03385 [cs], December 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. arXiv:1603.05027 [cs], July 2016.

[14] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861 [cs], April 2017.

[15] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. arXiv:1608.06993 [cs], January 2018.

[16] H.A. Qazi, U. Jahangir, B.M. Yousuf, and A.Noor. Human action recognition using sift and hog method. In 2017 International conference on information and communication technologies, pages 6– 10, 2017.

[17] H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. International journal of computer vision, 119(3):219–238, September 2016.

[18] H. Wang, A. klaser, C. Schmid, and C.L. Liu. Dense trajectories and motion boundary descriptors for action recognition. International journal of computer vision, 103(1):60–79, May 2013.

[19] S. Danafar and N. Gheissari. Action recognition for surveillance applications using optic flow and SVM. In Y. Yagi, S.B. Kang, I.S. Kweon, and H. Zha, editors, Computer Vision – ACCV 2007, Lecture Notes in Computer Science, pages 457– 466, Berlin, Heidelberg, 2007. Springer.

[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In IEEE conference on computer vision and pattern recognition, June 2014.

[21] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. arXiv:1812.03982 [cs], October 2019.

[22] F. Xiao, Y.J. Lee, K. Grauman, J. Malik, C. Feichtenhofer. Audiovisual slowfast networks for video recognition. March 2020. arXiv: 2001.08740 version: 2.

[23] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: deep networks for video classification. arXiv:1503.08909 [cs], April 2015.

[24] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. arXiv:1412.0767 [cs], October 2015.

[25] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence, 35(1):221–231, 2013.

[26] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In IEEE conference on computer vision and pattern recognition, July 2017.

[27] K. Liu, W. Liu, C. Gan, M. Tan, and H. Ma. T-C3D: Temporal convolutional 3D network for real-time action recognition. AAAI conference on artificial intelligence, 32(1), April 2018.

[28] X. Shi, Z. Chen, H. Wang, D.Y. Yeung, W.k. Wong, and W.c. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. arXiv:1506.04214 [cs], September 2015.

[29] W. Luo, W. Liu, and S. Gao. Remembering history with convolutional LSTM for anomaly detection. In 2017 IEEE International conference on multimedia and expo (ICME), pages 439–444, Hong Kong, Hong Kong, July 2017. IEEE.

[30] J. Donahue and L.A. Hendricks and S. Guadarrama and M. Rohrbach and S. Venugopalan and K. Saenko and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In CVPR, 2015.

[31] K.K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. Machine vision and applications journal MVAP-D-12-00032, 2012.

[32] J. Ahmed, M.D. Rodriguez and M. Shah. Action mach: A spatio-temporal maximum average correlation height filter for action recognition. Computer vision and pattern recognition, 2008.

[33] K. Soomro and A.R. Zamir. Action recognition in realistic sports videos. Computer vision in sports. springer international pub- lishing, 2014.

[34] J. Luo, J. Liu and M Shah. Recognizing realistic actions from videos "in the wild". IEEE International Conference on Computer Vision and Pattern Recognition(CVPR), 2009.

[35] A.R. Zamir, K. Soomro and M. Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. CRCV- TR-12-01, 2012.

[36] S.C. Neoh, L. Zhang, K. Mistry, M.A. Hossain, C.P. Lim, N. Aslam, and P. Kinghorn. Intelligent facial emotion recognition using a layered encoding cascade optimization model. Applied soft computing, 34:72–93, 2015.

[37] K. Mistry, L. Zhang, S.C. Neoh, C.P. Lim, and B. Fielding. A micro-ga embedded pso feature selection approach to intelligent facial emotion recognition. IEEE transactions on cybernetics, 47(6):1496–1509, 2016.

[38] B. Fielding and L. Zhang. Evolving image classification architectures with enhanced particle swarm optimisation. IEEE access, 6:68560–68575, 2018.

[39] T.Y. Tan, L. Zhang, and C.P. Lim. Adaptive melanoma diagnosis using evolving clustering, ensemble and deep neural networks. Knowledge-based systems, 187:104807, 2020.

[40] T. Lawrence, L. Zhang, C.P. Lim, and E.-J. Phillips. Particle swarm optimization for automatically evolving convolutional neural networks for image classification. IEEE access, 9:14369–14386, 2021.

[41] L. Zhang, C.P. Lim, Y. Yu, and M. Jiang. Sound classification using evolving ensemble models and particle swarm optimization. Applied soft computing, page 108322, 2021.

[42] T. Lawrence, L. Zhang, K. Rogage, and C.P. Lim. Evolving deep architecture generation with residual connections for image classification using particle swarm optimization. sensors, 21(23):7936, 2021.

[43] S. Slade, L. Zhang, Y. Yu, and C.P. Lim. An evolving ensemble model of multi-stream convolutional neural networks for human action recognition in still images. Neural computing and applications, pages 1–27, 2022.

[44] H. Xie, L. Zhang, and C.P. Lim. Evolving cnn-lstm models for time series prediction using enhanced grey wolf optimizer. IEEE access, 8:161519–161541, 2020.

[45] P. Kinghorn, L. Zhang, and L. Shao. A region-based image caption generator with refined descriptions. Neurocomputing, 272:416– 424, 2018.

[46] B. Fielding and L. Zhang. 2020. Evolving deep DenseBlock architecture ensembles for image classification. Electronics, 9(11), p.1880.

[47] T.Y. Tan, L. Zhang, and C.P. Lim. Intelligent skin cancer diagnosis using improved particle swarm optimization and deep learning models. Applied Soft Computing, 84, p.105725. 2019.

[48] T.Y. Tan, L. Zhang, C.P. Lim, B. Fielding, Y. Yu, and E. Anderson. Evolving ensemble models for image segmentation using enhanced particle swarm optimization. IEEE access, 7, pp.34004-34019. 2019.

[49] P. Kinghorn, L. Zhang and L. Shao. A Hierarchical and Regional Deep Learning Architecture for Image Description Generation, Pattern Recognition Letters. 2019.

[50] L. Zhang and C.P. Lim. Intelligent optic disc segmentation using improved particle swarm optimization and evolving ensemble models. Applied Soft Computing, 92, p.106328. 2020.

[51] S.C. Neoh, W. Srisukkham, L. Zhang, S. Todryk, B. Greystoke, C.P. Lim, A. Hossain and N. Aslam. An Intelligent Decision Support System for Leukaemia Diagnosis using Microscopic Blood Images, Scientific Reports, 5 (14938). 2015.

[52] W. Srisukkham, L. Zhang, S.C. Neoh, S. Todryk and C.P. Lim. Intelligent Leukaemia Diagnosis with Bare-Bones PSO based Feature Optimization, Applied Soft Computing, 56. pp. 405-419. 2017.

[53] T. Tan, L. Zhang, S.C. Neoh, and C.P. Lim, C.P. Intelligent Skin Cancer Detection Using Enhanced Particle Swarm Optimization, Knowledge-Based Systems. 2018.