Grenze International Journal of Engineering and Technology, June Issue

Deep Learning-based Video Classification using CNN and LSTM

Prasant Kumar[1] and Praveen Ailawalia[2]
[1-2]Department of Mathematics, Chandigarh University, Gharuan, Punjab, India - 140413
Email: prasantk2510@gmail.com, praveen.e14523@cumail.in

***Abstract*— Video classifying is a difficult field of computer vision, as successfully utilizing spatial and temporal features is essential. In this study, we describe a method that leverages dl methods for identifying human actions in videos. First, we implement a ConvLSTM model that jointly utilizes convolutional operations within the LSTM cells to capture both spatiotemporal features. Additionally, we implement a Longterm Recurrent Convolutional Network (LRCN) that applies a CNN to drawn out spatial features and then uses an LSTM for temporal modeling. We conduct experiments on a subset of the UCF50 dataset to illustrate both models performance in the classification task and the results show both models achieved better performance than expected in classification and the LRCN model slightly outperformed the ConvLSTM in accuracy. The paper includes a meticulous description of the preprocessing pipeline and the architectural design of the models for training and evaluation. Lastly, we provide some ideas for improvement and future work.**

***Index Terms*— Video classification, human action recognition, convolutional neural networks, long short-term memory, ConvLSTM, LRCN.**

I. INTRODUCTION

Identifying human actions in videos is a difficult problem in computer vision that deals with recognizing actions by humans across a sequence of video frames. This task is very important in many fields, including intelligent surveillance systems, video indexing, sports analysis, and human-computer interaction applications. The earlier approaches relied on hand designed features such as "Histogram of Oriented Gradients" (HOG), motion trajectories or optical flow [1] and they tend to fail when there are changes in viewpoint, scale and complex backgrounds.

Human activity recognition from videos is a difficult problem in machine vision that deals with identifying human activities from a sequence of video frames. The problem is highly valuable in many application domains including intelligent surveillance systems, video indexing, sports analysis, and human-computer interaction. Many of the traditional approaches rely on handcrafted features like HOG, motion trajectories, or optical flow [1]. While handcrafted features can be effective, they often suffer from strong variations like image viewpoint, scale, and complex backgrounds.

This work studies human action recognition through implementation of two popular deep learning techniques with the UCF50 dataset [3], which is a comprehensive and realistic collection of YouTube videos containing 50 action class categories. The first approach implements ConvLSTM layers that embed convolutional feature extraction inside LSTM cells, facilitating the modeling of spatial and temporal correlations.

Grenze ID: 01.GIJET.12.2.506

The second approach implements LRCN architecture, where convolutional layers extract spatial information and the temporal information is modeled with LSTM layers - the time-distributed wrapper allows the LSTM layers to process the sequences while the convolutional layers extract temporal features from the space. Both models were trained on the classes selected and separate action classes were defined from retrieved videos which contribute to normalization, spacing and all other preprocessing is required for modelling. We were able to successfully develop models that showed potential in learning task about the action recognition, but the process is not yet finished. Throughout the thesis, we were capable of demonstrating the importance of combining convolutional and recurrent layers for a single task when classifying videos, suggesting that the combination has a real-world applicability for human action recognition when tasks require understanding of a dynamic scene.
The contributions of this paper are:

- Design and implementation of ConvLSTM and LRCN architectures focused on activity recognition.
- Thorough preprocessing pipeline containing frame extraction, resizing, normalization, and sequence sampling.
- Training and validation on a realistic dataset, with proposed experimental analysis.
- Comparison of the performance of both architectures as well as the pros and cons of each approach.

II. Literature Review

Activity classification has evolved greatly a recent shift away from handcrafted feature engineering approaches has been observed and advanced dl techniques. In the early days of action recognition, the techniques most commonly used manually designed features that attempted to learn spatiotemporal representations from video inputs after a number of steps. Some of the most significant handcrafted features are HOG, HOF, and dense trajectories, which captured complex spatial structures and patterns of motion in video frames [5]. However, the common techniques at the time (manual features) has its own limitations, its manual dependencies for feature design, lack of robustness to complex variation in real video data and it had very poor or narrow generalizability to different datasets.
With the rise of dl, CNNs were able to utilize rich spatial features of individual video frames. CNNs also have the ability to learn representations that are hierarchical and thus are more discriminative and adaptive than handcrafted features. RNNs, namely LSTM networks, were used to learn temporal dependence between frames, as they are capable of capturing long-term temporal dynamics which are important for inferring actions in videos [6]. An example is the two-stream CNN.[7] that separates spatial and temporal processing in two streams so that they can be combined later, ultimately presenting a stronger recognition accuracy.
Donahue et al. developed the LRCN into the a deep neural network framework [8]. Their LRCN approach is innovative because it utilizes CNNs to extract spatial features, and it uses the LSTM model to process the sequence of video frames over time, in an fully trainable model. The LRCN model can learn spatial features and temporal features in a unified manner, allowing for a more robust representation to recognize action in videos.
Shi et al. [9] proposed ConvLSTM, which inserts convolution operations inside the LSTM cells in order to learn spatiotemporal dependencies for precipitation nowcasting. Currently, ConvLSTM is also being considered for video classification due to its ability to simultaneously model spatial and temporal features at a temporal step within recurrent units.
Recently, other work has used 3D CNNs to directly model spatiotemporal features [10] or explored transformer-based models that have shown promising results [11]. However, LSTM-based models are still popular because they are interpretable and very efficient when applied to moderately sized datasets.

III. Methodology

The methodology used in this research paper is:

A. Dataset Description

The UCF50 dataset [4] is a benchmark for human action recognition comprised of realistic videos that were gathered from YouTube. It contains 50 action categories, which in total have an average of 133 videos per category, approximately 199 frames per video, an average frame resolution of 320x240 pixels, and frame rate of 26 fps. For this research, we used four classes – Walking With Dog, TaiChi, Swing and HorseRace.

B. Data Preprocessing

Effective preprocessing is the key to ensure deep learning models converge properly through efficient representation learning.

- Frame Extraction and Sampling: Videos are loaded with OpenCV's VideoCapture function. As they can have different video lengths, the program samples a fixed sequence length of 20 frames that occur evenly spaced throughout the length of the video.
- Resizing: All of the extracted frames are resized to 64×64 pixels each to reduce computational burden, as well as maintain the key features.
- Normalization: All pixel values are scaled in range between 0 and 1 divide them into 255 to ensure faster convergence.
- Label Encoding: Class labels are transformed by onehot-encoding, using Keras utilities.

The preprocessing pipeline was implemented to ensure that all video data is supplied in a uniform input shape of (20, 64, 64,
3).

C. Dataset Creation

Each of the classes mentioned was iterated over with frames preprocessed for each video in the dataset. Videos getting less than 20 frames were purposely avoided for uniformity. The final dataset is composed of the features (frame sequences), labels (one-hot vectors), and pathway for the video file(.mp4). Additionally, the dataset is split into train (75%) and test (25%) while remembering to randomize in order to prevent bias.

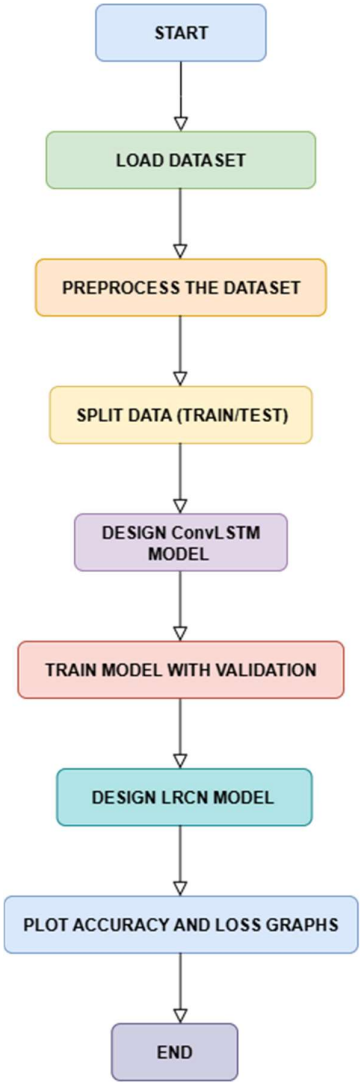

Fig. 1. Algorithm

IV. Model Design

A. ConvLSTM Architecture

ConvLSTM [8] is an extension to the original LSTM concept, in which fully connected operations are substituted with convolutions in the input, forget, and output gates to catch the spatial correlations across feature maps in addition to temporal dependencies since it learned from 3D inputs of shape (frames, height, width, channels).

The architecture implementation for the ConvLSTM model used 4 ConvLSTM2D layers with increasing filter sizes (4, 4, 8, 14, 16) as well as 3D max-pooling and dropout layers to alleviate overfitting and reduce computational complexity. The output of the last layer was flattened and subsequently fed into a dense layer using softmax activation to produce one of four class predictions. (Table 1)

Key Hyperparameters:

- Kernel size: (3,3)
- Activation: tanh
- Recurrent dropout: 0.2
- Sequence length: 20 frames
- Input shape: (20, 64, 64, 3)

TABLE I: CONVLSTM MODEL ARCHITECTURE

Layer Type	Output Shape	Parameters
ConvLSTM2D (filters=4)	(20, 62, 62, 4)	1,024
MaxPooling3D	(20, 31, 31, 4)	0
ConvLSTM2D (filters=8)	(20, 13, 13, 14)	11,144
MaxPooling3D	(20, 7, 7, 14)	0
ConvLSTM2D (filters=16)	(20, 5, 5, 16)	17,344
MaxPooling3D	(20, 3, 3, 16)	0
Flatten	(2880)	0
Dense (softmax)	(4)	11,524

The model contains approximately 44,524 trainable parameters.

B. LRCN Architecture

The LRCN architecture [2] decomposes the feature extraction process into spatial and temporal components. Spatial features are extracted by TimeDistributed CNN layers, utilizing convolutional layers with ReLU activation, maxpooling, and dropout for regularization on a per-frame basis. Feature representations are produced from the CNN layers in the form of flattened tensors, which are then inputted into an LSTM layer with 32 units for spatial-temporal relationship modeling.

The LSTM output is linked to a dense softmax classifier layer for action category predictions. The LRCN architecture allows spatial and temporal components to be trained end-to-end together.

Key Hyperparameters:

- Conv2D kernel size: (3,3)
- Activation: ReLU
- Dropout rate: 0.25
- Sequence length: 20 frames
- Input shape: (20,64,64,3)

TABLE II: LRCN MODEL ARCHITECTURE

Layer Type	Output Shape	Parameters
Conv2D (16 filters)	(20, 64, 64, 16)	448
MaxPooling2D	(20, 16, 16, 16)	0
Conv2D (32 filters)	(20, 16, 16, 32)	4,640
MaxPooling2D	(20, 4, 4, 32)	0
Conv2D (64 filters)	(20, 4, 4, 64)	18,496
MaxPooling2D	(20, 2, 2, 64)	0
Conv2D (64 filters)	(20, 2, 2, 64)	36,928
Flatten	(20, 64)	0
LSTM (32 units)	(32)	12,416
Dense (softmax)	(4)	132

The model has approximately 73,060 trainable parameters.

V. Experiments and Results

A. Training Setup

Both models were constructed using TensorFlow and Keras libraries. The Adam optimizer was utilized along with categorical cross-entropy loss for multi-class classification. We employed early stopping with a patience of 10-epochs (ConvLSTM) and 15 epochs (LRCN) to prevent overfitting.

- Both models compiled with categorical crossentropy loss and Adam optimizer.
- EarlyStopping callback with validation loss monitored with a patience of 10 and 15 epochs respectively for ConvLSTM and LRCN.
- Batch size: 4
- Epochs: max 50 for ConvLSTM, max 70 for LRCN.
- Validation split: 20% from training data.

B. Performance Metrics

We present accuracy and loss on training models, the training dataset, validation and test datasets are recorded during training and validation to determine convergence. Was accomplished using training history before being applied or evaluated on an unseen testing set to gauge generalization of the models.

C. ConvLSTM Results

The training results of the ConvLSTM model indicated that the training accuracy reached 100% by epoch 20. The validation accuracy fluctuated around 91.78%. The test accuracy achieved was 80.33%. The training for each epoch takes about 150 seconds due to the complexity of the ConvLSTM model (474,075 trainable parameters). The model learned, exhibited quick convergence, and was somewhat overfitted due to applying dropout layers to reduce variance. The train and validate loss and accuracy plots indicate the model was able to learn to reduce loss and improve its accuracy while resulting in minimal overfitting from the addition of dropout and application of early stopping (figure 1 and 2).

D. LRCN Results

The LRCN model training results indicated a training accuracy of 100% by epoch 37, while the validation accuracy reached 91.78%. The test accuracy of the LRCN model resulted in an accuracy of 92.6%. Each epoch trained more quickly than the ConvLSTM, averaging approximately 13 seconds per epoch, probably because of the more simplistic recurrent unit. The validation loss and accuracy curves not only indicated that the LRCN model generalized better than the ConvLSTM model , but they also showed consistent convergence with good validation loss and accuracy values and little or no overfitting as seen from figure 3 and 4.

E. Discussion

The LRCN model outperformed ConvLSTM with regard to test accuracy and training computational performance. The separate spatial and temporal modeling of LRCN allows for more flexible feature extraction. ConvLSTM does joint modeling, which is computationally more expensive and more likely to overfit on small datasets. All evidence of both models indicate the importance of jointly modeling the CNN and LSTM components for video classification.

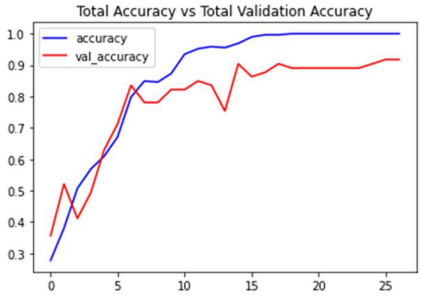

Fig. 2. Total Accuracy Vs Total Validation accuracy Curves for ConvLSTM

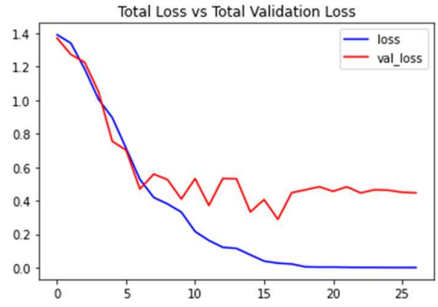

Fig. 3. Total Accuracy Vs Total Validation accuracy Curves for ConvLSTM

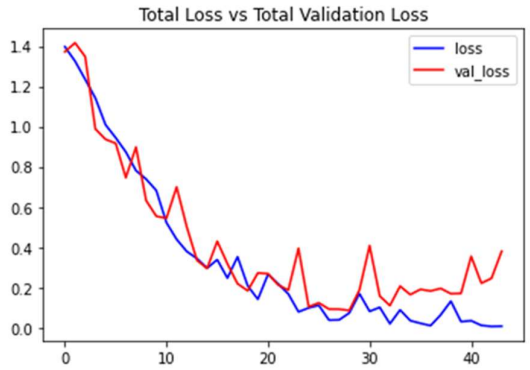

Fig. 4. Total Accuracy Vs Total Validation accuracy Curves for LCRN

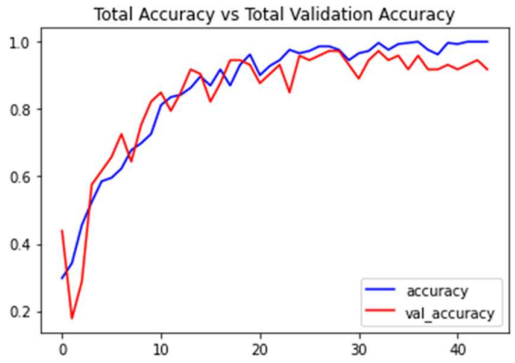

Fig. 5. Total Loss vs Total Validation Loss for LCRN

VI. Conclusion and Future Work

A. Conclusion

In this study, we demonstrated two dl architectures, ConvLSTM and LRCN, for video classification using the UCF50 dataset. Both models were effective in capturing spatiotemporal dynamics dependencies that have proven useful for human activity recognition. The observed results showed that the LRCN model performed more efficient than the ConvLSTM by attaining a max. test accuracy of 92.6% and a max. test accuracy of 80.3%, respectively. A possible reason for LRCNs better performance compared to ConvLSTM is splitting feature extraction (spatial) from sequence modeling (temporal), which increased the accuracy of the classification and reduced the computational expense. The results illustrated how applicable CNN-LSTM hybrid architecture for video classification can be in real-world applications such as surveillance, sports analytics, and human-computer interactions.

B. Future Work

Future research can extend the findings of this study in a number of ways. First, by evaluating the models on broader and more varied datasets such as UCF101 or Kinetics we would also be able to understand the scalability and generalization of the models. Second, by utilizing pre-trained CNN architectures (e.g. ResNet, DenseNet) for spatial feature extraction the models' representational power could be enhanced, and aspects of spatial classification may improve. Third, could leverage transformer architectures and attention mechanisms for a better modeling of time, particularly to allow for long-range communication rather than only relying on recurrent layers. Last, as models reach maturity, focus/importance should shift to real-time deployment and model optimization on edge devices via model compression, pruning, and quantization to enable efficient action recognition on resource constrained devices.

References

[1] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," ACM Computing Surveys, vol. 43, no. 3, pp. 1–43, 2011.
[2] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[3] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2015, pp. 4694–4702.
[4] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," arXiv preprint arXiv:1212.0402, 2012.
[5] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, "Action recognition by¨ dense trajectories," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2011, pp. 3169–3176.
[6] S. Donahue et al., "Long-term recurrent convolutional networks for visual recognition and description," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2015, pp. 2625–2634.
[7] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in Advances in Neural Information Processing Systems, 2014, pp. 568–576.

[8] S. Donahue et al., ”Long-term recurrent convolutional networks for visual recognition and description,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2015, pp. 2625–2634.
[9] X. Shi et al., ”Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in Advances in Neural Information Processing Systems, 2015, pp. 802–810.
[10] D. Tran et al., ”Learning spatiotemporal features with 3D convolutional networks,” in Proc. IEEE International Conference on Computer Vision, 2015, pp. 4489–4497.
[11] A. Arnab et al., ”ViViT: A video vision transformer,” in Proc. IEEE International Conference on Computer Vision, 2021, pp. 6836–6846.