# EARLY ACTION PREDICTION USING VGG16 MODEL AND BIDIRECTIONAL LSTM

Manju D[1], Dr. Seetha M[2], Dr. Sammulal P[3]
*[1] Assistant Professor, Dept.of CSE.GNITS, Hyderabad, India*
*[2] Professor & HOD, Dept. of CSE,GNITS, Hyderabad, India*
*[3]Professor, Dept.of CSE, JNTUH CEJ, Hyderabad, India*

**Abstract: Action prediction plays a key function, where an expected action needs to be identified before the action is completely performed. Prediction means inferring a potential action until it occurs at its early stage. This paper emphasizes on early action prediction, to predict an action before it occurs. In real time scenarios, the early prediction can be very crucial and has many applications like automated driving system, healthcare, video surveillance and other scenarios where a proactive action is needed before the situation goes out of control. VGG16 model is used for the early action prediction which is a convolutional neural network with 16 layers depth. Besides its capability of classifying objects in the frames, the availability of model weights enhances its capability. The model weights are available freely and preferred to used in different applications or models. The VGG-16 model along with Bidirectional structure of Lstm enables the network to provide both backward and forward information at every time step. The results of the proposed approach increased observation ratio ranging from 0.1 to 1.0 compared with the accuracy of GAN model.**

**Keyword: Accuracy, Bi-directional LSTM, observation ratio, Prediction, VGG16 model.**

## Introduction

Now-a-days, there is tremendous need for security monitoring as the crime rate and notarial activities are rapidly growing and taking new shapes which are unpredictable. Constant monitoring of such activities manually through security cameras is a very tedious task and also it is sometimes difficult to immediate action after detecting such criminal activity. There have been many research activities carried out to improve event detection and recognition techniques for visual monitoring applications, but there has been very little effort to accommodate these techniques in real time application. Deep learning is a class of machine learning that plays a pivotal role in solving many challenging tasks of different applications like computer vision, digital forensics, speech processing, NLP, object detection etc. Early action prediction is an approach to predict an action from a live video streaming beforehand, which is crucial in many online applications, such as predicting a malicious actions in the monitoring system, unusual scenarios such as fall detection, behavior recognition, or identification of object posture changes. With the growing technology there is also visible advancement in crime ratio and preventing strategies to avoid crime. Some of the major conspicuous
crimes are witnessed over the world includes street and theft crime. Surveillance is enabled to handle such situations by constant monitoring through CCTV cameras. The footage captured by security cameras can be accessed remotely by an authenticated users and agencies and will also serve as testimonial after the crime scene. Contradicting there is no existing intelligent method to identify or detect the person or specific object before the crime actually take place. It is quite difficult to manually monitor lengthy videos and to detect any abnormal activities from the CCTV footage. The drawback in present security monitoring is taken granted by criminals and is exploited based on their needs.

In this paper, our aim is to focus on prominent model VGG16 for early action frame prediction from surveillance videos where usually background remains constant. Early action detection has primary applications in predicting anomalous events where change in motion pattern and object appearance deviates from ideal patterns. During this process or model, video frames are considered as temporal patterns or time series in contrast to some models were goal of reconstruction is to learn model by reconstructing frames of a video, our approach goal is to model the VGG16 along with Bi-directional LSTM that predicts the action frames similar to that of CNN + LSTM, that is shown in figure 1. The common problem faced by all these established models is the representation learning which includes feature extraction or transformation of input training data that helps in predicting or detecting anomaly event. The procedure is shown in Fig.1

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

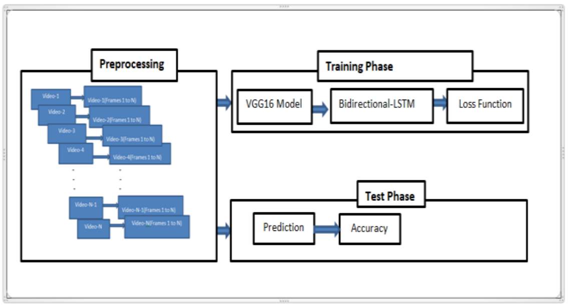


Fig. 1. Early action prediction process

In the figure, it's shown that preprocessing is done on a video by dividing it in to frames. These frames are used for training and testing purpose.

The activation function used in the hidden layers is custom activation. To improve the performance of the proposed approach used Root Mean Squared Error (RMSE), Mean squared error (MSE), Mean Absolute Error (MAE) the commonly used metrics. In simple terms, RMSE measures the average magnitude of the residuals or error. Mathematically, it is computed as the square root of the average of squared differences between predicted and actual values.

The paper is divided into the following sections, Section 2 overviews of existing methods, proposed work in Section 3, implementation results are shown in Section 4 and conclusion in Section 5.

## LITERATURE SURVEY

Many authors have carried out work on different models and used different datasets in early action prediction, where work can be focused on spatio-temporal features to improve accuracy by recognizing the actions accurately by which prediction can be done efficiently. This system develops a multi-stage LSTM architecture that leverages context-aware and action-aware features, and introduces a novel loss function that encourages the model to predict the correct class as early as possible [1]. They designed novel ranking losses to penalize the model on violation of such monotonicity that are used for classification loss in training of LSTM models [2]. The paper focuses on the localization of human action for spatio-temporal movement during a look at fencing video from UCF-101-24 dataset. For classification used CNN model and tend to style an inspired, on-line model for action label construction from single shot multi-box detector which resulted in performance. Two important advances are presented in the paper. First, used CNNs in real-time SSD (Single Shot Multi Box Detector) to lapse and identify detection boxes that potentially contain an action of interest in each video frame. Secondly, from the SSD frame level detections developed an original and efficient on-line algorithm to incrementally create and mark action tubes [3]. The main focus is on the identification of early events that have not been completed. To prevent crimes and stop the dangerous activities that they conduct, behavior prediction is necessary. For this, two dynamic bag-of-words and Integral bag-of-words methods were developed that recognize human activities in a sequential order [4]. The frame work used for detecting ongoing action recognition is MSRNN. In this training of the soft label is done which includes executions by partial intervention and subsequence is given during test process [5]. In order to solve the problem of action prediction, the generative adversarial network was introduced by increasing the accuracy of observed videos by reducing the distance between partially observed videos to the maximum one [6]. The framework used for detecting ongoing action recognition is MSRNN. In this training of the soft label is done which includes executions by partial intervention and subsequence is given during test process [7]. The early predictor system called GAN Predictor is used. In the method, the images created by the generator of specific GAN model which are registered and manually tested to determine if a mode anomaly is present [8]. A quick and precise deep-learning approach was proposed to perform localization and prediction of real-time action. In order to localize many acts and predict their groups in real time, proposed method that uses

convolutionary neural networks. The technique begins with the use of presence and motion detection networks known as "you only look once" (YOLO networks) to use a two-stream model to localize and distinguish behavior from RGB frames and optical flow frames [9]. An innovative method for identification and prediction of human behavior is different software for computer vision, including video monitoring, human, home entertainment that needs real time and online approximations. Suggest a method that is used for joint motion data sources for identifying and predicting behavior in online and in real time, linear latent spaces operating. The strategy is based on an approach that Supervised and has dimensionality reduction [10].

## PROPOSED METHOD

A spatio-temporal features are extracted from a video and detects motion changes that occurs in a video.VGG16 is an Intelligent convolutional model with a capability to learn from frames provided during training with 9 convolutional layers, 4 maxpooling layers and 3 fully connected layers that learns how to correctly classify based on the observed patterns. This is one of the most important architecture and is preferred because of the uniformity. It can easily handle some millions of parameters easily which is difficult to handle by other models. In the architecture of the VGG16 model initially it is represented by two cov1, 2 layers of fixed size 224 x 224 RGB image, Max-pooling is performed over a 2×2 pixel window, with stride

2. It is shown in Fig 2.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

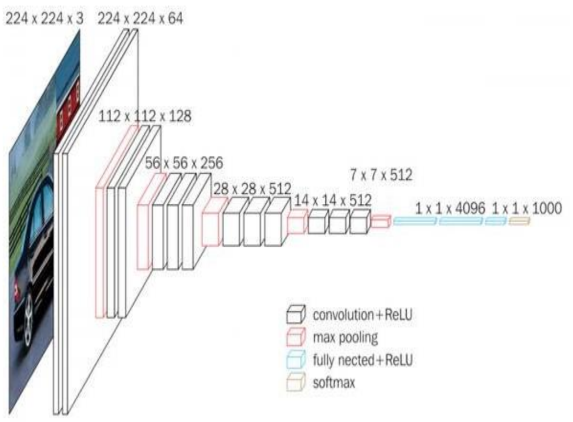


Fig. 2. VGG16 model

The thirteen convolutional layers with three max pooling layers, this process goes on untill the dense layer. There are multiple convolutional layers. Initially, the size of the image is resized to 224X224, then comes to 112X112, then to 56X56, 28X28 and 14X14 and 7, finally fully connected layers with activation function as Relu, softmax and custom layer are used. In this approach the output of the action classification is given as an input to the Bidirectional LSTM model. Modification is done on the Bidirectional LSTM model by adding an extra layer called custom layer, where this layer helps in using the activation function during the back propagation.

The core concept in back-propagation is fine-tuning of the weights of a neural net based on the previous epoch or iteration. Proper weight tuning ensures lower rates of error, rendering the model accurate by increasing its generalization.

Bidirectional LSTM can run the inputs in two directions, one from back to forth and other from forth to back and this is what it differentiates between Bidirectional Lstm and unidirectional Lstm. That is you retain information from the future in the LSTM that runs backwards and can be able to store information from both back and forth at any a particular point of time using the two hidden states combined. Bidirectional LSTMs display very accurate results as they can better grasp the data. Based on the error rate or loss obtained inthe previous iterations, it is the practise of fine-tuning the weights of a neural net. Proper weight tuning means that error rates are smaller, making the model accurate by increasing its generalisation. For that introduced a customer layer where need to have a build method that takes a argument, need to specify shape of the input data. For generating the weight corresponding to input shape and set it in the kernel. It uses a constant initializer to build the weight. Then for implementation considered UCF-101 dataset as shown in fig 3, which is an 8GB dataset of realistic action recognition videos. The process started by importing sequential from keras model, conv2D and Maxpool2D and flatten for used for converting into a single linear vector and Dense in the final layer. Model<-Sequential gives the neural network a sequential network. Then the training and test set is provided with 70% training and 30% testing. Metric values are recorded at the end of each epoch on the training dataset as well as on the testing dataset.

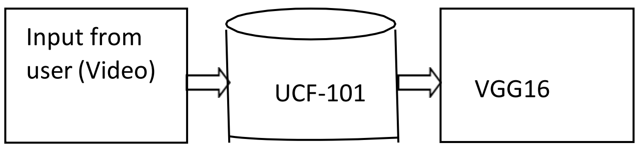


Fig. 3. Shows that the UCF-101 dataset is given as an input to the VGG16 model.

**Pseudo code for proposed method**

```
1. Pseudo Code: VGG16+Bidirectional LSTM
2. Procedure:
3. Reading the input videos
4. def VGG16_Extraction[video_input_file_path]:
     Model Vgg16()
vidcap VideoCapture(video_input_file_path) Frame vidcap.read()
Fm Resize(Frame , (224, 224), interpolation=cv2.INTER_AREA)

input img_to_array(Fm) input preprocess_input(input) feature model.predict(input)
5. Classification: Frames are classified
6. VGG16_Extracction, Labels
7. Initialize Keras backend as K

// VGG16 model used for spitting the data in to the training and testing data.
8. (trainX,testX,trainY,testY) ←
     train_test_split(VGG16_featureslables_size=0.3)
9. def model:

      // Model
      Initialization
      Model ←
               Sequent
      ial()
1.   Adding LSTMlayer ←
     (LSTM(512,kernel_size=3,activation=CustomActiv
     ation))

     Compile ←model.compile('Adam',
     'sparse_categorical_crossentropy',metrics=['acc
     uracy','mae','mse', 'rmse'])

     Model_Fit ←
     model.fit(trainX,trainY,batch_size=BatchSize,
     epochs=150, validation_data=[testX,testY])

     Model_Prediction ←
                              model.predict_classes(test
     Y)

2.   End function
```

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

3. **def** CustomActivation(X,beta=5):

return (X * K.sigmoid(X) * beta) - 1

Initially start with importing all the libraries and packages to implement VGG16 and Bidirectional Lstm models. From keras.preprocessing.image import img_to_array to transform the frames into arrays. Numpy is used to deal with arrays and has functions for linear algebra, fourier transformation, and matrix domain action. Model groups layers with training and inference features into an object. For creating VGG16 architecture use the following commands. Load the vgg16 model in to the variable model and use model.layers.pop() to remove the layers. Given model.inputs and model.layers[-1], output as parameters to model and load into model to get the output of the last layers. For extracting the VGG16 features define a function that takes video as parameter from the dataset and converts the video into frames. Next frames are resized, converted into array and passed as parameter to pre_process() method for preprocessing. Then input is passed to the predict() function in to the model in order to predict the class values for each instance in the array and flattened using ravel(). This is stored in the feature variable. This process is applied for all the frames and features are appended into features variable. Finally get an array of features from this function. Reading videos from the dataset specified by input data directory path and append them to all video file paths list. Then get the count of directories and list of videos from this function. Then import train_test_split method from sklearn to split inputs into random
train and test subsets. Take train and test subsets returned from train_test_split method into x_train,y_train and x_test, y_test respectively.

Now, send each video in all_video_file_paths list as a parameter to extract_vgg16_features method and store the return values (features and frames) into x and train_frame respectively. Append each x to x_train_samples list and each train_frame to train_frames list. By the end of the loop, have all the features of respective frames of all videos in x_train_samples and train_frames. Apply similar process as above for testing and obtain frames as test_frames and their features as xtest_samples, xtrain and xtest values are updated as array of x_train_samples and xtest_samples respectively. After pickling xtrain and xtest as xtrain_samples and xtest_samples consider first element in them as frame and append it to xtrain_frames_list. Using mean function on xtrain_frames_list get xtrain_expected_results. By applying similar process for xtest_samples get xtest_expected_results. For labelling each action uses test_labels dictionary. Each frame in Ytrain give label in accordance with the length of the test_labels. Load sequential model into model and add bidirectional LSTM with dropout 0.7 in one and dropout 0.8 in the other. After adding Bidirectional Lstm layers add dense layer with 512 as size. Then add the custom layer by modifying custom activation function as special activation and add dropout and dense layers again. Finally, add softmax Activation function and compile with loss as categorical_crossentropy and optimizer as rmsprop. Used batch size as 512 and generate batches of this size using generate method with parameters x_samples and y_samples. Then load previously saved xtrain, xtest ,ytrain and ytest python files into xtrain, ytrain,xtest, ytest using load method. By passing xtrain and ytrain as parameters to generate_batches method get array of batches of xtrain and ytrain hold that in train_gen variable. Similarly for xtest and ytest hold in test_gen variable. The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset. One epoch means that every sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. Here considered number of echos as 150. After execution, its observed that loss in minimum and accuracy is quite good.

During this process of the working model used train_test_split function for splitting a single dataset for two different purposes: training and testing. The testing subset is for building model. To evaluate the model's output, the testing subset is for using the model on unknown data to gauge the performance. The build_model() prepares the training data, defines the model, and fits the model on the training data, returning the fit model ready for making predictions. As discussed the first layer in a Sequential, a linear stack of layers that tells each layer has exactly an input layer and one output layer. Activations may be used either through the activation layer, or through the activation statement that all forward layers endorse. Generally VGG16 model is used for classification and a custom activation function to make 0 or 1 predictions is used. A metric is a feature used to calculate the model performance. Metric functions are similar to loss functions. Mean Absolute Error (MAE) computes the mean absolute error between the labels and predictions. Mean squared error (MSE) finds the average squared difference between y_true and y_pred and even Root Mean Squared Error computes root mean squared
error metric between y_true and y_pred. In the training dataset epochs, represents the number of training iterations which is 150 epochs. In deep networks the choice of activation functions has a huge influence on the dynamics of training and the success of tasks. Rectified Linear Unit is currently the most efficient and commonly used activation function but instead of Relu used custom activation. It is defined as the sigmoid function and β is either a constant or a trainable parameter. The degree of interpolation can be controlled by the model if β is set as a trainable parameter. when $\beta \rightarrow \infty$, the sigmoid component approaches a 0-1 function.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

## RESULTS AND DISCUSSIONS

UCF101 dataset comprises of realistic videos collected from YouTube. It contains 101 action categories, with 13320 videos in total. UCF101 gives the largest diversity in terms of actions and in the presence of huge variations of camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc. To start the work used front end as keras and back end as Tensor flow. Keras is an open-source library which provides artificial neural networks with a Python interface. Keras serves as a Tensor Flow library GUI. Keras uses Tensor Flow 2.0's high-level API, an open, highly efficient framework to solve machine learning issues with a focus on modern deep learning. For learning the environment, Bidirectional-LSTM model is used, which in turn helps for predicting the next frames given the past frames as input and to evaluate the quality of predicted frame Mean Square Error is used. For the experimentation, epochs are used to measure the MSE of Training and Test dataset. On changing the different parameters like epochs and batch size above 250, best results can be achieved. Batch size tells how many frames are loaded. The loss function used is MSE; it is the simplest of all the errors which is defined as error in the divergence of prediction from actual rating and the activation function used.

MSE: An estimator's Mean Squared Error (MSE) calculates the average square error, i.e. the mean squared difference between the expected values and the true value. It is non-negative at all times and values similar to zero are better. The MSE is the second moment of the error and thus contains the estimator's variance as well as its bias.

MSE=1/N (∑iN=1(Yi-Y^i)$^2$)

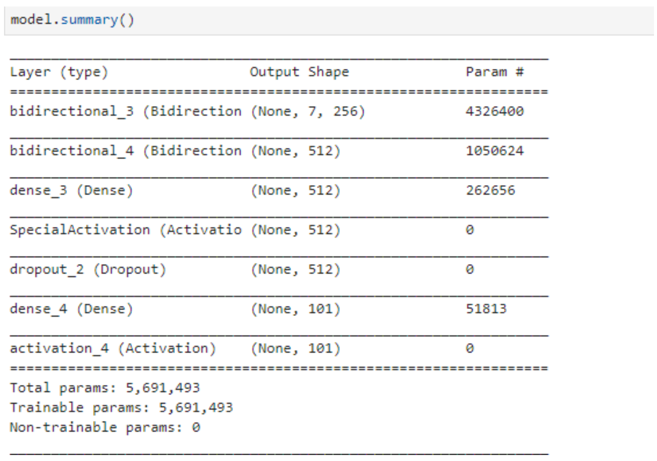
Yi-True values
Y^i- Estimated values
N-Total Number of Observations
Value tends to zero when estimated values are closer to the true values.

MAE: Mean Absolute Error (MAE) calculates the error between measurement of expected versus observed.

ACCURACY: Accuracy is 89% when 10% of the frames are observed, 92% when 50% of the frames are observed and 95% when 100% of the frames are observed. Accuracy of the model is calculated based on the number of correct predictions and total number of observations.

Number of correct predictions /
Accuracy= Total number of observations

If the number of estimated values that are correct are more, than the accuracy is more.

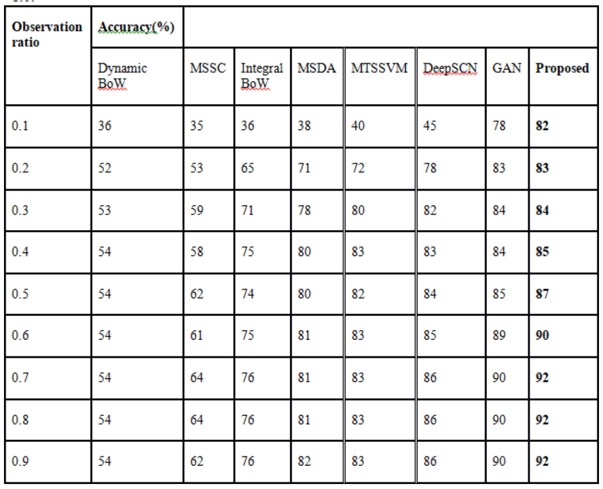


```
model.summary()

_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
bidirectional_3 (Bidirection (None, 7, 256)            4326400   
_________________________________________________________________
bidirectional_4 (Bidirection (None, 512)               1050624   
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656    
_________________________________________________________________
SpecialActivation (Activatio (None, 512)               0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 101)               51813     
_________________________________________________________________
activation_4 (Activation)    (None, 101)               0         
=================================================================
Total params: 5,691,493
Trainable params: 5,691,493
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. It is shown in the model.summary, that there are 512 hidden neurons, the trainable parameters are 5,691,493. Training a model on video data is done to get predictions for each frame.

TABLE I. PREDICTION APPROACHES TESTED ON UCF DATASET WITH DIFFERENT OBSERVATION RATIOS FROM 0.1 TO 1.0.



| **Observation ratio** | **Accuracy(%)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Dynamic BoW | MSSC | Integral BoW | MSDA | MTSSVM | DeepSCN | GAN | **Proposed** |
| 0.1 | 36 | 35 | 36 | 38 | 40 | 45 | 78 | **82** |
| 0.2 | 52 | 53 | 65 | 71 | 72 | 78 | 83 | **83** |
| 0.3 | 53 | 59 | 71 | 78 | 80 | 82 | 84 | **84** |
| 0.4 | 54 | 58 | 75 | 80 | 83 | 83 | 84 | **85** |
| 0.5 | 54 | 62 | 74 | 80 | 82 | 84 | 85 | **87** |
| 0.6 | 54 | 61 | 75 | 81 | 83 | 85 | 89 | **90** |
| 0.7 | 54 | 64 | 76 | 81 | 83 | 86 | 90 | **92** |
| 0.8 | 54 | 64 | 76 | 81 | 83 | 86 | 90 | **92** |
| 0.9 | 54 | 62 | 76 | 82 | 83 | 86 | 90 | **92** |

The prediction comparison results are presented in table. In the table our suggested method, outperformed significantly well compared to the GAN and DeepSCN for the different observation ratios because the sufficient action sequences are trained properly and it's able to perform the prediction correctly. It is observed that the observation ratio from 0.1 to 1.0 for the VGG16 model is increased. It is also noted that our method, obtained best accuracy prediction when 70% of action sequences are observed and it maintained 92% consistently for 80%, 90% and 100 % of frames as there is no change in the action sequences in some video clips.

## CONCLUSION

In this paper it is observed that VGG16 network acquired rich feature representation which is combined with Bidirectional LSTM for early action prediction. The VGG16 model works extremely well in terms of accuracy. From the experimental

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

results, it was shown that the performance of VGG16+Bidirectional LSTM is based on the quality of recognizing the actions, the probability of predicting the actions and on the epochs. It's observed that more reliable future motions are captured using Bidirectional dynamics in videos. Our experimental results on the UCF101 dataset have demonstrated the effectiveness of the proposed method. The loss value can be reduced by increasing the epochs. The given method improves the accuracy compared to the other approaches in the literature and it is concluded that the accuracy of the given method achieves up to 92% over GAN model. This work can be extended further with variations of LSTM Models.

## REFERENCES

[1]. Mohammad Sadegh Aliakbarian1,3 , Fatemeh Sadat Saleh1,3 , Mathieu Salzmann2 , Basura Fernando1 , Lars Petersson1,3 , Lars Andersson3, "Encouraging LSTMs to Anticipate Actions Very Early BY,ICCV,2017

[2]. S. Ma, L. Sigal and S. Sclaroff, "Learning Activity Progression in LSTMs for Activity Detection and Early Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016

[3]. Singh, Gurkirt & Saha, Suman & Sapienza, Michael & Torr, Philip & Cuzzolin, Fabio. (2017). Online Real-Time Multiple Spatiotemporal Action

[4]. 10.1109/ICCV.2017.393.

[5]. M. S. Ryoo, "Human activity prediction: Early recognition of ongoing

[6]. Activities from streaming videos," 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 1036-1043, doi:10.1109/ICCV.2011.612 6349.

[7]. J. Weng, X. Jiang, W. Zheng and J. Yuan, "Early Action Recognition with Category Exclusion using Policy-based Reinforcement Learning," in IEEE Transactions on Circuits and Systems for Video Technology, doi: 10.1109/TCSVT.2020.2976789.

[8]. Wang, Dong, Yuan Yuan, and Qi Wang. "Early Action Prediction With Generative Adversarial Networks." IEEE Access 7 (2019): 35795–35804.

[9]. J. Hu, W. Zheng, L. Ma, G. Wang, J. Lai and J. Zhang, "Early Action Prediction by Soft Regression," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 11, pp. 2568-2583, 1 Nov. 2019, doi: 10.1109/TPAMI.2018.2863279.

[10]. Chenkai Guo, Dengrong Huang, Jianwen Zhang, Jing Xu , Guangdong Bai , Naipeng Dong "Early prediction for mode anomaly in generative adversarial network training: An empirical study". Elsevier,Information Sciences, Volume 534, September 2020, Pages 117-138.

[11]. Ahmed Ali Hammam, Mona M. Soliman, Aboul Ella Hassanien "Real-Time Multiple Spatiotemporal Action Localization and Prediction Approach using Deep Learning". Neural Networks Volume 128, August 2020, Pages 331-344

[12]. VictoriaBloom, VasileiosArgyriou, DimitriosMakris, "Linear Latent Low Dimensional Space for Online Early Action Recognition and Prediction" Pattern Recognition ,Volume 72, December 2017, Pages 532-547

[13]. Mukherjee, Subham & Ghosh, Spandan & Ghosh, Souvik & Kumar, Pradeep & Roy, Partha. Predicting Video-frames Using Encoder-convlstm 10.1109/ICASSP 2019.8682158, IEEE 2019.

[14]. Wei Henglai, Xiaochuan Yin and Penghong Lin, "Novel Video Prediction For Large-Scale Scene Using Optical Flow".arXiv: abs/1805.12243 (2018).

[15]. Liang, Xiaodan, Lisa Lee, Wei Dai and Eric P. Xing, "Dual Motion GAN for Future-Flow Embedded Video Prediction," 2017 IEEE International Conference on Computer Vision (ICCV), Oct. 2017.

[16]. W Byeon, Q Wang, RK Srivastava, P Koumoutsakos, "ContextVP: Fully Context-Aware Localization and Prediction. 365 Video Prediction",European Conference on Computer Vision

[17]. William Lotter and Gabriel Kreiman and David Cox," Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning" 2016, arXiv: 1605.08104[ cs.LG]

[18]. Xu Jingwei, Ni Bingbing, Yang Xiaokang," Video Prediction via Selective Sampling", Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, Pages 1712--1722

[19]. Villegas, Ruben, Jimei Yang, Seunghoon Hong, Xunyu Lin and Honglak Lee. "Decomposing Motion and Content for Natural Video Sequence Prediction." (2017).

[20]. Wonmin Byeon & Qin Wang, Rupesh Kumar Srivastava, & Petros Koumoutsakos, "Fully Context-Aware Video Prediction" (2017)

[21]. Liu Ziwei, Raymond A. Yeh, Xiaoou Tang, Yiming Liu and Aseem Agarwala. "Video Frame Synthesis Using Deep Voxel Flow." 2017 IEEE International Conference on Computer Vision (ICCV) (2017)

[22]. Marc Oliu, Javier Selva and Sergio Escalera. "Folded Recurrent Neural Networks for Future Video Prediction." ECCV (2018).

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

[23]. Villegas, Ruben, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin and Honglak Lee. “Learning to Generate Long-term Future via Hierarchical Prediction.” ICML (2017).

[24]. Pauline Luc, Camille Couprie, Yann Lecun, Jakob Verbeek. “Predicting Future Instance Segmentation by Forecasting Convolutional Features” ECCV-European Conference on Computer Vision, (2018).

[25]. Liang, Junwei & Jiang, Lu & Niebles, Juan Carlos & Hauptmann, Alexander & Li, Fei Fei. “Peeking into the Future: Predicting Future Person Activities and Locations in Videos” CVPR (2019).

[26]. Reda, Fitsum A., Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao and Bryan Catanzaro. “SDCNet: Video Prediction Using Spatially-Displaced Convolution.” ECCV (2018).

[27]. Metz, Luke, Niru Maheswaranathan, Brian Cheung and Jascha Sohl-Dickstein. “Meta-Learning Update Rules for Unsupervised Representation Learning.” ICLR (2019).

[28]. Emily L Denton et al., “Unsupervised learning of disentangled representations from video,” in Advances in Neural Information Processing Systems, 2017.

[29]. Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu, “Deep feature consistent variational autoencoder,” in Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on. IEEE, 2017.

[30]. Unsupervised Hierarchical Video Prediction conference paper at ICLR 2018.

[31]. Nelly Elsayed, Anthony S Maida, and Magdy Bayoumi. Empirical activation function effects on unsupervised convolutional lstm learning. In 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), pages 336–343. IEEE, 2018

[32]. J. Hou, X. Wu, J. Chen, J. Luo, and Y. Jia, “Unsupervised deep learning ofmid level video representation for action recognition,” in AAAI, 2018.

[33]. S. Lai, W.-S. Zhang, J.-F. Hu, and J. Zhang, “Global-local temporal saliency action prediction,” IEEE Transactions on Image Processing, vol. 27, no. 5, pp. 2272–2285, 2018..

[34]. LSTM Networks-The math of Intelligent by Siraj Raval.

[35]. Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang, “Exploiting feature and class relationships in video categorization with regularized deep neural networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 2, pp. 352–364, 2018.

[36]. G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari, “Charades-ego: A large-scale dataset of paired third and first person videos,” arXiv preprint arXiv: 1804.09626, 2018.

[37]. N.Lee,W.Choi,P.Vernaza,C.B.Choy,P.H.Torr,andM .Chandraker, “Desire: Distant future prediction in dynamic scenes with interacting agents,” in CVPR, 2017.

[38]. D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in ICCV, 2015.

[39]. Y. Kong, Z. Tao, and Y. Fu, “Deep sequential context networks for action prediction,” in CVPR, 2017.

[40]. https://towardsdatascience.com/denoising-autoencoders-explained-dbb8 2467fc2

[41]. P.Mohanaiah, p.Sathyanarayana, L.Gurukumar : Image Texture Feature Extraction using GLCM Approach, Volume 3, Issue 5,ISSN 2250-3153 (2013).

[42]. C. Feichtenhofer, A. Pinz, and R. P. Wildes, “Spatiotemporal multiplier networks for video action recognition,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017, pp. 7445–7454.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731