Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 3, 2022, 423-431.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Video Prediction Using Multi-Scale Deep Neural Networks

Nima Shayanfar, Vali Derhami* and Mehdi Rezaeian

Computer engineering department, Yazd University, Yazd, Iran.

Article Info

Article History:
Received 04 December 2021
Revised 14 March 2022
Accepted 03 July 2022

DOI:10.22044/jadm.2022.11415.2305

Keywords:
Deep Neural Networks, Convolutional Autoencoder, Video Prediction, Multi-scale Processing.

**Corresponding author: vderhami@yazd.ac.ir (V. Derhami).*

Abstract

In video prediction, it is expected to predict the next frame of a video by providing a sequence of input frames. Whereas numerous studies exist that tackle frame prediction, a suitable performance is not still achieved, and therefore, the application is an open problem. In this work, multi-scale processing is studied for video prediction, and a new network architecture for multi-scale processing is presented. This architecture is in the broad family of autoencoders. It is comprised of an encoder and decoder. A pretrained VGG is used as an encoder that processes a pyramid of input frames at multiple scales simultaneously. The decoder is based on the 3D convolutional neurons. The presented architecture is studied using three different datasets with varying degrees of difficulty. In addition, the proposed approach is compared with two conventional autoencoders. It is observed that using the pretrained network and multi-scale processing results in a performant approach.

1. Introduction

Videos are a sequence of still frames. Individually, each frame consists of a series of pixels. By showing the sequence of frames in a predetermined order and a predetermined delay, an observer would perceive the sequence as video or motion picture. Based on this definition, there are two dependencies in videos. First, the spatial dependency or intra-frame dependency between the pixels of a frame. This dependency is formed by the depicted scene of the frame. Therefore, in a frame, the values of adjacent pixels are dependent. The complete set of these adjacent pixel dependencies are spatial context. The second dependency is called temporal or inter-frame dependency. This dependency is formed between the corresponding pixels of two consecutive frames [1]. In its most naïve form, temporal dependency is a function of object displacement in the scene among consecutive frames. The assumption here is that the video is continuous in time. This assumption can sometimes be broken, e.g. key frames in videos. The complete set of these temporal dependencies between consecutive frames of a video is the temporal context.

Video prediction is the prediction of a single video frame from a timeseries of video frames. In other words, having frames F_{t-1} to F_{t-n}, it is desirable to predict frame F_t. The approach that would fully exploit the spatial and temporal context would have better predictions.

Video prediction application is a relatively young field, and was a non-application prior to the introduction of deep neural networks and performant hardware, e.g. powerful graphical processing units. Powerful hardware made heavy workloads possible in acceptable timeframes. On the other hand, deep neural networks made the process of manual feature generation easier, and in some cases obsolete. This has revolutionized the process of designing prediction pipelines, and has changed the classic and time-consuming process of design, implementation and deployment of learning approaches.

Video prediction is important from two viewpoints. First, it powers many applications such as video anomaly detection [2], weather forecasting by satellite imagery [3], robotics [4] and hand gesture recognition [5] among others. Second, in almost all instances of video

prediction, a broad family of deep networks named convolutional autoencoder are used. This set of architectures has some key properties. Their input and output are the same, and they attempt to reconstruct the input in the output with some preconditions and modifications. Therefore, they can be categorized as self-supervised methods. Therefore, video prediction application is one of the toy problems to gauge the performance of these architecture. It is to be noted that the focus of this paper is on videos and in addition to spatial context, temporal context is also considered.

Prior to deep neural networks, vast amount of information in a frame virtually prevented from predicting or estimating the next frames. As it was mentioned, an acceptable prediction would require to process interframe and intraframe dependencies and decoding these data without employing big neural networks is challenging. In reviewing the deep network architectures, three separate categories of approaches can be discerned.

In the first category, recurrent networks are employed. These networks (especially the ones that use the relatively newer LSTM and GRU [6]) are efficient in modeling long-term temporal dependencies. One of the most prominent of such approaches belongs to Shi and colleagues [3]. The main obstacle in these networks is the ability to simultaneously model temporal and spatial dependencies. To this end, Shi *et al*. [3] has introduced an LSTM variant that can also model the intraframe dependencies. They have used the new ConvLSTM neuron to predict weather forecasting videos. Lotter has also studied the effect of neural networks' loss functions, and has deduced that using and merging multiple loss functions in the network would lead to superior results [5, 6]. Although approaches based on recurrent networks and their hybrid models has seen relative success, they also have some issues. The network parameters in a modest sized network can get massive relatively easily. This leads to two side-effects. First, their learning is slow, and second, considering the hardware limitations; the number of layers is limited that would lead to a limited performance.

The second category is GAN-based approaches. While the discriminative approaches learn the decision boundary, the generative approaches learn the underlying cause of the boundaries. These approaches learn based on a zero-sum game. The main benefit of these approaches for video prediction is their ability to adapt to uncertainty in future frames [8]. In other words, when there is the possibility of multi-modal prediction, discriminative approaches predict the mean of possibilities, and therefore, reduce their error, while generative approaches can potentially predict all possibilities. For example, Jin *et al*. [9] have used GANs for video prediction and video parsing. They have made a clear distinction between the two, and have separated the video prediction and video parsing steps. In the first step, the frames are predicted. Afterwards, the same network architecture is also used to parse the frames. In another example, Walker *et al*. [10] have used GAN to predict frames. Their main motivation is not to predict pixels. Therefore, in the first step, they have predicted the pose of active objects in the frame, and in the second step, they have used the pose prediction through a conditional GAN to reconstruct the next frame. Their visual results might show the need for more performant approaches. Overall, the GAN-based approaches have good performances but at the same time, their training is challenging and can easily diverge and the mode collapse is a serious problem [11]. In addition, the generated frames are sharp but unrealistic in a closer inspection [12].

The third category is based on convolutional neurons. The 2D convolutional neurons are classically used for modeling intra-frame dependencies and their numerous hybrid approaches even in other categories. On the other hand, these neurons do not have the capability to process temporal context. This can be remedied by using the 3D convolutional neurons. As an example, Van Amersfoort *et al*. [13] have predicted image transforms to transform frame t to frame $t+1$ instead of predicting pixel intensities. Their approach suffers from pixel blur. Some approaches prefer to preprocess the input through the use of foreground/background segmentation [14], and then proceed to only use foreground or process the two segments individually. For example, Vonderick *et al*. [12] have separated the foreground and background predictions, and process each in individual network streams. This approach allows the network to focus on the important foreground, and lowers the overall error. In a more general viewpoint, many approaches prefer to use some form of recurrent networks for frame prediction.

In actuality, many approaches are hybrid approaches, but the categorization gives us the capability to study the pros and cons of each category separately. This paper's proposed approach can be categorized in the convolutional (3^{rd} category), which will be reviewed in the next section.

The main contribution of this paper is presenting a convolutional autoencoder architecture for video prediction. While many studies attack the problem through the use of recurrent networks, it will be shown that a purely convolutional approach is possible and even better. To this end, the new architecture is compared with other related works. In order to compare these architectures, various datasets are used, namely moving MNIST [15], flying things [16] and KITTI [17]. These datasets have varying degrees of challenges, and would paint a comprehensive picture of the approach that shows the importance of multi-scale processing. In addition, the proposed architecture's effectiveness and performance is shown to be superior compared to the previous approaches.

The rest of the paper is organized as follows. Section 2 discusses the basic concepts. Section 3 introduces the proposed approach, Section 4 will discuss the results, and Section 4 concludes the paper**.**

2. Basic Concepts

Figure 1 shows the overall architecture of autoencoders. This architecture consists of an encoder and a decoder. The encoder transforms the input to a compressed feature representation, and the decoder transforms (or reconstructs) the feature representation back to an output that aims to be the same as the original input data. In the encoding phase, the feature maps are usually compressed to lower the width and height of feature maps, whereas the count of feature maps is increased. In the decoding phase, the width and height are increased, while feature map count is decreased.

The decoder's output dimensionality is the same as the original input frame. All autoencoders share this property [12, 16].

Where the layers of an encoder consist of convolutional layers, it is called a convolutional autoencoder. These layers would let the autoencoder model spatial context, and make the overall architecture suitable for image or sound processing. In order to consider the temporal context, one possible approach is to feed multiple input frames to the network. For example, by feeding 5 video frames as input to the network (and having the capability to process them), the temporal dependencies are considered [19].

Autoencoders are considered self-supervised algorithms, in the sense that the input data and label data are the same.

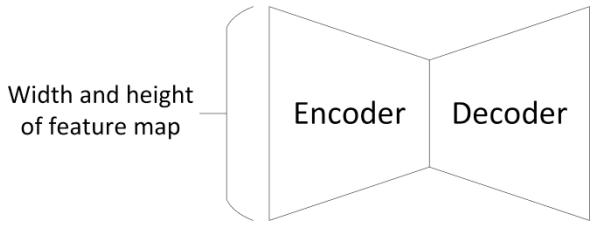

Figure 1. Overall architecture of autoencoders.

3. Proposed Approach

Figure 2 shows the overall architecture of proposed network. Like many other convolutional autoencoders, the architecture consists of an encoder and a decoder.

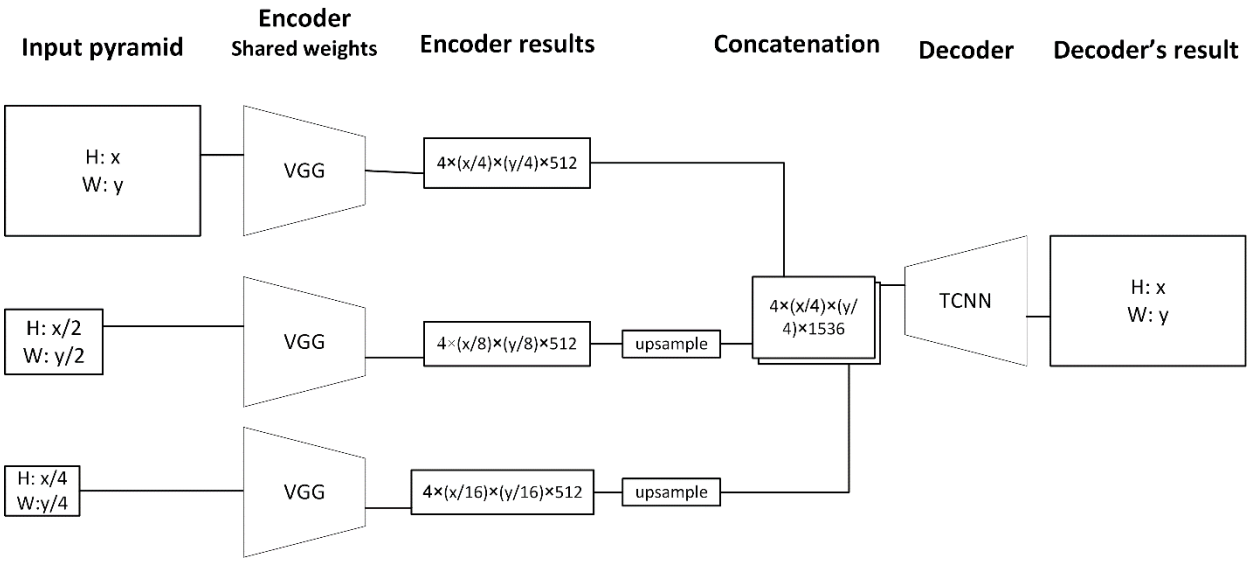

Figure 2. Proposed network architecture.

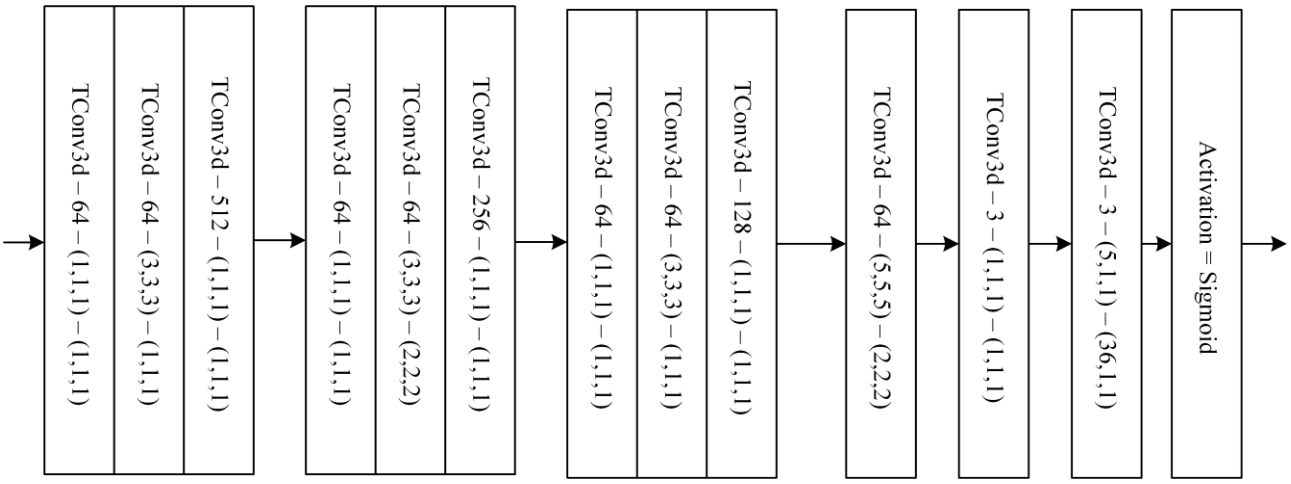

Figure 3. Decoder architecture.

In the encoder, the architecture consists of three separate input streams. These streams are fed by a pyramid of input images. Each stream's input is a sequence of 4 frames. In other words, F_{t-1} to F_{t-4} are considered as input, and are shown by $\boldsymbol{X}_t$. In addition, the input pyramid has a 1/2 multiplier, and each image is half the previous one in each dimension. The total count of encoder streams is equal to the multiplication of pyramid levels and input frame count. Therefore, in Figure 2, there are 12 total encoder streams.

The input streams use the bottom part of VGG network [20] with pretrained and shared weights. The bottom part contains only convolutional layers that would lead to flexibility of input dimensionality. In other words, due to the fully convolutional VGG, input dimensionality can vary (as in each level of input pyramid), and VGG would process the inputs accordingly. Furthermore, the weight sharing between encoder streams enable the architecture to increase its encoder stream count with minimum performance penalty. The multi-stream approach helps the algorithm process the inputs in multiple scales explicitly.

There are some notable points that should be mentioned. First, due to limitations imposed by VGG, the inputs consist of RGB channels. This forces the conversion of grayscale frames to RGB ones. Second, VGG is not designed to consider temporal context, whereas the inputs are comprised of several frames (4 in this article). In the encoder, this is not dealt with. In other words, each input frame in the encoder is processed separately, and all the resulting feature maps are concatenated.

Due to the pyramid of images, the input frames are in different sizes. Therefore, their encoder output is also different in size. In order to have uniform encoder output sizes, second pyramid stream in upsampled by a factor of 2, and the third pyramid stream is upsampled by a factor of 4.

Our proposed decoder is shown in Figure 3 . The decoder has a single stream and is fed by the concatenation of encoders' feature maps. For the decoder to be able to process multiple frames at the same time and access the temporal context, the decoder uses 3D convolutional layers. The decoder consists of 3 main blocks and several independent layers. The blocks gradually decrease the feature map count, while increasing its height and width. Additionally, similar to inception module [21] and through the use of convolutional layers with filter size of (1,1,1) features are reduced, and the overall weight count has been reduced. Furthermore, the decoder will output estimation of frame F_t, which is also called $\boldsymbol{Y}_t$. The training label ($\hat{\boldsymbol{Y}}_t$) is F_t that makes the approach self-supervised.

Finally, the loss function is the binary crossentropy function, and the L2 weight loss is also employed.

4. Results

The proposed approach's results are presented with three datasets, which are ordered by their challenge levels. These are moving MNIST, flying things and KITTI. The moving MNIST and KITTI are used in various video prediction literature to gauge the performance of algorithm [20–23], whereas the flying things is not commonly used but can be a middle ground in terms of image complexity.

Moving MNIST [15] is a set of video clips, which show two numbers moving in an image frame, bouncing off the edges of the image, and overlapping in some other cases. This dataset is basic and fairly easy. Each frame is a 64 by 64

grayscale image. Each video clip consists of 20 frames, and there is a total of 10000 clips. The authors have divided these into 6000 training sequences, 1000 validation sequences, and 3000 test sequences.

Flying things [16] is a subset of SceneFlow dataset. The set of images of each video clip contains not only the frame but also the optical flow, which is not used in this article. The images are created by computer graphics; therefore, they are not from the real world. The frames show objects that are translated and rotated with 6 degrees of freedom in the scene. In addition, the camera is also rotating, though it does not have translation. The dataset creators have divided the sequences into the training and test subsets.

KITTI [17] uses a set of cameras, which are mounted on a driving car. Most clips show the translation and rotation of car (camera), and generally, the objects in the scene have little movements. This dataset is grayscale, and consists of real-world scenes. This dataset is also divided into the training and test sets by the dataset creators. A small subset of the training set is chosen as the validation set by the authors.

In the experiments, in order to report the performance, two metrics are used, namely mean squared error and mean of structural similarity [26]. Mean structural similarity is a full reference metric (in image quality assessment, full reference metrics are metrics that use both the original and degraded images to measure the performance. These metrics are usually slow but at the same time better metrics) used to measure the amount of degradation between two images (original and degraded or in this case reconstructed image). It is first presented by Wang *et al*. [26]. This metric's values are between -1 and 1. The value of 1 represents a perfect reconstruction.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y_i}) \tag{1}$$

$$\mu_x = \frac{1}{M}\sum_{j=1}^{M} x_j \tag{2}$$

$$\sigma_x^2 = \frac{1}{M}\sum_{j=1}^{M}(x_j - \mu_x)^2 \tag{3}$$

$$\sigma_{xy} = \frac{1}{M}\sum_{j=1}^{M}(x_j - \mu_x)(y_j - \mu_y) \tag{4}$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5}$$

$$c_1 = (k_1 L)^2, c_2 = (k_2 L)^2 \tag{6}$$

In the above equations, $\boldsymbol{Y}_i$ is the result of the network using the i[th] sample, whereas $\hat{\boldsymbol{Y}}_i$ is the i[th] sample's label or in other words the next frame. x and y are two corresponding 11x11 windows that are on $\boldsymbol{Y}_i$ and $\hat{\boldsymbol{Y}}_i$. N is total sample count and M is total pixel count (121). In addition, c_1 and c_2 are two small variables that stabilize the equation when the denominator is zero or very small. k_1 and k_2 are 0.01 and 0.03 respectively, and L (the dynamic range) is 255. The result of (5) is a structural similarity image that is the same size and dimension as the network output, and each pixel is a representation of degradation between network's output and label image. The mean structural similarity measure is the mean of this matrix, and is more precisely referred to as MSSIM.

The training is done by Adam's gradient descent algorithm with a learning rate of 0.0001. In addition, the network uses L2 regularization with the rate of 0.0005 and dropout probability of 0.5.

In the first experiment, the proposed approach is tested with Moving MNIST dataset, and is compared with a network completely comprised of ConvLSTM [3]. Table 1 shows the results of this experiment. The compared works are some of the most recent studies in video prediction. The proposed approach and [3] represent two different frameworks for video prediction. The proposed approach receives a limited frame history, in this case, a sequence of 4 frames, while ConvLSTM can potentially save and extract infinite temporal dependencies through the use of its recurrent gates. The proposed approach has a 20% lead compared to ConvLSTM. It is shown in Table 1 that performance lead holds for more recent recurrent-based networks such as [20–22].

Table 1. Result comparison of proposed approach using moving MNIST dataset.

	Structural similarity (MSSIM)%	Mean squared error (MSE)
Derivable memory [27]	N/A	0.044
Z-Net [24]	87.7	N/A
PredRNN [23]	86.7	N/A
PredRNN++ [22]	89.8	N/A
ConvLSTM [3]	71.3	0.014
3D U-Net	85.9	0.013
Proposed approach	**90.5**	**0.009**

Three points stand out in Table 1. First, in the video prediction research, the mean squared error is presented in two ways; on one hand it can be the result of image matrix whose elements are

integers between 0 and 255, and on the other hand, it can be the result of image matrix whose elements are real numbers between 0 and 1. In the first case, the mean squared error is a big number, whereas in the second case, it is a number much smaller than 1. In this work, the mean squared error is presented in the small form (for normalized images). Second, in the table, some values are not presented by the original authors but there are other researchers that have reported their own results. Where it was possible to obtain these results it has been done, otherwise their value is not available. Third, in this experiment and all the next ones, ConvLSTM and 3D U-Net is implemented by the authors according to their respective articles.

The second experiment studies the results of the proposed approach on the KITTI dataset and compares the proposed approach with another common convolutional autoencoder named U-Net [18]. The U-Net architecture was proposed for still images at its inception. However, there are applications that have replaced the 2D convolutional layers with 3D ones, and have therefore included the temporal context in their model [28]. Thus, comparing the proposed approach with this architecture can give a bigger picture of the approaches. The U-Net results in Table 2 is implemented with the 3D approach. As it is evident, the proposed approach has a superior effectiveness.

In a more general term and similar to the proposed approach, U-Net also has an encoder and a decoder, and is also multi-scale. The main benefit of the proposed approach compared to U-Net is the pretrained VGG that affects convergence speed and effectiveness. In addition, the fact that the decoder has access to multi-scale features simultaneously is also beneficial.

Table 2. Result comparison of proposed approach using KITTI dataset.

	Structural similarity (MSSIM)%	Mean squared error (MSE)
ConvLSTM [3]	41.7	0.093
3D U-Net	47.3	0.050
Proposed approach	53.1	0.025

The third experiment studies the results of flying things dataset in Table 3. Although this dataset is not commonly used in video prediction works, it can be a middle ground (in terms of dataset complexity) for this application. Therefore, the results are reported.

Table 3. Results of proposed approach using flying things dataset.

	Structural similarity (MSSIM)%	Mean squared error (MSE)
ConvLSTM [3]	56.73	0.058
3D U-Net	66.61	0.028
Proposed approach	71.92	0.011

Figure 4. Visual results. Top left: KITTI result, Bottom left: KITTI label, Top right: flying things result, Bottom right: flying things label.

Figure 4 shows the visual results of the KITTI dataset along with the flying things dataset.

One of the key aspects of measuring video prediction approaches' performance is their long-term effectiveness. In other words, suppose that a sample with 4 consecutive frames is predicted. In the next step, the prediction with the previous 3 frames are fed into the algorithm as the new input, and this approach is repeated many times. The result of this experiment is a set of predictions that only the first four predictions have access to the original dataset samples, and the rest of the input test samples are comprised of partial or complete prediction frames. Yet, in other words, let us assume that a video clip consists of 20 frames named $\boldsymbol{X}_1 \dots \boldsymbol{X}_{20}$. In the first step, the input sample is $\boldsymbol{X}_1 \dots \boldsymbol{X}_4$ and the output is $\boldsymbol{Y}_5$.

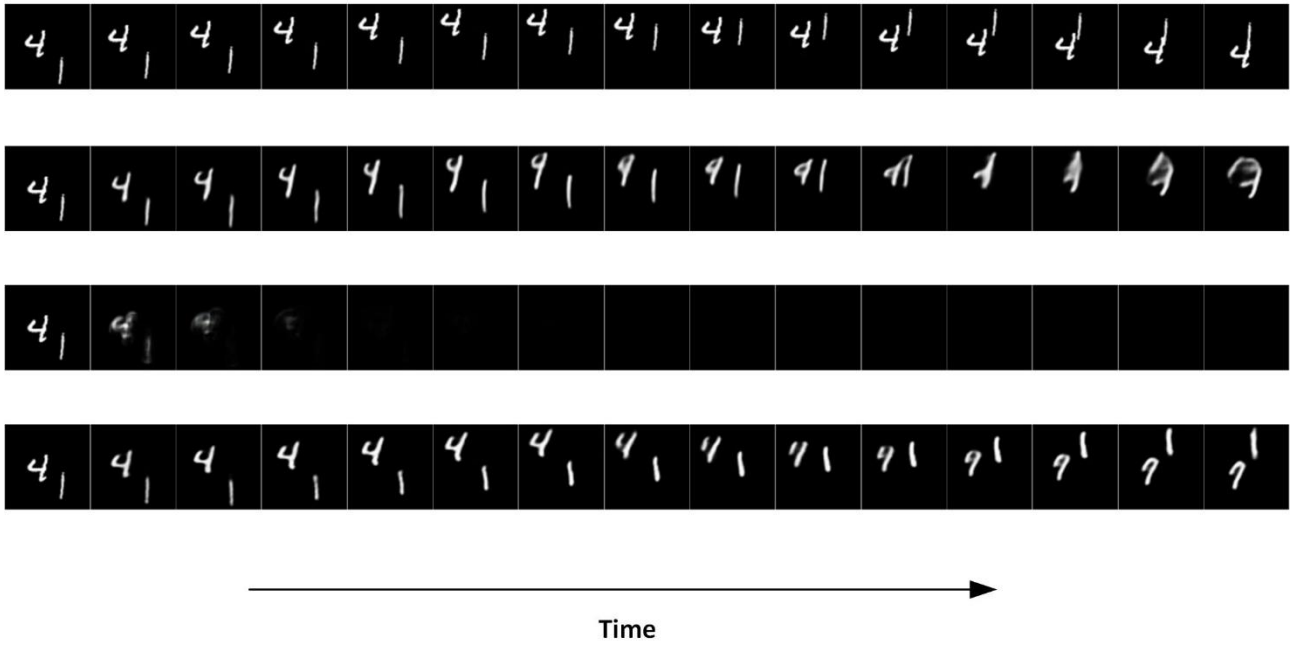

Figure 5. Visual results of long-term experiment. Time progresses to the right. Top row: dataset input, Second row: ConvLSTM results, Third row: U-Net results, Bottom row: proposed approach.

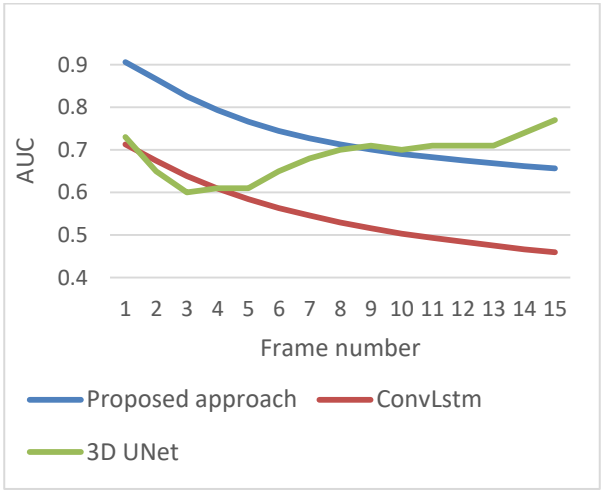

Figure 6. Performance comparison of ConvLSTM and proposed approach in long-term experiment.

This output is compared with $\widehat{\boldsymbol{Y}}_5$ (that is equal to $\boldsymbol{X}_5$), and its performance is measured by MSE and MSSIM. For the second time, the sample consists of $\boldsymbol{X}_2$ ... $\boldsymbol{X}_4, \boldsymbol{Y}_5$. For the third time, it consists of $\boldsymbol{X}_3, \boldsymbol{X}_4, \boldsymbol{Y}_5, \boldsymbol{Y}_6$, and likewise. This method of prediction allows the approach to be benchmarked in the long term. The experiment shows the performance results of long-term tests. They are shown in Figure 6. This experiment is done with the moving MNIST dataset.

Figure 5 shows the visual results of long-term experiment, and compares it with the ConvLSTM and U-Net approach. ConvLSTM and the proposed approach's architectures are different, and represent two completely different approaches to video prediction. The ConvLSTM uses recurrent layers and predicts with accumulation of memory, whereas the proposed approach is memoryless and uses 2D and 3D Convolutional layers. On the other hand, the U-Net's architecture is more similar to the proposed approach (it is based on convolution layers).

The experiment shows interesting results. U-Net destroys the input much sooner than the proposed approach (and ConvLSTM). Therefore, it tends to become an image that is zero in every pixel. However, at the same time, an all zero image yields a relatively large SSIM (Figure 6) because after all the two images (label and result) have much in common. In other words, in the comparison, U-Net becomes a naïve prediction much sooner than the other two approaches. As it is evident, in the long-term prediction, the proposed approach preserves the visual properties of the objects better and has less blur. However, this prediction is only better than a naïve prediction for 8 frames. One can determine this threshold by looking at crossing point of U-Net and proposed approach in Figure 6.

5. Conclusion

In this work, in order to predict video frames, a new multi-scale framework was introduced. In this framework, a new convolutional autoencoder architecture was used. Our proposed deep neural network architecture consisted of three VGG stream to encode the input sample. Streams make the multi-scale processing of inputs possible. The decoding was done by transposed Conv3D layers. These layers had also access to the encoders'

feature maps in multiple scale. The combination of spatiotemporal and multi-scale processing lead to a better performance.

In order to compare this framework, three datasets were used, named Moving MNIST, KITTI and Flying things, and the approaches performance was measured by structural similarity and mean squared errors metrics.

The proposed multi-scale convolutional approach was compared with the recurrent-based approaches on moving MNIST dataset, which lead to at least 0.7% boost to MSSIM. This can be attributed to multi-scale processing of the inputs. In addition, from the results, one can deduce that unlike applications that work on text and sound, video prediction does not benefit from long term history. After all, video prediction largely requires estimating displacing objects by using their speed, and in the strictest definition, deducing speed does not require more than two frames.

In another experiment, the proposed approach was compared with the 3D U-Net on KITTI dataset, and showed 5.7% advantage.

In spite of the results of the proposed approach, it can probably be further optimized. In particular, we are working on the loss functions that decrease blur. In addition, while the proposed framework's performance is considerably improved, the resulting network architecture is quite big (in terms of parameter count), and this results in an increased run time. Using more compact architectures in place of VGG could probably remedy this. The sensitivity test of input sample's frame count can also be mentioned as a future work.

References

[1] C. Zhang and J. Kim, "Modeling Long- and Short-Term Temporal Context for Video Object Detection," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 71–75, doi: 10.1109/ICIP.2019.8802920.

[2] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection - A new baseline," in *Computer Vision and Pattern Recognition*, 2018, pp. 6536–6545.

[3] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[4] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *International Conference on Learning Representations*, 2016, pp. 1–14.

[5] W. Lotter, G. Kreiman, and D. Cox, "Unsupervised Learning of Visual Structure using Predictive Generative Networks," in International Conference on Learning Representations - *Workshop track*, 2016, pp. 1–12.

[6] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," in *International Conference on Learning Representations*, 2017, pp. 1–18.

[7] S. Oprea *et al.*, "A Review on Deep Learning Techniques for Video Prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–26, 2020, doi: 10.1109/TPAMI.2020.3045007.

[8] X. Jin *et al.*, "Video Scene Parsing with Predictive Feature Learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5580–5588.

[9] J. Walker, K. Marino, A. Gupta, and M. Hebert, "The Pose Knows: Video Forecasting by Generating Pose Futures," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3352–3361, doi: 10.1109/ICCV.2017.361.

[10] M. Jamaseb Khollari, V. Derhami, and M. Yazdian Dehkordi, "Variational Generative Adversarial Networks for Preventing Mode Collapse," *Computational Intelligence in Electrical Engineering*, Vol. 13, No. 3, 2021, doi: 10.22108/isee.2021.129742.1495.

[11] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating Videos with Scene Dynamics," in *Neural Information Processing Systems (NIPS)*, 2016, pp. 613–621.

[12] J. van Amersfoort, A. Kannan, M. Ranzato, A. Szlam, D. Tran, and S. Chintala, "Transformation-Based Models of Video Sequences," *arXiv preprint arXiv:1701.08435*, pp. 1–11, 2017.

[13] L. A. Lim and H. Yalim Keles, "Foreground segmentation using convolutional neural networks for multiscale feature encoding," *Pattern Recognition Letters*, Vol. 112, pp. 256–262, 2018, doi: 10.1016/j.patrec.2018.08.002.

[14] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised Learning of Video Representations using LSTMs," in *Proceedings of Machine Learning Research*, 2015, Vol. 37, pp. 843–852, doi: citeulike-article-id:13519737.

[15] N. Mayer *et al.*, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048, doi: 10.1109/CVPR.2016.438.

[16] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070, doi: 10.1109/CVPR.2015.7298925.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 1–8, doi: 10.1007/978-3-319-24574-4_28.

[18] M. Sabokrou, M. Fathy, Z. Moayed, and R. Klette, "Fast and accurate detection and localization of abnormal behavior in crowded scenes," *Machine Vision and Applications*, Vol. 28, No. 8, pp. 965–985, 2017, doi: 10.1007/s00138-017-0869-8.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015, pp. 1–14.

[20] C. Szegedy *et al.*, "Going Deeper with Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[21] Y. Wang, Z. Gao, M. Long, J. Wang, and P. S. Yu, "PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning," in *International Conference on Machine Learning*, 2018, pp. 5123–5132.

[22] Y. Wang, M. Long, J. Wang, Z. Gao, and P. S. Yu, "PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs," in *Neural Information Processing Systems*, 2017, pp. 880–889.

[23] J. Zhang, Y. Wang, M. Long, W. Jianmin, and P. S. Yu, "Z-order recurrent neural networks for video prediction," in *Proceedings - IEEE International Conference on Multimedia and Expo*, 2019, pp. 230–235, doi: 10.1109/ICME.2019.00048.

[24] R. Mahjourian, M. Wicke, and A. Angelova, "Geometry-Based Next Frame Prediction from Monocular Video," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 1700–1707, doi: 10.1109/IVS.2017.7995953.

[25] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, Vol. 13, No. 4, pp. 600–612, 2004, doi: 10.1109/TIP.2003.819861.

[26] V. Patraucean, A. Handa, and R. Cipolla, "Spatio-temporal video autoencoder with differentiable memory," 2016.

[27] T. Wang *et al.*, "MSU-Net: Multiscale Statistical U-Net for Real-Time 3D Cardiac MRI Video Segmentation," *Lecture Notes in Computer Science*, Vol. 11765, pp. 614–622, 2019, doi: 10.1007/978-3-030-32245-8_68.

پیش‌بینی ویدیو با استفاده از شبکه‌های عصبی عمیق چندمقیاسه

نیما شایان‌فر، ولی درهم* و مهدی رضائیان

دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.

ارسال ۲۰۲۱/۱۲/۰۴؛ بازنگری ۲۰۲۲/۰۳/۱۴؛ پذیرش ۲۰۲۲/۰۷/۰۳

چکیده:

در پیش‌بینی ویدیو، هدف پیش‌بینی قاب بعدی با وجود دنباله‌ای از قاب‌های ورودی است. با اینکه تحقیقات زیادی در مورد پیش‌بینی قاب انجام شده است، کارایی بالا دور از دسترس بوده و این موضوع باعث شده که همچنان این کاربرد، زمینه مورد مطالعه باشد. در این تحقیق، پردازش چندمقیاسه برای پیش‌بینی ویدیو بررسی شده و معماری شبکه جدیدی برای پردازش چندمقیاسه ارایه شده است. این معماری از یک رمزگذار و رمزگشا تشکیل شده است. بخش رمزگذار، از VGG از پیش آموزش دیده شده تشکیل شده است که هرمی از قاب‌های ورودی را به صورت همزمان در چند مقیاس پردازش می‌کند و کدگشا بر اساس نرون‌های پیچشی سه بعدی طراحی شده است. معماری ارایه شده با استفاده از سه مجموعه دادگان با درجه سختی متفاوت آزموده شده است. به علاوه، روش ارایه شده با دو خودرمزگذار متداول مقایسه شده است. استفاده از شبکه از پیش آموزش دیده و پردازش چندمقیاسه، باعث کارا بودن نتایج می‌شود.

کلمات کلیدی: شبکه‌های عصبی عمیق، خودرمزگذار پیچشی، پیش‌بینی ویدیو، پردازش چندمقیاسه.