Self-supervised Video Prediction

CudaVision - Learning Computer Vision on GPUs

Karapetyan Ani and Tsaturyan Vladimir

Universität Bonn
ani.karapetyan@uni-bonn.de, Matrikelnummer: 3299535,
vladimir.tsaturyan@uni-bonn.de, Matrikelnummer: 3299527

Abstract. This lab report presents our implementation and evaluation of a neural network model (Fig. 3) for self-supervised video prediction. Transfer learning is used to construct the network based on a pretrained ResNet-18 (He et al., 2016) backbone. The model has been tested on Moving MNIST (Srivastava et al., 2015) and Soccer Robot datasets. A detailed description of the network architecture, data collection and preprocessing, as well as training procedure and results, is provided in the report. The implementation of the project together with some sample results are available at the following link: http://bit.ly/3bOkXtX.

1 Introduction

Deep learning methods have made an enormous advance in the field of artificial intelligence by introducing an end-to-end learning approach that eliminates the need for traditional hand-designed feature extraction mechanisms (LeCun et al., 2015). Many computer vision problems, such as image classification, object detection, action recognition, etc., have been widely studied in the context of deep learning, and many high-performing solutions have been found. A relatively new direction of research is concerned with the video prediction problem.

Video prediction is the problem of predicting the video's next frames given some previous frames. Let us formally define this problem (Oprea et al., 2020). We are given a sequence of n video frames $X = (X_{t-n+1}, \ldots, X_{t-1}, X_t)$ up to time step t, where $X_t \in R^{C \times H \times W}$ with C, H and W denoting the number of channels, height and width of the frames respectively. The task is to predict the next m frames $\hat{Y} = (\hat{Y}_{t+1}, \hat{Y}_{t+2}, \ldots, \hat{Y}_{t+m})$ that make up a meaningful and realistic continuation of the given video part.

On the one hand, there is no labeled data, but on the other hand, some of the frame sequences extracted from videos can be used as input and the others as target outputs for training. Therefore, the task of future frame prediction fits nicely into the context of self-supervised learning paradigm.

Video prediction is a very challenging problem since the network should model the world's dynamics and find powerful representation for its spatio-temporal characteristics. Factors such as sudden camera movements, zooming in or out, lighting condition changes, object deformations give rise to the need for rather complex models and rich datasets.

Video prediction is essential for many real-world applications, such as weather forecasting (Shi et al., 2015); dynamic environment prediction for drones, mobile robots, autonomous vehicles (Lotter et al., 2016; Villegas et al., 2019); etc. Moreover, the captured deeper layer representations can be helpful for some supervised tasks, such as action recognition from videos, where there is a lack of labeled data.

The existing deep networks can be classified into three main categories: Convolutional Models, Recurrent Models, and Generative Models. A detailed summary and review of existing datasets and deep learning approaches for video prediction task can be found in (Oprea et al., 2020).

In this project, we consider a problem of predicting the next $m = 3$ frames, given $n = 3$ previous frames. In this regard, we have implemented an unpublished model *NimbRoPred*. This model was trained and tested on two datasets, namely Moving MNIST (M-MNIST) (Srivastava et al., 2015) and Soccer Robot datasets. The latter has been created by extracting frame sequences from a number of YouTube videos.

2 Related Work

A number of methods have been proposed for solving the video prediction problem. Let us briefly summarize some of the RNN based approaches.

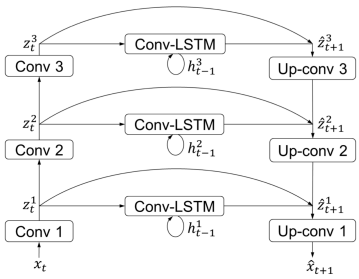

Fig. 1. Overview of VLN. x_t is the frame at time t. $\hat{x}_{t+1}$ is the predicted current frame at time $t+1$. Figure from (Cricri et al., 2016).

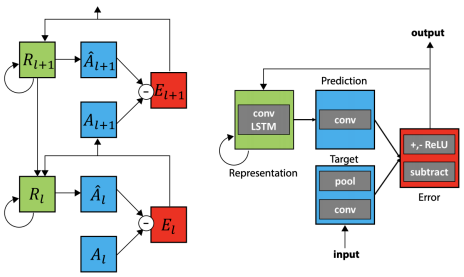

Fig. 2. Overview of PredNet. R_l is a recurrent representation layer, A_l is an input convolutional layer, $\hat{A}_l$ is a prediction layer and E_l is an error representation. Figure from (Lotter et al., 2016).

Video Ladder Networks (VLN) (Cricri et al., 2016) has become one of the most famous neural networks existing in this area. Fig.1 shows the architecture of the network as well as the data flow. VLN is an encoder-decoder network with recurrent lateral connections in terms of convolutional LSTM (Conv-LSTM) blocks (Shi et al., 2015) at all levels. Such recurrent connections allow the decoder to use spatio-temporal summaries generated by different encoder layer, making

more accurate predictions. Unfortunately, due to standard convolutional layers, the model cannot deal with location-dependent features. The model was trained and evaluated using binary cross-entropy loss on the M-MNIST dataset achieving state-of-the-art performance.

Another model, created approximately at the same time as VLN, is Predictive Coding Network (PredNet) (Lotter et al., 2016). Fig.2 shows the architecture and the data flow of the network. PredNet consists of several vertically stacked convolutional LSTM blocks. Each of them is connected by the propagation of error computer at each level. The model was trained with $l1$ and $l2$ losses and evaluated using Mean Squared Error (MSE) and the Structural Similarity Index Measure (SSIM) (Wang et al., 2004) on the synthetic data as well as on natural image sequences datasets.

One of the latest models is Z-Order RNN (Znet) (Zhang et al., 2019). In Znet, the updates of the hidden states are done along a z-order curve. This is done to enhance the consistency of mirrored layer features, since according to the authors, ordinary hidden state transitions do not fully use the homogeneity of the input and the output space in video prediction. Additionally, to increase uncertainty and make results less blurry, an adversarial training approach was used: Znet-Predictor learned to imitate the behavior of the Znet-Probe. For training, Znet-Predictor uses previously generated frames, whereas Znet-Probe uses actual frames. The outputs of each encoder (Znet-Predictor and Znet-Probe) are concatenated and fed into a 3D CNN Discriminator. The discriminator attempts to distinguish between the intermediate feature maps of these two networks. The model was evaluated using MSE and SSIM on the M-MNIST as well as natural image sequences datasets, achieving great results and even outperforming some of the state-of-the-art approaches.

3 *NimbRoPred* Architecture

The suggested *NimbRoPred* network architecture (Fig. 3) belongs to the class of Recurrent Models and has been motivated by Video Ladder Network (VLN) (Cricri et al., 2016) and NimbRoNet2 (Rodriguez et al., 2019) architectures, as well as the newly proposed Location Dependent Convolution (Azizi et al., 2018).

First, it has been inspired by VLN in terms of being a recurrent encoder-decoder network with convolutional blocks that aid in extracting useful spatio-temporal representations from videos. Moreover, similar to NimbRoNet2, NimbRoPred also adopts a pre-trained ResNet-18 encoder. However, in contrast to NimbRoNet2, the suggested network has a much deeper decoder part. Both VLN and NimbRoPred are extended by recurrent residual connections between different layers of encoder and decoder in terms of convolutional recurrent blocks that constitute the memory of the networks. In contrast to VLN, NimbRoPred employs convolutional GRU (convGRU) blocks (Siam et al., 2016) instead of convolutional LSTM blocks in its lateral connections.

The last Global Average Pooling and fully-connected layers are removed from the pre-trained ResNet-18 since it has initially been designated for image classi-

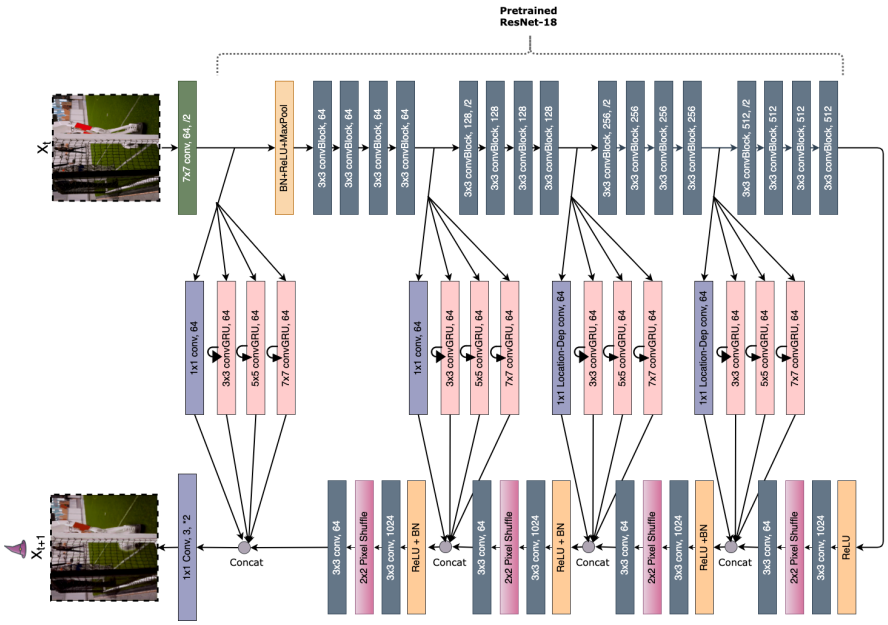

Fig. 3. NimbRoPred architecture. The skip connections in ResNet-18 are not depicted for simplicity.

fication tasks. The decoder uses PixelShuffle (Shi et al., 2016) layers for upsampling. There are four recurrent residual blocks each consisting of three convGRU cells. These recurrent blocks concatenate the output of a single 1×1 convolutional layer with the outputs of the three convGRU cells. Two of the blocks descending from deeper layers of the encoder use a location-aware convolutional layer (Azizi et al., 2018) to aid in learning more location-dependent fine-grained details.

The NimbRoPred model is designed to be applied in an autoregressive manner. The input frames up to time t are fed into the network one at a time, modifying the network's hidden state while the intermediate outputs are being discarded. The output of the last forward pass is used as the first predicted frame, and the following predictions are generated by using the previous prediction as an input.

4 Datasets

Since ResNet-18 has initially been trained on ImageNet dataset (Russakovsky et al., 2015), the input RGB frames are normalized according to ImageNet data statistics, i.e. using mean-standard deviation normalization with $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$.

The M-MNIST dataset has been chosen as a starting point for evaluation of the model, which consists of short videos of two different randomly moving digits. The pregenerated dataset (*M-MNIST*) contains 10 000 samples each presenting a sequence of 20 image frames of size 64×64. Since the initial image frame pixels are in the range of [0, 255], we apply simple rescaling to the range [0, 1]. Additionally, as the model expects inputs of three channels, we convert the gray M-MNIST frames to RGB by duplicating the given single channel.

In order to evaluate the model on a real-life dataset, we used a number of YouTube videos depicting humanoid robots in a soccer-environment. We implemented a custom Dataset class in PyTorch for on-demand loading of video frame sequences from the given set of videos. The extracted frames are resized to a fixed size of 240×320.

5 Training

We have used the following hardware/software in this project: 1) Nvidia GeForce RTX 2080 GPU with 11GB memory; 2) PyTorch/CUDA; 3) Implementation of Location Dependent Convolution (*Location Dependent Convolution*); 4) Implementation of Convolutional GRU (*Convolutional GRU*); 5) Implementation of SSIM loss (*SSIM loss*).

A weighted combination of MSE and Structural Dissimilarity (DSSIM) is used:

$$Loss(Y, \hat{Y}) = \lambda * MSELoss(Y, \hat{Y}) + (1 - \lambda) * DSSIMLoss(Y, \hat{Y})$$

$$= \frac{1}{m} \sum_{i=1}^{m} [\lambda * MSE(Y_i, \hat{Y}_i) + (1 - \lambda) * DSSIM(Y_i, \hat{Y}_i)],$$

with

$$DSSIM(Y_i, \hat{Y}_i) = \frac{(1 - SSIM(Y_i, \hat{Y}_i))}{2},$$

where Y_i and $\hat{Y}_i$ are the i^{th} target and predicted frames, and $m = 3$ in our case. SSIM uses a sliding window approach to compute similarity between different image patches of two images based on several criteria. The sliding window size is set to 11×11. The coefficient λ in the Loss is set to 0.35 and 0.1 in case of M-MNIST and Soccer Robot datasets respectively to control the gain between the two loss parts. Since minimizing MSE loss tends to produce blurry averaged predictions and MSE is easier to optimize, DSSIM loss is given more weight in order to achieve more plausible-looking results.

The pre-trained ResNet-18 encoder has been fine-tuned together with the rest of the network.

Adam optimizer (Kingma and Ba, 2017) has been chosen since it is a state-of-the-art optimizer and has been proven to perform excellent for deep neural networks. The initial learning rate is set to 10^{-4} and $5 * 10^{-4}$ for M-MNIST and Soccer Robot datasets respectively. We added $L2$ regularization by setting the

weight_decay parameter value to 10^{-5}. The remaining parameters are left to their default values.

Moving MNIST dataset was split into training (72%), validation (18%), and testing (10%) datasets. A random 6-long subsequence of the 20-long sample sequence is chosen at each iteration for training. Validation and testing are performed using only the first 6 frames of the 20-long sequences.

For Soccer Robot dataset, we sample every second frame of the videos and uniformly sample around 10 700 frame sequences which are randomly split into training (80%) and validation (20%) datasets. Testing is done on 1700 sample sequences extracted from a single pre-reserved video.

Since it is recommended to use an input size of at least 224 for ResNet, we aim to train and test the final network on inputs of size 224×224 and 224×320 for M-MNIST and Soccer Robot datasets respectively, using bilinear interpolation in case of upsampling the original images.

Table 1. Details of progressive image resizing training method.

	M-MNIST			**Soccer Robot**			
	Step 1	Step 2	Step 3	Step 1	Step 2	Step 3	Step 4
Epochs	50	50	70	40	40	40	40
Input size	64×64	128×128	224×224	64×96	128×192	160×256	224×320
Batch size	64	16	8	64	16	8	4

For evaluating the model on both datasets, we used progressive image resizing approach (Brock et al., 2017; Yosinski et al., 2014) which has been used in some recent works (Rodriguez et al., 2019) to essentially reduce the training time and achieve better convergence. We employed 3 and 4-step training approaches that progressively increase the resolution of input frames for M-MNIST and Soccer Robot datasets respectively (Table 1). Since in this model, only the location-dependent convolutional layers depend on the input size, we insert them only at the last step of training. The final training for 170 (160) epochs took about 26 hours (5 days) of GPU time for M-MNIST (Soccer Robot) dataset. Fig.4 and Fig. 5 show the training and validation error curves during training process for M-MNIST and Soccer Robot datasets respectively. The bumps in the loss curves correspond to the points of increasing the input resolution.

Table 2. MSE, DSSIM and total losses on training, validation and test datasets.

	M-MNIST			**Soccer Robot**		
Data	MSE Loss	DSSIM Loss	Total Loss	MSE Loss	DSSIM Loss	Total Loss
Training	0.1536	0.0611	0.0935	0.0718	0.0772	0.0767
Validation	0.1553	0.0615	0.0943	0.0681	0.0793	0.078
Test	0.1598	0.0613	0.0958	0.1454	0.1675	0.1653

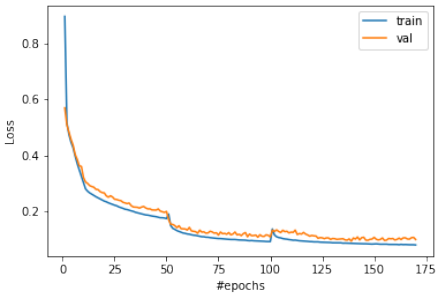

Fig. 4. M-MNIST: Graph of training and validation loss curves.

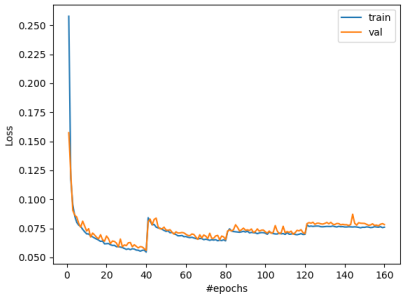

Fig. 5. Soccer Robot: Graph of training and validation loss curves.

6 Results

Table 2 contains the resulting performance metrics, namely MSE, DSSIM, as well as the total losses on training, validation and test datasets. Fig. 6 and Fig. 7 (Fig. 8 and Fig. 9) show sample prediction results for M-MNIST (Soccer Robot) dataset on training and test data respectively.

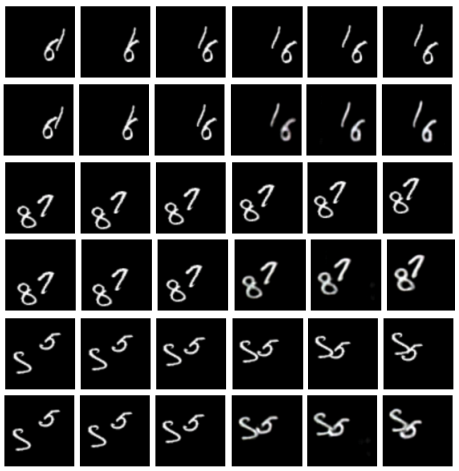

Fig. 6. M-MNIST sample results on training set: Input frames followed by ground truth (top)/predicted (bottom) frames.

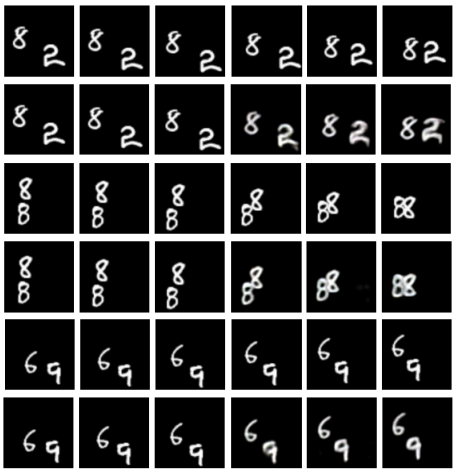

Fig. 7. M-MNIST sample results on test set: Input frames followed by ground truth (top)/predicted (bottom) frames.

Fig. 8. Soccer Robot sample results on training set: Input frames followed by ground truth (top)/predicted (bottom) frames.

Fig. 9. Soccer Robot sample results on test set: Input frames followed by ground truth (top)/predicted (bottom) frames.

7 Conclusion

The model has been implemented and evaluated on two datasets, namely a synthetic (M-MNIST) and a real-life (Soccer Robot) datasets. Progressive image resizing technique has been applied to essentially reduce the training time and achieve better generalization. In order to train the model on Soccer Robot dataset, i.e. a series of YouTube videos, the problem of efficient on-demand loading of frame sequences from videos has been addressed. It is shown that the model is able to achieve quite impressive results on M-MNIST dataset, where the motion of digits is deterministic and predictable enough. Although the results on Soccer Robot dataset are promising, the predictions for fast moving small objects, such as soccer balls, are mostly blurry. With longer training time, a larger and richer dataset, as well as automatic hyper-parameter tuning, the

model could achieve better performance. We conclude, that the problem of next frame prediction is a really challenging task even for such a simplified setting as predicting the next three frames of a short video in real-life scenarios.

In the future work, it may be worth experimenting with other pre-trained CNN backbones such as a ResNeXt (Xie et al., 2017) model. An adversarial loss can be added together with an additional random Z-vector input to give a generative nature to the model. Other perceptual similarity metrics such as Peak Signal-to-Noise Ratio (Fardo et al., 2016) can be employed in the loss to reduce blurriness of the predictions. A mixture of teacher forcing method (LeCun et al., 2015) and free-running inputs may be useful in getting faster convergence. The model should further be assessed by comparing it to some famous video prediction models. Moreover, the inference capabilities of the deployed model should also be evaluated on an edge device with and without GPU availability.

8 Acknowledgement

We want to thank our supervisor Hafez Farazi for his continuous guidance and invaluable advice throughout the completion of this lab project. We are also grateful for the opportunity of using the University's GPU-accelerated computers for our project experiments.

References

AIS-Bonn. *Location Dependent Convolution*. Website. https://github.com/AIS-Bonn/LocDepVideoPrediction/blob/master/vlnOrig.ipynb.

Azizi, Niloofar et al. (2018). "Location Dependency in Video Prediction". In: *CoRR* abs/1810.04937. arXiv: 1810.04937. URL: http://arxiv.org/abs/1810.04937.

Brock, Andrew et al. (2017). *FreezeOut: Accelerate Training by Progressively Freezing Layers*. arXiv: 1706.04983 [stat.ML].

Cricri, Francesco et al. (2016). "Video Ladder Networks". In: *CoRR* abs/1612.01756. arXiv: 1612.01756. URL: http://arxiv.org/abs/1612.01756.

Fardo, Fernando A. et al. (2016). *A Formal Evaluation of PSNR as Quality Measurement Parameter for Image Segmentation Algorithms*. arXiv: 1605.07116 [cs.CV].

happyjin. *Convolutional GRU*. Website. https://github.com/happyjin/ConvGRU-pytorch/blob/master/convGRU.py.

He, K. et al. (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG].

kornia. *SSIM loss*. Website. https://kornia.readthedocs.io/en/latest/_modules/kornia/losses/ssim.html.

LeCun, Yann et al. (May 2015). "Deep Learning". In: *Nature* 521, pp. 436–44. DOI: 10.1038/nature14539.

Lotter, William et al. (2016). "Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning". In: *CoRR* abs/1605.08104. arXiv: 1605.08104. URL: http://arxiv.org/abs/1605.08104.

M-MNIST. http://www.cs.toronto.edu/~nitish/unsupervised_video/. University of Toronto.

Oprea, Sergiu et al. (2020). *A Review on Deep Learning Techniques for Video Prediction*. arXiv: 2004.05214 [cs.CV].

Rodriguez, Diego et al. (2019). "RoboCup 2019 AdultSize Winner NimbRo: Deep Learning Perception, In-Walk Kick, Push Recovery, and Team Play Capabilities". In: *CoRR* abs/1912.07405. arXiv: 1912.07405. URL: http://arxiv.org/abs/1912.07405.

Russakovsky, Olga et al. (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3, pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

Shi, Wenzhe et al. (2016). *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network*. arXiv: 1609.05158 [stat.ML].

Shi, Xingjian et al. (2015). "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *CoRR* abs/1506.04214. arXiv: 1506.04214. URL: http://arxiv.org/abs/1506.04214.

Siam, Mennatullah et al. (2016). "Convolutional Gated Recurrent Networks for Video Segmentation". In: *CoRR* abs/1611.05435. arXiv: 1611.05435. URL: http://arxiv.org/abs/1611.05435.

Srivastava, Nitish et al. (2015). "Unsupervised Learning of Video Representations using LSTMs". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 843–852. URL: http://proceedings.mlr.press/v37/srivastava15.html.

Villegas, Ruben et al. (2019). "High Fidelity Video Prediction with Large Stochastic Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/f7177163c833dff4b38fc8d2872f1ec6-Paper.pdf.

Wang, Zhou et al. (2004). "Image Quality Assessment: From Error Visibility to Structural Similarity." In: *IEEE Trans. Image Processing* 13.4, 600–612.

Xie, Saining et al. (2017). *Aggregated Residual Transformations for Deep Neural Networks*. arXiv: 1611.05431 [cs.CV].

Yosinski, Jason et al. (2014). *How transferable are features in deep neural networks?* arXiv: 1411.1792 [cs.LG].

Zhang, J. et al. (2019). "Z-Order Recurrent Neural Networks for Video Prediction". In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 230–235. DOI: 10.1109/ICME.2019.00048.