Self-supervised Video Prediction

Lab Vision Systems - Learning Vision Systems on Graphics Cards MA-INF 4308

Leuschner Oliver, Scheffczyk Jan

Universität Bonn
s6jascef@uni-bonn.de, Matrikelnummer: 3242317
s6olleus@uni-bonn.de, Matrikelnummer: 3205025

Abstract. In this Lab we implemented and refined a convolutional neural network for supervised video prediction. Given a set of three consecutive frames the network predicts the next three frames. The network architecture has been proposed for the lab and extends on the Video Ladder Network by utilizing additional location dependent convolutions as well as exchanging the LSTM layers to more memory efficient GRU layers. Thus we will call this model Location dependent Video Ladder Network (L-VLN). In addition we explore the possibilities for transfer learning by using the hidden states of the video prediction network for classification. This report will give a short overview of the used technologies as well as an implementation.

1 Introduction

The task of video prediction consists of taking in a sequence of frames and generating succeeding frames from it. This is a challenging task, as the scene as well as it's physical rules have to be learned by a model to make sensible predictions. Furthermore the spatial layout of a scene has to be taken into account if a scene contains static and dynamic elements. The learned internal representation of a scene can additionally be used as features for classifying the scene or the actions performed in the scene. In recent years deep neural networks have advanced many computer vision tasks including video prediction. In this work we build on previous architectures and investigate whether contemporary advancements at different parts of the architecture supplement each other. In the video prediction task, three images are forwarded into the model which is then tasked to predict the subsequent three images. In the video classification task a full video is forwarded to the network. The hidden states of the ConvGRUs can then be used as inputs for a MLP with one hidden layer, classifying the video.

2 Related Work

Computer vision tasks such as object recognition have been dominated by convolutional neural networks in the recent years. Since the introduction of AlexNet

[KSH12] all new records on the ImageNet benchmark have been achieved by convolutional neural network [AIS19]. These models require enormous amounts of computational power and training data which is typically only available to large corporations such as Facebook, Google or Microsoft [Ten20]. However, these pre-trained networks can be used in numerous applications even beyond object recognition which makes state-of-the-art accuracy available to individuals and projects with much smaller data sets by utilizing transfer learning [Tan+18].

The network architecture proposed for this lab utilizes the concept of transfer learning by basing their model on ResNet-18 [He+15]. This network uses fewer weights in comparison to other networks in the ImageNet benchmark and is therefore suited for extensions by limited computational resources. Inspired by the Video ladder network (VLN) [Cri+16] lateral connections between different resolutions are used with residual layers. As ConvGru [Sia+16] reduces both computation and memory and video prediction of the few frames does not require as much long-term memory they replace the original LSTM layers. Furthermore as shown in [AFB18] location dependent convolution can help to reduce a prediction loss and video production tasks.

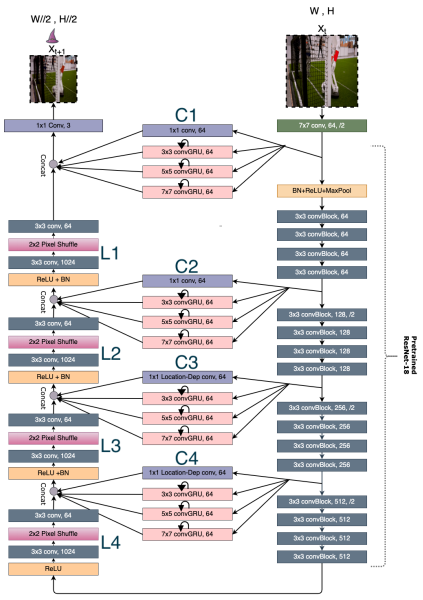


Fig. 1: L-VLN model overview as proposed for this Lab

3 Methods

The network was implemented using pytorch and the pytorch lightning library. The project is structured into loading the data (utils/UCFLoader), defining the models (LVLN_model, one_frame_model, one_frame_model_full_res) and running (main) and evaluating (sample, one_frame_sample) these models. This can be seen in fig. 2. Additionally tensorboard and weights and biases is used for monitoring the training progress.

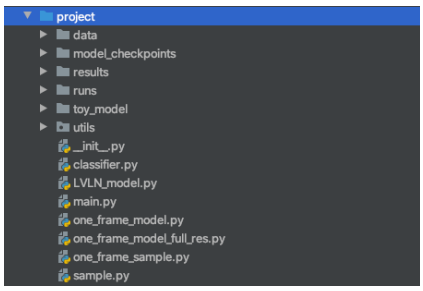


Fig. 2: Setup of the project

The training and validation datasets are loaded from UCF101 videos that have been split into clips of sequences of six frames. We've experimented with both adapting the framerate of the video, and thus effectively increasing the time step in between each successive frame, as well as the dataset size by adapting the step size between clips, effectively reducing the size of the dataset. We settled on first training the network on 25fps and then reducing the fps after four epochs to 15 and then finally 10fps. The intuition being that the network firstly learns to correctly reproduce a scene and then focus on integrating movement by lowering the fps. Varying the step size between clips did not show any significant difference. To ensure a 0 fractional part during the integer divisions present in the four up/downsampling layers we rescaled the input to a multiple of 8 while maintaining the aspect ratio. Therefore frames are rescaled to have a height of 128 and a width of 160. Most recurrent neural networks in pytorch take the full sequence as input, however as we generate frames that will then be used as input we augment the ConvGRU implementation accordingly. Furthermore we adapted the UCF101 loader to allow for transformations of all video frames and implemented a loader for sampling, testing and debugging that loads a single video sequence.

The first three frames are consecutively forwarded through the model. This gives the model the opportunity to learn the temporal features of the scene and differentiate between static and dynamic content. As depicted in figure 3(a), feeding these first three input frames through the network will generate the first predictive frame which will be compared to the fourth input frame. For the last two predictions the previous predictions are used as model input. The

loss function is then used to measure the difference between predicted and real frames. When only using DSSIM as a loss function the network correctly learns the contour of objects in the scene. Moderate L1 and L2 loss are used to ensure correct colors. After testing SGD, Adagrad and Adam we found that Adam significantly outperforms the other options and therefore utilizes a variation AdamW [KB14]. The architecture can be divided into three major parts. A

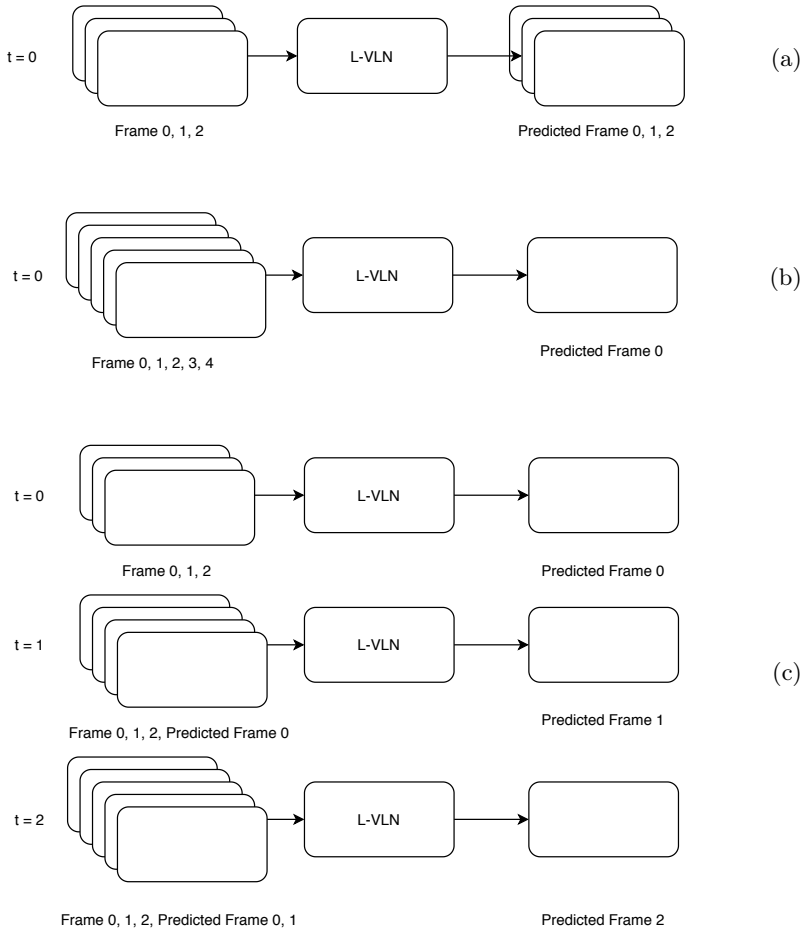


Fig. 3: (a) shows the training and predicition routine as described in the task. (b) shows our modified training and (c) our modified prediction routine.

ResNet18 component with frozen weights, encoding the input, where the output after each block of convolutions is saved. The decoding part that consists of four blocks, L1 to L4 fig. 1, upsampling the frames again. These blocks consist of firstly a batch norm and ReLU activation layer, followed by a convolutional layer,

a pixel shuffle layer tasked with upsampling the image and another convolutional layer. Lastly we've implemented four blocks, C1 to C4 fig. 1, working as latent connections between the encoding and decoding parts of the network. These blocks contain 3 ConvGRUs that learn the temporal features of the scene and one convolution layer. In the deeper two blocks the convolution is exchanged with a location dependent Convolution [AFB18] in order to encode additional location dependent features. We trained the network on a Nvidia 1080Ti GPU with a batch size of 16. One batch entry consists of a sequence of 6 frames. The training progress is continually saved to the disk and can therefore be stopped and restarted at any time.

4 Implementation Details

To confirm, check and debug the L-VLN we first implemented a simplified model. As a simplified dataset we use the horizontal translation of a circle to check if the temporal ConvGRU modules correctly capture and predict the movement. We augment this sample by shifting and mirroring the sequence to get a training set of 5 samples. Both models succeed at learning the motion, as can be seen in figure 4.

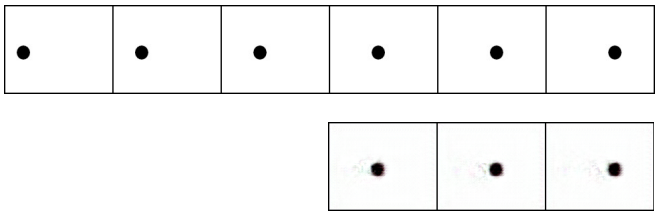


Fig. 4: (a) depicts the three input frames and the teacher images used to compare the predictions in (b) against. This example took about 500 training steps on the augmented circle dataset.

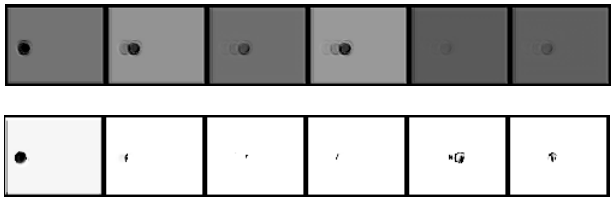


Fig. 5: A single filter of a ConvGRU throughout the training process, before and after training the model

Further we confirm that the hidden state captures and retains salient information throughout the time series, see Figure 5. After verifying that all components work as expected we then test whether the L-VLN can predict movement on subsets of length 5 (fig.11), 20 (fig. 12), 40 (fig. 13) of the UCF101 dataset.

These experiments showed us that we require a training loss of less than 0.1 to achieve results as above. Finally we proceeded to train on 25% of the full dataset for 140 epochs (300.000 training steps) which took us 6 days[1] on our hardware, fig. 8 (b). More samples can be found in the appendix 15. During training the pre-trained weights from the ResNet component where not updated.

Unfortunately the L-VLN model fails to predict movement within the images on larger datasets and instead only blurs the regions containing movement. The loss curve does not indicate that results will significantly improve within a reasonable timeframe since the loss hovers far above 0.8.

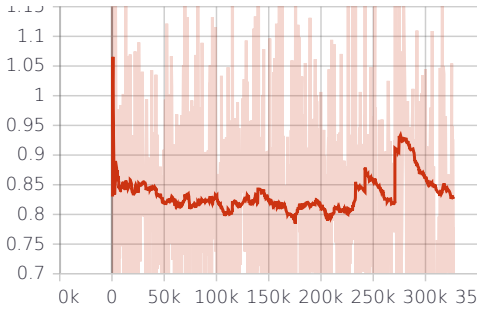
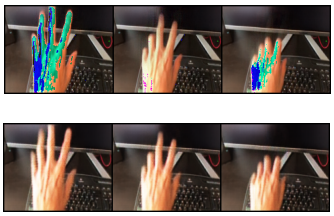


Fig. 6: Training loss of the full L-VLN model over a 300k training step period.

To avoid artifacts from values outside the defined 0, 1 range for rgb values we clip all images to this range before displaying, see fig.7. By ensuring that the coefficients of the three different losses sum to one their ratio can be changed without affecting the learning rate.



(a)

(b)

Fig. 7: shows result with (b) and without (a) clipping.

[1] And roughly 18€ in electricity cost

4.1 Simplifications

The previous experiments suggest that either the problem complexity is too high for the proposed L-VLN model, or we lack the computational resources to adequately train the 31 million parameters. We therefore propose to reduce the problem complexity by predicting only a single frame at a time, given a sequence of varying length. For training we use seeding sequences of 5 frames to predict the sixth frame. However, the model works independent of the length of the seeding sequence. This allows us to append predictions to the seeding frames in order to predict a second and finally third frame. Therefore we can predict three frames similarly to the previous approach. Further to reduce the computational requirements we reduced the input size further, down to 64 by 96 pixels. The proposed L-VLN model returns a predicted image of half the resolution of the input image. In order to compute the loss the output has to be upsampled or the ground truth frames have to be downsampled. As we're already working with fairly small images downsampling them further masks small motion in the process. Therefore we chose to upsample instead before computing the loss. We added a pixel shuffle [Shi+16] before the output that is responsible for the upsampling. While this works well for images in the training set it results in a notable grid pattern on novel data. To ensure that these artifacts are not solely based on the added pixel shuffle we also tried a bilinear and nearest neighbour upsampling, see fig. 9, which resulted in similar but not as severe grid structures. Otherwise this approach manages to predict motion, even on unseen data, unlike the previous attempt fig. 8 (c).

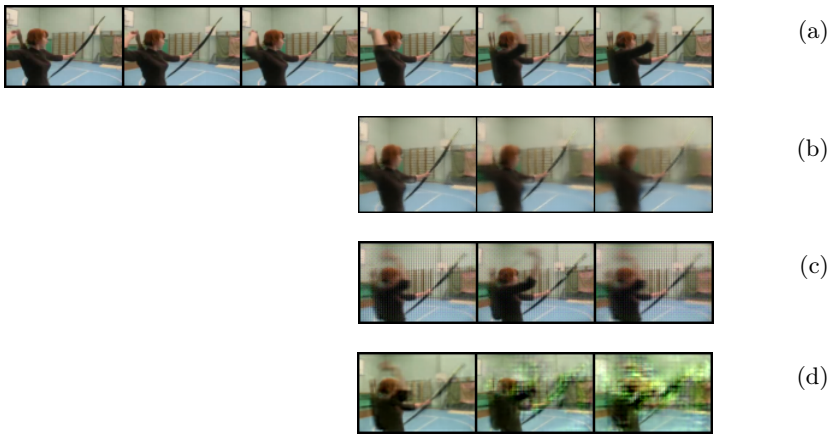


Fig. 8: (b) shows results after training for 300.000 training steps using the original approach, (c) and (d) show the results after training with the single frame approach and the modified architecture respectively.

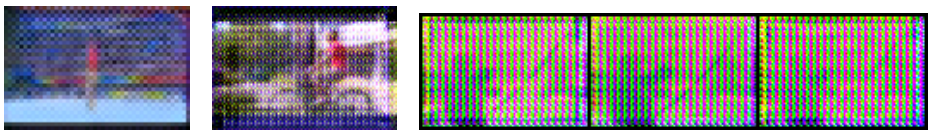


Fig. 9: Artifacts using nearest neighbor upsampling and pixel shuffle. Occurence and intensity increase as training progresses. Therefore early stopping was used for all other one frame samples in this section.

4.2 Full resolution improvements

The grid structures appear to be caused by the upsampling. Therefore we tried to adjust the model, fig. 14, such that it inherently predicts frames of the same resolution. This also requires to reduce the filter, size as the ConvGRU modules are now working on full resolution images. The training uses the same one frame prediction approach as described in section 4.1. While this does address and solve the grid structures we suspect that effective increase of resolution inside the model would again require more computational resources than we have available. See figure 8 (d).

5 Classification

For the video classification task we use our pretrained video prediction model from the previous task, fig. 3(a), as base component. The frames of a video get propagated through the network but we disregard the output. After all frames are forwarded the states of the ConvGRUs are extracted as features for the classification. They are then used as input for a fully connected neural network with one hidden layer and one output layer. Since concatenating all hidden states of all ConvGRUs results in over 1 billion parameters it is computationally not feasible to do so. We then experimented with different strategies for choosing which filters to use as input to the fully connected layers. We reasoned that the hidden states in the deepest latent connection, which are part of the C4 block, should contain the most compressed representation of the sequence. We compared this to subsampling all hidden states. Finally we combined both approaches by sampling all of the deepest filters and additionally taking a subsample of the other hidden states. In addition we tried Batch Normalization and Dropout for regularization, but removed Dropout since it decreased accuracy. As a loss function we use Cross entropy loss and optimize the parameters with Adam. We test the accuracy on the validation set and reach an accuracy of up to 59% on the full validation set. The learning progress can be seen in fig 10.

6 Conclusion

We have shown that the L-VLN architecture is, in principle, able to predict motion in video frames as shown with both a moving circle and small training

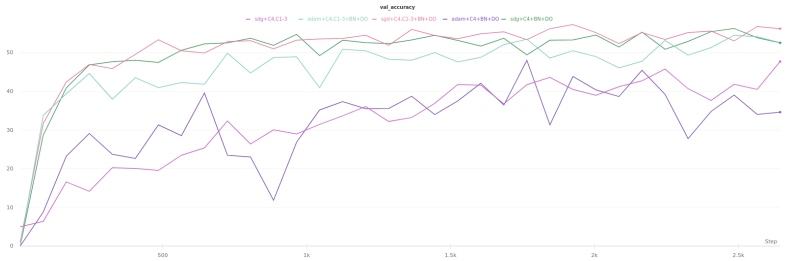


Fig. 10: Comparison of validation accuracy between different normalization methods (BN batch normalization, DO dropout 0.3). Validation has been performed every 0.03 epochs on 5% of the validation set.

sets. We explored multiple simplifications to make training on the full dataset feasible. This allowed us to achieve rudimentary motion prediction with only 6 days of training on a single GPU. Then we tried to adopt the model to address the grid artifacts and possibly improve sharpness. For this we proposed a slight alteration of the L-VLN model. Finally we show that the hidden states of the ConvGRU model can be used for action classification.

References

[AFB18] Niloofar Azizi et al. "Location Dependency in Video Prediction". en. In: *Artificial Neural Networks and Machine Learning – ICANN 2018*. Ed. by Věra Kůrková et al. Vol. 11141. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 630–638. ISBN: 978-3-030-01423-0 978-3-030-01424-7. DOI: 10.1007/978-3-030-01424-7_62. URL: http://link.springer.com/10.1007/978-3-030-01424-7_62 (visited on 10/08/2020).

[AIS19] AISmartz. *CNN Architectures Timeline (1998-2019)*. en. Oct. 2019. URL: https://www.aismartz.com/blog/cnn-architectures/ (visited on 10/08/2020).

[Cri+16] Francesco Cricri et al. "Video Ladder Networks". en. In: *arXiv:1612.01756 [cs, stat]* (Dec. 2016). arXiv: 1612.01756. URL: http://arxiv.org/abs/1612.01756 (visited on 10/08/2020).

[He+15] Kaiming He et al. "Deep Residual Learning for Image Recognition". en. In: *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 10/08/2020).

[KB14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". en. In: *arXiv:1412.6980 [cs]* (Jan. 2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 10/08/2020).

[KSH12] Alex Krizhevsky et al. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

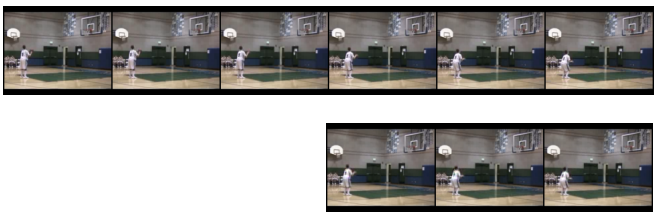
[Shi+16] Wenzhe Shi et al. "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network". en. In: *arXiv:1609.05158 [cs, stat]* (Sept. 2016). arXiv: 1609.05158. URL: http://arxiv.org/abs/1609.05158 (visited on 10/08/2020).

[Sia+16] Mennatullah Siam et al. "Convolutional Gated Recurrent Networks for Video Segmentation". en. In: *arXiv:1611.05435 [cs]* (Nov. 2016). arXiv: 1611.05435. URL: http://arxiv.org/abs/1611.05435 (visited on 10/08/2020).

[Tan+18] Chuanqi Tan et al. "A Survey on Deep Transfer Learning". en. In: *arXiv:1808.01974 [cs, stat]* (Aug. 2018). arXiv: 1808.01974. URL: http://arxiv.org/abs/1808.01974 (visited on 10/08/2020).

[Ten20] Tensorflow. *How Hugging Face achieved a 2x performance boost for Question Answering with DistilBERT in Node.js*. en. May 2020. URL: https://blog.tensorflow.org/2020/05/how-hugging-face-achieved-2x-performance-boost-question-answering.html (visited on 10/08/2020).

7 Appendix



(a)

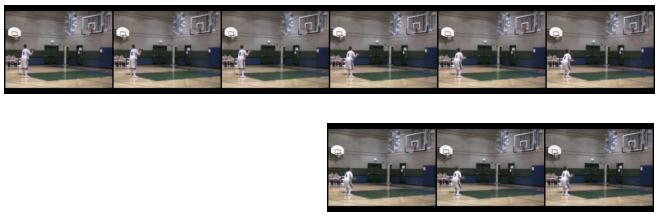
(b)

Fig. 11: (a) depicts the three input frames and the teacher images used to compare the predictions in (b) against. This example was trained on a trainingset of size 5.

(a)

(b)

Fig. 12: (a) depicts the three input frames and the teacher images used to compare the predictions in (b) against. This example was trained on a trainingset of size 20.



(a)

(b)

Fig. 13: (a) depicts the three input frames and the teacher images used to compare the predictions in (b) against.This example was trained on a trainingset of size 40.

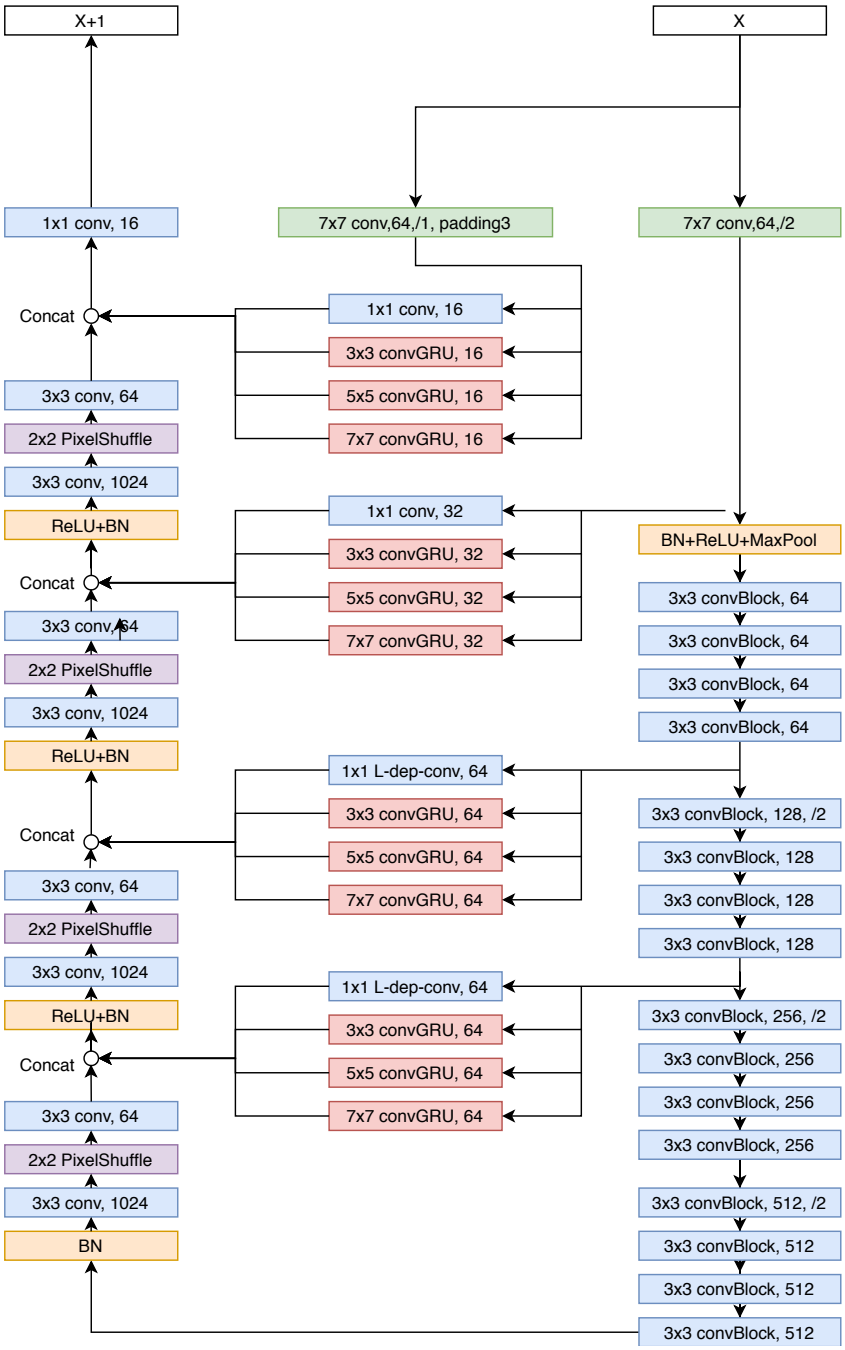


Fig. 14: L-VLN architecture modified for improved full resolution prediction and reduced memory consumption

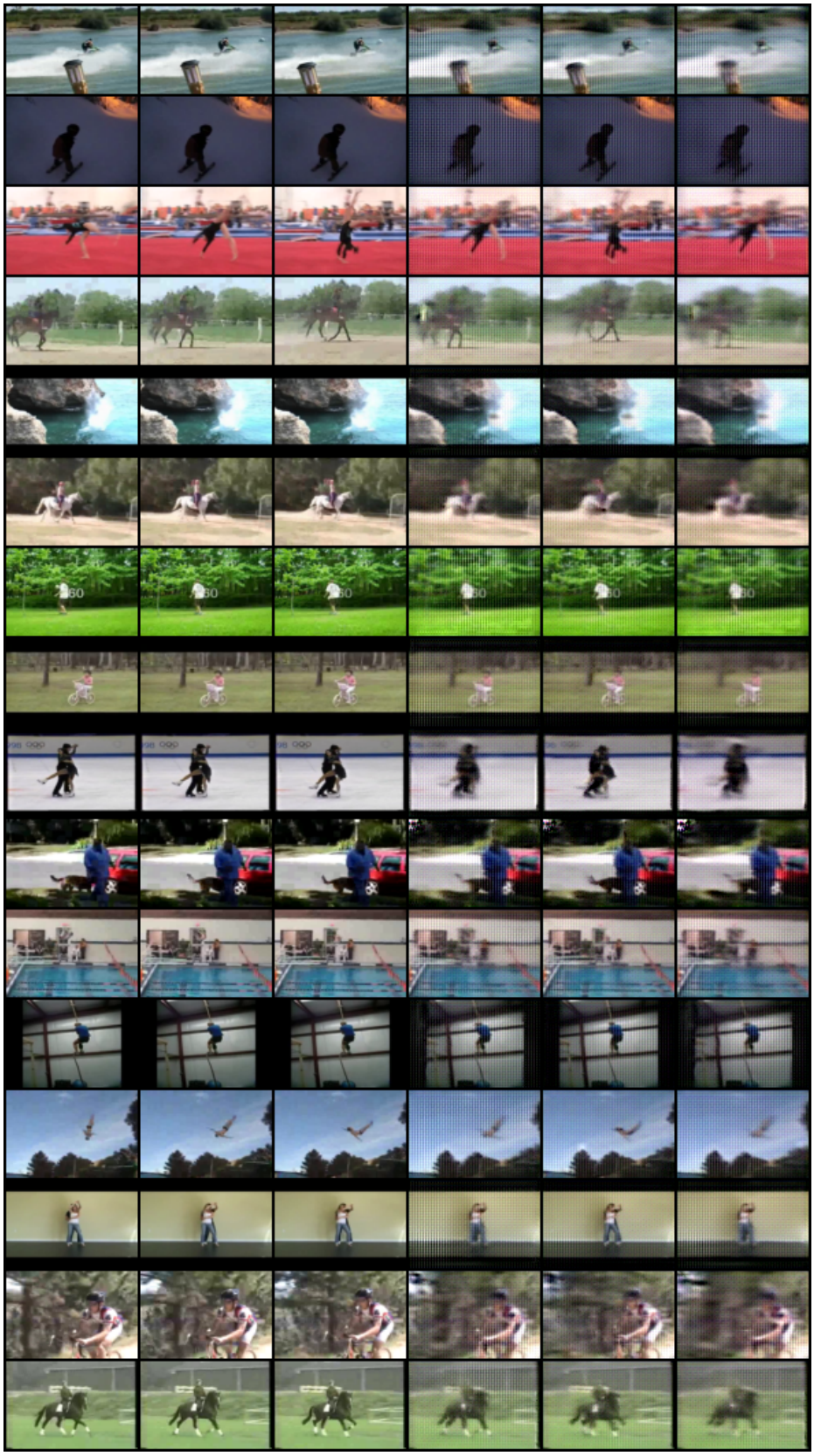


Fig. 15: Samples generated by the one frame approach