Bachelorthesis

Comparing different state-of-the-art solutions for image prediction using time-series analysis

Sören Dittrich
soeren.dittrich@uni-hildesheim.de

September 2020

Abstract

This thesis compares different state-of-the-art solutions for image prediction. Key aspect of the work is the comparison of different versions of the ConvLSTM [13]. To be able to compare those different versions, an image prediction architecture, more explicit PredNet [8], is implemented as baseline architecture. This architecture uses the ConvLSTM module as recurrent sub-module. This sub-module is then changed during the experiments with another implementation (PredRNN [17]). Other comparisons are then performed theoretically The thesis introduces the reader about image prediction, convolutional LSTM's and other necessary parts to be able to follow. It then describes the PredNet architecture and idea and how this differs from other image prediction architectures. Then the performed experiments are described in-depth. Afterwards it gives a comprehensive discussion part, where the different sub-module performances are discussed very deeply, to understand the experimental results. Lastly there is a conclusion, which sums everything up.

Contents

1 Scientific questions

1. What different types of image prediction architectures exist?
2. What different types of recurrent modules exist?
3. How important is the choice of the recurrent module for the runtime and performance of the algorithm?

2 Introduction

The thesis will compare different state-of-the-art solutions for image-/video-prediction 2.4. The main module of the solutions, which is the core aspect of this work, is the LSTM (Long short-term memory). 2.7. This module was invented by Hochreiter and Schmidhuber [5] in 1997 and is used heavily in the field of image-& video-prediction since then, e.g. in Srivastava et. al. [14]. During the time the module got many different add-ons and changes, which are described in different papers ([11], [8], [17], [16] and many more.). To have a valid comparison, I implement three different state-of-the-art solutions for image-/video-prediction ([13], [11] and [8]). All of them use the Shi et. al. ConvLSTM [13] (Or a slightly different version) as recurrent sub-module, which is changed during the experiments with another, more advanced solution named PredRNN [17]. The algorithms are re-implemented in PyTorch [10], as well as the „standard"ConvLSTM and PredRNN. This introduction will cover the basic knowledge, the reader should have to follow the rest of the thesis.

2.1 Deep Learning

2.2 Backpropagation

2.3 Backpropagation through time (BPTT)

2.4 Image Prediction / Video Prediction

Image-/Video-prediction is a field inside machine learning, where the key is to predict future images, given a sequence of image. The image sequence X is of length n, $(x_0, \ldots, x_{n-1})$. One possible use-case is the one-frame prediction, where one predicts x_n, given the the sequence X. Another common use-case is multi-frame prediction, where the key is to predict t many frames into the future. This is often performed using sequence-to-sequence learning [15]. Obviously the first frames look much better then the last frames, as ground-truth is missing, and the predicted frames are only approximated, which means they contain a certain level of error.

2.5 Autoencoder

The autoencoder is a network architecture, which simply consists of two neural networks chained together. The first network is called „Encoder". This part gets the input x and outputs the „code"h. Often the output layer of the „Encoder"is named bottleneck-layer. The second network is called „Decoder". It gets the „code"h as input and outputs $\hat{x}$. This architecture is often used for reconstruction, where $x = \hat{x}$. To prevent the architecture to simply copy the input directly to the output (which would be an interpolation, and not the goal of an autoencoder.), there are different techniques to have the autoencoder to instead approximate the output, given the „code"h. The simplest autoencoder architecture is the so named „undercomplete"; autoencoder [4], in which the output of the bottleneck-layer h is smaller then the input x. Therefore the architecture needs to learn how to extract useful features from the input x, because it is not able to simply copy the input x to the output $\hat{x}$. There are many different ideas of using an autoencoder architecture, which are described more in-depth in Goodfellow et. al. [4].

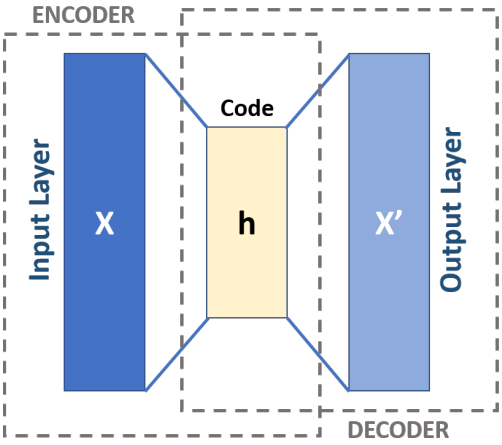

Figure 1: Autoencoder schema [9]

2.6 RNN

RNN (Recurrent neural network)

2.7 LSTM

LSTM (Long Short-term Memory) [5] is a form of RNN, which avoids a critical problem of standard RNN: Saving **Long-term dependencies** [4]. The architecture consists of different submodules, an inpute-gate, forget-gate, cell-state and output-gate.

$$i_t = \sigma(w_{x_i}x_t + w_{h_i}h_{t-1} + w_{c_i}c_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(w_{x_f}x_t + w_{h_f}h_{t-1} + w_{c_f}c_{t-1} + b_f) \tag{2}$$

$$c_t = f_t c_{t-1} + i_t tanh(w_{x_c}x_t + w_{h_c}h_{t-1} + b_c) \tag{3}$$

$$o_t = \sigma(w_{x_o}x_t + w_{h_o}h_{t-1} + w_{c_o}c_t + b_o) \tag{4}$$

$$h_t = o_t tanh(c_t) \tag{5}$$

w is the weight of the layer, σ the sigmoid function, b the layer bias. h_t is the output, in RNN's the output is often denoted as hidden.

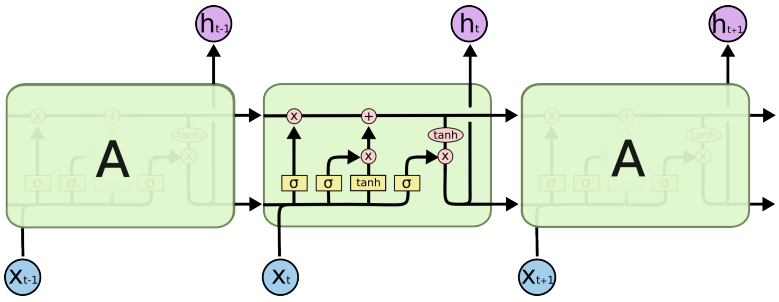

Figure 2: LSTM Architecture [1]

2.8 ConvLSTM

The convolutional LSTM, invented by Shi et. al. [13] is a LSTM using convolutional layer instead of fully connected ones. Therefore the formulas looks very similar to the ones in section 2.7.

$$i_t = \sigma(x_t * w_{x_i} + h_{t-1} * w_{h_i} + w_{i_b}) \tag{6}$$

$$f_t = \sigma(x_t * w_{x_f} + h_{t-1} * w_{h_f} + w_{f_b}) \tag{7}$$

$$\tilde{c_t} = tanh(x_t * w_{x_{\tilde{c}}} + h_{t-1} * w_{h_{\tilde{c}}} + w_{c_{\tilde{b}}}) \tag{8}$$

$$c_t = \tilde{c_t} \odot i_t + c_{t-1} \odot f_t \tag{9}$$

$$o_t = \sigma(x_t * w_{x_o} + h_{t-1} * w_{h_o} + w_{o_b}) \tag{10}$$

$$h_t = o_t \odot tanh(c_t) \tag{11}$$

$*$ is the commonly used sign for the convolution operation.
$\odot$ is the hadamard product (point-wise multiplication).

2.9 PyTorch

3 Image Prediction Architectures

This section will describe a range of state-of-the-art architectures for image prediction. Image prediction is a very broad field, but almost all state-of-the-art solutions for image prediction share one common part, the recurrent module. LSTM's are the most used modules in image prediction, as they are able to store information over a long period of time, despite the standard RNN (recurrent neural network). All algorithms described here have a different way to perform image prediction, but all use a type of LSTM to store the time-series information. It is very common for this type of papers, that the authors start their experiments by using a synthetic dataset. In the most cases this is MovingMNIST [7] and then afterwards performing tests on natural images. For natural images, the authors often use action-recognition datasets, because the camera is fixed and only objects/people in the scenery are moving. Another approach is using natural image datasets, where the camera is also moving through the scenery, for example Kitti dataset [3]. This dataset consists of natural images from different car drives through the city and residential area of Karlsruhe in Germany. In this dataset not only objects in the scenery are moving, but also the camera has a self-motion, which can be very tricky for image prediction algorithms to „understand“.

3.1 LSTM Autoencoder

The paper „Unsupervised Learning of Video Representations using LSTMs“by Srivastava et. al. [14] is using the standard LSTM 2.7 in an autoencoder architecture 2.5 for image reconstruction and future image prediction. The architecture is often used as a baseline in newer and more advanced architectures, because it consists of the standard LSTM as recurrent module. Due to the fact, that the LSTM module is not able to handle multi-dimensional data as is, the images need to be reshaped at the input and also at the output again. The authors use MovingMNIST [7] as synthetic dataset, where every image is of size $(64 \times 64 \times 1)$. Therefore the image is

vectorized into $(64 \cdot 64 \times 1) = (4096 \times 1)$. This MovingMNIST implementation consists of two digits inside every frame. The authors input 10 images and output the next 10 images. The model is end-to-end differentiable and trained using BPTT (backpropagation through time) [18]. For the synthetic dataset, the model is trained using cross-entropy loss with logits 2.9, for natural image datasets using MSE (mean-squared error) [19].

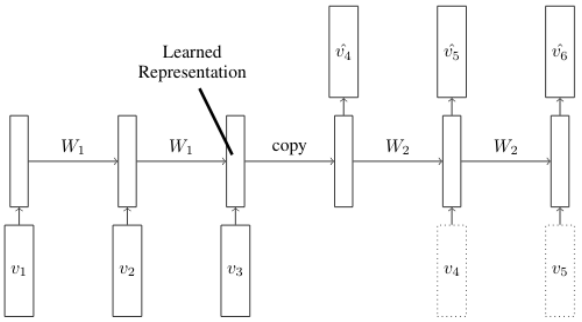

Figure 3: Architecture for future image prediction [14]

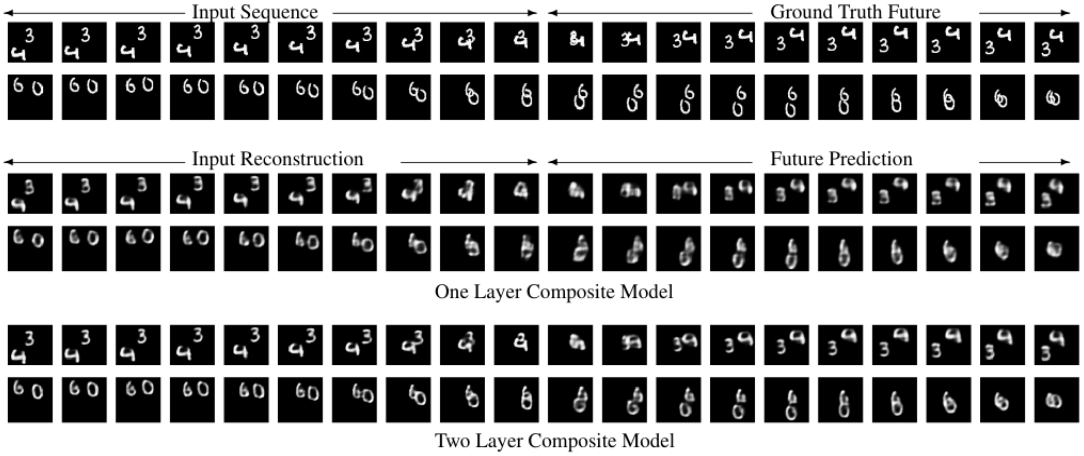

Figure 4: Results for MovingMNIST [14]

3.2 ConvLSTM Autoencoder

The paper „Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting“by Shi et. al. [13] is using a similar architecture as Srivastava et. al. in section 3, but instead of using the standard LSTM, they use a novel ConvLSTM 2.8. The ConvLSTM used in this architecture is with peephole **??**. This architecture outperforms the implementation of Srivastava et. al., because it „captures spatiotemporal correlations better“. This model is, same as LSTM Autoencoder 3.1 end-to-end differentiable and trained using BPTT. It also uses the cross-entropy loss with logits for the synthetic dataset (MovingMNIST) experiment. In this MovingMNIST implementation, every frame consists of three digits. As in the architecture above, the authors here input 10 images and output 10.

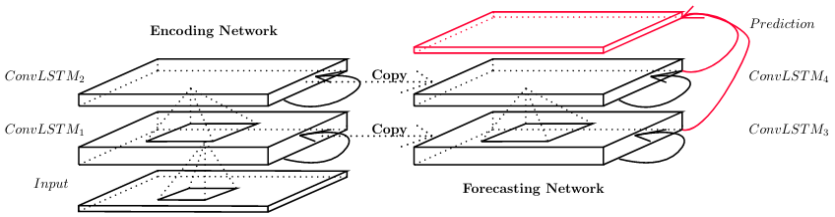

Figure 5: Future image prediction model [13]

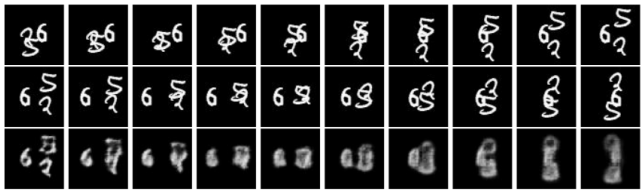

Figure 6: Results for MovingMNIST. **First Row:** 10 input images, **Second Row:** Ground truth next images, **Third Row:** Prediction of 3-layer implementation [13]

3.3 Spatio-temporal Video Autoencoder

The paper „Spatio-temporal Video Autoencoder With Differentiable Memory“by Patraucean et. al. [11] describes a more complex architecture, in which the authors nest a temporal autoencoder inside a spatial autoencoder. The spatial autoencoder is a simple undercomplete autoencoder 2.5, where the decoder uses nearest-neighbor upsampling to get the output image size correct. The temporal autoencoder consists of a ConvLSTM (A ConvLSTM without peephole **??**, which works as the temporal encoder and and an optical flow convolutional module, which works as the temporal decoder. The network idea is to insert the image sequence X, which will create an optical flow map. This optical flow map is then applied on the last given image, to shift every pixel to it's new position. This will create the next image. The idea behind this is given in „Spatial Transformer Networks“by Jadeberg et. al. [6]. The model is end-to-end differentiable and trained using BPTT. The authors also used MovingMNIST as synthetic dataset and use the cross-entropy loss with logits as reconstruction error for it. The MovingMNIST implementation is the same as in LSTM Autoencoder 3.1 (With two digits per frame.). In contrast to the other algorithms, this architecture is only capable of doing one-frame prediction as is. The authors input 19 images and output 1 image.

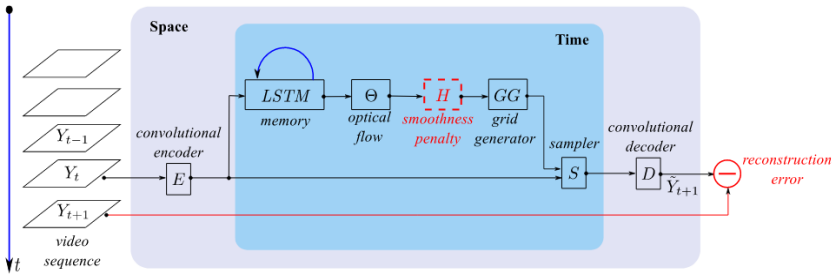

Figure 7: Spatio-temporal Video Autoencoder [11]

Figure 8: Results for MovingMNIST. **conv:** Is a simple convolutional autoencoder, without recurrent module, **fcLSTM:** LSTM Autoencoder 3.1, **cLSTM:** ConvLSTM Autoencoder 3.2, **cLSTM-flow:** Spatio-temporal Video Autoencoder with extra output of flow-map. [11]

3.4 PredNet

The paper „Deep Predictive Coding Networks For Video Prediction And Unsupervised Learning“by Lotter et. al. [8] composes an architecture, which is informally named **PredNet**. It describes a network architecture based on the concept of „predictive coding“[12], [2]. It is the baseline for the experiments performed in section 7. The network consists of an arbitrary amount of layers, the amount of layers (depth of the network) can be treated as a hyperparameter. Every layer consists of an input module ${A_l}^t$, prediction module $\hat{A_l}^t$, representation module ${R_l}^t$ and error module ${E_l}^t$. l is the corresponding layer, t the timestamp.

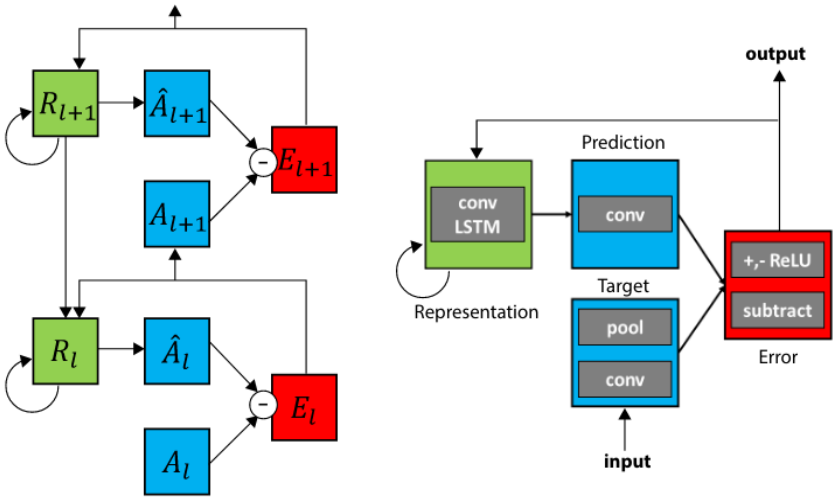

Figure 9: **Left:** PredNet architecture with two layers, **Right:** „Module operations for case of video sequences“[8]

$$A_l{}^t = \begin{cases} x_t & l = 0 \\ MaxPool(ReLU(Conv(E_{l-1}^t))) & l > 0 \end{cases} \tag{12}$$

$$\hat{A}_l^t = ReLU(Conv(R_l^t) \tag{13}$$

$$E_l^t = [ReLU(\hat{A}_l^t - A_l^t); ReLU(A_l^t - \hat{A}_l^t)] \tag{14}$$

$$R_l^t = ConvLSTM(E_l^{t-1}, R_l^{t-1}, Upsample(R_{l+1}^t) \tag{15}$$

3.5 PredRNN

The paper „PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs“by Wang et. al. [17].

4 Implementation

5 Methodology

6 Training

The training section will cover the aspects of different training types.

7 Experiments

The experiments performed on the implemented PredNet with ConvLSTM and PredNet with PredRNN will be described here. Also other theoretical comparisons will be covered in this section.

8 Discussion

9 Conclusion

10 Explanation

Erklärung über das selbstständige Verfassen von „Comparing different state-of-the-art solutions for image prediction using time-series analysis“
Ich versichere hiermit, dass ich die vorstehende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der obigen Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich in jedem einzelnen Fall durch die Angabe der Quelle bzw. der Herkunft, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet und anderen elektronischen Text-und Datensammlungen und dergleichen. Die eingereichte Arbeit ist nicht anderweitig als Prüfungsleistung verwendet worden oder in deutscher oder in einer anderen Sprache als Veröffentlichung erschienen. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschung behandelt werden.

Datum, Ort Unterschrift

References

[1] *Understanding LSTM Networks.* https://colah.github.io/posts/2015-08-Understanding-LSTMs/. – Accessed: 2020-07-13

[2] Friston, Karl: A Theory of Cortical Responses. In: *Philosophical transactions of the Royal Society of London. Series B, Biological sciences* 360 (2005), 05, S. 815–36

[3] Geiger, Andreas ; Lenz, Philip ; Stiller, Christoph ; Urtasun, Raquel: Vision meets Robotics: The KITTI Dataset. In: *International Journal of Robotics Research (IJRR)* (2013)

[4] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning.* MIT Press, 2016. – http://www.deeplearningbook.org

[5] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-term Memory. In: *Neural computation* 9 (1997), 12, S. 1735–80

[6] Jaderberg, Max ; Simonyan, Karen ; Zisserman, Andrew ; Kavukcuoglu, Koray: Spatial Transformer Networks. In: *CoRR* abs/1506.02025 (2015). – URL http://arxiv.org/abs/1506.02025

[7] Lecun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[8] Lotter, William ; Kreiman, Gabriel ; Cox, David D.: Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. In: *CoRR* abs/1605.08104 (2016). – URL http://arxiv.org/abs/1605.08104

[9] Massi, Michela: *Autoencoder schema — Wikipedia, The Free Encyclopedia.* 2019. – URL https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_schema.png. – [Online; accessed 21-Juli-2020]

[10] Paszke, Adam ; Gross, Sam ; Massa, Francisco ; Lerer, Adam ; Bradbury, James ; Chanan, Gregory ; Killeen, Trevor ; Lin, Zeming ; Gimelshein, Natalia ; Antiga, Luca ; Desmaison, Alban ; Kopf, Andreas ; Yang, Edward ; DeVito, Zachary ; Raison, Martin ; Tejani, Alykhan ; Chilamkurthy, Sasank ; Steiner, Benoit ; Fang, Lu ; Bai, Junjie ; Chintala, Soumith: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H. (Hrsg.) ; Larochelle, H. (Hrsg.) ; Beygelzimer, A. (Hrsg.) ; Alché-Buc, F. d(Hrsg.) ; Fox, E. (Hrsg.) ; Garnett, R. (Hrsg.): *Advances in Neural Information Processing Systems 32.* Curran Associates, Inc., 2019, S. 8024–8035. – URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[11] Patraucean, Viorica ; Handa, Ankur ; Cipolla, Roberto: Spatio-temporal video autoencoder with differentiable memory. In: *CoRR* abs/1511.06309 (2015). – URL http://arxiv.org/abs/1511.06309

[12] Rao, Rajesh P. N. ; Ballard, Dana H.: Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. In: *Nature Neuroscience* 2 (1999), Jan, Nr. 1, S. 79–87. – URL https://doi.org/10.1038/4580. – ISSN 1546-1726

[13] SHI, Xingjian ; Chen, Zhourong ; Wang, Hao ; Yeung, Dit-Yan ; Wong, Wai-kin ; WOO, Wang-chun: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In: Cortes, C. (Hrsg.) ; Lawrence, N. D. (Hrsg.) ; Lee, D. D. (Hrsg.) ; Sugiyama, M. (Hrsg.) ; Garnett, R. (Hrsg.): *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, S. 802–810. – URL http://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowca pdf

[14] Srivastava, Nitish ; Mansimov, Elman ; Salakhutdinov, Ruslan: Unsupervised Learning of Video Representations using LSTMs. In: *CoRR* abs/1502.04681 (2015). – URL http://arxiv.org/abs/1502.04681

[15] Sutskever, Ilya ; Vinyals, Oriol ; Le, Quoc V.: Sequence to Sequence Learning with Neural Networks. In: *CoRR* abs/1409.3215 (2014). – URL http://arxiv.org/abs/1409.3215

[16] Wang, Yunbo ; Gao, Zhifeng ; Long, Mingsheng ; Wang, Jianmin ; Yu, Philip S.: PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning. In: *CoRR* abs/1804.06300 (2018). – URL http://arxiv.org/abs/1804.06300

[17] Wang, Yunbo ; Long, Mingsheng ; Wang, Jianmin ; Gao, Zhifeng ; Yu, Philip S.: PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. In: *NIPS*, 2017

[18] Werbos, Paul: Backpropagation through time: what it does and how to do it. In: *Proceedings of the IEEE* 78 (1990), 11, S. 1550 – 1560

[19] Zhao, H. ; Gallo, O. ; Frosio, I. ; Kautz, J.: Loss Functions for Image Restoration With Neural Networks. In: *IEEE Transactions on Computational Imaging* 3 (2017), Nr. 1, S. 47–57