TaylorSwiftNet: Taylor Driven Temporal Modeling for Swift Future Frame Prediction Supplemental Material

Saber Pourheydari*[1]
m.saberpourheydari@gmail.com
Emad Bahrami*[1]
bahrami@iai.uni-bonn.de
Mohsen Fayyaz*[1, 2]
mohsenfayyaz@microsoft.com
Gianpiero Francesca[3]
gianpiero.francesca@toyota-europe.com
Mehdi Noroozi[4]
m.noroozi@samsung.com
Juergen Gall[1]
gall@iai.uni-bonn.de

[1] Computer Vision Group
University of Bonn
Bonn, Germany
[2] Microsoft
Berlin, Germany
[3] Toyota Motor Europe
Brussels, Belgium
[4] Samsung AI
Cambridge, UK

* indicates equal contribution

In the following sections, we present additional ablation studies, more details of our method, and a comparison to analytical derived derivatives.

1 Additional Ablation Studies

1.1 Recurrent DCBs

As mentioned in the paper, the DC blocks can also be implemented as recurrent DCBs (RDCB). In contrast to DC blocks, RDC blocks share their weights. As shown in Table 1, DC blocks perform slightly better. We also found that DC blocks are more stable during training.

Method	Moving MNIST	Traffic BJ	SST	Human 3.6M
DCB	0.965	0.992	0.978	0.910
RDCB	0.964	0.971	0.977	0.906

Table 1: SSIM for models with DCBs and RDCBs.

1.2 Runtime Comparison

We report the runtime of our model for Human 3.6M in Tab. 2. We used an NVIDIA Titan RTX GPU.

	run-time (ms)	Parameters (M)	GMACs	SSIM
PhyDNet [1]	30	11	76	0.901
ours	**21**	11	**61**	**0.910**

Table 2: Computation cost and runtime.

	1^{st}	2^{nd}	3^{rd}	4^{th}
$d^i \sin/dt^i$	0.05077	−0.99871	−0.05077	0.99871
ours	0.05083	−0.99993	−0.05017	0.99012
$d^i \cos/dt^i$	−0.9987	−0.0507	0.9987	0.0507
ours	−0.9991	−0.0506	0.9900	0.0504
$d^i \exp/dt^i$	4.5722	4.5722	4.5722	4.5722
ours	4.5723	4.5726	4.5721	4.5730

Table 3: Comparing the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} order derivatives of three functions with the estimated derivatives.

2 Comparison to Analytical Derivatives

For video data, the function $\mathcal{F}_{\mathcal{H}_t}$ and thus the ground-truth terms of the Taylor series are unknown. In order to analyze how accurately our network can learn the terms of the Taylor series, we use three functions where we can analytically derive the derivatives. The results in Table 3 demonstrate that the DC blocks are able to learn derivatives.

3 Implementation Details

3.1 Encoder and Decoder

We provide the details of the encoder and decoder for each dataset in Tables 4-7. The models share common blocks. We define each convolutional layer as: [input channel, output channel], [kernel height, kernel width, kernel depth], [stride over height, stride over width, stride over depth]. We also define each residual block as:

$$ResBlock = \begin{bmatrix} [C,C],[3,3,3],[1,1,1] \\ [C,C],[3,3,3],[1,1,1] \end{bmatrix}$$

Furthermore, Figures 3 and 4 visualize the baselines 'Point Estimate (Expand)' and 'Point Estimate (Flatten)' from Table 2 of the paper.

3.2 Training

We use Adam [2] with a learning rate of 0.0001 to optimize the model through 4K epochs. For the SST dataset, we train our model in 1K epochs. To control the learning rate, we use a scheduler to reduce the learning rate by a factor of 0.5 in case of a plateau over the SSIM metric on the training set. Following the previous state-of-the-art methods [1], we use MSE as the loss function.

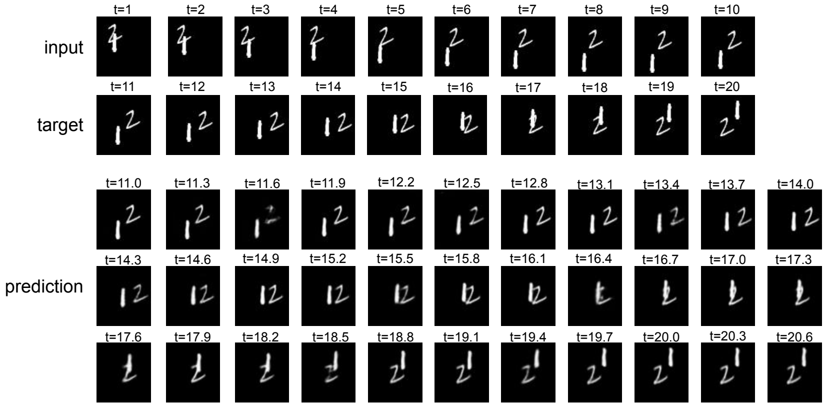

Figure 1: Predicting future frames at higher temporal resolution. Input are the 10 observed frames and target are the future ground-truth frames. τ is increased by 0.3 instead of 1.

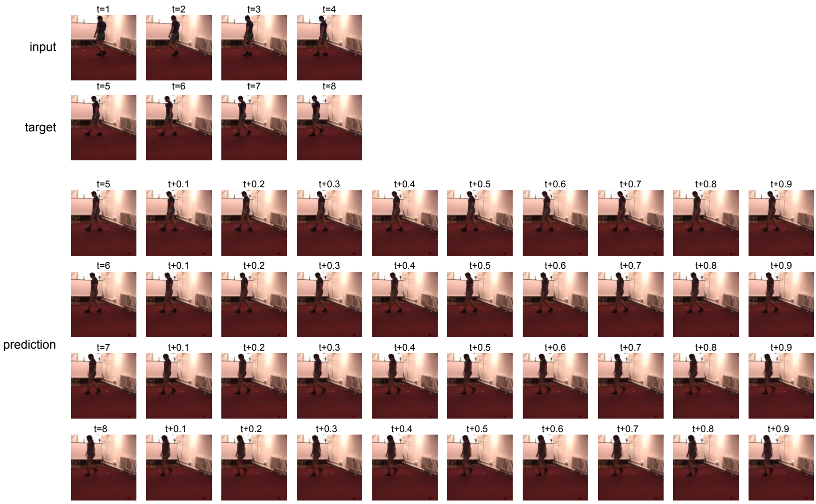

Figure 2: Predicting future frames at higher temporal resolution. Input are the 4 observed frames and target are the future ground-truth frames. τ is increased by 0.1 instead of 1.

4 Forecasting at Different Temporal Resolutions

Since our model forecasts frames using a continuous representation, we do not need to stick to the framerate of the observation. In Fig. 1, we show qualitative results on Moving MNIST for the future temporal steps $t+\tau \in \{11, 11.3, 11.6, ..., 20.6\}$, i.e., we increase the framerate by $1/0.3$. Note that we do not re-train our model for this experiment. As it can be seen, our TaylorSwiftNet smoothly predicts intermediate frames. The sharp digits and their accurate location clearly demonstrate the continuous temporal modeling capability of our model. We provide more qualitative results of the continuous temporal modeling capability of our method for Human 3.6M in Fig. 2.

References

[1] Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

stage	layer	output size
raw	-	$1 \times 10 \times 64 \times 64$
Encoder	[1, 8], [1, 3, 3], [1, 1, 1]	$8 \times 10 \times 64 \times 64$
	[8, 16], [1, 3, 3], [1, 1, 1]	$16 \times 10 \times 64 \times 64$
	[16, 32], [1, 3, 3], [1, 2, 2]	$32 \times 10 \times 32 \times 32$
	[32, 32], [1, 3, 3], [1, 1, 1]	
	[32, 64], [1, 3, 3], [1, 2, 2]	$64 \times 10 \times 16 \times 16$
	[64, 128], [1, 3, 3], [1, 1, 1]	$128 \times 10 \times 16 \times 16$
	[128, 128], [3, 3, 3], [1, 1, 1]	
	$\text{ResBlock}_2 \times 2$	
	$\text{ResBlock}_3 \times 2$	
	$\text{ResBlock}_4 \times 2$	
	$\text{ResBlock}_5 \times 2$	
Decoder	[128, 64], [1, 3, 3], [1, 1, 1]	$64 \times 10 \times 16 \times 16$
	[64, 32], [1, 3, 3], [1, 2, 2]	$32 \times 10 \times 32 \times 32$
	[32, 32], [1, 3, 3], [1, 1, 1]	
	[32, 16], [1, 3, 3], [1, 2, 2]	$16 \times 10 \times 64 \times 64$
	[16, 8], [1, 3, 3], [1, 1, 1]	$8 \times 10 \times 64 \times 64$
	[8, 1], [1, 3, 3], [1, 1, 1]	$1 \times 10 \times 64 \times 64$

Table 4: Model architecture with a modified 3DResNet encoder for Moving MNIST.

stage	layer	output size
raw	-	$2 \times 4 \times 64 \times 64$
Encoder	[2, 32], [1, 3, 3], [1, 1, 1]	$32 \times 4 \times 32 \times 32$
	[32, 64], [1, 3, 3], [1, 2, 2]	$64 \times 4 \times 16 \times 16$
	[64, 128], [1, 3, 3], [1, 1, 1]	$128 \times 4 \times 16 \times 16$
	[128, 128], [3, 3, 3], [1, 1, 1]	
	$\text{ResBlock}_2 \times 2$	
	$\text{ResBlock}_3 \times 2$	
	$\text{ResBlock}_4 \times 2$	
	$\text{ResBlock}_5 \times 2$	
Decoder	[128, 64], [1, 3, 3], [1, 1, 1]	$64 \times 4 \times 16 \times 16$
	[64, 32], [1, 3, 3], [1, 2, 2]	$32 \times 4 \times 32 \times 32$
	[32, 2], [1, 3, 3], [1, 1, 1]	$2 \times 4 \times 32 \times 32$

Table 5: Model architecture with a modified 3DResNet encoder for Traffic BJ.

stage	layer	output size
raw	-	$1 \times 4 \times 64 \times 64$
Encoder	[1, 16], [1, 3, 3], [1, 1, 1]	$16 \times 4 \times 64 \times 64$
	[16, 32], [1, 3, 3], [1, 2, 2]	$32 \times 4 \times 32 \times 32$
	[32, 32], [1, 3, 3], [1, 1, 1]	
	[32, 64], [1, 3, 3], [1, 2, 2]	$64 \times 4 \times 16 \times 16$
	[64, 128], [1, 3, 3], [1, 1, 1]	$128 \times 4 \times 16 \times 16$
	[128, 128], [3, 3, 3], [1, 1, 1]	
	$\text{ResBlock}_2 \times 2$	
	$\text{ResBlock}_3 \times 2$	
	$\text{ResBlock}_4 \times 2$	
	$\text{ResBlock}_5 \times 2$	
Decoder	[128, 64], [1, 3, 3], [1, 1, 1]	$64 \times 4 \times 16 \times 16$
	[64, 32], [1, 3, 3], [1, 2, 2]	$32 \times 4 \times 32 \times 32$
	[32, 32], [1, 3, 3], [1, 1, 1]	
	[32, 16], [1, 3, 3], [1, 2, 2]	$16 \times 4 \times 64 \times 64$
	[16, 1], [1, 3, 3], [1, 1, 1]	$1 \times 4 \times 64 \times 64$

Table 6: Model architecture with a modified 3DResNet encoder for SST.

stage	layer	output size
raw	-	$3 \times 4 \times 64 \times 64$
Encoder	[3, 16], [1, 3, 3], [1, 1, 1]	$16 \times 4 \times 64 \times 64$
	[16, 32], [1, 3, 3], [1, 1, 1]	$32 \times 4 \times 64 \times 64$
	[32, 64], [1, 3, 3], [1, 2, 2]	$64 \times 4 \times 32 \times 32$
	[64, 64], [1, 3, 3], [1, 1, 1]	
	[64, 128], [1, 3, 3], [1, 2, 2]	$128 \times 4 \times 16 \times 16$
	[128, 256], [1, 3, 3], [1, 1, 1]	$256 \times 4 \times 16 \times 16$
	[256, 256], [3, 3, 3], [1, 1, 1]	
	$\text{ResBlock}_2 \times 2$	
	$\text{ResBlock}_3 \times 2$	
	$\text{ResBlock}_4 \times 2$	
	$\text{ResBlock}_5 \times 2$	
Decoder	[256, 128], [1, 3, 3], [1, 1, 1]	$128 \times 4 \times 16 \times 16$
	[128, 64], [1, 3, 3], [1, 2, 2]	$64 \times 4 \times 32 \times 32$
	[64, 64], [1, 3, 3], [1, 1, 1]	
	[64, 32], [1, 3, 3], [1, 2, 2]	$32 \times 4 \times 64 \times 64$
	[32, 16], [1, 3, 3], [1, 1, 1]	$16 \times 4 \times 64 \times 64$
	[16, 3], [1, 3, 3], [1, 1, 1]	$3 \times 4 \times 64 \times 64$

Table 7: Model architecture with a modified 3DResNet encoder for Human 3.6M.

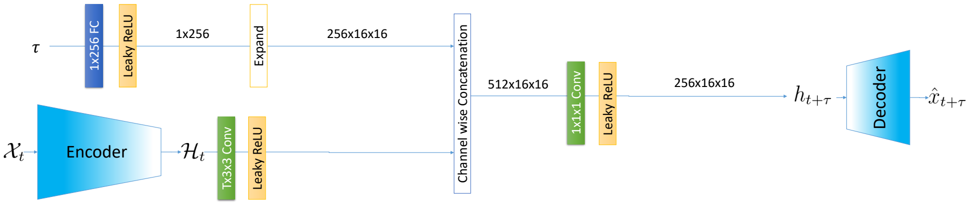

Figure 3: Architecture of the baseline ‘Expand’.

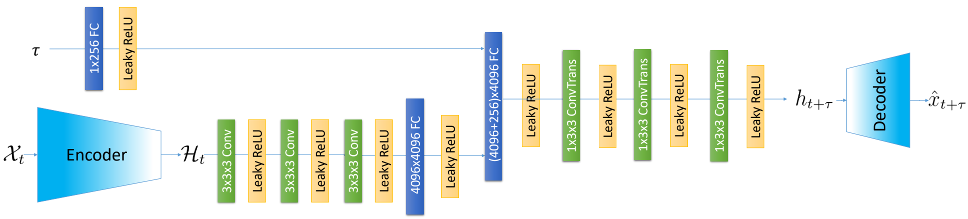

Figure 4: Architecture of the baseline ‘Flatten’.