Paper ID 10989: Convolutional Tensor-Train LSTM for Spatio-Temporal Learning. We thank all reviewers for their valuable feedback. Reviewers found that this work is solid, the main idea is interesting and clearly presented. Below is the point-by-point responses to the reviewers.

R1-Q1. Comparison to Yang et. al. [40]: R1 appreciates the difference between ours and [40], especially the unique link between tensor train and HO-CRNN. There are two major differences between our work and [40]: **(1)** while [40] relies on the classic tensor-train decomposition (TTclassic) based on [18], our *convolutional tensor-train decomposition* (CTTD, Eq.(6)) factorizes the tensor with convolutions instead of inner products; **(2)** [40] compresses only input-hidden weights within one time-step in LSTM. Our CTTD compresses spatio-temporal interactions over time. This is a new design of tensor-train decomposition for spatio-temporal data. We add a new comparison between ours and [40] in the table below. For fair comparison, we applied the core idea of [40] to ConvLSTM (baseline) as follows: $[\mathcal{I}(t); \mathcal{F}(t); \tilde{\mathcal{C}}(t); \mathcal{O}(t)] = \sigma(\text{TTclassic}(\{\mathcal{W}_i\}) * \mathcal{X}(t) + \mathcal{K} * \mathcal{H}(t-1))$. From the comparison, we observe that our model outperforms [40] on MNIST and KTH (except LPIPS on KTH) with similar number of parameters.

Dataset	TTclassic [40]		ConvLSTM		Conv-TT-LSTM	
(predicted frames)	SSIM	LPIPS	SSIM	LPIPS	SSIM	LPIPS
MNIST (10)	0.890	59.09	0.882	67.13	**0.915**	**40.54**
MNIST (30)	0.817	125.2	0.806	140.1	**0.840**	**90.38**
KTH (20)	0.900	**120.1**	0.903	137.1	**0.907**	133.4
KTH (40)	0.874	**163.5**	0.876	201.3	**0.882**	191.2
Params.	2.20M		3.97M		2.69M	

Furthermore, we believe our work is orthogonal to [40] — [40] can be used to further compress our model by decomposing each factor $\mathcal{G}(i)$ (in Eq.(6)) with TTclassic.

R1-Q2. Efficiency evaluation. We agree that there is a trade-off between FLOPs and latency. Therefore, we introduce two algorithms in Appendix A. While Alg. 2 significantly decreases the complexity in FLOPs, it also lowers the degree of parallelism. However, Alg. 1 shows how our model can be parallelized. Ideally, these two algorithms can be combined using CUDA multi-streams (execute multiple kernels in parallel): use Alg. 1 for the beginning iterations of i and Alg. 2 for the later ones (the beginning ones have smaller kernel sizes). In our current implementation, we use Alg. 2 to reduce the GPU memory requirement. The run-time of current implementation is 27.3 mins (37.83 GFLOPs) for Conv-TT-LSTM and 26.2 mins (55.83 GFLOPs) for ConvLSTM (per epoch on KTH). We will add the run-time and this discussion in the final version.

R2-Q1. Motivation of higher-order RNN. While vanilla RNN can be a universal approximator or Turing complete theoretically, there is no guarantee that the model will find the optimal solution. R2 believes that in practice, the vanishing/exploding gradients prevent RNNs from learning higher-order interaction. Our proposed model addresses the vanishing/exploding gradient problem by incorporating long-term dependencies with higher-order RNNs [8].

R2-Q2. Use case of higher-order RNNs. The applications in our experiments requires future prediction. To predict the most possible future, understanding the long-term dynamics is essential. The early activity recognition task requires a model to understand the dynamics of the video, so the model can predict an activity at the early stage. For the video prediction tasks, the model learns to predict 10 frames during training. In testing, the model further predicts 30-40 frames. Optical flow features (based on short-term dynamics) are not sufficient to make such predictions.

R2-Q3. Add ablation studies. The choice of a 12 layer baseline. The paper already includes the detailed ablation studies in Table 3 and Appendix B.6: single v.s. higher-order model, the necessity of convolutions in CTTD, 4 v.s. 12 layers, the benefit of scheduling tricks, etc. The results show that Conv-TT-LSTM consistently outperforms ConvLSTM baseline under all scenarios. Necessity of a deep model for video prediction has been already discussed in [43].

R2-Q4. Many scheduling tricks. These scheduling tricks are used for both our model and the baseline ConvLSTM, so our performance improvement is NOT due to the scheduling tricks (also shown by the aforementioned ablation studies). These scheduling tricks are commonly used for prediction and early activity recognition tasks [19,23-24].

R2-Q5. Pre-processing conflicts with the paper's motivation (higher-order interaction). The reviewer misunderstood the difference between *order in RNNs* and *order of a tensor decomposition*. The former refers to the number of previous time-steps used at each update, while the latter denotes the number of factors in tensor decomposition. In fact, we propose the pre-processing module mainly to decouple these two concepts (explained in Lines 136-146), and the order of tensor decomposition only controls the complexity of the mapping function Φ.

R2-Q6. Include videos in the appendix. '10989_result_videos' in the supplementary file already includes the videos.

R3-Q1. Analysis of "implicit regularizer" leading to "generalized models". The relationship between low-rank regularizer and generalized models is discussed in Line 40-42, and analyzed theoretically in [11, 12]. Intuitively, consider a linear model $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{q \times p}$ is factorized as $\mathbf{A} = \mathbf{U}\mathbf{V}$ with $\mathbf{U} \in \mathbb{R}^{q \times r}$, $\mathbf{V} \in \mathbb{R}^{r \times p}$, $r < \min(p, q)$. The factorization implicitly creates a bottleneck $\mathbf{h}$ since the model can be evaluated as $\mathbf{h} = \mathbf{V}\mathbf{x}$ and $\mathbf{y} = \mathbf{U}\mathbf{h}$. Since $\mathbf{h}$ has lower dimension than $\mathbf{x}$, the bottleneck filters out redundant information, leading to more generalized models.

R3-Q2. Difference between ConvLSTM [4] and ConvLSTM (baseline). ConvLSTM [4] is the original model proposed by [4]. We re-implemented ConvLSTM of [43] as our baseline. It has exactly the same model size, skip connections and uses the same training strategies as our Conv-TT-LSTM. We will add the clarification in the paper.