Appendix - Motion Graph Unleashed: A Novel Approach to Video Prediction

Anonymous Author(s)
Affiliation
Address
email

1 Implementation Details

We implement the video prediction system using PyTorch [1] and conduct end-to-end training on a single NVIDIA A100 GPU. We use AdamW optimizer [2] during the training. The initial learning rate is set to $1e^{-3}$ and decayed to $1e^{-5}$ following a cosine decay scheduler [3]. There are only a few hyperparameters to adjust for the system when training on different datasets. The adjustments are mainly based on the resolution of the frame. Those hyper-parameters include i) image feature length (Image feat.), which is the parameter for the image encoder; ii) Tendency feature length (Tendency feat.), which is the length of the tendency feature in node initialization; iii) location feature length (Location feat.), which is fixed to 4 for all datasets, indicating the length of the location feature in node initialization; iv) the number of graph views, indicating view number in motion graph feature learning; v) k, indicates the number of the dynamic vectors embedded in each node, the number of the temporal edges per node and the output dynamic vectors per pixel; vi) training epoch is the training related parameters; vii) the reconstruction loss, which follows the popular setting of the SOTA methods on each dataset. In Table 1, we demonstrate the hyper-parameter setting for each dataset.

Please note that we did not especially tune the parameters for each dataset. When adjusting the parameters, we consider more about the training efficiency instead of the performance. Therefore, our setting is likely *not* the optimal choice. For example, in DMVFN [4], the training on Cityscapes and KITTI are both 300 epochs, we observe that our system can achieve comparable performance with only 100 and 200 epochs respectively, we thus stay with this configuration.

Dataset	Image feat.	Tendency feat.	Location feat.	Number of Graph views	k	Epoch	Loss
UCF Sports	16	16	4	4	10	300	MSE
Cityscapes	16	32	4	4	10	100	L1 + Lpips
KITTI	16	32	4	4	8	200	L1 + Lpips

Table 1: Hyper-parameter setting for each dataset.

2 Network Architecture

The proposed video prediction system includes three major components, the image encoder, the motion graph interaction module, and the motion upsampler. Here we demonstrate the detailed architecture of each component for reproduction needs.

Image Encoder: Figure 1 shows the inner structure of the image encoder in the proposed system. C_{img} is related to the image feature length in Table 1. Each convolution layer will come with a Leaky ReLU layer [5] as the activation layer.

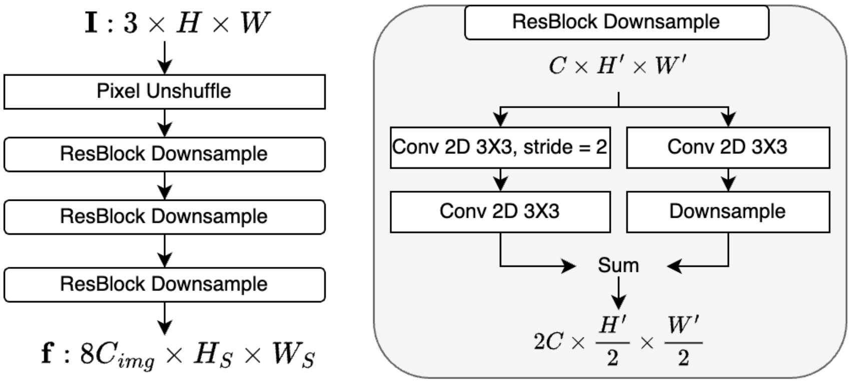

Figure 1: Architecture of the image encoder

Motion Graph Interaction Module In Figure 2 we demonstrate the inner structure of the spatial and temporal message passing in the motion graph interaction module. C_{node} equals the sum of the tendency feature length and the location feature length in Table 1.

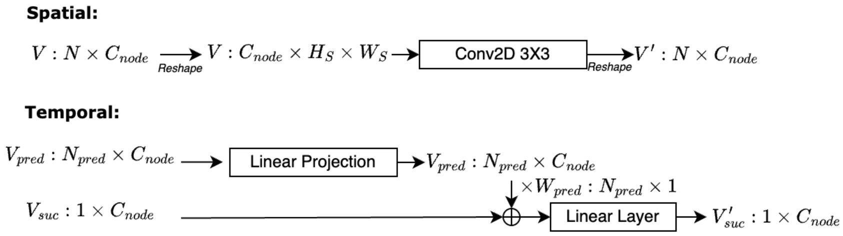
Spatial:
$V : N \times C_{node}$
Reshape
$V : C_{node} \times H_S \times W_S$
Conv2D 3X3
Reshape
$V' : N \times C_{node}$
Temporal:
$V_{pred} : N_{pred} \times C_{node}$
Linear Projection
$V_{pred} : N_{pred} \times C_{node}$
$\times W_{pred} : N_{pred} \times 1$
$V_{suc} : 1 \times C_{node}$
Linear Layer
$V'_{suc} : 1 \times C_{node}$

Figure 2: Inner structure of spatial and temporal block in motion graph interaction module

Motion Upsampler Figure 3 illustrates the inner structure of the motion upsampler as well as the motion decoder. The implementation of the decoder is a single 2D convolution layer with a kernel size of 1.

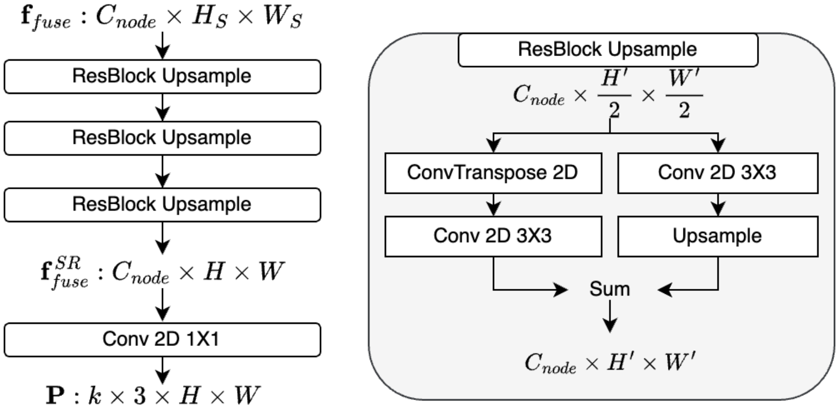
$\mathbf{f}_{fuse} : C_{node} \times H_S \times W_S$
ResBlock Upsample
ResBlock Upsample
ResBlock Upsample
$\mathbf{f}_{fuse}^{SR} : C_{node} \times H \times W$
Conv 2D 1X1
$\mathbf{P} : k \times 3 \times H \times W$
ResBlock Upsample
$C_{node} \times \frac{H'}{2} \times \frac{W'}{2}$
ConvTranspose 2D
Conv 2D 3X3
Conv 2D 3X3
Upsample
Sum
$C_{node} \times H' \times W'$

Figure 3: Inner structure of the motion upsampler and the motion decoder.

3 Additional Quantitative Evaluation

On UCF Sports MMVP split [6], the validation dataset has been divided into three categories: the easy (SSIM $\leq$ 0.9), intermediate (0.6 $\leq$ SSIM $<$ 0.9), and hard subsets (SSIM $<$ 0.6), which take up

66%, 26%, and 8% of the full set respectively. Due to the page limitation, we put the comparison and evaluation on each categories here in Table 2.

Table 2: Performance comparison on the UCF Sports MMVP split.

Method	Full set			Easy (SSIM $\geq$ 0.9)			Intermediate (0.6 $\leq$ SSIM < 0.9)			Hard (SSIM < 0.6)			Model Size↓
	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS↓	SSIM ↑	PSNR ↑	LPIPS ↓	
STIP [7]	0.8817	28.17	0.1626	0.9491	30.65	0.1066	0.8351	23.97	0.2271	0.4673	15.97	0.4450	18.05M
SimVP [8]	0.9189	29.97	0.1326	0.9664	32.87	0.0584	0.8845	25.79	0.1951	0.6267	18.99	0.5600	3.47M
MMVP [6]	0.9300	30.35	0.1062	0.9674	33.05	0.0580	0.8970	26.29	0.1569	0.7203	**20.84**	0.3510	2.79M
Ours	**0.9314**	**30.49**	**0.0823**	**0.9685**	**33.23**	**0.0444**	**0.8978**	**26.36**	**0.1348**	**0.7264**	20.83	**0.2320**	**0.60M**

4 Extensive Ablation Study

In this section, we add two ablation studies to help the audience better interpret the design of the motion graph.

Number of the predicted dynamic vectors per pixel: In the proposed system, we set the number of the predicted dynamic vectors per pixel to k, which is identical to the number of the dynamic vectors embedded by each node and the temporal edge of each node. This design ensures the flexibility of the predicted motion to have multiple modes compared to the optical-flow-based method which only allows each pixel to have a single future motion. The comparison between the first two rows of Table 3 showcase that allowing multiple predicted dynamic vectors can largely improve the performance. Meanwhile, if we control the number of the predicted dynamic vectors, as demonstrate by the comparison between the first and third row of the Table 3, we see that when the motion graph embeds more past motion modes, the performance will also has significant improvements.

k	Predicted Vectors #	Full set			Hard (SSIM $<$ 0.6)			Memory ↓
		SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	
1	1	0.8742	26.34	0.1527	0.5394	17.34	0.5032	0.98G
1	10	0.9199	29.87	0.1042	0.6408	19.44	0.3706	1.97G
10	1	0.9212	30.00	0.0877	0.6546	19.63	0.3261	1.38G

Table 3: Ablation study on the number of the predicted vectors. The experiments are conducted on UCF Sports MMVP splits. The listed results are from the models trained for 100 epochs (models were all trained for 300 epochs in the main manuscript).

Motion Graph Interaction Module The design of the motion graph interaction module are following the intuition that both spatial connection and temporal connection should benefit the graph learning. Here we also show the experimental results in Table 4 that both spatial and backward edges are beneficial to the final performance.

Spatial	Backward	PSNR ↑	MS-SSIM ×100 ↑	LPIPS×100 ↓
×	×	21.55	87.06	9.85
✓	×	21.64	87.25	9.83
✓	✓	21.71	87.70	9.50

Table 4: Ablation study on graph interaction module. The experiments are conducted on KITTI and metrics show evaluation on the $t + 1$ results.

5 Failure case demonstration

The video prediction is always a challenging problem. Especially for those video sequences with abrupt motion which can be hardly indicated by the previous video frames. The proposed method formulates the video prediction as a motion prediction problem and outperforms most of the existing methods by using motion graph to better capture the motion hints from the input frames. However, when evaluating the qualitative results, we still find some failure cases that require additional research efforts to solve. In Figure 4, we showed typical failure cases in UCF Sports dataset. We notice that most of the failures cases are in the action of kicking and diving, which usually include fast, unpredictable motion that requires stronger video understanding capability of the model.

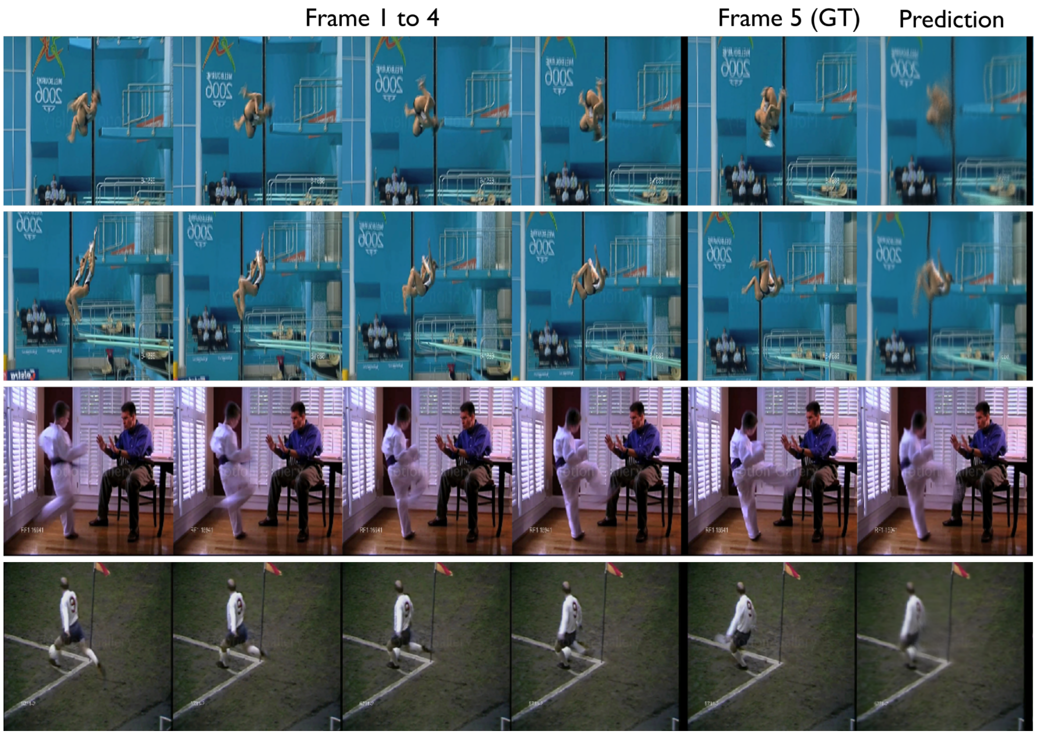

Figure 4: Failure cases in UCF Sports Dataset

6 Node Feature Visualization

To better understand the initialization of the node embedding, here we visualize the tendency feature and the location feature. We first extract the tendency feature and location feature of each node in the motion graph and apply a K-means clustering to the extracted features. For the tendency feature, we set the cluster number to 2; and for the location feature, we set the cluster number to 4 for better visualization.

From Figure 5, we see that using the learned tendency feature, the system should be able to distinguish the dynamic areas from the static areas. If we further enlarge the cluster number, we can see more clearly that the tendency features embed the different motion patterns of each feature patch in the frame. For the location feature, in the paper, we have shown that removing the location feature from

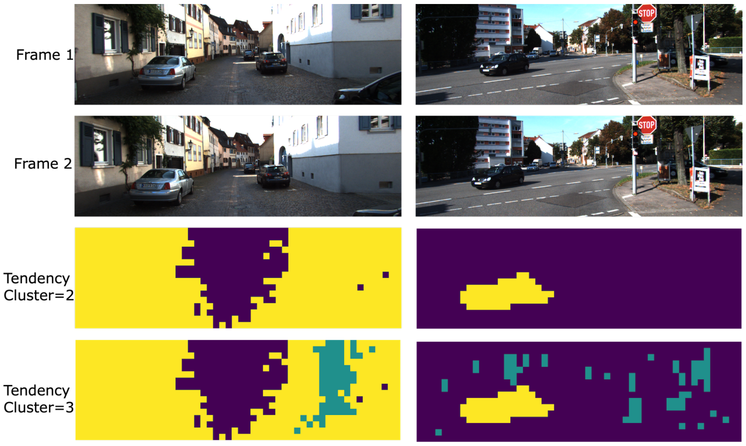

Figure 5: Tendency feature visualization using KITTI dataset

the node initialization will result in a performance drop. From Figure 6 we observe that the location feature may contain information that is related to the camera projection mode. For cityscapes and the KITTI, which use wide-range cameras, the clustering pattern of the location feature is very different from the UCF Sports whose projection mode is possible to be orthogonal projection.

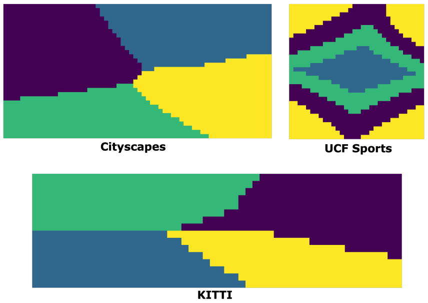

Figure 6: Locaiton feature visualization on three datasets.

References

[1] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[3] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[4] Xiaotao Hu, Zhewei Huang, Ailin Huang, Jun Xu, and Shuchang Zhou. A dynamic multi-scale voxel flow network for video prediction. *arXiv preprint arXiv:2303.09875*, 2023.

[5] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

[6] Yiqi Zhong, Luming Liang, Ilya Zharkov, and Ulrich Neumann. Mmvp: Motion-matrix-based video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4273–4283, 2023.

[7] Zheng Chang, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. Stip: A spatiotemporal information-preserving and perception-augmented model for high-resolution video prediction. *arXiv preprint arXiv:2206.04381*, 2022.

[8] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3170–3180, 2022.