Visual Forecasting by Imitating Dynamics in Natural Sequences
Supplementary Material

Kuo-Hao Zeng[†‡] William B. Shen[†] De-An Huang[†] Min Sun[‡] Juan Carlos Niebles[†]
[†]Stanford University [‡]National Tsing Hua University
{khzeng, bshen88, dahuang, jniebles}@cs.stanford.edu sunmin@ee.nthu.edu.tw

1. Model Architecture and Training Detail

We give the details for both model architecture and training setting in this section. For simplicity, let Ck denotes a Convolutional layer with k filters, B denotes batch normalization layer, R denotes a RELU layer, D denotes a Dropout layer with dropout rate of 50%, $3C$ denotes a $3D$ convolutional layer, dC denotes a deconvolutional layer, F denotes a fully connected layer and S denotes a Sigmoid function. For example, $3C64BR$ means a $3D$ convolutional layer with 64 filters coming with a batch normalization layer, and a RELU layer. We implement all of our models in PyTorch[2].

Future Frame Generation. The overall model architecture for experiment of future frames generation can be found in Figure 1. In this experiment, we need to employ ϕ^{-1}, because the output of the policy network must be pixel-wise. Also, since we want to generate next frame based on the previous motion cue, we utilize a $3D$ convolutional neural network as the network ϕ. The input for this network ϕ is 3 consecutive frames. Further, we utilize a $2D$ Deconvolutional neural network as network ϕ^{-1}. As time progresses, we continuously replace the input of network ϕ with previous last two input frames and the last generated frame so that we make sure that the next generated frame is conditioned on previous motion and visual state. Thus, the policy network π for this experiment consists of network ϕ and network ϕ^{-1}. For adversarial training and imitation learning, we implement the discriminator by the same architecture as the network ϕ. The detail of model architecture is listed in as follows.

Network ϕ:

3C64R - 3C128BR - 3C256BR - 3C512BR - 3C512R

Network ϕ^{-1}:

2dC512BR - 2dC256BR - 2dC128BR- 2dC64BR - 2dC1 - S

Policy Network π:

ϕ - ϕ^{-1}

Discriminator:

3C64R - 3C128BR - 3C256BR - 3C512BR - 3C1 - S

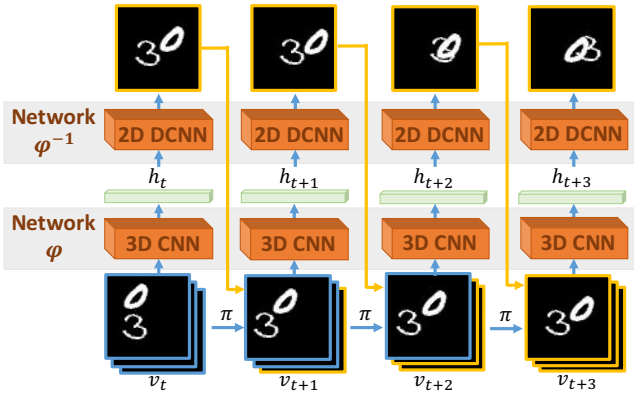

Figure 1. Model for future frames generation. We implement network ϕ using a $3D$ convolutional neural network and network ϕ^{-1} using a $2D$ deconvolutional neural network. Thus, the policy network π consists of network ϕ and network ϕ^{-1}.

where $3C64BR$ means a $3D$ convolutional layer with 64 filters coming with a batch normalization layer, and a RELU layer.

During training, we use 10^{-2} and 10^{-3} as initial learning rates for policy network and discriminator, respectively. We then decay both learning rates by multiplying those with 0.1 for every 25 epochs. We have tried the batch size as 128, 64, and 32, but we do not find major different between different setting. We set batch size as 32 in final. Inspired by [6], we use leaky-RELU with slope of 0.2 as our nonlinearity function. In addition to imitation learning, we also apply $L1$ loss function and typical generative adversarial loss function to jointly optimize model.

Action Anticipation. The overall model architecture for action anticipation experiments can be found in Figure 2. In this experiment, we utilize ResNet-101 [5] as network $\phi\cdot$. Further, we use autoencoder as the policy network $\hat{\pi}$. The most important difference here is that the model does not need a network ϕ^{-1} since we only want to learn the dynamics in the representation space. The detail of this change can be found in Section.3.3 in the main paper. Following the

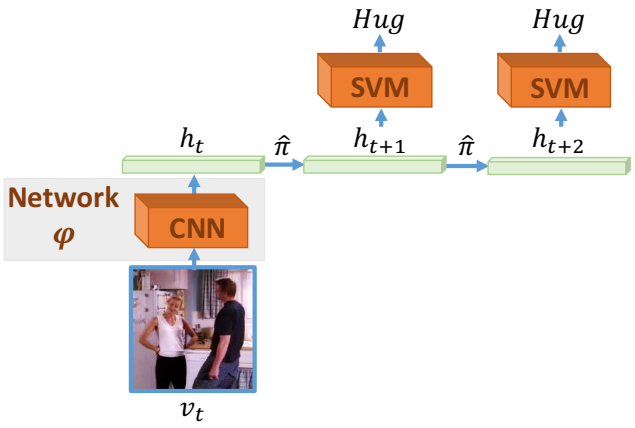

Figure 2. Model for action anticipation. We implement network ϕ using ResNet-101 and policy network π using autoencoder. We further implement the linear classifier on the anticipated representation using linear-SVM with hyperpameter C as 1.

policy network $\hat{\pi}$, the model propagates the current visual representation into future. In unsupervised representation learning stage, we train our model on a large scale video dataset [4]. In classification learning stage, we further learn a linear-SVM[1] [1] on the representation which is anticipated by the representation one second before action duration[2] to classify which action will happen. At linear-SVM training stage, all the parameters of ResNet-101 and policy network $\hat{\pi}$ are fixed. The detail of model architecture is as follows.

Network ϕ:

ResNet-101

Network ϕ^{-1}:

None

Policy Network π:

F1024BDR - F512BDR - F256BDR - F512BDR - F1024BDR - F2048

Discriminator:

F1024BDR - F512BDR - F1 - S

where $F1024BDR$ means a fully connected layer with 1024 filters coming with a batch normalization layer, a dropout layer, and a RELU layer.

During training, we use 10^{-4} as initial learning rate for both policy network and discriminator. We then decay both learning rates by multiplying those with 0.1 for every 20 epochs. We set batch size as 16 and use the same nonlinearity function and dropout rate as those used in task of future frame generation. We also utilize $L1$ loss function and generative adversarial loss function to jointly train models. Different from the future frames generation experiment, we fix the weights of ResNet-101 in first 5 epochs. After 5 epoch, we train the entire model end-to-end.

[1]Although we try $0.01, 0.1, 1, 10, 100$ as hyperparameter C for training linear-SVM, we find that the performance is not sensitive to it. For fair comparison, we set hyperparameter C as 1 for our method and all baseline models.

[2]Following [11], we also train our linear-SVM on the representation extracted from the frame within the action duration which is called Adaptation in [11].

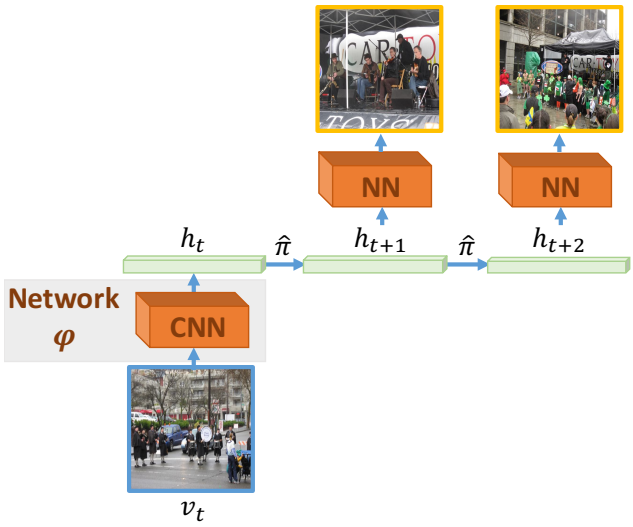

Figure 3. Model for storyline forecasting. We implement network ϕ with ResNet-101 and policy network π with autoencoder. During testing, we conduct retrieval task by applying nearest-neighbor algorithm (NN) on the forecasted representation. The distance function used by a NN module is absolute-value norm.

Storyline Forecasting. The overall model architecture for experiment of storyline forecasting can be referred in Figure 6. Expect for NN module (nearest-neighbor algorithm), we almost use same architecture as it used in experiment of action anticipation. To conduct retrieval task, we replace the linear classifier used in experiment of action anticipation by a NN module with distance function of absolute-value norm. The training detail is same as those used in experiment of action anticipation.

2. Policy Gradient with Roll out

We train our policy network by using policy gradient. Inspired by [7], we employ the Monte Carlo (MC) search to compute intermediate time step costs for a sequence. It can be founded in Figure 3 in the main paper. The Monte Carlo (MC) search is also known as roll out [12] in reinforcement learning. The idea is that when the model sample a action conditioned on current state, we let the model keeps sampling next action based on current policy network until the end of trajectory so that we can compute a excepted cost on entire trajectory for this intermediate action. For each intermediate action, it can roll out multiple times. Let's define R as number of roll out trajectories. The Eq. (7) in the main paper can be rewritten as follows.

Dataset split	# Training	# Validation	# Testing
Short-term	3643	445	454
Long-term	4381	553	557

Table 1. VIST dataset split.

$$\mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log(\pi_\theta(v_{t+1}|v_t))Q(v_t, v_{t+1})] \quad (1)$$

$$= \frac{1}{R}\sum_{r=1}^{R}[\nabla_\theta \log(\pi_\theta(v^r_{t+1}|v^r_t)Q(v^r_t, v^r_{t+1})] \quad (2)$$

We set R as 1 in the all experiments.

3. Detail of Turing Test

We conduct the Turing Test [8] for evaluating the task of future frame generation. We hire 7 in-house annotators to conduct the evaluation. For each annotator, we randomly assign a generated sequence and a real sequence and ask this annotator to recognize which sequence is a natural sequence. We generate 128 sequences using each method and let each annotator conduct the Turing Test for all 128 sequences and all methods. Finally, we first average the error rate[3] over all 128 sequences, and then average the averaged error rate again over all annotators for every method.

4. Details of Storyline Experimental Setup

We follow [9] and split the storyline evaluation into two goals: the first is short-term, where the image is visually similar consecutive image in the album. The second goal is long-term prediction, where prediction is for the next representative event involving large visual appearance change. We split the short-term and long-term sets by the visual similarity between the representation of input photo and the target photo. We first calculate the mean absolute-value norm distance between the representation of input photo and the target photo over entire training set. Then, we set this value as a threshold so that we obtain short-term storyline by taking the storyline with the lower absolute-value norm distance and obtain long-term storyline by taking the storyline with the larger absolute-value norm distance. In experiment, the average absolute-value norm is 0.34. The final dataset split can be found in Table 1.

5. Qualitative Results

We give more qualitative results for three experiments we conduct in the main paper. It is worthy to note that those three different experiments possess different abstraction of dynamics in the natural sequences.

[3]The error rate means how many times the annotator recognize the generated sequence as a natural sequence.

Future Frames Generation. The results are in Figure 4. We show the results of ground truth, our method, GAN (generative adversarial network) [3], LSTM [10], $L1$ loss respectively. It can be observed that our approach always try to generate consistent frames along the time. It is because our approach is not only optimized to generate a realistic frame for each time step, but also optimized to generate a natural sequence through time.

Action Anticipation. The results are in Figure 5. We show seven correct anticipated examples on top section and two incorrect anticipated examples on bottom section. For each row, the first figure is the input frame and the second figure is the frame retrieved by using the nearest-neighbor algorithm with distance function of absolute-value norm. The third figure shows the probability for each action class and the forth figure shows the ground truth action class. It can be found that our method can anticipate correct action class in the future by propagating the visual representation from current to future. The retrieved figure also shows that our anticipated representation is close to the representation of the frame within the action duration. The two incorrect examples show the failure cases which our method propagate the visual representation to the wrong future and it results in the wrong prediction.

Storyline Forecasting. The result are in Figure 6. We show 6 correct examples on top 6 rows and two incorrect examples on bottom two rows. For each row, the first figure shows the input photo, second figure shows the ground truth target photo, and the following figures show the results forecasted by our method, nearest-neighbor (NN), and deep regression [11]. The strong baseline NN suffer from forecasting next photo based on appearance cue of input photo. On the other hand, because our model learns the natural dynamics within a storyline, it can forecast the next photo by predicting the changing of semantic meaning based on the semantic meaning of the input photo.

References

[1] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. 2

[2] Facebook. Pytorch. http://pytorch.org, 2017. 1

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 3

[4] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes, 2015. 2

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

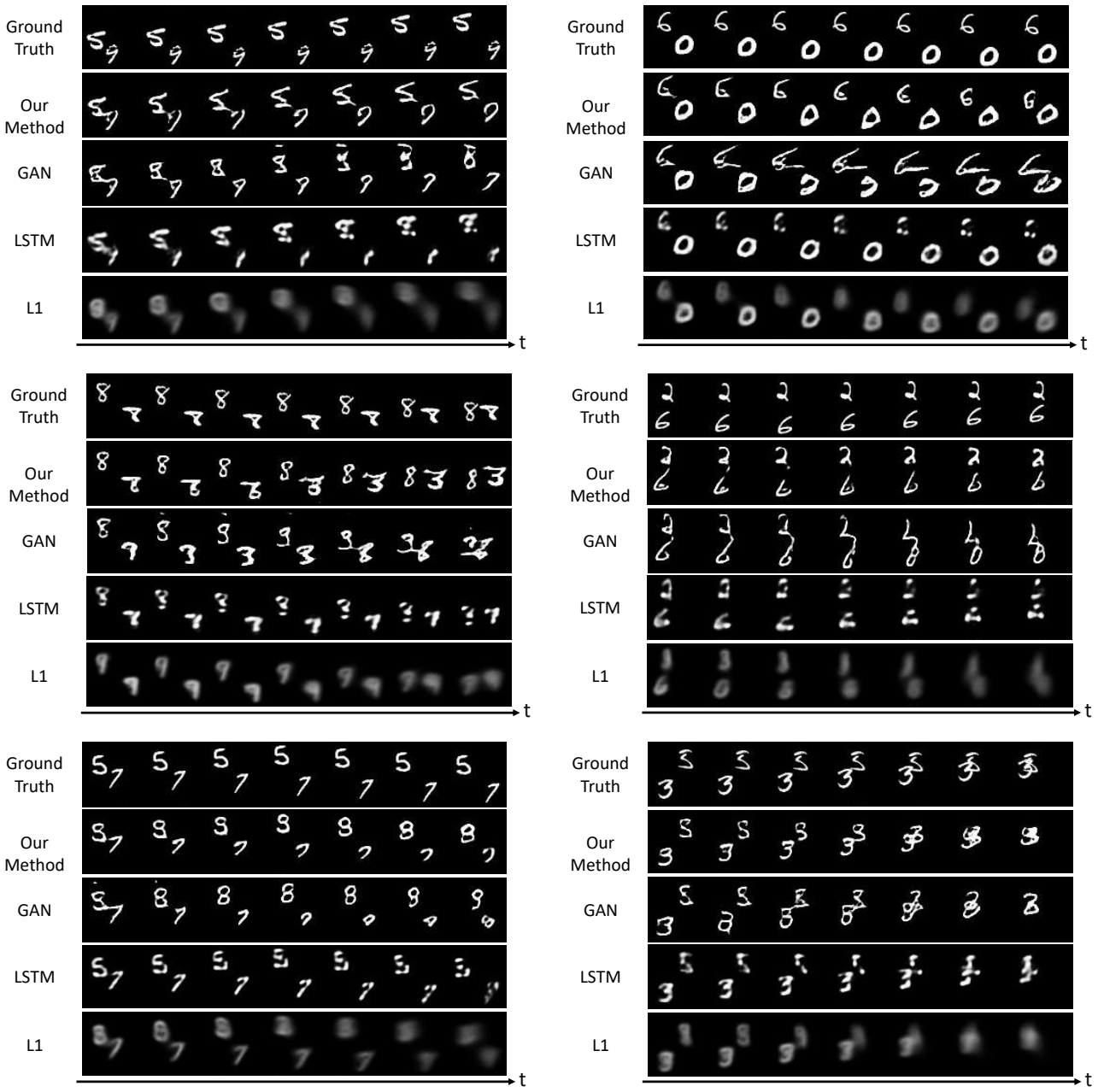

Figure 4. Qualitative results on the moving MNIST dataset. Our model trained with IRL aims to minimize long-term expected cost over the entire sequence. This results in having consistent high-quality frame prediction across time step, and avoids the compounding error problem of baselines.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1

[7] J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky. Adversarial learning for neural dialogue generation. In *EMNLP*, 2017. 2

[8] G. Oppy and D. Dowe. The turing test. 2003. 3

[9] G. A. Sigurdsson, X. Chen, and A. Gupta. Learning visual storylines with skipping recurrent neural networks. In *ECCV*, 2016. 3

[10] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015. 3

[11] C. Vondrick, H. Pirsiavash, and A. Torralba. Antici-

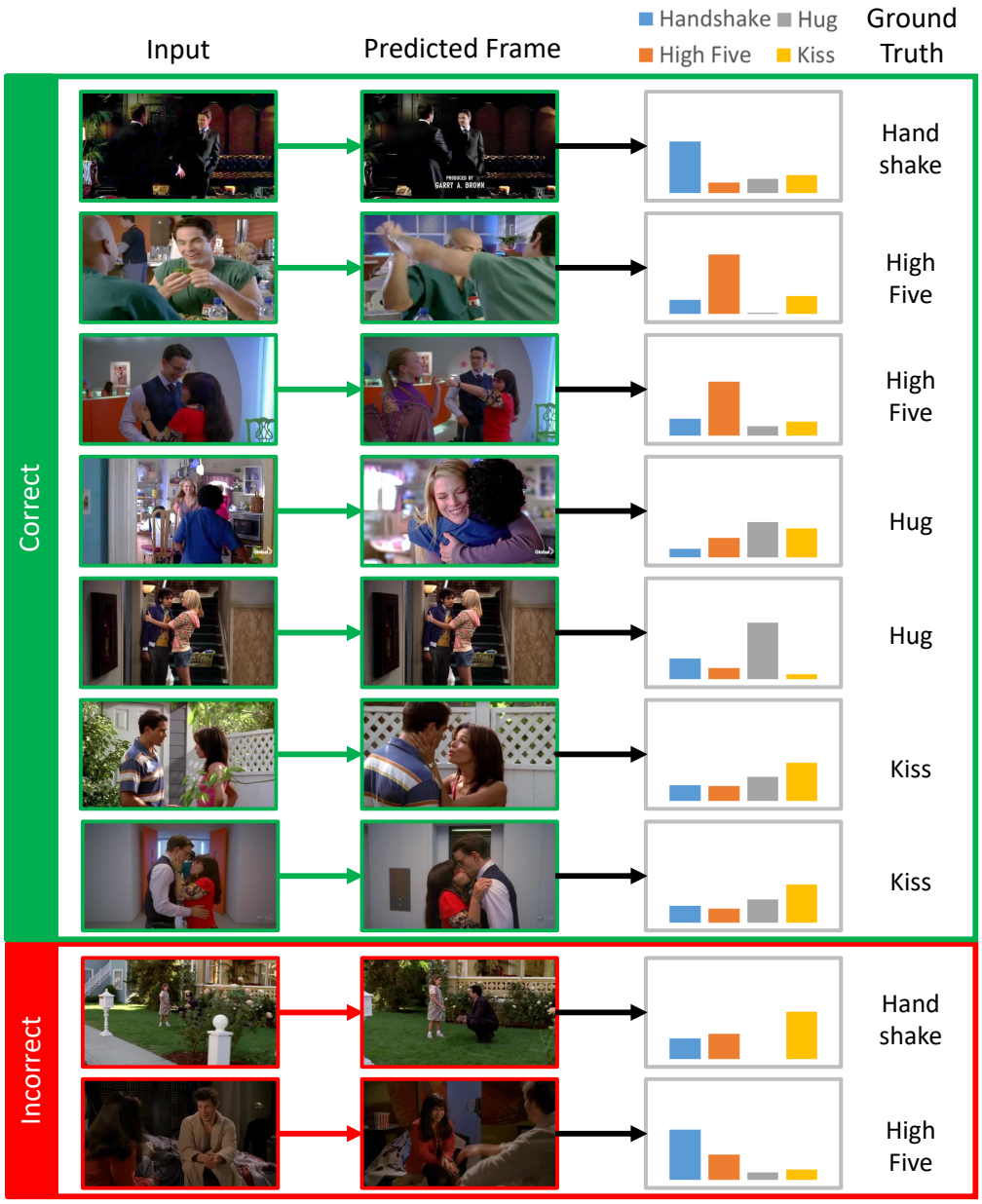

Figure 5. Single step action prediction results. Given an input frame one second before the the interaction takes place, our model is able to robustly predict the future frame and improve significantly over the baselines for action prediction. The frame visualization is retrieved by nearest neighbor image of the predicted deep representation.

pating visual representations from unlabeled video. In *CVPR*, 2016. 2, 3

[12] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: sequence generative adversarial nets with policy gradient. In *AAAI*, 2017. 2

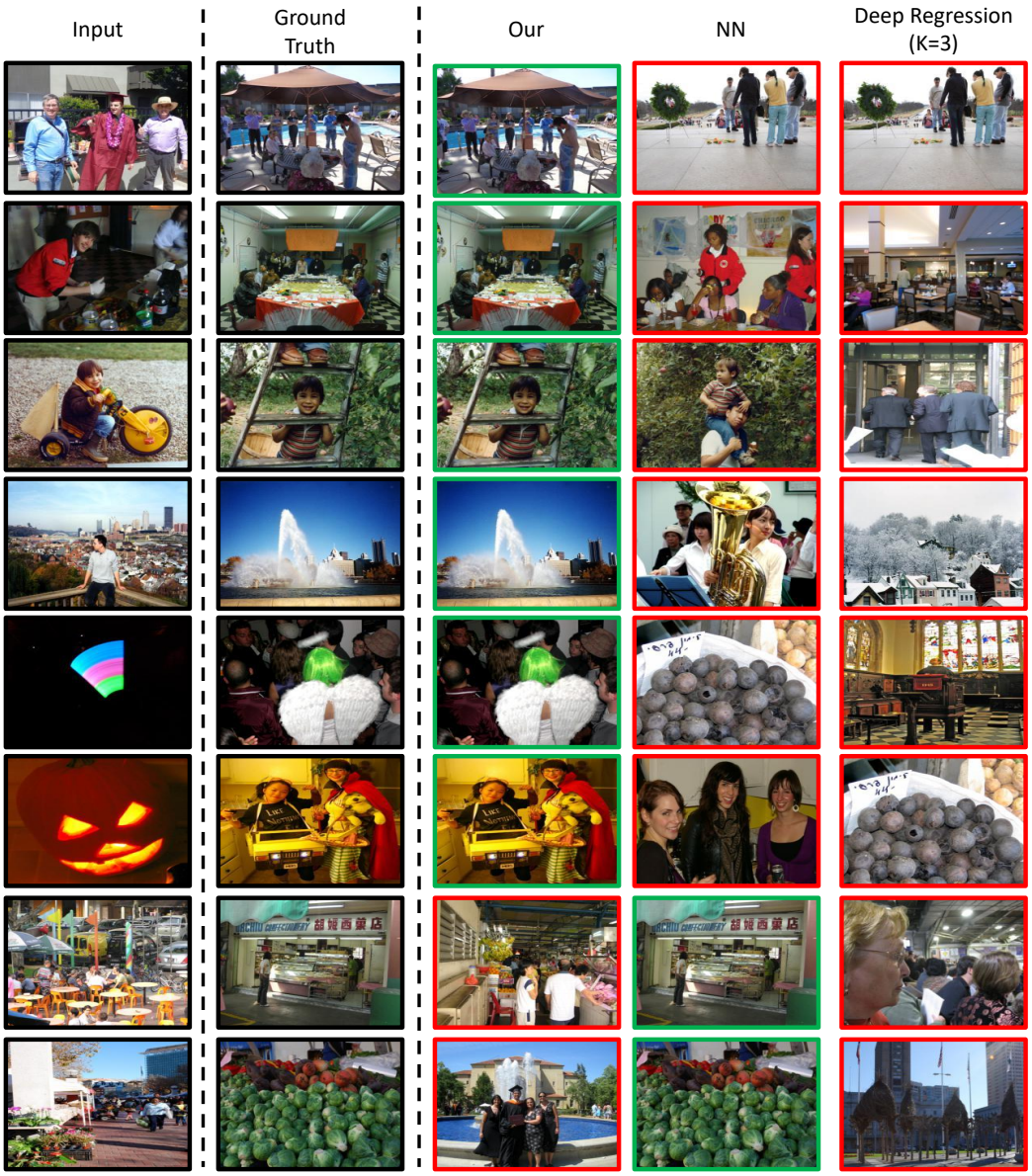

Figure 6. Qualitative results of storyline forecasting. Green boxes indicate correct prediction of the next image in the storyline, and red boxes indicate incorrect predictions. Our imitation learning based framework can best capture the semantics by imitating storylines generated by human. On the other hand, baselines focusing on appearance matching or reconstruction loss is harder to successfully predict semantics in the storylines.