ROVR: Reinforcement Optimized Video Reconstruction

Aaditya Prasad [1] Arnuv Tandon [1] Arjun Vikram [1]

[1]Department of Computer Science

Introduction

Video-to-video translation presents two primary challenges: establishing the **sequence** in which to process frames from a source video, and determining the **contextual frames** that assist in editing a given frame. Attention mechanisms appear prime for these tasks but are rarely used in video to video translation due to their $(np)^2$ time complexity, with n representing the number of frames and p the square patches per frame.

To address these shortcomings of attention, we introduce ROVR: a video to video translation architecture that achieves sub-quadratic time complexity. We leverage dual actor policy networks to determine a) the specific frame to be edited at any given timestep, and b) the context frames from the source video that should be incorporated in the editing process. Once these frames of interest are determined, a unet is used to condition on context images and edit the selected frame. Finally, we train two critic networks to help us improve our system via PPG.

Prior Work

Prior work in video to video translation attempts to resolve this issue of quadratic time complexity by introducing inductive biases in the method used to select the sequence in which to process frames, as well as contextual frames.

Wang et. al. produced Vid2Vid, a multi-frame generator that conditions on prior frames to generate an accurate, spatiotemporally consistent output. This method suffers from the assumption that *only* prior frames are useful for performing style translation on a selected frame.

Liang et. al. tackle the problem of style transfer in art by introducing a frame selection process which maximizes the amount of new information provided for style transfer at each step. Despite these advances, the method suffers from limited generalizability due to its heavy reliance on inductive bias in the frame selection process.

Existing methods attempt to avoid inductive biases in the frame selection process by utilizing attention on a low-dimensional version of the source video.

Huang et. al. jointly leverage an aggressive feature extraction network and attention in order to perform few shot video to video translation. While this method is free of inductive biases, attention still runs in quadratic time complexity – rendering the method infeasible for longer, higher resolution videos.

Local Network Results

A key metric to determine our system's value is how much agentic sampling improves local network performance over a baseline. The baseline we chose is the sampling method from Vid2Vid, i.e. conditioning on the previous two frames. We trained two local networks; one with conditioning images chosen per the baseline and one with conditioning images chosen per our agents.

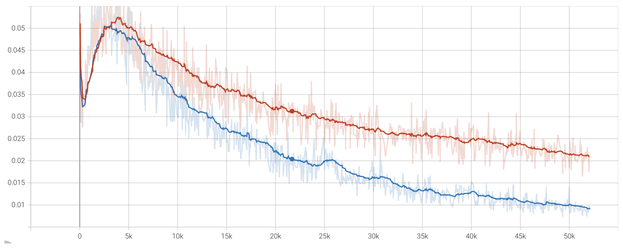

Figure 1. Total Loss of Local Networks vs. Sampling Strategy

Sequential Sampling Agentic Sampling

The above graph shows the total loss of our two networks as they trained. The agentic system quickly learns to perform much better as it searches across the state space for the next conditions.

Task and Dataset

We chose the task of video in-painting across a masked video as a testbed to train and evaluate our proposed approach. This task is demanding as it requires consideration of context from both nearby and distant frames to ensure accurate in-painting and preservation of spatiotemporal consistency.

In dataset preparation, we apply randomly sized masks that shift around the dimensions of the frame throughout a given video. This design encourages our policy to select context frames that are not the immediate neighbors of the frame we have selected to edit at timestep t. This is desirable as we want our policy to model attention and understand both long and short range dependencies.

We use the RealVSR dataset for videos. For simplicity, each video is 25 frames and has dimension 3 x 256 x 256.

MDP

- **State** (s_{i_t}): Source video with the $t-1$ edits made in prior time-steps. Each frame $\bar{X}_i$ is processed through a pre-trained feature extractor (ϕ), reducing dimensionality from (256, 256, 3) to (16, 16, 3). These extracted features $\phi(\bar{X}_i)$ are then composed into an (80, 80, 3) tensor. The first policy network is additionally fed in an LSTM-encoded history of actions, while the second policy network is fed a full resolution version of the target frame $\bar{X}_{i_t}$
- **Action** ($a_t = \{i_t, c_{t1}, c_{t2}\}$): Our first policy network (π_1) takes s_{1_t} and predicts the next frame index to edit i_t. Our second policy network takes s_{2_t} to produce the indices of two context frames c_{t1} and c_{t2}. The three indices produced, our next frame to edit and context indices, make up a_{1_t} and a_{2_t}.
- **Transition**: Our image-to-image translation network (Γ) takes as input the frame to edit $\bar{X}_{i_t}$, and conditions on context frames $\bar{X}_{c_{t1}}$ and $\bar{X}_{c_{t2}}$, to produce a frame $\Gamma(\bar{X}_{i_t})$ in the target domain. This new edited frame in the target domain replaces its respective frame in the source domain, thus updating our state to s_{t+1}. At this point, the LSTM encoded history will also have been updated with the actions chosen by our actors.

Our feature extractor ϕ is a ResNet-50. Both policy networks, π_1 and π_2 and the image-to-image translation network Γ are U-Nets.

Global Network Results

To see if our network could learn to prioritize a global reward, rather than one given at every time step, we tested to see if our network could recover a greater portion of the original video's optical flow than the sequential baseline. Below is a visualization of how the two methods chose context frames.

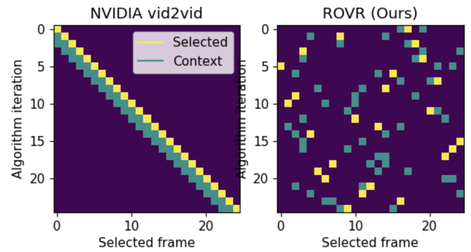

Figure 2. Sequential vs. Agentic Sampling Strategies

To compare results, we measured optical reconstruction:

$$\mathcal{O} = \frac{\varphi(\Gamma(\bar{X})) - \varphi(X)}{\varphi(\bar{X}) - \varphi(X)}$$

where $\varphi(\cdot)$ was RAFT, a network trained to predict optical flow. Over 20 rollouts, we found that the baseline method recovered 47.8% of optical flow, while our agent's sampling method recovered **65.4%**

Reward and Optimization Strategies

Local Network Loss

We aim to optimize the distance between the original ground truth image and the reconstructed image Γ outputs. We do this by combining simple supervised loss with a learned perceptual loss.

$$\mathcal{L}(X, \Gamma(\bar{X})) = \sum_l \frac{\lambda}{W_l H_l} \sum_{w,h} \left\| w_l \odot \left(\phi(X)^l_{wh} - \phi(\Gamma(\bar{X}))^l_{wh} \right) \right\|_2^2 + \frac{1-\lambda}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \left(X_{i,j} - \Gamma\left(\bar{X}\right)_{i,j} \right)^2$$

where $X, \bar{X}$ are the ground truth and corrupted images and ϕ is a pretrained model, in this case VGG, used for discerning perceptual features. λ is a hyperparameter that we grow exponentially from 0 to 0.9 throughout pretraining of the local network, as MSE loss helps force early learning but introduces unwanted behaviors such as feature averaging, while perceptual loss optimizes for higher level feature similarity.

PPG

To improve our policy networks, we calculate a dense reconstruction reward at each time step. At the last time step, we also add a global optical flow reward that minimizes distance between the optical flows of the original and reconstructed videos.

$$\mathcal{R} = \mathcal{L}(\bar{X}, X) - \mathcal{L}(\Gamma(\bar{X}), X)$$

We also train critic networks V_1, V_2 to predict rewards-to-go from states. We compute Advantages with $\hat{A}_l = V_l - r$ and then optimize the following objective for both actor critic pairs.

$$\mathcal{L}_l^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_{l_t}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_{l_t} \right) \right]$$

The $r_t(\theta)$'s in the objective above denote the probability ratio $r_t(\theta) = \frac{\pi_{l_\theta}(a_t|s_t)}{\pi_{l_{\theta_{\text{old}}}}(a_t|s_t)}$, where $\pi_{l_{\theta_{\text{old}}}}$ denotes the policy network(s) used to compute the last rollout. Finally, the critics are regressed to the true rewards-to-go by minimizing the following

$$\mathcal{L}_{V_l} = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{R}_i - V_l(s_i))^2$$

Next Steps and Improvements

While our agents do perform better than the baseline on a simple masking strategy, their training is often unstable. When we made the task more complex by adding brightness/noise corruptions to our original masking strategies, we found that our actor/critic networks failed to train well entirely. We have some ideas on how to fix this:

1. **Separate the training of the policy networks from each other.** We already pretrain our local network, but it is likely that both of our policy networks get a bad signal while their counterpart is still learning. Thus it may help to hardcode "expert" π_1^*, π_2^* and then pretrain the real policy networks with the expert of their counterpart, so they have full agency over the reward signal.
2. Similar in spirit to (1), we could develop experts π_1^*, π_2^* **and use IL to pretrain our policy networks directly.** Then, after they are initialized in this way, their exploration during PPG may gather better results.
3. We use separate actor and critic parameters because empirically, PPG has shown drastic performances in sample efficiency over PPO. However, we may test **moving back to PPO** and sharing actor/critic parameters in case one's learned features can help the other understand some complicated region of the state/action landscape.
4. Finally, if both of these methods fail to produce stable training, we may have to look at using an **off policy algorithm** such as DQN. An off policy algorithm would increase sample efficiency, which is a minor constraint, but more importantly it would let us pretrain our critic networks on expert demonstrations. This is impossible with an on policy algorithm because the critic networks need to be trained on the same policy as the one being rolled out.