

# GT-SVJ: Generative-Transformer-Based Self-Supervised Video Judge For Efficient Video Reward Modeling

## Supplementary Material

Perturbation Type	Sampling Probability	
	Real Video	Generated Video
Temporal Slice Swap	0.04	0.05
Noisy Segments	0.10	0.15
Frame Shuffle	0.18	0.25
Frame Drop	0.18	0.25
Patch Swap	0.20	0.30

Table A.1. **Sampling probabilities assigned to each perturbation type based on gradient-based difficulty analysis.** Note that a fixed probability of 0.3 is reserved for contrasting real videos with model-generated videos, and hence the probabilities of perturbations for real videos sum to 0.7.

### A. Perturbing Real Videos

To explicitly train our reward model to recognize subtle temporal and spatial inconsistencies in videos, we introduce a diverse set of perturbations that simulate realistic failure modes observed in video generative models. These perturbations are designed to degrade semantic coherence, temporal smoothness or visual fidelity while preserving overall structure – thereby creating hard negative samples that encourage the model to learn fine-grained consistency cues rather than relying on trivial artifacts.

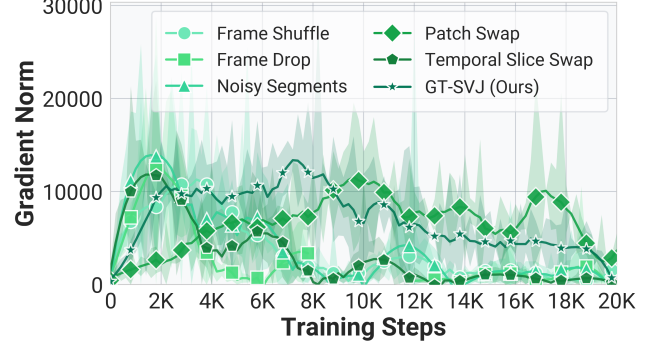
For training, each real video is paired with a randomly chosen perturbed counterpart produced through one of five operators: frame shuffle, frame drop, noisy segments, patch swap, or temporal slice swap. We show representative examples of these perturbations in the supplementary video `supp_video_perturbations.mp4`.

#### A.1. Adaptive Perturbation Sampling

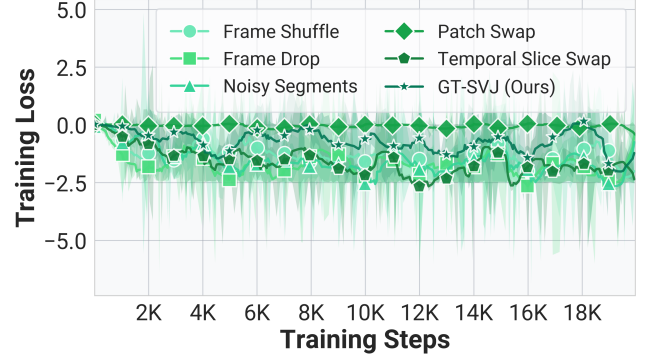
At each training step, rather than uniformly sampling negatives from all five types of perturbations to augment our training dataset, we assign a fixed probability of selection based on a perturbation’s *relative difficulty*, defined by how easily the discriminative model can learn to distinguish the perturbed sample in isolation. Difficulty is quantified through the magnitude of the gradient of the loss with respect to the model parameters when trained on a single perturbation type. Perturbations that induce lower initial gradients (*i.e.*, are harder to detect) are sampled with higher probability to ensure sufficient exposure during training, while easier perturbations receive lower weights to avoid gradient domination early on (See Fig. A.1).

Formally, let  $\mathcal{L}_p$  denote the loss corresponding to perturbation type  $p$ , and let  $\nabla_{\theta}\mathcal{L}_p$  be the gradient with respect to model parameters  $\theta$ . Then we obtain the sampling probability  $\mathcal{P}$  for perturbation  $p$  based on

$$\mathcal{P}(p) \propto \frac{1}{|\nabla_{\theta}\mathcal{L}_p|}. \quad (\text{A.1})$$



(a) Average gradient norm during training from using each perturbation type individually.



(b) Training loss curves from using each perturbation type individually.

Figure A.1. **Effect of using each perturbation type individually to generate negative samples.** We show the ease or difficulty of training the discriminative model when using each perturbation type individually.

In practice, we track the gradients for each perturbation type across independent training runs and quantify its effective difficulty based on the rate at which its gradient norm decays. Specifically, we monitor  $|\nabla_{\theta}\mathcal{L}_p(t)|$  over optimization steps and apply early stopping when the gradient norm falls below a small threshold  $\epsilon = 1500$ , indicating saturation. The decay time

$$\tau_p = \min\{t : |\nabla_{\theta}\mathcal{L}_p(t)| \leq \epsilon\} \quad (\text{A.2})$$

serves as a proxy for perturbation difficulty, where smaller values correspond to easier perturbations that are rapidly learned in isolation.

We employ an inverse decay-based heuristic to determine perturbation sampling probabilities, assigning higher weights to distortions whose gradients decay more slowly over training. This design choice prevents harder-to-learn perturbations from being under-represented and ensures sustained exposure to challenging negative samples. After normalization, the resulting probabilities

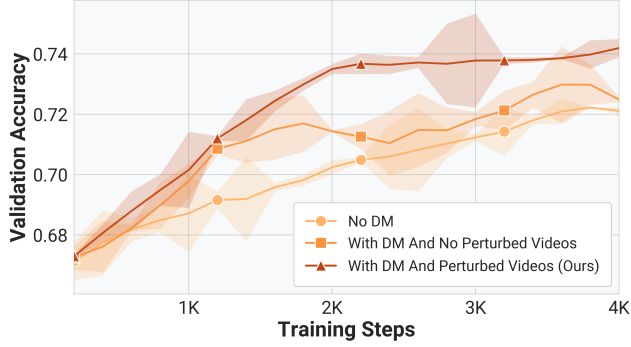


Figure A.2. **Effect of using all types of perturbed videos in training the discriminative model (DM).** We observe significantly faster convergence to 2-3% better validation alignment accuracies to human preferences.

are fixed for the remainder of training to enable stability and reproducibility across runs. We report the empirically calibrated sampling distributions used for all experiments in Tab. A.1.

Although all perturbation types exhibit comparable peak gradient magnitudes during early training, we observe substantial variation in their gradient decay dynamics. Perturbations whose gradients vanish rapidly contribute limited long-term learning signals and are consequently down-weighted. In contrast, perturbations that maintain persistent gradients are prioritized, enforcing continuation of optimization and improved discriminative robustness.

## A.2. Impact on Learning

These perturbations collectively encourage the model to develop sensitivity to both spatial realism and temporal coherence. As shown in Fig. A.1, perturbation types that produce visually plausible yet semantically inconsistent videos lead to smoother loss curves but smaller gradients, justifying their greater difficulty and higher sampling frequency for data augmentation. This design effectively acts as a curriculum, guiding the model from easy-to-detect artifacts toward more nuanced inconsistencies.

We also show the collective benefits of using all the perturbations in training our discriminative model in Fig. A.2. Adding perturbed videos to training consistently leads to faster convergence to 2-3% better alignment accuracies with human preferences during inference.

## B. Qualitative Results

We show qualitative results of GT-SVJ and the different baselines on video preference in `supp_video_results.mp4`. We show the performance on human preference alignment for each method on five randomly selected video pairs from each of the three benchmark datasets, GenAI-Bench [15], MonteBench [38], and VideoReward-Bench [23].