INR-V: A Continuous Representation Space for Videos

Anonymous authors
Paper under double-blind review

Abstract

Images are considered a complete signal, whereas videos are usually broken down into a set of temporally coherent images. Consequently, an image space is leveraged for various video-based tasks, such as novel video generation and future video segment prediction. This limits the expressivity of videos to only image-based operations needing network designs to obtain temporally coherent trajectories in the image space. We propose INR-V, a video representation network that learns a continuous latent space directly for videos. INR-V regards videos as complete units parameterized by implicit neural representations (INRs), a multi-layered perceptron with only a few thousand parameters. A meta-network is then used to predict these few thousand parameters allowing it to learn a continuous space over the neural representations. Later, the meta-network can generate novel neural representations by sampling diverse points over the learned space leading to novel videos. Interestingly, we find that conditional regularization and progressive weight initialization play a crucial role in obtaining INR-V. In this work, we analyze INR-V's several video-based properties. For instance, we show smooth interpolation of coherent videos between two videos by traversing along their latent points in the underlying video space. Moreover, learning a video space allows the network to directly invert an unseen video to its latent point in the latent space. We show the various applications of video inversion. Lastly, INRs learn a continuous signal independent of the input dimension letting INR-V generate multi-resolution videos (like 32×32 or 100×100) directly at inference without any finetuning or architectural changes. We conduct several comparisons and evaluate each of the properties, ultimately demonstrating the potential of a continuous representation space for videos.

1 Introduction

Learning to generate complex spatio-temporal videos from simple distributions is a challenging problem in computer vision that has been recently addressed in various ways Tian et al. (2021); Tulyakov et al. (2017); Clark et al. (2019); Skorokhodov et al. (2021); Ding et al. (2019); Yu et al. (2022); Yan et al. (2021). State-of-the-art (SOTA) works Skorokhodov et al. (2021); Tian et al. (2021); Yu et al. (2022) treat video generation as a task of generating a sequence of temporally coherent frames. Although such networks have advanced the SOTA to generate high-quality frames (such as carefully crafted eyes, nose, and mouth for talking-head videos), they come with a major limitation: They rely on an image space. This limits the application of the learned space to image-based operations such as animating images and editing on frames. Direct operations on videos, such as interpolating intermediate videos between two videos and generating future segment of a video, become difficult. This is because such operations require learning the set of frame and motion constraints and ensuring that they are coherently learned.

We propose that videos should be represented as a single unit instead of being broken into a sequence of images. One can learn a latent space where each latent point represents a complete video. However, with existing video generator architectures, such representations are difficult. Firstly, such a video generator would be made of several 3D convolution operations. As the dimension and length of the video increase, such an architecture would become drastically computationally expensive. Secondly, videos are high-dimensional signals spanning both spatial and temporal directions. Representing such a highly expressive signal by a single latent point would require complicated generator architectures and a very high-dimensional latent space.

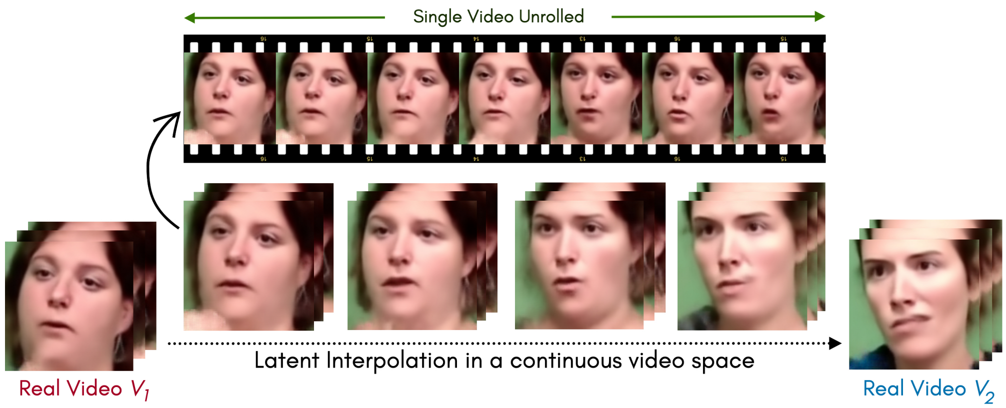

Figure 1: Demonstrating the continuity of the video space learned by INR-V by interpolating novel videos between two real videos V_1 and V_2. Note that content (identity, hair) and motion (pose, expressions) gradually transition as we traverse the trajectory in the latent space between V_1 and V_2's latents.

Instead, videos can be parameterized as a function of space and time using implicit neural representations (INRs). Any point in a video V_{hwt} can be represented by a function $f_\theta(h, w, t) \rightarrow RGB_{hwt}$ where t denotes the t^{th} frame in the video and h, w denote the spatial location in the frame and RGB denotes the color at the pixel position $\{h, w, t\}$. Here, the dynamic dimension of videos (a few million pixels) is reduced to a constant number of weights θ (a few thousand) required for the parameterization. A network can then be used to learn a prior over videos in this parameterized space. This can be obtained through a meta-network that learns a function to map from a latent space to a reduced parameter space that maps to a video. A complete video is thus represented as a single latent point.

We propose INR-V, a video generator network with a continuous video representation space based on learning an implicit neural representation for videos. It is illustrated in Fig. 2. INR-V is made of key elements that, when combined, makes it ideal for video representation: (1) Its INR is free of any convolutional layers and relies on traditional multi-layered perceptrons (MLPs), leading to very few parameters (a found thousand) when compared to the existing SOTA architectures (millions of parameters) Skorokhodov et al. (2021); Tian et al. (2021). (2) Having very few parameters, INR's weights can be populated using a secondary meta-network called hypernetwork Ha et al. (2016) that learns a continuous function over the INRs by getting trained on multiple video instances. (3) It is trained on a deterministic distance loss, such as Euclidean or Manhattan distance. This allows INR-V to learn the exact requirements of a coherent video directly from the ground truth video instances.

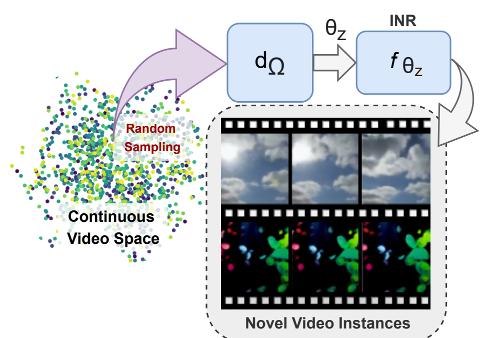

Figure 2: **Overview of INR-V:** INR-V learns a continuous video space by first parameterizing videos as implicit neural representations denoted by f_{θ_z}, where z denotes a unique video instance V_z. Next, a meta-network based on hypernetworks denoted by d_Ω is used to learn a continuous representation over the neural representations. d_Ω is conditioned by an underlying continuous video space where each point denotes the condition for a complete video.

Hypernetworks have seen wide applications in grahics Sitzmann et al. (2020; 2021); Chiang et al. (2021); Sitzmann et al. (2019); however, they have seldom been used for videos. Hypernetworks are notoriously unstable to train, especially on the parameterizations of highly expressive signals like videos. Thus, we propose a key prior regularization and a progressive weight initialization scheme to stabilize the hypernetwork training allowing it to scale quickly to more than 30,000 videos. As we show in the experimental section, INR-V demonstrates an expressive and continuous video space by getting trained on these datapoints. The learned prior enables several downstream tasks such as random video generation. It also demonstrates unique properties such as video inversion, allowing tasks like future segment prediction and video inpainting directly at a video level. Finally, as shown in Fig. 1, INR-V showcases smooth interpolation of novel videos between two videos by traversing the path between their latent points. Interpolation morphs different identities and motions and generates coherent videos. Interestingly, the properties demonstrated in this work are not enforced at training but are natural outcomes of the continuous video space. To summarize, our contribution in this work is as follows:

1. We propose considering videos as a single unit and learning a continuous latent space for videos where each latent point represents a complete video.
2. We propose INR-V, a video representation technique that parameterizes videos using INRs, bringing down the dimension of a video from a dynamic few million to a constant few thousand. INR-V uses a hypernetwork as a meta-network to learn a continuous space over these parameterizations.
3. We demonstrate the benefit of a key regularization and progressive weight initialization scheme to stabilize the hypernetwork training. We scale the hypernetworks to more than 30,000 video points enabling it to learn a continuous meaningful latent space over the INRs.
4. Lastly, we demonstrate key properties of the learned video space, such as video interpolation, video inversion, and so on, by conducting several experiments and evaluations.

2 Related Work

Video Generation. Video generation aims to produce novel videos from scratch. It falls under the paradigm of 'video synthesis' that encompasses several categories, including (1) Video prediction Luc et al. (2020); Moing et al. (2021); Walker et al. (2021): that predicts the next set of frames given the current frames, (2) Frame interpolation Park et al. (2021); Niklaus & Liu (2020); Niklaus et al. (2017); Zhang et al. (2021): that interpolates frames between given frames of a video. These tasks generate the unseen portion of the video based on the context of the seen portion. On the other hand, video generation produces videos without any expressive prior context, making the task more challenging. The complexity of the problem has led to a plethora of works in this area Tian et al. (2021); Tulyakov et al. (2017); Skorokhodov et al. (2021); Clark et al. (2019); Ding et al. (2019); Yu et al. (2022). VideoGPT Yan et al. (2021) tackled this challenge by first reducing the raw videos of up to 128×128 dimension to a quantized space. It then trained a transformer architecture to model a prior over the quantized space. Our architecture is conceptually similar to VideGPT, which used a likelihood-based generative model to learn a video prior. However, VideoGPT operates on a quantized space that is discontinuous, making the prior less expressive. INR-V, on the other hand, models a continuous video space. VideoGPT also consists of 3D convolution layers making the model computationally expensive for larger videos. INR-V is a simple MLP based on a continuous parameterization scheme of INRs, making it agnostic to the video dimension. This allows scaling to multiple resolutions (64×64 or 256×256) at inference without any architectural changes or finetuning. More recent works StyleGAN-V Skorokhodov et al. (2021), DIGAN Yu et al. (2022), and MoCoGAN-HD Tian et al. (2021) are a GAN-based setup that model videos as a temporally coherent trajectory over an image space.

Hypernetworks. Hypernetworks Ha et al. (2016) were introduced as a metafunction that initializes the weights for a different network called the primary network. Hypernetworks have been widely used for several purposes, starting from compression Nguyen et al. (2022); Gao et al. (2021), few-shot learning Sendera et al. (2022); Lamb et al. (2021), continual learning von Oswald et al. (2019), architecture search Zhang et al. (2018), language modeling Suarez (2017), meta-learning Zhao et al. (2020). We use hypernetworks to populate our primary video generation network, an MLP parameterizing different video instances.

Implicit Neural Representations. In this paradigm, a continuous signal is represented by a neural network. INRs have had wide adaptations in 3D Computer Vision Park et al. (2019); Genova et al. (2019); Sitzmann et al. (2018); Mescheder et al. (2018); Sitzmann et al. (2021); Mildenhall et al. (2020) and Computer Graphics Guo et al. (2021); Yao et al. (2022). Recently, INR was adopted for images Skorokhodov et al. (2020) and videos Chen et al. (2021); Sitzmann et al. (2020); Yu et al. (2022); Chen et al. (2022). INR-GAN Skorokhodov et al. (2020) first showed the application of INRs in generating high-quality images by replacing the generator component of StyleGAN2 Karras et al. (2019) with an MLP-based INR. It then used a hypernetwork to populate the INR. Unlike INR-GAN, which is trained using a stochastic discriminator, INR-V relies on a deterministic distance-based loss to train the hypernetwork. SIREN Sitzmann et al. (2020) proposed periodic activation functions for INRs as a replacement for ReLU activation to parameterize many different data types like images, videos, sounds, and 3D shapes, with fine details. NeRV Chen et al. (2021) designed an implicit function as a continuous function of time and used convolution blocks at each time step to parameterize discrete frames showcasing an improved frame quality over SIREN. Recently, VideoINR Chen et al. (2022) was proposed that used INRs for video superresolution. DIGAN Yu et al. (2022) incorporated INRs made of MLP layers for video generation. It consisted of two separate networks that generated spatial and temporal codes for generating videos in a frame-wise fashion. StyleGAN-V Skorokhodov et al. (2021) also incorporated INRs and relied on continuous non-periodic positional encodings for each timestep of a video. Like NeRV, StyleGAN-V used traditional convolution operations for frame-by-frame video generation. Both DIGAN and StyleGAN-V used a GAN set up to train the video generators. INR-V is based on MLPs with ReLU activation trained in a fashion similar to Light Field Networks (LFNs) Sitzmann et al. (2021). LFNs proposed a novel neural scene representation for novel view synthesis and trained a hypernetwork over multiple object instances using distance-based losses like Euclidean or Manhattan distance. Like LFNs and INR-GAN, INR-V parameterizes the entire signal (a video) using INRs and relies on a single hypernetwork to generate the INRs. However, unlike LFNs and INR-GAN, INR-V encodes a denser representation of a volumetric 3D $\in \mathbb{R}^3$ data making hypernetwork training more challenging.

3 INR-V: Implicit Neural Representation for Video Synthesis

Each video instance V_n consists of pixels at locations (h, w) at t^{th} frame. We have a particular parameter vector θ_n that is used by a network f to generate the value of the color RGB_{hwt} for that pixel location (h, w, t). We need to learn a network d with parameters Ω that predicts the parameters θ_n for a particular video V_n. Here, d is a hyper-network. The overall approach to train the network is illustrated in Fig. 3.

3.1 Hypernetwork for Modeling Multiple Video Instances

As f_θ implicitly stores a single video signal, any new video would need its own implicit function. Let f_{θ_n} denote the implicit function for a given video $\{V_n\}_{n=1}^N$ where N is the total number of available videos in the training dataset $\mathcal{D}$. Each of these implicit functions, f_{θ_n} can be modeled using a neural network trained on each pixel value of the video V_n. Thus, implicit functions minimize the following objective:

$$L(\theta_n) = \frac{1}{T}\frac{1}{W}\frac{1}{H}\sum_{t=1}^{T}\sum_{w=1}^{W}\sum_{h=1}^{H}(f_{\theta_n}(h, w, t) - RGB_{hwt})^2 \tag{1}$$

Generating a novel video V_z translates to generating a novel implicit function f_{θ_z} that represents a meaningful video. Let us consider f_{θ_z}, an unseen sample from an underlying distribution Φ. Each point in the distribution Φ denotes an implicit function of a meaningful video. To randomly sample f_{θ_z}, we make use of a meta-network to learn the distribution Φ.

We use a hypernetwork d_Ω as a meta-network to parameterize f_θ, such that $d_\Omega(m_n) = \theta_n$ for video instance V_n. Here m_n is a a d-dimensional point in the latent space, say τ, and serves as an instance code for V_n. d_Ω is conditioned by m_n. Given enough number of samples N, d_Ω should be able to learn a representation over Φ constrained on τ. As can be seen in Fig. 3, a latent code m_n is fed to the hyper-network d_Ω which generates a set of parameters θ_n. The parameters θ_n are then used to initialize network f_{θ_n} to generate V_n.

Let us consider τ as a meta-distribution such that $\{m\}_\mathcal{D} \in \tau$. At the time of inference, m_z can be randomly sampled from τ. As d_Ω has learned a valid representation over Φ constrained on τ, m_z enables d_Ω to generate a meaningful implicit function $f_{\theta_z} \in \Phi$. Sampling from τ can be made straight forward by making sure τ is regularized during training. At the time of training, Ω and $\{m_n\}_{n=1}^N$ are optimized together. θ is a non-learnable parameter and f_θ is initialized as the output of d_Ω. The following objective is optimized:

$$L(\Omega, m) = \frac{1}{N}\frac{1}{T}\frac{1}{W}\frac{1}{H}\sum_{n=1}^{N}\left(\sum_{t=1}^{T}\sum_{w=1}^{W}\sum_{h=1}^{H}(f_{\theta_n}(h, w, t) - RGB_{ijk})^2\right) \quad \text{and} \quad \theta_n = h_\Omega(m_n) \tag{2}$$

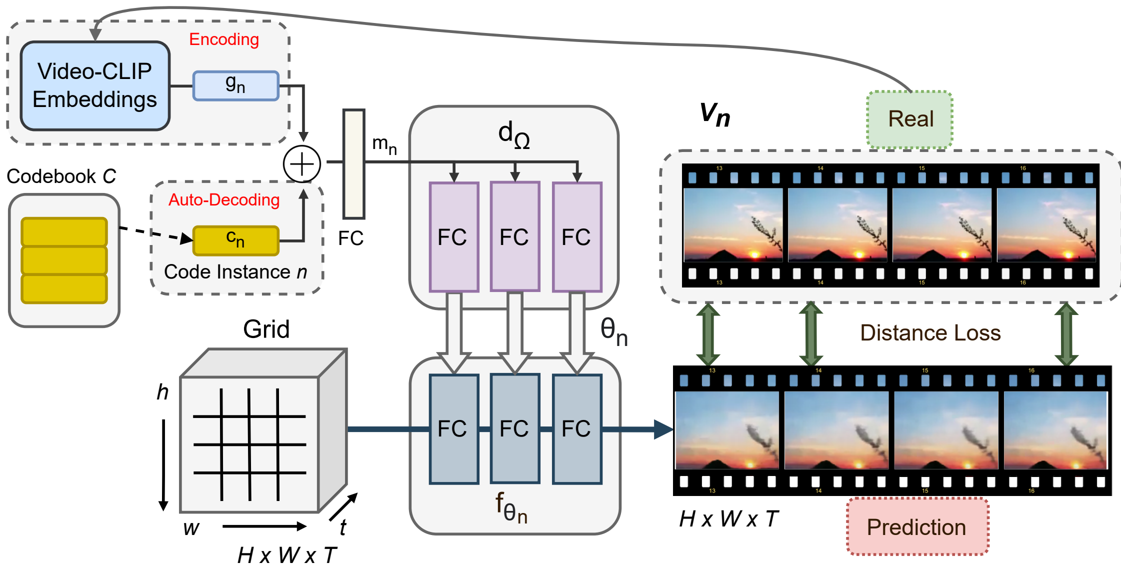

Figure 3: **Architecture of INR-V:** Any video instance V_n is represented by its corresponding implicit neural representation, an MLP, f_{θ_n}. f_{θ_n} takes a grid as input denoting the pixel positions of the video encoded using periodic positional encodings. It then generates the pixel values for all the positions. f_{θ_n} is initialized by a meta-network called hypernetworks denoted as d_Ω composed of a set of MLPs. d_Ω is conditioned by an instance code m_n unique to every video instance V_n. m_n is trained by combining (1) auto-decoding framework to regress to a code c_n and (2) encoding-framework to regularize the space using CLIP embedding that generates V_n's semantic code g_n. At the time of inference, m_n is randomly sampled from an underlying learned distribution τ.

3.2 Regularizing τ for Hypernetwork Conditioning

To generate a novel video, a random latent m_z is sampled from the latent space τ. d_Ω is then conditioned on m_z generating an implicit function $f_{\theta_z} \in \Phi$. In a standard hypernetwork training, m_n is optimized in an auto-decoding framework as given in Eqn. 2. However, given the complexity of the signal V (a 3D volumetric representation) that d_Ω has to model, $\{m\}_\mathcal{D}$ can collapse to a single point if τ is not regularized at the time of training, bringing the expressiveness of d_Ω down to a single implicit function. We regularize τ by leveraging pretrained CLIP Radford et al. (2021) designed for generating semantically meaningful embeddings for images. We design Video-CLIP that encodes an entire video V_n to a vector g_n. As shown in Fig. 4, Video-CLIP first generates the image-level CLIP embeddings. These embeddings are then passed through a bi-directional GRU. The mean of the hidden state outputs of the final layer produces g_n. The regularized instance code m_n is now given as:

$$m_n = \phi(c_n, g_n) \tag{3}$$

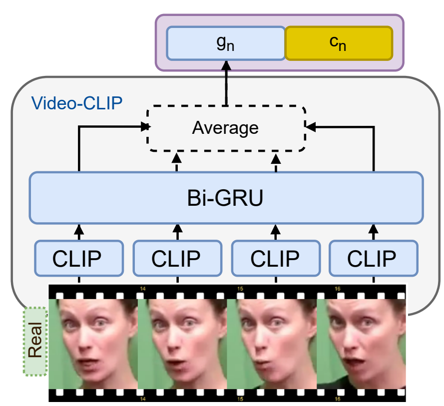

Figure 4: **Video-CLIP**: Encoding a video V_n to a latent vector g_n by using image-level CLIP encodings.

where c_n is the instance code of V_n optimized in an auto-decoding fashion at the time of training, and ϕ is a neural network. CLIP regularization encourages the latent codes to be spaced sufficiently apart by leveraging predefined semantic encoding. This also helps in avoiding a mode collapse. We observe through our experiments that CLIP regularization leads to a faster convergence with the implicit functions preserving finer video details. Please find the ablation on CLIP regularization in Appendix A.1.

3.3 Progressive Training

A video is a dense 3D volume mandating its neural representation to model every single point in the volume. Although implicit representations have a constant number of parameters made of only a few layers of MLP in our case, learning a meta-function using a hypernetwork over such dense representations is challenging. As a result, if not appropriately initialized, the hypernetworks can easily collapse to a single representation despite CLIP regularization. Moreover, a sub-optimal hypernetwork initialization could result in a significantly longer convergence period. To tackle this challenge, we adopt a progressive initialization scheme. Firstly, the training is divided into multiple stages. Each stage, denoted by $\{l\}_{l=1}^{\mathcal{K}}$ where $\mathcal{K}$ is the total number of stages, is made of a subset of the training dataset $\mathcal{D}$. The number of samples N_l in each stage l is given as:

$$N_l = \begin{cases} N_{l-1} + \epsilon_l & l > 1 \\ \mathcal{C} & l = 1 \end{cases} \tag{4}$$

where $\mathcal{C}$ is a constant and ϵ_l denotes the number of additional samples for l^{th} stage. Each step l consists of $\{V_n\}_{n=1}^{N_l}$ datapoints that is computed as:

$$\{V_n\}_{n=1}^{N_l} = \{V_i\}_{i=1}^{N_{l-1}} + \{V_j\}_{j=N_{l-1}}^{N_{l-1}+\epsilon_l} \tag{5}$$

where the order of set $\{V\}$ is maintained across the training stages. At the start of the training, the model is trained on $\mathcal{C} < 10$ examples. This allows the hypernetwork to quickly adapt to the handful of examples and initialize the weights. However, jumping from $\mathcal{C}$ to $\sim$ 30,000 samples cause the network to collapse again. Thus, we adapt the network progressively to the given examples. A vital step in this method is reusing the instance codes c_n learned at a stage l in the next stage $l+1$. If not done, this pushes the hypernetwork to re-learn all the instances. Instead, we focus on teaching the datapoints to the hypernetwork in stages by first fitting on a set of examples and then using the experience to learn more examples. Reusing the weights of the previous stages allows the hypernetworks to retain the previously seen examples.

4 Experiments

Experimental Setup: We perform our experiments on (1) How2Sign Faces Duarte et al. (2020), (2) SkyTimelapse Xiong et al. (2017), (3) Moving-MNIST Srivastava et al. (2015), and (4) RainbowJelly[1]. Real video samples of each dataset is visualized in the Appendix Fig. 11. How2Sign Duarte et al. (2020) is a full-body sign-language dataset consisting of 11 signers. The signers have elaborate facial expressions, mouth, and head movements. We modify How2Sign to How2Sign-Faces by cropping the face region out of all the videos and randomly sample 10,000 talking head videos, each of at least 25 frames, dimension 128×128. SkyTimelapse Xiong et al. (2017) consists of scenic videos focusing on sky changes. It is made of 1803 videos, each at least 25 frames long. The videos are first center-cropped to 360×360 from an original dimension of 360×620 and then resized to the 128×128 for training. Moving-MNIST Srivastava et al. (2015) is a video dataset made of moving MNIST classes containing a total of 10,000 datapoints. Each video is 20 frames long. RainbowJelly is a single underwater video capturing colorful jellyfishes. The video is first extracted into frames which are then divided into videos of 25 frames each, making a total of 34,526 videos. Similar to SkyTimelapse, the videos are first center cropped to 360×360 and then resized to 128×128.

All experiments are performed on 2 NVIDIA-GTX 2080-ti GPUs with 12 GB memory each. All models, except INR-V, are trained at a resolution of 128×128. To make training computationally efficient, INR-V is trained on a lower resolution of 100×100 videos. Based on INRs, INR-V can infer directly at multiple resolutions (please refer section 5.2). For evaluations and comparisons, INR-V is inferred at 128×128 like the other models. The training setup and model architecture are in Appendix A.2[2]

4.1 Comparing INR-V with Single-INR

INR-V uses hypernetworks to learn a distribution over the INRs of videos. A single hypernetwork d_Ω can initialize the INRs for multiple videos $\{V_n\}$ based on their respective instance codes m_n. Thus, measuring if

[1] https://www.youtube.com/watch?v=P8Bit37hlsQ

[2] The codebase, dataset, and pretrained models will be publicly released.

Dataset	Single-INR $\mathcal{E} \downarrow$	Single-INR $\text{PSNR}_{50} \uparrow$	Single-INR $\text{SSIM}_{50} \uparrow$	INR-V $\mathcal{E}_{50} \downarrow$	INR-V $\text{PSNR}_{50} \uparrow$	INR-V $\text{SSIM}_{50} \uparrow$	INR-V $\text{PSNR}_{\text{FULL}} \uparrow$	INR-V $\text{SSIM}_{\text{FULL}} \uparrow$
How2Sign-Faces	4.83	29.72	0.925	8.29	25.69	0.850	25.84	0.869
SkyTimelapse	4.69	36.19	0.943	5.87	33.69	0.931	33.94	0.924
Moving-MNIST	3.57	37.26	0.978	6.06	29.81	0.949	29.54	0.975
RainbowJelly	4.17	35.93	0.918	5.02	33.34	0.937	33.57	0.938

Table 1: Quantitative metrics on reconstruction quality. Comparison set is made of 50 videos per training dataset. INRs trained individually for each video is denoted as Single-INR. INR-V trains a single hypernetwork d_Ω to populate the INRs of all the videos in the training dataset. PSNR_{50} and SSIM_{50} are computed on the comparison set, $\text{PSNR}_{\text{FULL}}$ and $\text{SSIM}_{\text{FULL}}$ are computed on the entire training set. $\mathcal{E}$ is computed on videos with pixel range $[0, 255]$.

Method	How2Sign-Faces	SkyTimelapse	Moving-MNIST	RainbowJelly
MoCoGAN-HD	396.53	321.44	296.95	1856.21
DIGAN	165.89	135.60	144.97	408.19
StyleGAN-V	94.64	**85.05**	109.85	1227.70
INR-V	161.68	153.42	103.24	**356.98**
+ Denoising	**87.22**	-	**47.28**	-

Table 2: FVD_{16} metrics computed on random videos generated by the respective models.

d_Ω generates the INR functions f_θ accurately is crucial. We evaluate this using a set of 50 randomly sampled videos from the training dataset. Each video is first trained to fit a single INR function f_{θ_n} using Eqn. 1 denoted as Single-INR. Next, the INRs of these 50 videos are populated using a pretrained hypernetwork d_Ω trained on the entire dataset. We measure the reconstruction quality with PSNR, SSIM, and the error as:

$$\mathcal{E} = \left(\frac{1}{50} \sum_{n=1}^{50} \frac{1}{HWT} (V_n^{'} - V_n)^2 \right)^{\frac{1}{2}} \tag{6}$$

where $V_n^{'}$ denote the video generated using the implicit function f_θ. Single-INR was optimized for 750 steps using Eqn. 1 taking ~ 5.56 minutes for each video (~ 4.63 hours for 50 videos). Table. 1 presents quantitative metrics on the videos reconstructed using Single-INR and INR-V. $\text{PSNR}_{\text{FULL}}$ computes the PSNR on the entire training dataset, PSNR_{50} computes the metric on the selected 50 videos for comparison. As can be seen, although hypernetwork d_Ω is trained on huge datasets, it performs comparably with Single-INR. For RainbowJelly, it even outperforms Single-INR in SSIM metric and performs at par on SkyTimelapse. This indicates that d_Ω has learned to accurately generate INRs for complex spatio-temporal signals.

4.2 Comparing INR-V with SOTA video generation networks

Overview: Fig. 5 and Table 2 present qualitative and quantitative comparisons respectively between MocoGAN-HD Tian et al. (2021), DIGAN Yu et al. (2022), StyleGAN-V Skorokhodov et al. (2021), and INR-V. All models were trained from scratch. As we train the models on smaller datasets of $\sim$ 10,000 datapoints, MoCoGAN-HD is trained on StyleGAN2-ADA Karras et al. (2020) backend. For each model, the best-performing checkpoint is selected for comparison.

Evaluation: As can be seen in Fig. 5, INR-V generates novel videos with coherent content and motion. MoCoGAN-HD fails to maintain the identity in a single video instance. For quantitative evaluation, we use the Frechet Video Distance (FVD) metric as implemented by StyleGAN-V. FVD_{16} is computed on 2048 videos of 16 frames sampled at a resolution of 128×128. As can be seen in Table 2, INR-V outperforms the existing networks on Moving-MNIST and RainbowJelly and performs comparably on the remaining datasets.

Enhancing INR-V's Visual Quality Enhancing image and video quality has been an area of extensive research Yang et al. (2021); Chu et al. (2020); Liang et al. (2022); Chadha et al. (2020) with many breakthroughs. We propose that video generation can be partitioned into two stages (1) generating coherent

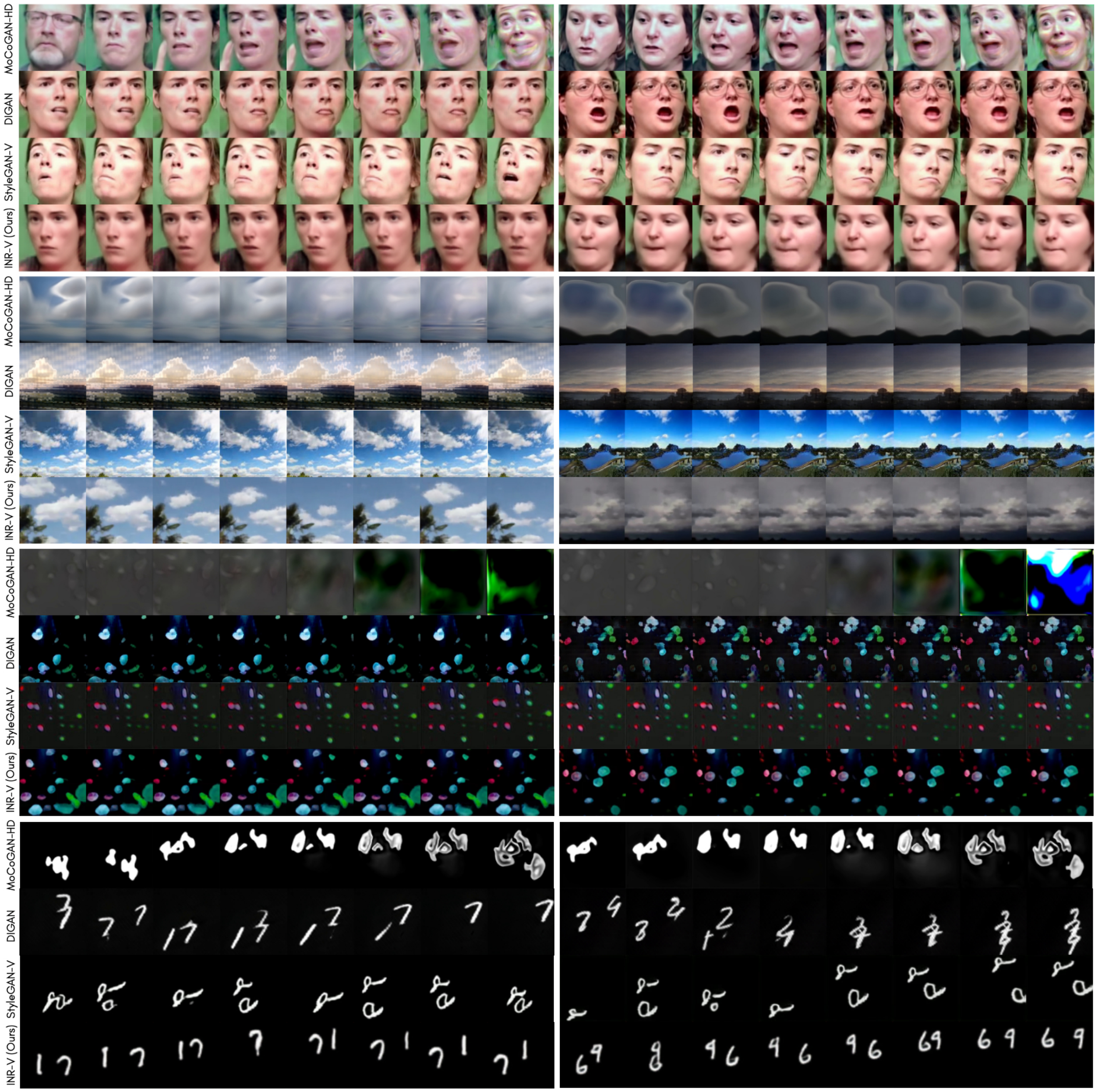

Figure 5: Examples of random videos generated on, from top to bottom, How2Sign-Faces Duarte et al. (2020), SkyTimeLapse Xiong et al. (2017), RainbowJelly Soomro et al. (2012), and Moving-MNIST Srivastava et al. (2015). For Moving-MNIST, every 2^{nd} frame of 20 frames long videos are shown. For other datasets, every 3^{rd} frame of 25 frames long generated are shown. Moving-MNIST and How2Sign-Faces are passed through VQVAE2 denoising network as described in Section 4.2

content and motion (2) enhancing the visual quality. Note that, in the present work, our effort has been on (1) to propose a novel continuous representation space for videos. Here, we demonstrate (2) by developing a simple denoising network using a standard VQVAE2 Razavi et al. (2019). We train VQVAE2 as a frame-by-frame denoising autoencoder making one minor change: Instead of reconstructing the given low-quality input, we use the high-quality frame for computing the error. The low-quality inputs are the intermediate video instances reconstructed by INR-V during training. We train denoising VQVAE2 on How2Sign-Faces and Moving-MNIST. Appendix Fig. 10 demonstrates the results of the denoising network on blurry instances

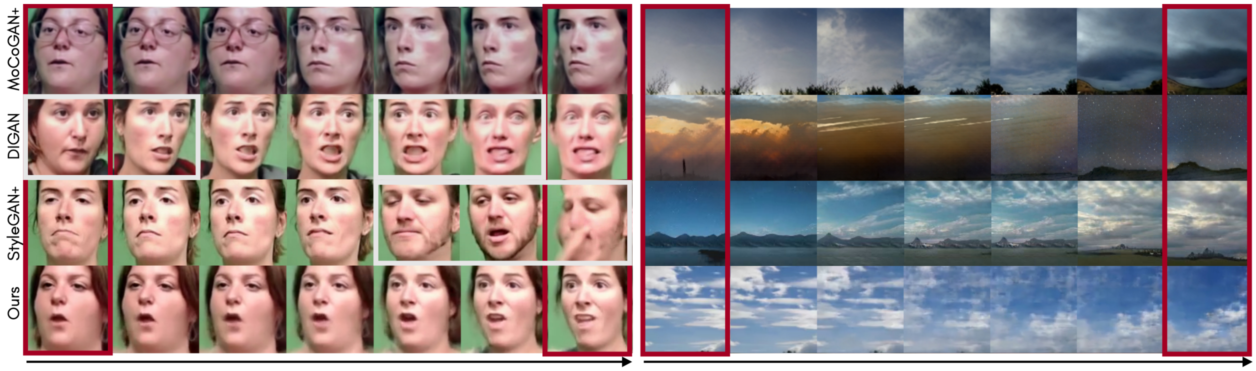

Figure 6: **Each cell displays 12^{th} frame of 25 frames long generated videos**. The videos demonstrate interpolation between two given videos (in red boxes) by traversing along a trajectory in the latent space connecting the latent points of the given videos. Here, MoCoGAN+ and StyleGAN+ denote MoCoGAN-HD and StyleGAN-V. White boxes indicate a sudden transition in content (e.g. identity) or motion (e.g. pose).

generated by INR-V. As can be seen from the quantitative metrics in Table. 2, using a simple additional denoising network improves the network's performance by $\sim 2\times$.

5 Applications of the continuous video space learned by INR-V

INR-V learns a continuous latent representation for videos allowing complex spatio-temporal video signals to be represented using a single latent point. In this section, we showcase the advantage of such a latent space through several demonstrated properties and comparisons. We also benchmark several tasks based on the inversion of 256 videos on How2Sign-Faces using full and incomplete video context.

5.1 Video Interpolation

MoCoGAN-HD	DIGAN	StyleGAN-V
100.00	89.43	95.24

Table 3: Video interpolation user study: % of times INR-V interpolation was preferred over existing models.

Given two videos V_1 and V_2, a continuous video space should be able to make a gradual transition between the two videos such that every point along the trajectory between the two (1) produces a meaningful video and (2) shares content and motion properties from V_1 and V_2. We demonstrate this property in Fig. 1 and Fig. 6 with Slerp interpolation. Each cell in Fig. 6 demonstrate the 12^{th} frame of the 25 frames long videos. As can be seen, INR-V observed a gradual change in motion (pose, mouth movements, expressions, cloud shift) and content (identity, visibility of sun). The interpolated videos are spatio-temporally coherent (best seen in videos added in the supplementary). Appendix Fig. 15 and Fig. 16 demonstrate the spatio-temporal transition on How2Sign-Faces and SkyTimelapse. As we represent an entire video in a single point in the continuous video space, interpolation is a natural operation that can be performed with INR-V.

Comparisons: Existing models have different motion and content codes. To interpolate videos, intermediate content codes were interpolated between two videos by Slerp interpolation. INR-V does not have separate motion and content vectors; thus, videos can be interpolated directly using given video's latent points. As shown in Fig. 6, INR-V has a gradual transition in motion and content. For How2Sign-Faces, StyleGAN-V abruptly changes motion (cell 5-7), and DIGAN abruptly switches identity (cell 1-2, cell 5-6). This effect is highlighted in white boxes. This is expected as both of these architectures operate in the image space, and thus a gradual spatio-temporal transition is harder to achieve. We performed a user study on 30 users to qualitatively evaluate the interpolation quality of INR-V and report the metrics in Table. 3. INR-V interpolation was randomly shown against either of the other three models. The users provided their preference on which interpolation looked smoother in terms of transition in content and motion. INR-V was preferred at least 85% more than all the SOTA video generation methods. This demonstrates the continuous

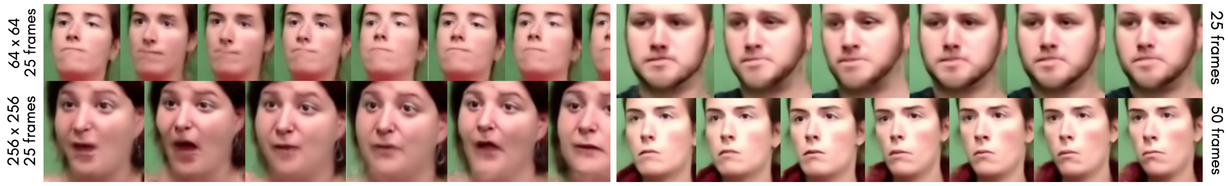

Figure 7: **INR-V direct inference on multiple resolutions and frame length**. INR-V trained on only 25 frames long 100×100 videos. Novel videos of multiple resolutions (64×64, 128×128, 256×256) and video length (50) are directly generated on the trained model without any architectural change or finetuning. The images are not upto scale, please refer Appendix Fig. 17 for scaled representation.

nature of the video space learned by INR-V. Please refer to the added supplementary videos for additional results and comparison.

5.2 Multi-Resolution and Multi-Length Inference

An underlying property of INRs is a continuous representation of the signal given as $f_\theta(h, w, t) \rightarrow \text{RGB}$. This enables the model to understand a continuous property of the signal making it agnostic to the dimension. Thus, one can infer a model on multiple resolutions and video length without changing the model's architecture and without any finetuning. In Fig. 7 we show INR-V trained on videos of only 100×100 resolution with 25 frames per video, generating random videos of multiple resolutions and lengths, maintaining the content and motion quality of the output. Additional qualitative results are added in the Appendix Fig. 17 and supplementary videos. Table 4 presents the FVD_{16} metrics on 2048 random videos generated at multiple resolutions on RainbowJelly. Qualitative results are added in the supplementary videos. As can be seen, the difference in FVD_{16} is statistically insignificant Unterthiner et al. (2018).

	128×128	256×256	360×360
INR-V (Ours)	**356.98**	290.43	342.14

Table 4: FVD_{16} metrics on random video generation at multi-resolution on INR-V. Training was done on only 100×100 dimensional videos of 25 frames. Inference was taken directly on multiple resolutions without any finetuning or architectural changes.

	$2\times$ 200×200		$3\times$ 300×300		$3.6\times$ 360×360	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	31.53	0.884	32.13	0.915	32.31	0.920
VideoINR	31.59	0.883	33.01	0.913	-	-
INR-V (Ours)	28.62	0.892	29.17	0.894	29.05	0.896

Table 5: Quantitative metrics on video superresolution using INR-V and SOTA superresolution networks on video instance seen at the time of training. INR-V was trained at 100×100 video resolution.

Additionally, we compare INR-V with existing SOTA superresolution techniques Chen et al. (2022) in Table 5 on RainbowJelly. 2048 videos are first randomly selected from the training dataset. A pretrained INR-V model trained on the entire dataset is then used for comparison. As can be seen, INR-V performs comparably with SOTA methods in the task of superresolution.

5.3 Video Inversion and its applications

Inversion has been widely adopted in many applications prominently for images. StyleGAN2 Karras et al. (2019) is extensively used for image inversion enabling many downstream image editing tasks such as changing the emotion, age, or gender of a given face. However, inverting a complete video was a complex task before needing the inversion of many motion and content codes. In INR-V, inverting a complex spatio temporal video signal can be achieved by optimizing a single latent code through a simple optimization objective:.

$$\underset{m_z}{\text{argmin}} \frac{1}{T}\frac{1}{W}\frac{1}{H}\sum_{T}^{t=1}\sum_{W}^{w=1}\sum_{H}^{h=1}(f_{\theta_z}(h, w, t) - RGB_s)^2 \quad \text{where} \quad \theta_z = h_\Omega(m_z) \tag{7}$$

where m_z is the latent point for a video instance V_z. Fig 8 shows the qualitative demonstration of INR-V inversion trained on How2Sign-Faces for a video outside of the training dataset $\mathcal{D}$.

Video Completion: Key categories of 'video synthesis' include future prediction (future frames prediction), completing the video between frames (frame interpolation), and predicting the missing part of the video (video inpainting). In INR-V, a video V_z that is represented by a single latent code m_z can be generated without any additional knowledge. Thus, all the above operations can be performed using a modified optimization operation based on Eqn 7 on the seen part of the video given as:

$$\underset{m_z}{\text{argmin}} \frac{1}{S}(f_{\theta_z}(h_s, w_s, t_s) - RGB_s)^2 \quad \text{and} \quad \theta_z = h_\Omega(m_z) \tag{8}$$

where S is the number of context points, h_s, w_s, and t_s are the context points of V_z seen at the time of optimization. With the optimized m_z, the full video can simply be generated back with INR-V. Fig. 8 demonstrates the results for the various operation on a video outside of $\mathcal{D}$ with ~ 2.5 minutes of optimization on a single 12 GB NVIDIA GTX 2080ti GPU. As can be seen, the network is able to regress to a latent corresponding to the given identity while preserving finer details like spectacles, mouth shape, pose, etc. In the case of 'Video Inpainting', the network understands the person's pose. For 'Frame Prediction', although the pose does not match the ground truth, the overall video is coherent. In 'Frame Interpolation', the model is able to generate a coherent context between two frames, including the pose, expressions, identity, and mouth movements. In 'Sparse Inpainting', we randomly set 25% of all the video pixels as the context points for optimization. Even with very sparse context, INR-V is able to regress to the correct specifications, including the finer content details, pose, and motion.

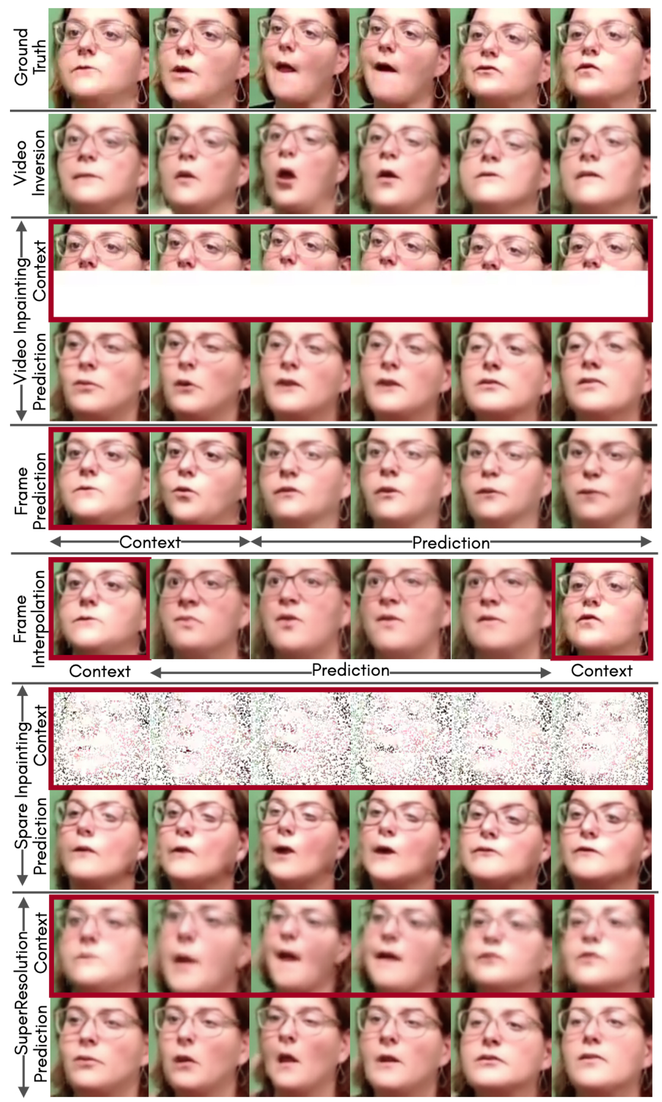

Figure 8: **Video Inversion and it's applications.** INR-V can be directly used for several tasks by simply inverting a video to its latent point based on the given context. Here, we demonstrate some qualitative results.

Video Superresolution through inversion: Video Superresolution is the task of enhancing the resolution of a given video. Recent works such as Chu et al. (2020); Liang et al. (2022); Sajjadi et al. (2018); Chadha et al. (2020); Wang et al. (2019); Chen et al. (2022) have made significant progress in video superresolution, showcasing 4× enhancement. INR-V can directly superresolve seen video instances as showcased in Table. 5. For unseen instances, combining the capability of video inversion and multi-resolution video generation, INR-V can superresolve a given video V_z of a lower resolution (say 32×32) simply as following: (1) Invert V_z at the smaller resolution to obtain m_z. (2) Render V_z from m_z directly at a higher resolution (say 256×256). In Fig. 8, we demonstrate the qualitative results on a video outside the training dataset. Like the other inversion tasks, the video was optimized at 32×32 for ~ 2.5 minutes on a single 12 GB NVIDIA-GTX 2080ti GPU. The inverted video was then superresolved at a scale factor of 8× to 256×256. Please note that we do not solve the task of superresolution but rather show superresolution as a potential application of our work.

Quantitative Evaluation: To quantify the performance of INR-V, we prepare a comparison set by randomly sampling 256 videos outside of the training set. We compare against StyleGAN-V only on the task of video inversion. StyleGAN-V generates full frames at once thus needing full frame context for backpropogation. DIGAN is based on INRs and can invert incomplete frame context; thus, we compare INR-V with DIGAN for the rest of the tasks except Superresolution. For superresolution, we compare with SOTA techniques Bicubic and VideoINR superresolution at a scale factor of $4\times$ from 32×32 to 128×128.

Task	Method	GT-ID ↑	TL-ID ↑	TG-ID ↑	Context-L1 ↓	PSNR ↑	SSIM ↑	Cost ↓
Inv.	DIGAN	0.652	0.953	0.9599	45.08	19.59	0.653	$\sim$ 4.25
	Style-V	**0.804**	**0.985**	**0.998**	42.16	19.65	0.665	$\sim$ 3.25
	INR-V	0.770	0.950	0.950	**5.25**	**21.21**	**0.773**	$\sim$ **2.75**
Inp.	DIGAN	0.628	**0.960**	**0.969**	45.80	-	-	$\sim$ 4.25
	INR-V	**0.758**	0.948	0.939	**4.83**	-	-	$\sim$ **2.75**
Pre.	DIGAN	0.603	0.940	0.928	40.26	-	-	$\sim$ 4.25
	INR-V	**0.703**	**0.946**	**0.932**	**4.72**	-	-	$\sim$ **2.75**
Int.	DIGAN	0.653	0.925	**0.921**	48.66	-	-	$\sim$ 4.25
	INR-V	**0.702**	**0.928**	0.905	**7.46**	-	-	$\sim$ **2.75**
Spr.	DIGAN	0.718	0.961	0.967	46.24	19.74	0.671	$\sim$ 4.25
	INR-V	**0.768**	**0.968**	**0.974**	**5.29**	**22.35**	**0.774**	$\sim$ **2.75**
Sup. $4\times$	Bicubic	0.808	0.923	0.903	-	28.36	0.906	-
	VideoINR	**0.939**	**0.982**	**0.974**	-	**32.86**	**0.957**	-
	INR-V	0.734	0.911	0.903	4.92	21.94	0.742	$\sim$ **2.75**

Table 6: Comparison of INR-V on various video inversion tasks: Video Inversion (Inv.), Video Inpainting (Inp.), Frame Prediction (Pre.), Frame Interpolation (Int.), Sparse Interpolation (Spr.), and Superresolution (Sup.). Comparison set is made of 256 videos outside of the training dataset. Metrics used for evaluation is explained in Sec. 5.3. Cost denotes the time to optimize a single video instance in minutes.

PSNR and SSIM metrics are used when the generated video is expected to match the ground truth. There is no single correct prediction for tasks like 'Future Frame Prediction', 'Frame Interpolation', and 'Video Inpainting'. Thus, we adopt the following metrics (1) Temporally Locally (TL-ID) and Temporally Globally (TG-ID) Identity Preservation, (2) Context-L1, and (3) Ground Truth Identity (GT-ID) Match. TL-ID and TG-ID were proposed in 'Stitch it in Time Tzaban et al. (2022)'. They evaluate a video's identity consistency at a local and global level. For both metrics, a score of 1 would indicate that the method successfully maintains the identity consistency of the original video. Context-L1 computes the L1 error on the inverted videos at the given context points. An error of 0 would indicate that the inversion is perfect. The error is computed in a pixel range of $[0, 100]$. GT-ID measures the match in identity between the ground truth and the inverted video. DeepFace [3] face features are extracted for both the videos, and the cosine similarity is computed between the extracted features.

As can be seen, INR-V outperforms all the existing networks in most of the metrics on video inversion and the proposed inversion tasks, except 'Superresolution', indicating the advantage and robustness of a continuous video space. Moreover, the optimization time is $1.75\times$ lesser than DIGAN. For the task of Superresolution, VideoINR performs the best. However, INR-V performs comparably despite inverting at a lower resolution of 32×32. Moreover, VideoINR superresolves a complete video. An accurate inversion opens a vast number of possibilities, such as inverting an incomplete video (missing frames or missing pixels in each frame due to video corruption) and then superresolving at a higher resolution.

6 Conclusion

We present INR-V, a continuous video representation network. Unlike existing architectures that extend superior image generation networks for generating videos one frame at a time, we use implicit neural representations to parameterize videos as complete signals allowing a meta-network to encode it to a single latent point. Given enough examples, the meta-network learns a continuous video space as demonstrated through video interpolation and inversion tasks. INR-V generates diverse coherent videos outperforming many existing video generation networks. INR-V opens the door to a multitude of video-based tasks and removes the dependency on an image generator. For instance, we showcase video inversion using a simple optimization objective. We propose several downstream tasks and observe that INR-V outperforms the existing works on a majority of these tasks. We demonstrate the advantages and potential of a continuous video space and hope to encourage research in this direction.

[3] https://github.com/serengil/deepface

References

Aman Chadha, John Britto, and M. Mani Roja. iSeeBetter: Spatio-temporal video super-resolution using recurrent generative back-projection networks. *Computational Visual Media*, 6(3):307–317, jul 2020. doi: 10.1007/s41095-020-0175-7. URL https://doi.org/10.1007%2Fs41095-020-0175-7.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos, 2021. URL https://arxiv.org/abs/2110.13903.

Zeyuan Chen, Yinbo Chen, Jingwen Liu, Xingqian Xu, Vidit Goel, Zhangyang Wang, Humphrey Shi, and Xiaolong Wang. Videoinr: Learning video implicit neural representation for continuous space-time super-resolution, 2022. URL https://arxiv.org/abs/2206.04647.

Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork, 2021. URL https://arxiv.org/abs/2105.13016.

Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé , and Nils Thuerey. Learning temporal coherence via self-supervision for GAN-based video generation. *ACM Transactions on Graphics*, 39(4), aug 2020. doi: 10.1145/3386569.3392457. URL https://doi.org/10.1145%2F3386569.3392457.

Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets, 2019. URL https://arxiv.org/abs/1907.06571.

Zihan Ding, Xiao-Yang Liu, Miao Yin, and Linghe Kong. Tgan: Deep tensor generative adversarial nets for large image generation, 2019. URL https://arxiv.org/abs/1901.09953.

Amanda Duarte, Shruti Palaskar, Lucas Ventura, Deepti Ghadiyaram, Kenneth DeHaan, Florian Metze, Jordi Torres, and Xavier Giro-i Nieto. How2sign: A large-scale multimodal dataset for continuous american sign language, 2020. URL https://arxiv.org/abs/2008.08143.

Shangqian Gao, Feihu Huang, and Heng Huang. Model compression via hyper-structure network, 2021. URL https://openreview.net/forum?id=Oc-Aedbjq0.

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions, 2019. URL https://arxiv.org/abs/1904.06447.

Yudong Guo, Keyu Chen, Sen Liang, Yong-Jin Liu, Hujun Bao, and Juyong Zhang. Ad-nerf: Audio driven neural radiance fields for talking head synthesis, 2021. URL https://arxiv.org/abs/2103.11078.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016. URL https://arxiv.org/abs/1609.09106.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2019. URL https://arxiv.org/abs/1912.04958.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020. URL https://arxiv.org/abs/2006.06676.

Angus Lamb, Evgeny Saveliev, Yingzhen Li, Sebastian Tschiatschek, Camilla Longden, Simon Woodhead, José Miguel Hernández-Lobato, Richard E. Turner, Pashmina Cameron, and Cheng Zhang. Contextual hypernetworks for novel feature adaptation, 2021. URL https://arxiv.org/abs/2104.05860.

Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. Vrt: A video restoration transformer, 2022. URL https://arxiv.org/abs/2201.12288.

Pauline Luc, Aidan Clark, Sander Dieleman, Diego de Las Casas, Yotam Doron, Albin Cassirer, and Karen Simonyan. Transformation-based adversarial video prediction on large-scale data, 2020. URL https://arxiv.org/abs/2003.04035.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space, 2018. URL https://arxiv.org/abs/1812.03828.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. URL https://arxiv.org/abs/2003.08934.

Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Ccvs: Context-aware controllable video synthesis, 2021. URL https://arxiv.org/abs/2107.08037.

Phuoc Nguyen, Truyen Tran, Ky Le, Sunil Gupta, Santu Rana, Dang Nguyen, Trong Nguyen, Shannon Ryan, and Svetha Venkatesh. Fast conditional network compression using bayesian hypernetworks, 2022. URL https://arxiv.org/abs/2205.06404.

Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation, 2020. URL https://arxiv.org/abs/2003.05534.

Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution, 2017. URL https://arxiv.org/abs/1708.01692.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. URL https://arxiv.org/abs/1901.05103.

Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation, 2021. URL https://arxiv.org/abs/2108.06815.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. URL https://arxiv.org/abs/1906.00446.

Mehdi S. M. Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution, 2018. URL https://arxiv.org/abs/1801.04590.

Marcin Sendera, Marcin Przewięźlikowski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Hypershot: Few-shot learning by kernel hypernetworks, 2022. URL https://arxiv.org/abs/2203.11378.

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings, 2018. URL https://arxiv.org/abs/1812.01024.

Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/b5dc4e5d9b495d0196f61d45b26ef33e-Paper.pdf.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020. URL https://arxiv.org/abs/2006.09661.

Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering, 2021. URL https://arxiv.org/abs/2106.02634.

Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images, 2020. URL https://arxiv.org/abs/2011.12026.

Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2, 2021. URL https://arxiv.org/abs/2112.14683.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012. URL https://arxiv.org/abs/1212.0402.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms, 2015. URL https://arxiv.org/abs/1502.04681.

Joseph Suarez. Language modeling with recurrent highway hypernetworks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf.

Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N. Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis, 2021. URL https://arxiv.org/abs/2104.15069.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation, 2017. URL https://arxiv.org/abs/1707.04993.

Rotem Tzaban, Ron Mokady, Rinon Gal, Amit H. Bermano, and Daniel Cohen-Or. Stitch it in time: Gan-based facial editing of real videos, 2022. URL https://arxiv.org/abs/2201.08361.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric challenges, 2018. URL https://arxiv.org/abs/1812.01717.

Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks, 2019. URL https://arxiv.org/abs/1906.00695.

Jacob Walker, Ali Razavi, and Aäron van den Oord. Predicting video with vqvae, 2021. URL https://arxiv.org/abs/2103.01950.

Xintao Wang, Kelvin C. K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks, 2019. URL https://arxiv.org/abs/1905.02716.

Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks, 2017. URL https://arxiv.org/abs/1709.07592.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers, 2021. URL https://arxiv.org/abs/2104.10157.

Tao Yang, Peiran Ren, Xuansong Xie, and Lei Zhang. Gan prior embedded network for blind face restoration in the wild, 2021. URL https://arxiv.org/abs/2105.06070.

Shunyu Yao, RuiZhe Zhong, Yichao Yan, Guangtao Zhai, and Xiaokang Yang. Dfa-nerf: Personalized talking head generation via disentangled face attributes neural rendering, 2022. URL https://arxiv.org/abs/2201.00791.

Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks, 2022. URL https://arxiv.org/abs/2202.10571.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search, 2018. URL https://arxiv.org/abs/1810.05749.

Youjian Zhang, Chaoyue Wang, and Dacheng Tao. Video frame interpolation without temporal priors, 2021. URL https://arxiv.org/abs/2112.01161.

Dominic Zhao, Seijin Kobayashi, João Sacramento, and Johannes von Oswald. Meta-learning via hypernetworks. 12 2020.

A Appendix

A.1 Ablation

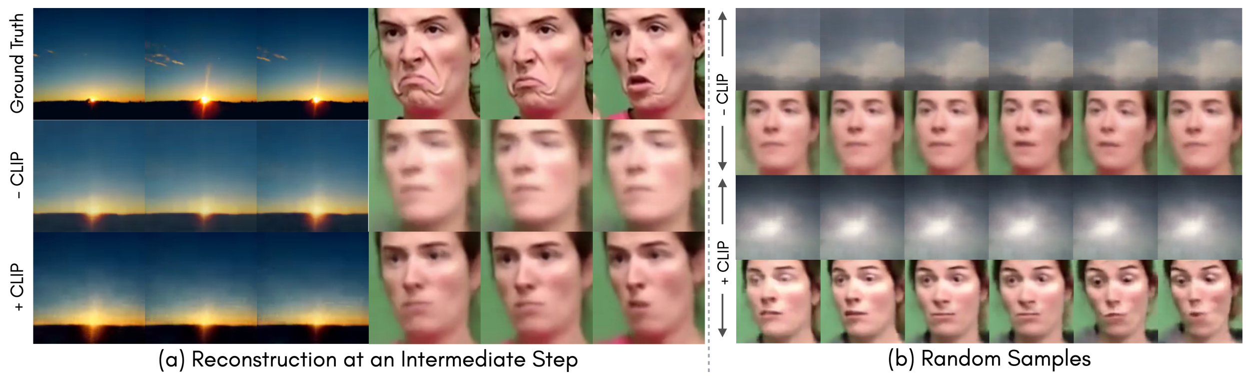

Figure 9: Qualitative results of CLIP regularization. Results are taken from an intermediate step after 30 hours of training on 2 NVIDIA GTX 2080ti GPUs. (a) Reconstruction of videos seen during training. CLIP regularization enables the meta-network to model the INRs with finer details. (b) Videos generated by randomly sampling. CLIP regularization improves the quality of the sampled videos and encourages variation in the implicit representations.

Figure 10: Denoising VQVAE2 reconstructions to enhance the visual quality of relatively blurry videos generated by INR-V. Please refer Sec. 4.2.

In this section, we compare the training time and the performance of INR-V (1) with/without CLIP regularization and (2) with/without 'progressive training'. Fig. 9 presents the qualitative results on INR-V with and without clip regularization after 30 hours of training on 2 NVIDIA GTX 2080ti GPUs. The reconstruction quality is much worse without CLIP. This is expected as Video-CLIP (see Fig. 4) assigns semantic meaning to the initialized codes for each video instance. Thus, the optimization is not completely from scratch. For random generations, we already see a motion emerging with expressive faces. In our observations, models trained with clip regularization in a progressive manner train faster. For instance, in RainbowJelly, the model trained progressively using CLIP regularization reached an FVD score of 440.98 in just 66 hours ($\sim$ 2.75 days). The model trained with CLIP regularization but without progressive training could reach an FVD of 526.15 after 76 hours ($\sim$ 3.25) of training. Without CLIP regularization, the model reached an FVD of 580.68 and 817.22 after 115 hours ($\sim$ 4.5 days) and 140 hours ($\sim$ 5.75 days) of training respectively. That is, the model without progressive training or CLIP regularization took the longest to train. The model reached an FVD_{16} of 356.98 after $\sim$ 6 days of training on the same setup with CLIP regularization and progressive training. The behavior was common across the datasets. However, SkyTimelapse made of only 1803 datapoints did not have enough datapoints for progressive training. Thus, the model was trained at once on all the datapoints. The final reported score of 153.43 was achieved using CLIP regularization.

A.2 INR-V Implementation Details

The implicit neural representation f_θ is an MLP with three 256-dimensional hidden layers. Each hidden layer is passed through ReLU activations. The hypernetwork d_Ω is a set of MLPs. Each MLP predicts the weights for a single hidden layer and the output layer of f_θ. Each MLP has three 256-dimensional hidden layers. CLIP embeddings are 512-dimensional vectors, Video-CLIP encodes the CLIP embeddings of each frame through three 512 dimensional, GRU layers. As shown in Fig. 4, Video-CLIP produces 512-dimensional video-level embedding g_n. c_n is a 512-dimensional context vector that is regressed in an auto-decoding fashion during training. ϕ is made of 3-hidden layers that takes a 1024-dimensional vector as input (concatenation of g_n and c_n) and produces m_n, a 128-dimensional instance code of V_n, as the input for d_Ω. The input to f_θ is a periodic positional encoding of $(\{h\}_{h=1}^{H}, \{w\}_{w=1}^{W}, \{t\}_{t=1}^{T})$ as implemented in Sitzmann et al. (2021). Adam optimizer is used with a learning rate of $1e-4$ during training and $1e-2$ during inversion tasks. No scheduler is used. Progressive training is done at a power of 10 where i^{th} stage is made of $\min(10^i, N)$ examples. $i = 0 \ldots \mathcal{K}$ such that $10^{\mathcal{K}+1} < N+1$, where N is the total number of training samples.Each stage except the last stage is trained until the reconstruction error reaches a threshold of $1e-3$.

A.3 Discussion

Limitations. Although INR-V has learned a powerful video space demonstrating several intriguing properties, the videos generated by the model are sometimes blurry. This is prominent when moving away to unseen points in the video space far from the seen instances. Fig. 10 demonstrates the enhancement on one such blurry sample. This is done by training a standard VQVAE2 network in a denoising fashion (please refer to Sec. 4.2). However, the entire process is broken into generating a relatively lower quality output and relying on a second network to improve its quality. A single end-to-end network capable of retaining the demonstrated powerful properties while generating high-quality videos is a potential future work.

Another limitation of INR-V is infinitely long video generation. Although coupling the content and time into a single latent has clear advantages, it removes the network's ability to leverage the temporal dimension separately and find infinitely long temporally coherent paths in the image space. This can be tackled by training INR-V to encode video segments of multiple lengths in a single space (1 to 50 or more frames long video segments). A temporally and semantically coherent trajectory between these video segments can then be learned. Such a generation technique would directly leverage video segments and potentially remove repetitions in the long videos. We believe that leveraging a video space for generating infinitely long videos at multiple resolutions presents an interesting and exciting direction for future research.

Broader Impact. The potential negative impact of our work is similar to existing image-based and video-based GANs: creating "photorealistic-deepfakes" and using them for malicious purposes. Our simple training strategy makes it easier to train a model which produces realistic-looking videos. However, this is partly addressed for the following reasons: (1) Even though our network produces diverse novel videos, the perceptual quality of our generated videos falls short of the existing state-of-the-art image-based generators that produce high-resolution images. (2) The availability of high-quality video datasets limits the intended malicious use of this codebase. Despite these limitations, we believe that the potential of our work far outweighs its limitations. A continuous video representation space offers tremendous applications in areas requiring video prediction, interpolation, and conditional video generation. E.g. pedestrian trajectory prediction is an important area of research for self-driving cars. Pedestrian trajectory prediction through future frame generation can serve to reduce accidents in fully-autonomous vehicles in the future. Similarly, conditional video generation can be used for synthesizing novel sign language videos that can be integrated into schools and universities to encourage and enable hard-of-hearing students.

A.4 Additional Qualitative Results

We encourage our readers to view the supplementary video results of INR-V. Fig. 11 presents the real video instances in the training set. Fig. 12 and Fig. 13 presents qualitative results on the reconstruction of video instances from different training datasets. Fig 14 presents random videos generated by INR-V on different datasets. Fig 15 and Fig. 16 present spatio-temporal view of video interpolations on How2Sign-Faces and

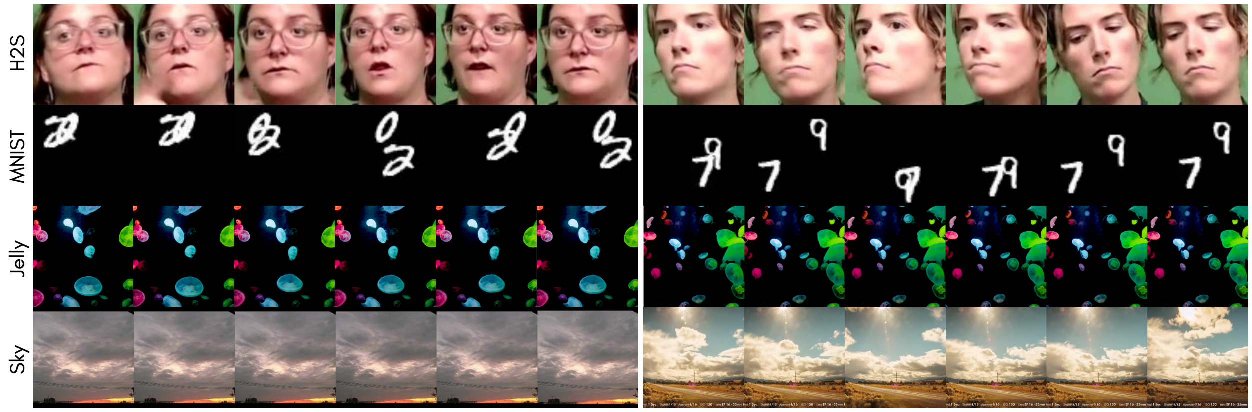

Figure 11: Examples of real videos instances of How2Sign-Faces (H2S) Duarte et al. (2020), Moving-MNIST (MNIST) Srivastava et al. (2015), RainbowJelly (Jelly), and SkyTimelapse (Sky) Xiong et al. (2017) datasets.

SkyTimelapse respectively. Fig. 17 presents the random generation of INR-V on multiple resolutions starting from 32×32 to 256×256 jumping a scale factor of $8\times$. The visualization is up to scale, and one can see the scale jump. INR-V can also be inferred at multiple frame rates. The supplementary videos include inferences at 50 frames. Fig. 18 - Fig. 23 present the qualitative results and comparisons on the proposed inversion tasks. Additional results on several inversion tasks can also be found in the supplementary videos.

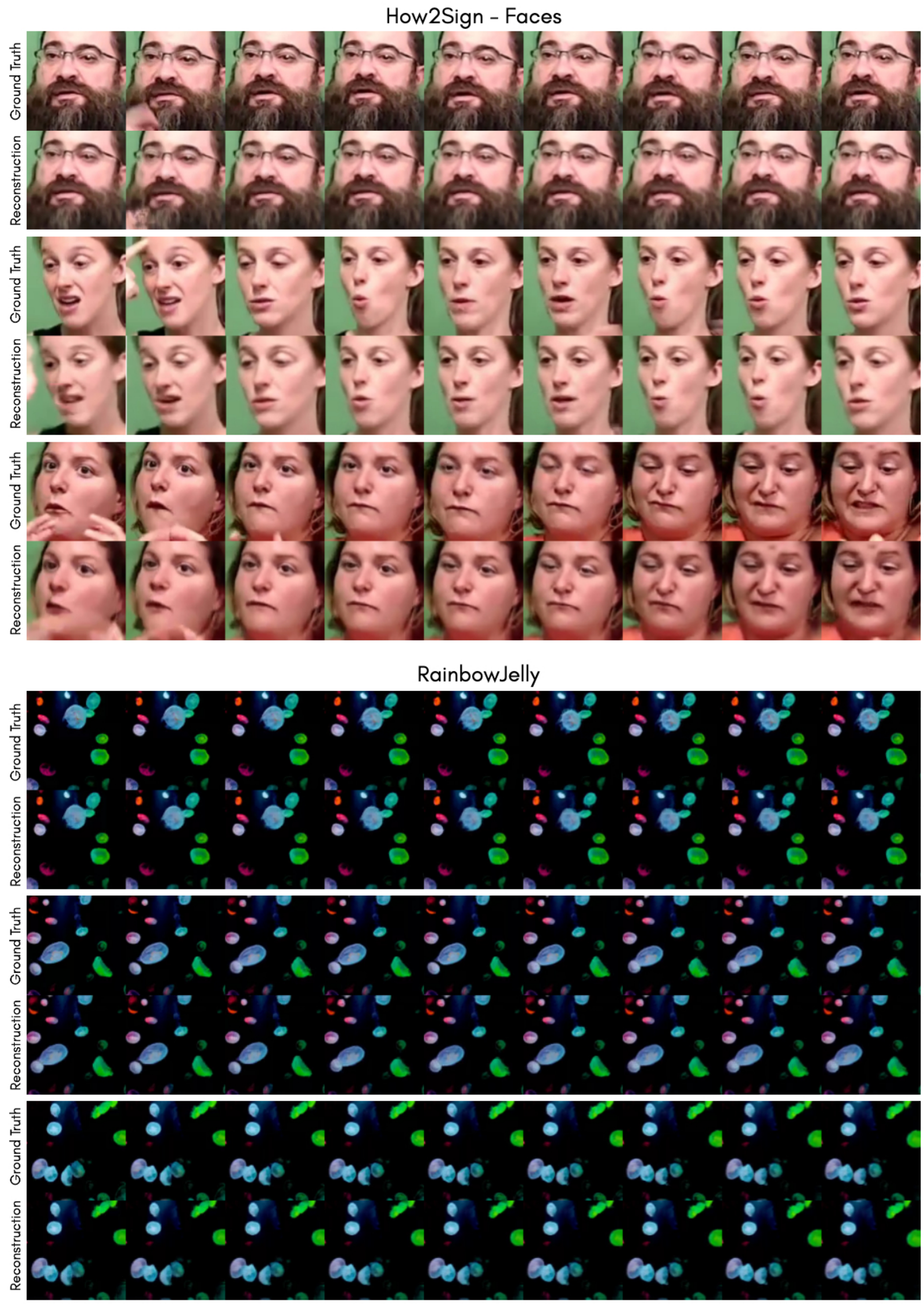

Figure 12: Examples of video instances in the training set reconstructed by INR-V.

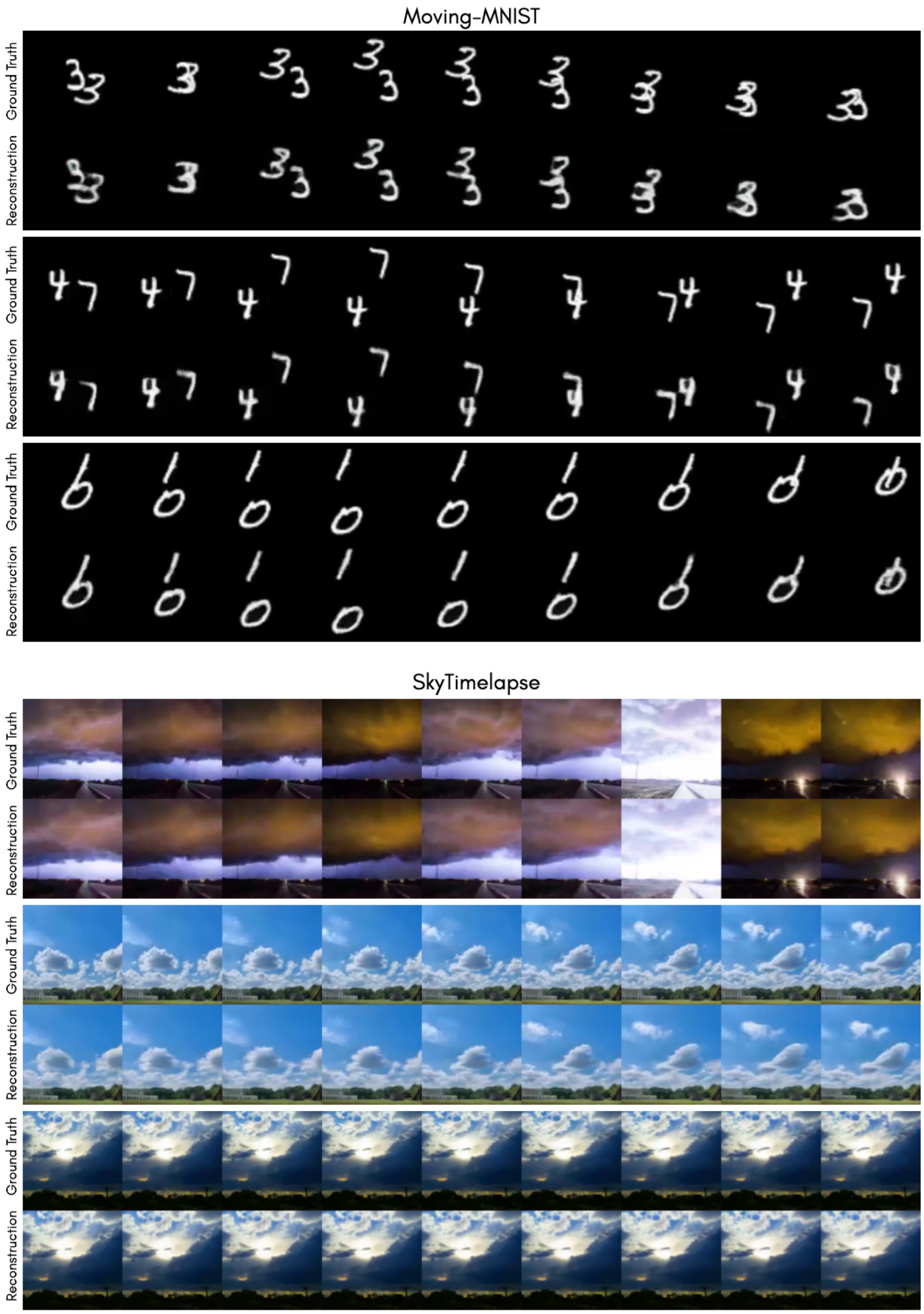

Figure 13: Examples of video instances in the training set reconstructed by INR-V.

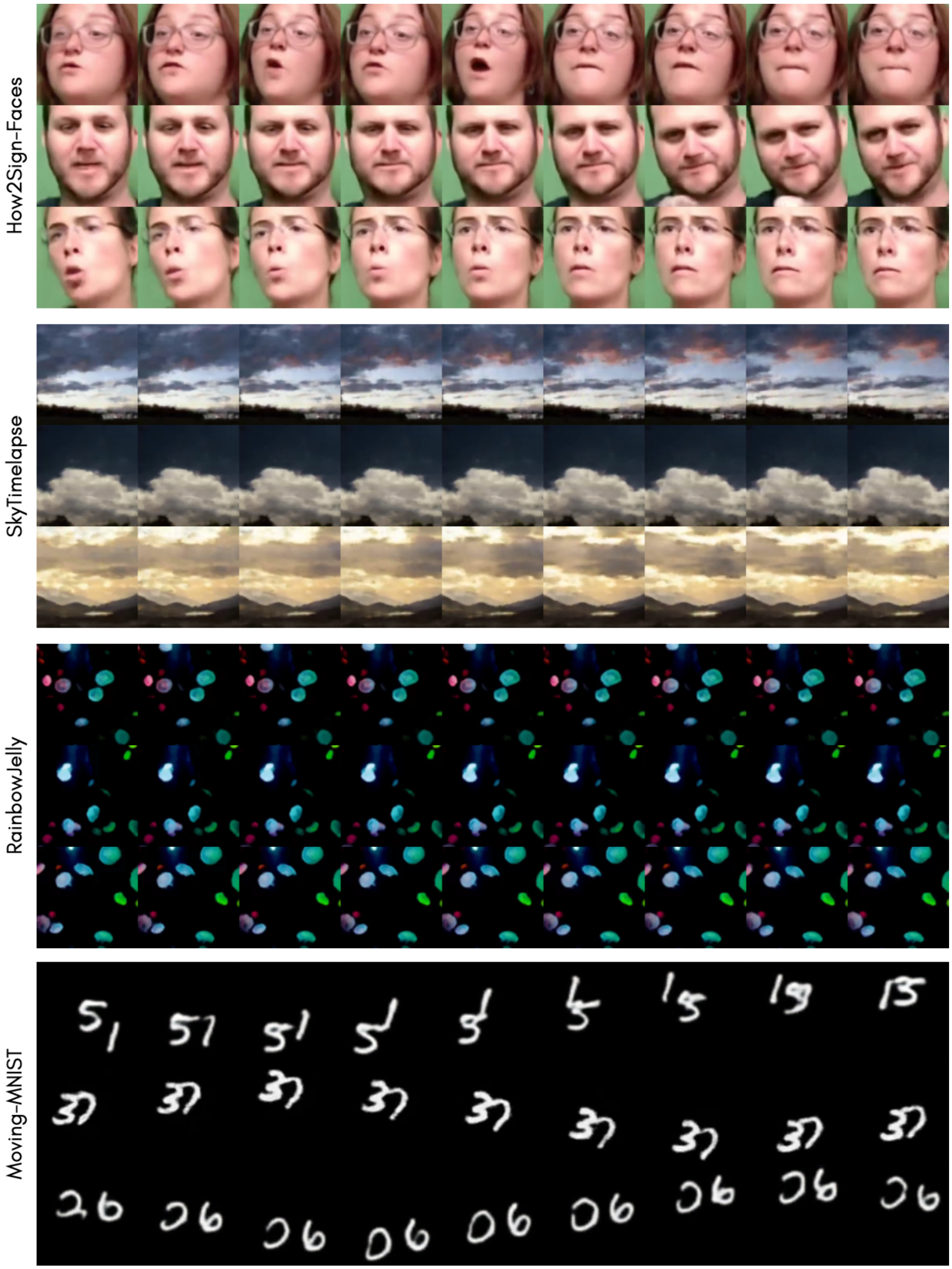

Figure 14: Examples of random videos generated by INR-V.

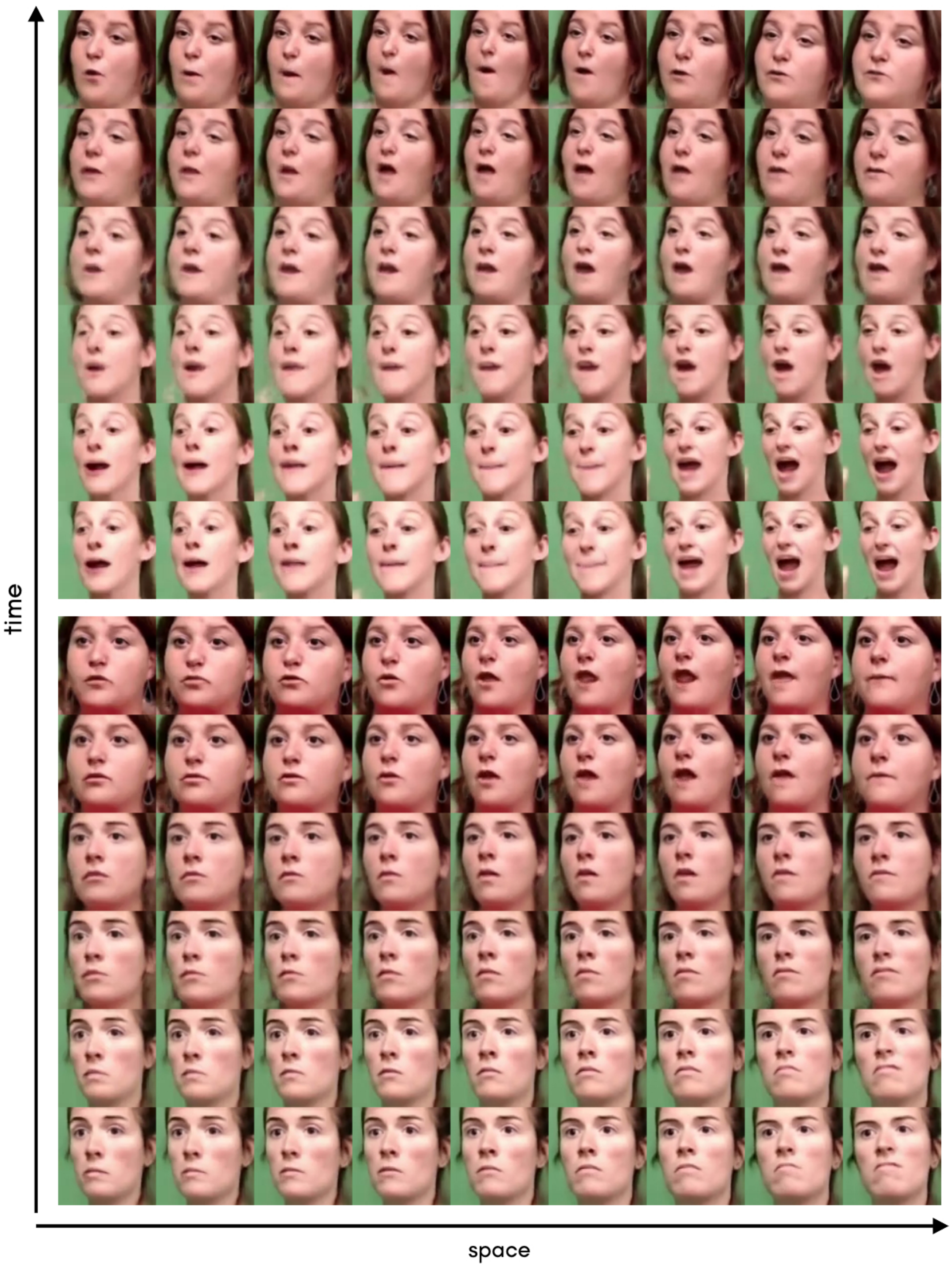

Figure 15: Examples of video interpolation in INR-V on How2Sign-Faces. Two latent points are sampled from the training dataset. Intermediate videos are then generated by sampling intermediate latent points using Slerp interpolation technique. We urge the readers to view the supplementary videos for best experience.

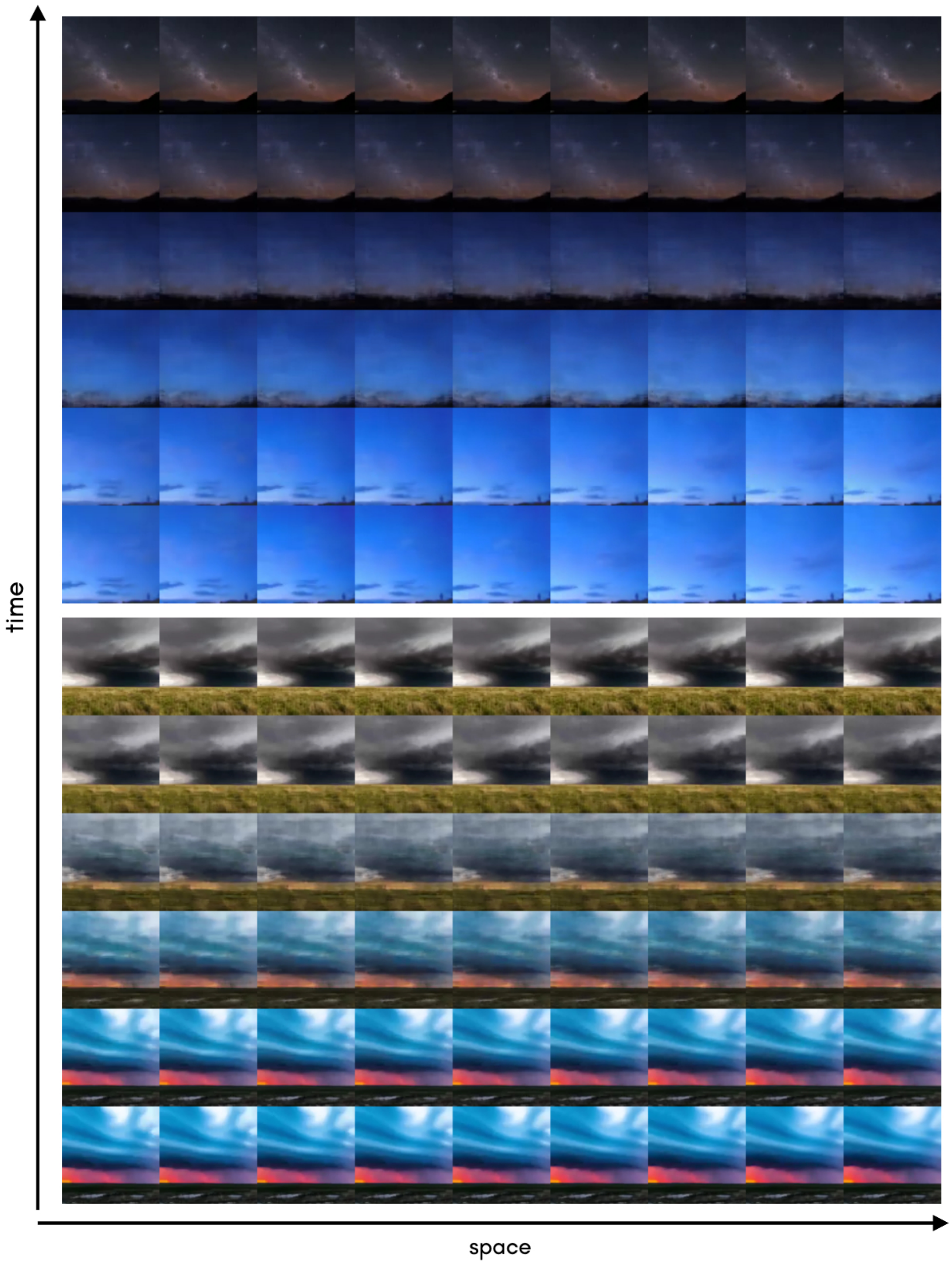

Figure 16: Examples of video interpolation in INR-V on SkyTimelapse. Two latent points are sampled from the training dataset. Intermediate videos are then generated by sampling intermediate latent points using Slerp interpolation technique. We urge the readers to view the supplementary videos for best experience.

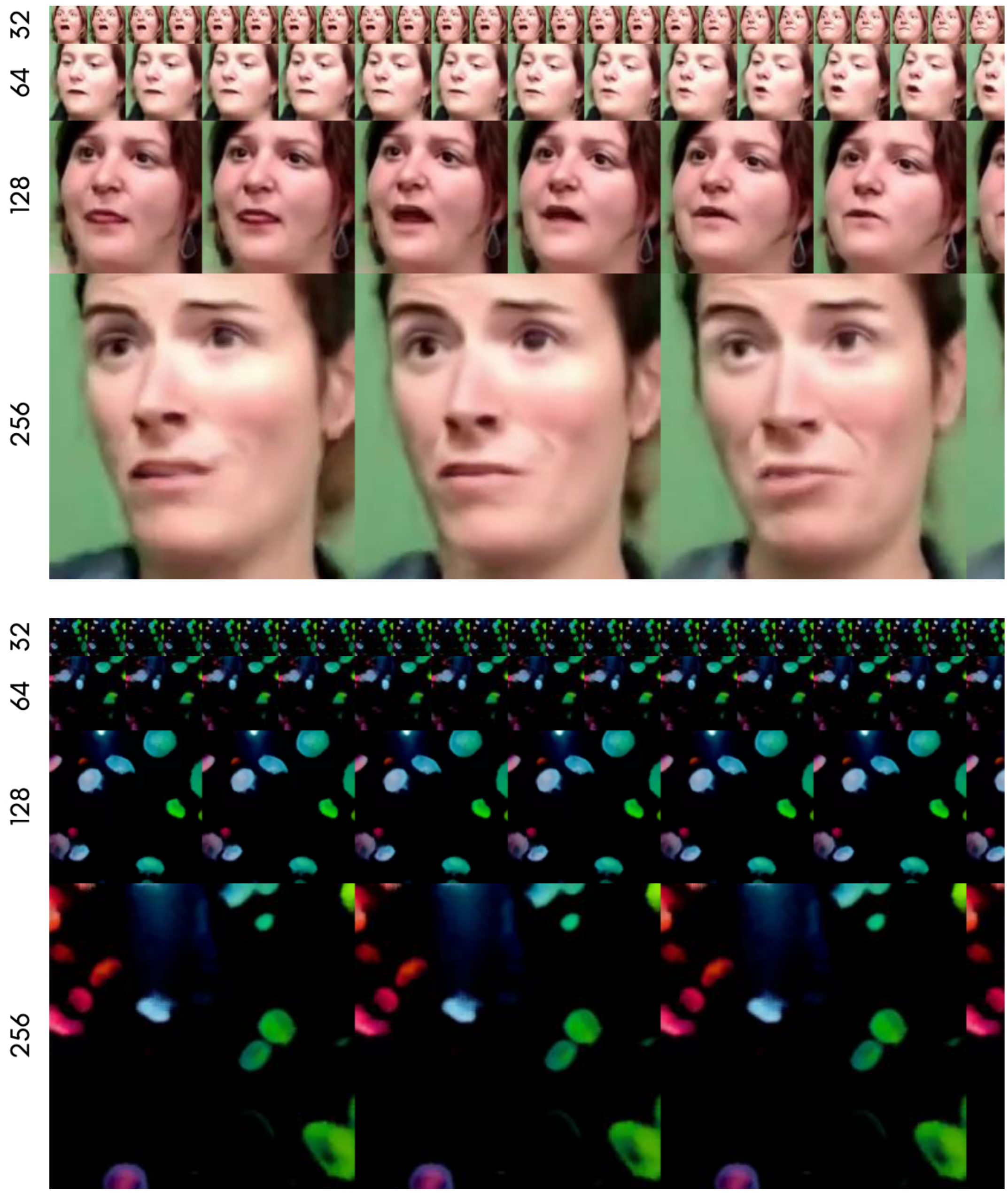

Figure 17: Examples of random videos generated by INR-V at multiple resolutions of 32×32, 64×64, 128×128, and 256×256 on How2Sign-Faces (top) and RainbowJelly (bottom). The videos are 25 frames long each. The videos are upto scale. INR-V was trained on videos of only 100×100 resolution. Please refer the supplementary videos for better experience and additional results on 50 frames long video generation.

Figure 18: Comparison of video inversion. Red boxes highlight the differences and matches between the ground truth (GT) and the various methods. To note, INR-V is able to preserve the finer mouth movements well.

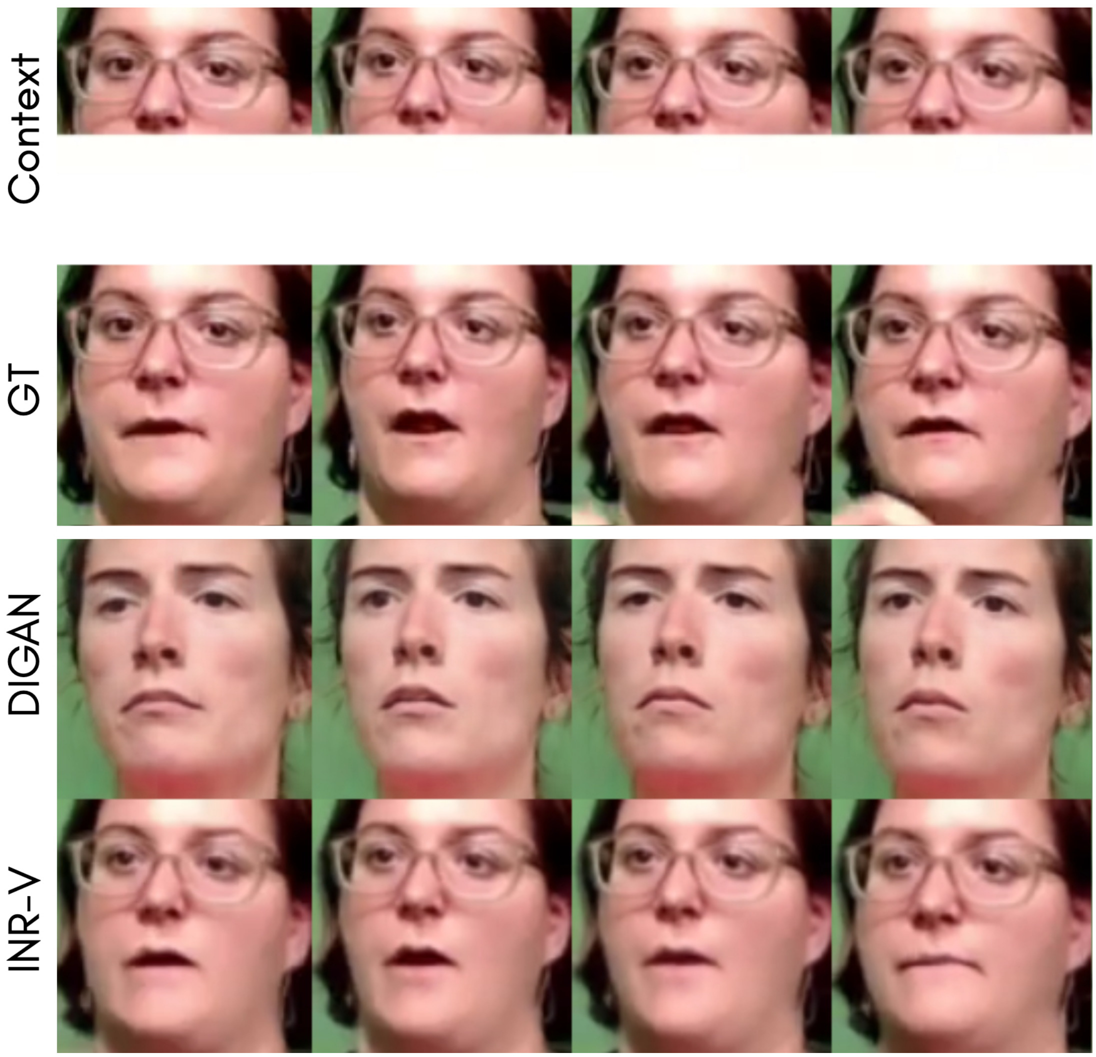

Figure 19: Comparison of half-context inversion in an inpainting setting. At the time of optimization, the model only sees the top half of the video. It then generates the full video back. There can be multiple correct predictions, we showcase one such prediction.

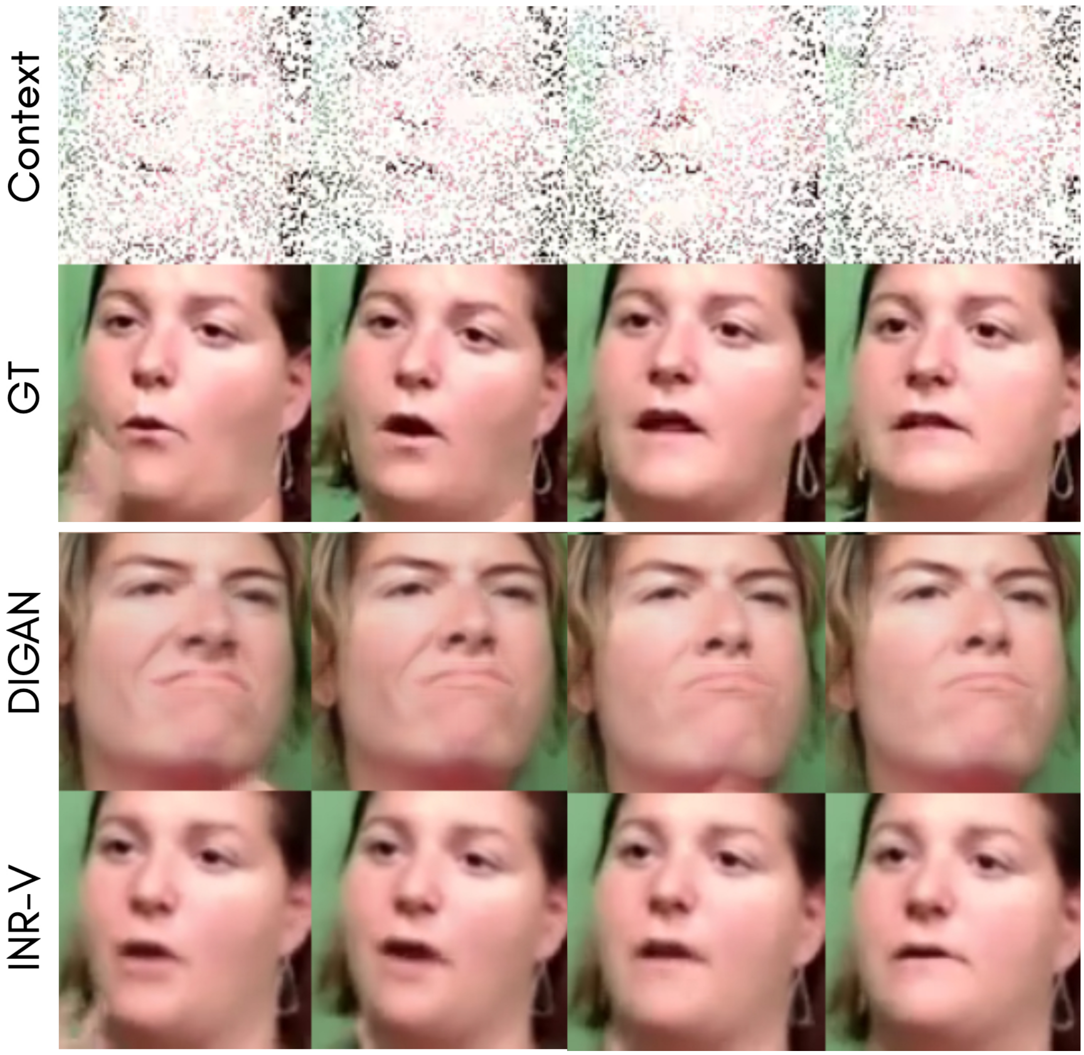

Figure 20: Comparison of half-context inversion in a sparse context setting. At the time of optimization, the model only sees 25% of the full video. INR-V preserve the identity including finer content details like earrings. It also preserves motion like pose and mouth movements.

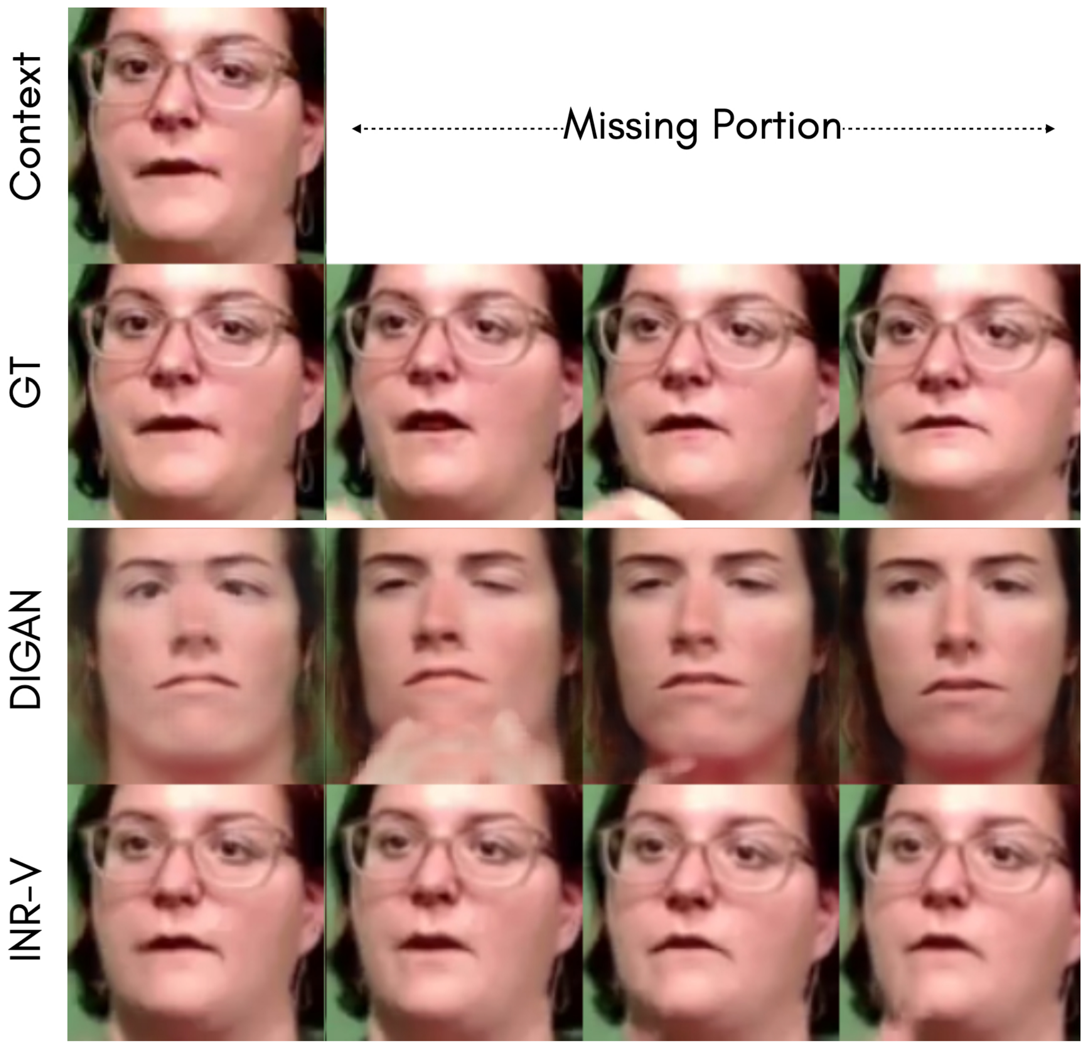

Figure 21: Comparison of half-context inversion in a future frame prediction setting. At the time of optimization, the model only sees the first 4 frames of the video. There can be multiple correct predictions given the identity is preserved across the video. We show one such example.

Figure 22: Comparison of half-context inversion in a frame interpolation setting. At the time of optimization, the model only sees the first and last frames of the video. As can be seen, the first and the last frame generated by INR-V match the context (pose, identity, mouth movements), whereas the intermediate frames are very different from the ground truth.

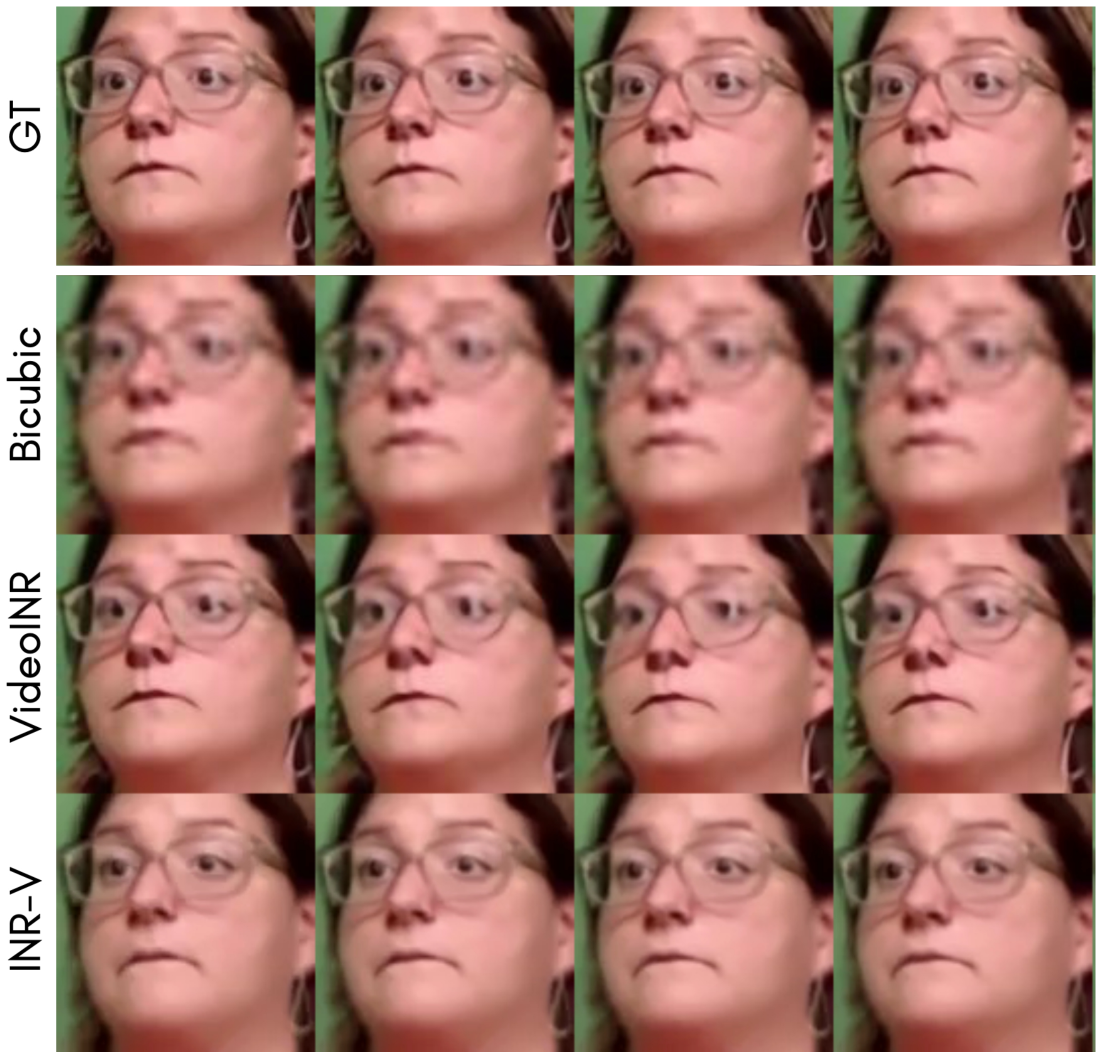

Figure 23: Comparison of video superresolution. A video of 32×32 is given as input to INR-V for optimization. Once the video is optimized, INR-V regenerates the video at a higher resolution of 128×128. VideoINR and Bicubic directly see the 32×32 video and superresolves it to 128×128. Here, INR-V is not influenced by the glaze on the spectacles and superresolves to a higher dimension closer to the ground truth.