

Supplementary Material for “HyperNVD: Accelerating Neural Video Decomposition via Hypernetworks”

Maria Pilligua^{1,2,*} Danna Xue^{1,2,*†} Javier Vazquez-Corral^{1,2}
¹ Universitat Autònoma de Barcelona ² Computer Vision Center

In this supplementary material, we present more quantitative results on unseen videos in Section A, highlighting our model’s superior adaptability and efficiency for new videos. More ablation results about using the MAE encoder embedding are presented in Section B. We also explain the network architectures, loss functions, and training data we use in Section C. Finally, we showcase additional qualitative results demonstrating video layer decomposition and editing capabilities in Section D.

A. More results on unseen videos

We present more results of fine-tuning the NVD model from the 15-video metamodel on unseen videos. In this setting, the parameters generated by the MAE encoder and hypernet are used to initialize the NVD model. This latter model is updated during fine-tuning. In Table 1, we compare the results of training the NVD model from metamodel (*i.e.* our HyperNVD) versus training from scratch (*i.e.* Hashing-nvd [1]). All the models are trained for 80k iterations, each video takes around 2 hours. The results show consistent improvements on all the unseen videos.

Table 1. Reconstruction quality (PSNR) on unseen videos when fine-tuning from our 15-video metamodel versus training from scratch.

Methods	scooter-black	car-shadow	bmX-trees	scooter-gray	mallard-water	drift-straight	rihno	paragliding
From scratch	26.12	30.76	29.20	30.68	29.12	29.50	32.66	32.54
From metamodel	28.52	31.74	30.11	31.57	29.99	30.24	32.97	32.96
Improvements	+2.40	+0.98	+0.91	+0.89	+0.87	+0.74	+0.31	+0.42

To demonstrate the acceleration of our method, the training curve of different videos is shown in Figure 1. Our model gets the same PSNR as Hashing-nvd [1] (the current SOTA both for efficiency and quality, also our baseline) in 40 minutes faster on an NVIDIA RTX 3090 GPU, reflecting a 30% improvement in training time (2 hours in total).

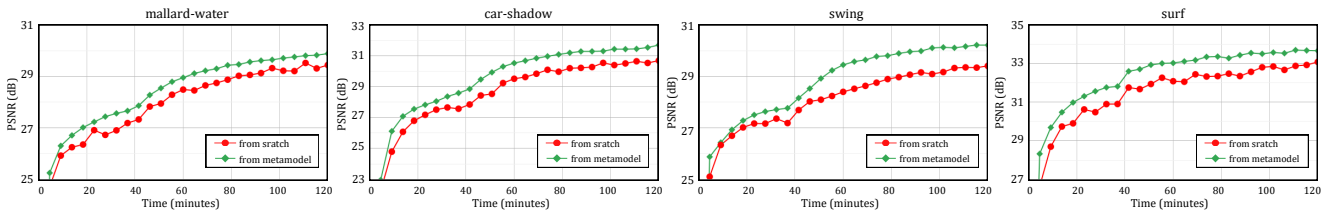


Figure 1. Fine-tuning the video decomposition model from our 15-video metamodel (HyperNVD) versus training from scratch (Hashing-nvd) on unseen videos.

*These authors contributed equally to this work.

†Corresponding author.

B. More ablation results

To further validate the generalization ability of the VideoMAE embeddings, we conduct an ablation on fine-tuning from the metamodel trained with 3 videos with different embeddings. Table 2 shows that VideoMAE still outperforms learned embeddings on unseen videos. Notably, VideoMAE Encoder does not add extra computational cost during fine-tuning, so it also improves the efficiency when adapting the HyperNVD to unseen videos.

Table 2. Ablations on different embeddings on unseen videos. Results are reported in PSNR.

Embedding	boat	bear	kite-walk
learned	32.62	30.98	31.88
MAE encoder	33.31	32.45	32.69

C. Implementation details

In this section, we present the detailed network architectures in Section C.1, our loss functions for training the network in Section C.2, and the videos we use for training the HyperNVD in Section C.3.

C.1. Architectures

The details of the architecture are presented in Table 3. As explained in the main paper, our network has three main components:

- (1) The MAE encoder reduces the output of VideoMAE from shape $(\#videos, 768, 1568)$ to a compressed embedding shape of $(\#videos, 768, 1)$ using 1-linear-layer encoder. To guarantee the compressed embedding maintains VideoMAE’s main features, the encoder is learned in combination with a decoder composed of a 1-linear layer that returns to the initial embedding shape $(\#videos, 768, 1568)$.
- (2) The main NVD model, whose architecture is shown in Figure 2 from the main paper, includes an alpha module and two Layer Modules (LM), each consisting of a mapping module, a texture module, and a residual module. Each of these modules is a 4-layer MLP (input layer, 2 hidden layers, output layer). The mapping module has an inner hidden dimension of 256 while all the rest have a hidden dimension of 64.
- (3) Our hypernet comprises 33 MLPs, each corresponding to a structural layer of the main net. Each of these MLPs has an input dimension of 768, a hidden layer of size 128, and outputs the predicted weights for its respective layer in the NVD model.

Table 3. Architecture details of our network. b stands for background, f stands for foreground.

Network	Module	Number of nets	Input dim	Output dim	Hidden layers	Hidden dim
MAE encoder	Encoder	1	$(\#videos, 768, 1568)$	$(\#videos, 768, 1)$	0	—
	Decoder	1	$(\#videos, 768, 1)$	$(\#videos, 768, 1568)$	0	—
NVD model	Mapping	2 (b&f)	$(\#points, 3)$	$(\#points, 2)$	3	256
	Texture	2 (b&f)	$(\#points, 2)$	$(\#points, 3)$	3	64
	Residual	2 (b&f)	$(\#points, 3)$	$(\#points, 1)$	3	64
	Alpha	1	$(\#points, 3)$	$(\#points, 1)$	3	64
Hypernet	1 Module	33	$(\#videos, 768)$	$(\#videos, \#weights)$	2	128

C.2. Training losses

The training objective of our Hypernet is the same as that of the previous neural video decomposition (NVD) models. Through these losses, we learn high-quality video reconstruction in a self-supervised manner. For this reason, we consider loss functions derived from previous research [1, 2]. We adhere to the same notation as outlined in the main paper. These losses include:

Reconstruction loss. The reconstruction loss serves as the primary objective for self-supervised learning, ensuring the quality of video reconstruction. This loss comprises two terms: one addressing the RGB values and the other focusing on gradients,

$$\mathcal{L}_{recon} = \lambda_r \mathcal{L}_{rgb} + \lambda_g \mathcal{L}_{grad}. \quad (1)$$

Here, λ_r and λ_g represent the weights, and are set to 5 and 1, respectively. The RGB term calculates the squared distance between the reconstructed color, \hat{c}^p , and the ground truth color, c^p , resulting in:

$$\mathcal{L}_{rgb} = \|\hat{c}^p - c^p\|_2^2, \quad (2)$$

and the gradient term \mathcal{L}_{grad} is given by:

$$\mathcal{L}_{grad} = \left\| \hat{d}_x - d_x \right\|_2^2 + \left\| \hat{d}_y - d_y \right\|_2^2, \quad (3)$$

where (\hat{d}_x, \hat{d}_y) and (d_x, d_y) are the spatial derivatives of the reconstructed image and the ground truth image, respectively.

Consistency loss. This loss ensures accurate motion representation supervised by optical flow. We expect corresponding pixels across the video to be mapped to the same point in the texture layer. For a pixel coordinate $p = (x, y, t)$, the corresponding point $p' = (x', y', t \pm 1)$, propagated using pre-computed forward or backward optical flow, is calculated as:

$$\begin{aligned} \mathcal{L}_{flow-p} = & \alpha^p \|\mathcal{M}_f(p) - \mathcal{M}_f(p')\| \\ & + (1 - \alpha^p) \|\mathcal{M}_b(p) - \mathcal{M}_b(p')\|, \end{aligned} \quad (4)$$

where \mathcal{M}_f and \mathcal{M}_b are the foreground and background mapping functions, which map p to the texture map coordinates (u_f^p, v_f^p) and (u_b^p, v_b^p) , respectively. α^p represents the predicted opacity value of p .

We also want the corresponding pixels to have the same alpha value:

$$\mathcal{L}_{flow-\alpha} = |\alpha^p - \alpha^{p'}|. \quad (5)$$

Then, the total consistency loss is given by:

$$\mathcal{L}_{flow} = w^p (\lambda_{fp} \mathcal{L}_{flow-p} + \lambda_{f\alpha} \mathcal{L}_{flow-\alpha}), \quad (6)$$

where we set $\lambda_{fp} = 0.01$ and $\lambda_{f\alpha} = 0.05$. w^p indicates whether the correspondence between the two points p and p' is consistent, based on the standard forward-backward flow consistency check. Specifically, $w^p = 1$ denotes consistent correspondence. To verify the reliability of the predicted optical flows, we perform a cycle mapping using the forward and backward flows. If the coordinate difference is smaller than 1 pixel, we consider it a reliable prediction and assign $w^p = 1$; otherwise, we set $w^p = 0$.

Sparsity loss. The sparsity loss is designed to prevent duplicate content across different texture layers.

$$\mathcal{L}_{sparsity} = \lambda_s \left\| (1 - \alpha^p) c_f^p \right\|_2^2, \quad (7)$$

where c_f^p represents the predicted color at position p for the foreground layer. We set $\lambda_s = 1$ in the experiments.

Residual consistency loss. This loss ensures smooth lighting conditions while preventing the residual estimator from capturing all color details. Formally, it is defined as:

$$\mathcal{L}_{res} = \lambda_{res-s} \mathcal{L}_{smooth} + \lambda_{res-r} R_{res}, \quad (8)$$

where $\lambda_{res-s} = 0.1$, $\lambda_{res-r} = 0.5$.

First, the loss ensures smooth lighting conditions by constraining the values at the same position on the texture coordinates to be positively correlated. For a small $k \times k$ patch at time t_1 and t_2 in the residuals r^p , we define the residual consistency loss using normalized cross-correlation as:

$$\mathcal{L}_{smooth} = \frac{(r_{t_1}^p - \mu_{r_{t_1}^p})(r_{t_2}^p - \mu_{r_{t_2}^p})}{\sigma_{r_{t_1}^p} \sigma_{r_{t_2}^p}} + \sigma_{r_{t_2}^p}^2, \quad (9)$$

where μ and σ are the mean and standard deviation of the corresponding patch, respectively. $\sigma_{rt_2}^2$ is a variance-smoothness term. We set k as 3.

Since only lighting changes are expected to be included in the residuals, a regularization term is applied to prevent the residual estimator \mathcal{R} from capturing color details:

$$R_{res} = \|\mathcal{R}(\cdot) - 1\|. \quad (10)$$

Alpha regularization loss. This additional regularization term ensures that the opacity map for each layer is clean and reliable, with lighting conditions properly embedded in each layer. It is defined using a BCE loss applied to the maximum values across all opacity maps:

$$\mathcal{L}_{\alpha reg} = \lambda_{\alpha reg} BCE(\max_{n \in \{0, \dots, N\}} \alpha_n), \quad (11)$$

where we choose $\lambda_{\alpha reg} = 0.1$.

Two extra losses are applied just in the initial iterations:

Rigidity loss. The rigidity loss is designed to prevent the texture layer from degrading into a simple color palette or developing a distorted layout. It enforces local rigidity in the mapping from pixel locations in the video to the texture layer. This loss is applied to both foreground and background mappings:

$$\mathcal{L}_{rigid} = \lambda_r (D(\mathcal{M}_b) + D(\mathcal{M}_f)), \quad (12)$$

where $\lambda_r = 0.001$, and the loss is applied only in the first 5,000 iterations. For a giving mapping \mathcal{M} , the rigidity term D is defined as a variant of symmetric Dirichlet term:

$$D(\mathcal{M}) = \|J_{\mathcal{M}}^T J_{\mathcal{M}}\|_F + \|(J_{\mathcal{M}}^T J_{\mathcal{M}})^{-1}\|_F, \quad (13)$$

where $J_{\mathcal{M}}$ is the Jacobian matrix of \mathcal{M} , it is given by:

$$J_{\mathcal{M}} = [\mathcal{M}(p_x) - \mathcal{M}(p) \quad \mathcal{M}(p_y) - \mathcal{M}(p)], \quad (14)$$

where $p_x = (x + 1, y, t)$, $p_y = (x, y + 1, t)$.

Alpha bootstrapping loss. This loss provides initial guidance for the alpha module to predict a reasonable opacity map for layer separation. We compute the binary cross entropy loss between the expected alpha value α^p of the pixel p and the corresponding value m^p in the reference mask.

$$\mathcal{L}_{\alpha boot} = \lambda_{\alpha} BCE(\alpha^p, m^p) \quad (15)$$

We set $\lambda_{\alpha} = 2$ and apply the alpha bootstrapping loss during the first 10,000 iterations.

Final loss. The addition of the above-mentioned losses gives the total loss:

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \mathcal{L}_{flow} + \mathcal{L}_{sparsity} + \mathcal{L}_{res} + \mathcal{L}_{\alpha reg} \quad (16)$$

C.3. Training dataset

In Table 4, we list the videos from the DAVIS dataset [3] used for training the models presented in Table 2 of the main paper. Please recap that Table 2 of the main paper demonstrate the capability of our HyperNVD to train with different numbers of videos without compromising largely the quality reconstruction.

D. More qualitative results

In this section, we present more visualization results of our approach to video decomposition and editing. A video demo is also provided in the supplementary materials to intuitively showcase the decomposition and editing results.

Figure 2 illustrates the video decomposition results. The three examples in the figure achieve a PSNR of approximately 30 dB and effectively differentiate between the foreground and background. Notably, the shadows on the rhino’s back are accurately decomposed into the residual map.

Table 4. Videos used for training our metamodel in Table 2 of the main paper.

Number	Videos
1	hike
3	hike , car-turn, blackswan
5	hike , car-turn, blackswan , bear, lucia
10	hike , car-turn, blackswan , bear, lucia , boat, bus, kite-walk, rollerblade, stroller
15	hike , car-turn, blackswan , bear, lucia , boat, bus, kite-walk, rollerblade, stroller , camel, car-roundabout, elephant, motorbike, train
	hike , car-turn, blackswan , bear, lucia , boat, bus, kite-walk, rollerblade, stroller , camel, car-roundabout, elephant, motorbike, train ,
30	bmx-bumps, bmx-trees, cow, drift-straight, flamingo, goat, horsejump-low, kite-surf, libby , paragliding, scooter-gray, soapbox, soccerball, surf, swing

Figure 3 demonstrates the video editing results. By modifying the foreground and background texture maps, we observe that elements added to the foreground —such as the bear’s scarf and the headphones and flowers on Lucia’s skirt— naturally adapt to the motion of the foreground. For these videos, where the backgrounds remain relatively stable, elements added to the background exhibit no significant motion and blend naturally.

Please note that in all current models —including ours— editing is restricted to non-transparent areas of the texture map due to the requirement of alpha-weighted compositing for blending layers. Edits applied to transparent regions will not appear in the final edited video. To add elements outside the object’s region that move along with the object, it is necessary to modify the opacity map. This limitation is inherent to video layer-based editing and is also present in previous methods. Despite this constraint, we believe that addressing such challenges could open avenues for future research and enable exciting new applications.

E. Limitation and discussion

In terms of limitations of our work, since our hypernetwork uses an MLP to predict parameters for each layer of the target network, resulting in a large parameter count, further structural optimization and techniques like Low-Rank Factorization [5] could help reduce the hypernetwork’s parameter size.

We have made some interesting observations. For example, using different initial masks does not significantly affect the final reconstruction quality, but finer masks guide the network to focus on object details. Therefore, in practical applications where precise editing of edges or related effects is required, using more defined masks is advisable. These can be improved with the latest segmentation methods, such as SAM2 [4].

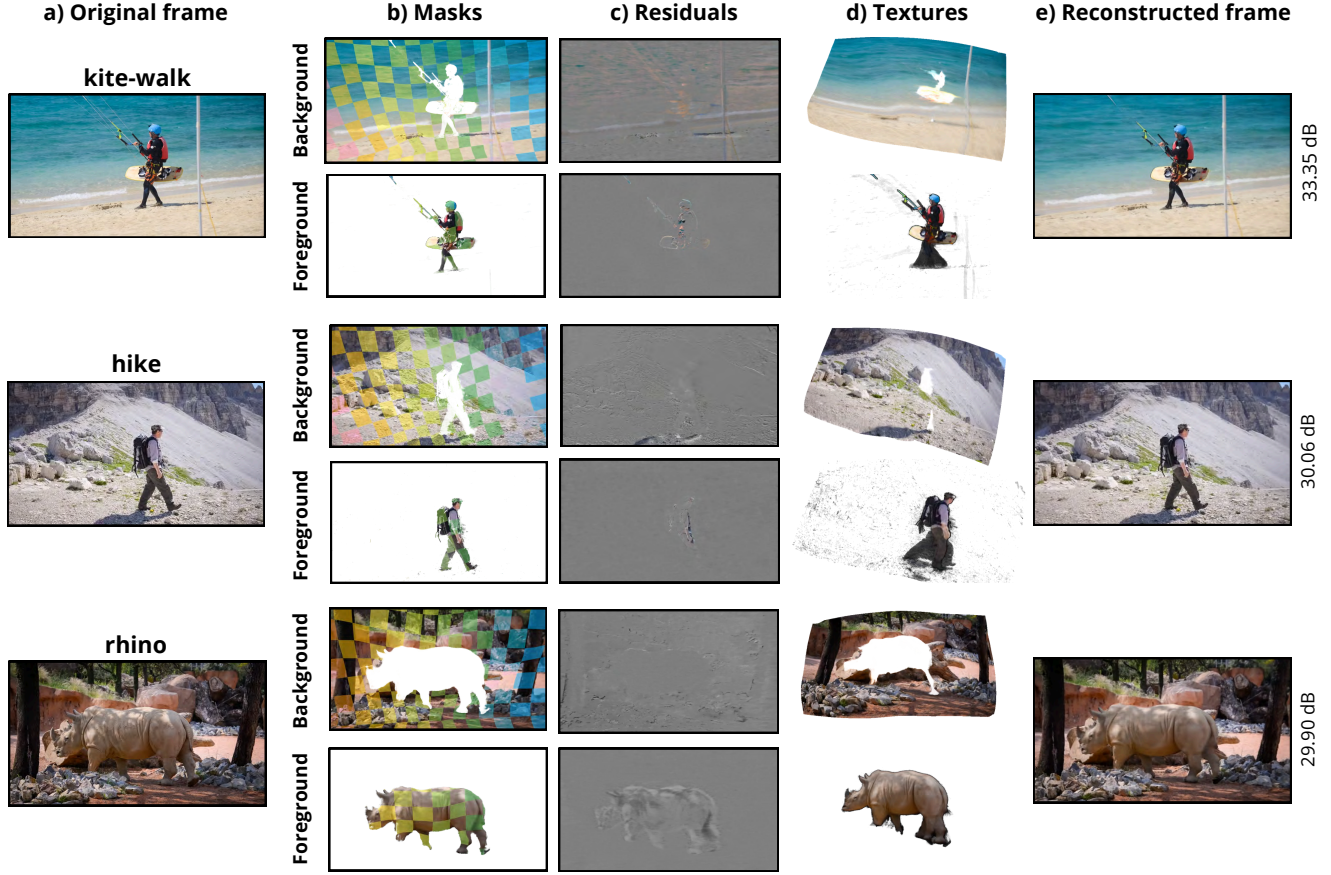


Figure 2. Qualitative results on the DAVIS dataset. Displayed are the a) original frames, b) predicted masks, c) residual maps, d) texture maps, e) reconstructed frames, and PSNR values for different videos. A color checkerboard overlay is applied to the masked areas to visualize texture transformations.

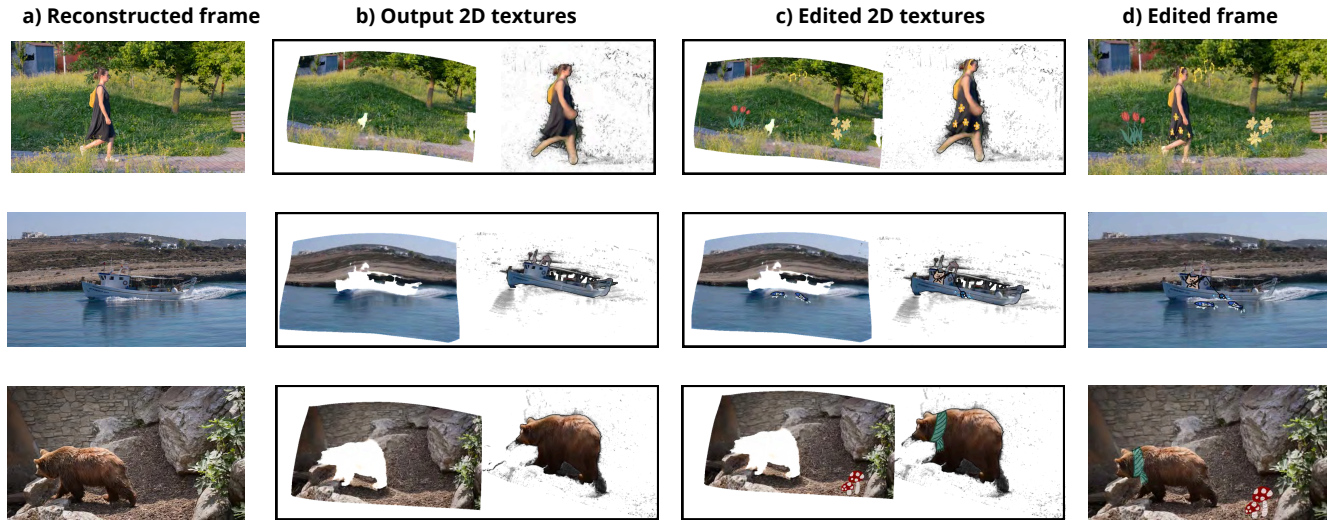


Figure 3. Edited video results. Displayed are the a) reconstructed frame with our HyperNVD, b) 2D texture maps, c) edited texture maps, and d) edited frame. We add doodles to both the foreground and background texture layers. The edited video maintains good visual quality and smooth transitions.

References

- [1] Cheng-Hung Chan, Cheng-Yang Yuan, Cheng Sun, and Hwann-Tzong Chen. Hashing neural video decomposition with multiplicative residuals in space-time. In *ICCV*, 2023. [1](#), [2](#)
- [2] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *ACM TOG*, 40(6):1–12, 2021. [2](#)
- [3] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. [4](#)
- [4] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. [5](#)
- [5] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *CVPR*, 2021. [5](#)