Is Overfitting Necessary for Implicit Video Representation?

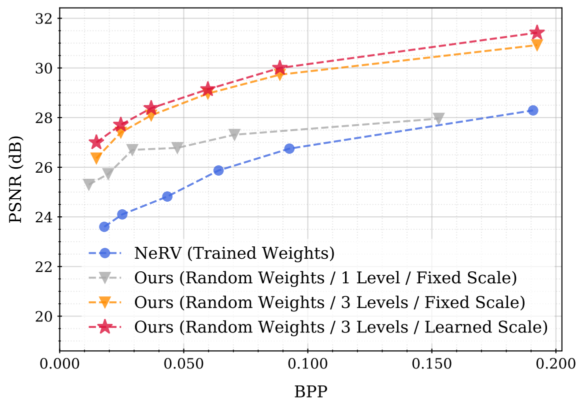
Hee Min Choi[*1] Hyoa Kang[*1] Dokwan Oh[1]

Abstract

Compact representation of multimedia signals using implicit neural representations (INRs) has advanced significantly over the past few years, and recent works address their applications to video. Existing studies on video INR have focused on network architecture design as all video information is contained within network parameters. Here, we propose a new paradigm in efficient INR for videos based on the idea of strong lottery ticket (SLT) hypothesis (Zhou et al., 2019), which demonstrates the possibility of finding an accurate subnetwork mask, called supermask, for a randomly initialized classification network without weight training. Specifically, we train multiple supermasks with a hierarchical structure for a randomly initialized image-wise video representation model without weight updates. Different from a previous approach employing hierarchical supermasks (Okoshi et al., 2022), a trainable scale parameter for each mask is used instead of multiplying by the same fixed scale for all levels. This simple modification widens the parameter search space to sufficiently explore various sparsity patterns, leading the proposed algorithm to find stronger subnetworks. Moreover, extensive experiments on popular UVG benchmark show that random subnetworks obtained from our framework achieve higher reconstruction and visual quality than fully trained models with similar encoding sizes. Our study is the first to demonstrate the existence of SLTs in video INR models and propose an efficient method for finding them.

1. Introduction

Implicit neural representation (INR) has become a promising tool to encode various multimedia signals in the last few



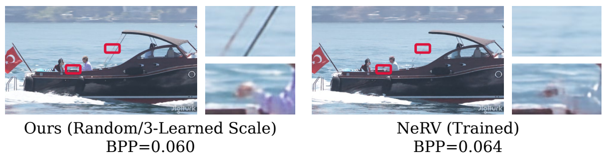


Figure 1. Reconstruction quality of decoded YachtRide video. The proposed random weight pruning method with learned scales achieves higher reconstruction quality than the method with fixed scale parameters as well as weight training of state-of-the-art implicit video representation models (NeRV) at various compression ratios. The reconstruction quality and compression ratio are measured by peak signal-to-noise-ratio (PSNR) and bits-per-pixel (BPP) metrics. As shown in the example frame, our random subnetworks encode fine details better (e.g., shroud and hand).

years. INR leverages on the possibility of using a neural network to fit a continuous function $f_\theta : \mathbb{R}^m \rightarrow \mathbb{R}^n$ that maps an m-dimensional input coordinates (e.g., pixel coordinates for an image) to an n-dimensional signal of interest (e.g., RGB color of the pixel) with network parameters θ. This continuous representation allows for compact and efficient encoding of target signals in various applications such as image (Sitzmann et al., 2020; Mehta et al., 2021; Chen et al., 2021b) and video representations (Chen et al., 2021a; Li et al., 2022; Zhang et al., 2022), 3D shapes (Genova et al., 2019; 2020), 3D scenes (Sitzmann et al., 2019; Jiang et al., 2020; Chabra et al., 2020; Peng et al., 2020) and 3D structures (Mildenhall et al., 2020; Niemeyer et al., 2020; Peng et al., 2021; Oechsle et al., 2021).

Due to its compact nature, one suitable application of INR is video representation. Indeed, encoding is saving video frames into neural network architecture and parameters

[*]Equal contribution [1]Samsung Advanced Institute of Technology, Samsung Electronics, Suwon, Republic of Korea. Correspondence to: Hee Min Choi <chm.choi@samsung.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

through model fitting. Once all video information is embedded within the model parameters, all image frames can be decoded by simple network inference, and the storing cost is proportional to the number of network parameters.

Most prior works on implicit video representation have focused on developing network architectures to improve parameter efficiency. Early methods (Sitzmann et al., 2020; Mehta et al., 2021) parameterize continuous mappings between spatial-temporal coordinates of each pixel and its RGB color value using deep networks. However, these pixel-wise representation approaches require enormous number of parameters to accurately represent target signals (Sitzmann et al., 2020; Mildenhall et al., 2020). Recently, NeRV (Chen et al., 2021a) proposes an image-wise implicit representation architecture that directly outputs each image frame given the associated frame index as input. It is shown that this reformulation boosts reconstruction quality with smaller amounts of parameters compared to the existing pixel-wise representations. In this work, we build a new video representation framework on the state-of-the-art image-wise representation model of NeRV.

Designing parameter-efficient implicit video representation can be also viewed as solving a model compression problem (Chen et al., 2021a). One of the most intriguing findings in network compression literature is strong lottery ticket hypothesis (Zhou et al., 2019), which demonstrates the feasibility of finding an accurate subnetwork in an over-parameterized random neural network for image classification tasks by training a weight mask called a supermask and properly scaling weights. Inspired by this, we propose a new image-wise implicit video representation framework that constructs a subnetwork in a randomly initialized model without weight training.

Our method is developed upon the algorithm of Okoshi et al. (2022) that improves the idea of supermask by training an overlay of multiple supermasks with a nested structure. Specifically, the edge-popup algorithm (Ramanujan et al., 2020) selects hierarchical subnetwork masks in a randomly initialized NeRV model (Chen et al., 2021a) without weight updates. Unlike Okoshi et al. (2022), a learnable scale parameter for each supermask is used instead of multiplying by the same fixed scale for all levels. We empirically demonstrate that this simple modification expands the parameter search space to sufficiently explore different types of sparsity patterns, leading the proposed algorithm to find better subnetworks (see Figure 1 and Table 1). We note that this is the first work to show the presence of strong lottery tickets in an implicit video representation model and introduce an efficient method for finding them.

Unlike existing implicit video representation methods (e.g., Sitzmann et al., 2020; Chen et al., 2021a; Li et al., 2022), it is not necessary to save model weights in our framework. In fact, encoded videos can be reconstructed from smaller amount of data. We only need to store the random seed used at train time, final *binary* supermasks and a few scale parameters to recover the compressed video.

Extensive experiments are performed on the popular UVG benchmark dataset (Mercat et al., 2020) for video compression. We demonstrate that the proposed algorithm improves the reconstruction quality of generated random subnetworks over that obtained from existing methods of finding supermasks in classification models. Moreover, the subnetworks selected by our framework perform better than fully trained models with similar encoding sizes, even though our method does not require weight training.

In summary, we made the following contributions:

- We propose a novel video representation framework that is based on strong lottery tickets in an INR model. This is the first work to show the existence of random subnetworks for non-classification tasks that output high resolution images (e.g., full HD images).
- We present a new algorithm to find strong subnetworks from randomly initialized video INR models without weight training.
- We show that video encoding quality of the random subnetworks obtained from our method outperforms that of counterpart networks selected by existing algorithms for finding supermasks in classification models as well as the performance of fully trained implicit video representation networks under various video sequences and compression ratio settings.

2. Related Work

Implicit Video Representation: Recently, there has been a growing interest in video INR, the idea of parametrizing video signals with neural networks. Most prior studies on implicit video representation have focused on developing efficient network architectures to improve encoding and decoding speed and/or compression ratio (e.g., Chen et al., 2021a; Li et al., 2022; Kim et al., 2022; Chen et al., 2022). The pioneering approaches (Sitzmann et al., 2020; Mildenhall et al., 2020) use pixel-wise representation that maps each spatial-temporal coordinate to its pixel value. A follow-up work, NeRV (Chen et al., 2021a), proposes an alternative image-wise representation model that directly outputs an image given the corresponding frame index as input. With this simple design idea, NeRV achieves state-of-the-art quality of video encoding. In this paper, we build a new framework for implicit video representation on this architecture.

Neural Network Pruning: The goal of neural network pruning is to simplify a large model by reducing the number of parameters without significant deterioration in its

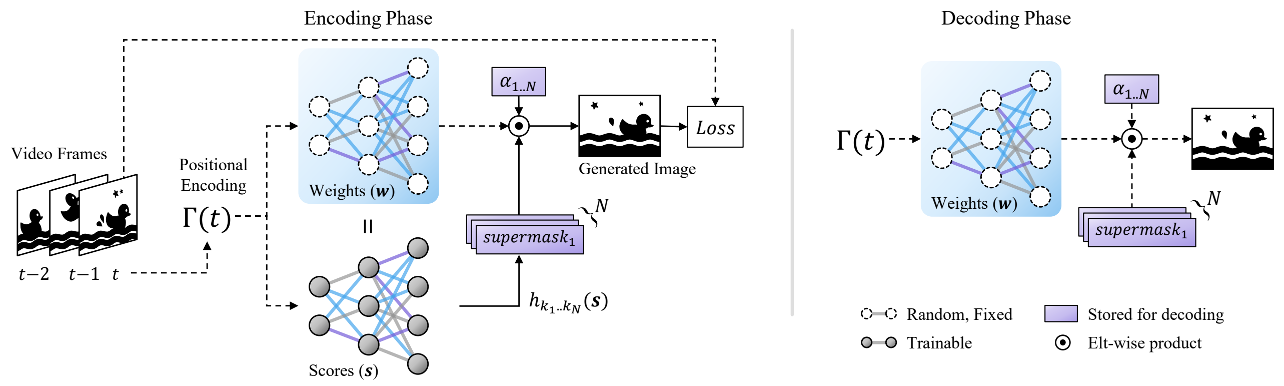


Figure 2. Overview of the proposed video representation framework. In the encoding phase, our video representation framework finds multiple supermasks and corresponding scale parameters in an implicit video representation network without training its weights. In the forward path, scores s of the randomly initialized network weights w are sorted, and N masks are assigned to the weights using step functions h_{k_n}'s. The threshold scores of the step functions are determined by pre-defined densities k_n's of nonzeros in the supermasks. The masks are then scaled by learnable parameters α_n's and accumulated to calculate the output of the network. In the backward path, the mean-squared error loss between the ground truth video frames and the network outputs is computed, and only the scores s and scales α_n's are updated by back-propagation. As all network parameters are fixed and random, we can recover the encoded images by simple network inference using random seed used at train time, N final supermasks and learned scale parameters α_n's in the decoding phase. This differs from the existing implicit video representation methods, which require storing model weights and biases.

performance. The traditional iterative pipeline of "overfit, prune, re-train" greatly reduces computation at inference, but involves considerable computational cost at training. To alleviate this problem, a substantial body of work has been proposed (see Hoefler et al., 2021, for survey). Lottery Ticket Hypothesis (Frankle & Carbin, 2018) is a recent and noteworthy discovery in this area. It suggests that an over-parameterized dense neural network, initialized randomly, includes a subnetwork that is initialized in such a way that it can achieve similar level of test accuracy as the original network when trained separately.

Strong Lottery Tickets: Zhou et al. (2019) discovers that lottery tickets can be found without weight updates. This is referred to as Strong Lottery Ticket Hypothesis, and the corresponding binary mask for subnetwork selection is called *supermask*. A recent work, Hidden Networks (Ramanujan et al., 2020), improves the algorithm of Zhou et al. (2019) by removing its stochasticity. A subsequent study, Multi-coated Supermasks (Okoshi et al., 2022), further refines Hidden Networks by training multiple nested supermasks. Moreover, there has been considerable research on the theory and algorithms related to the strong lottery ticket idea (see e.g., Malach et al., 2020; Pensia et al., 2020; Orseau et al., 2020; Diffenderfer & Kailkhura, 2021; Sreenivasan et al., 2022a;b). Despite being popular, there are currently limited findings on non-classification tasks. While a recent study (Yeo et al., 2023) has shown the existence of strong lottery tickets in generative models, their ability to produce high-resolution images is not yet satisfactory. In this paper, we move towards developing an implicit neural video representation framework that can effectively process high-resolution images by leveraging strong lottery tickets.

3. Method

3.1. Framework Overview

This section proposes a new video representation framework that finds strong lottery tickets (SLTs) in a video INR network (see Figure 2 for overall architecture). We use an image-wise representation model of Chen et al. (2021a) $f_\theta : \mathbb{R} \to \mathbb{R}^{3\times H\times W}$ that maps a frame index to the associated RGB image with network parameters θ. Our goal is to find hierarchical multi-level supermasks and proper scale parameters such that the resulting subnetwork fits to the target video without weight training, and this process corresponds to video encoding. Once model fitting is done, decoding is simple network inference as its weights and biases encode all video frames. Different from existing INR approaches (e.g., Sitzmann et al., 2020; Chen et al., 2021a; Li et al., 2022), we do not need to save the model parameters for decoding as a random number generator can recover untrained weights and biases, leading to significant reduction in model size. In the next section, we describe our algorithm for finding SLTs in the video INR model.

3.2. Multi-level Supermasks with Learnable Scales

Multi-level Supermasks: Our algorithm is developed upon Okoshi et al. (2022) that proposes a method for finding SLTs using multi-level supermasks with a hierarchical structure. For completness, we recap details of multi-level supermasks. The main idea is to score the importance of each weight and apply the edge-popup algorithm (Ramanujan et al., 2020) to pick weights with high scores according to a pre-determined density at each supermask level. To be specific, scores

$s = \{s_i\}_{i=1}^{D_l}$ are assigned for randomly initialized weight parameters $w^{\text{rand}} = \{w_i^{\text{rand}}\}_{i=1}^{D_l}$ in each layer (say, layer l) of the network. Here, D_l denotes the number of weights in the l-th layer. At each forward path, scores, $s = \{s_i\}_{i=1}^{D_l}$, are sorted in the decreasing order:

$$\begin{aligned} &\text{sort}(s) = (s'_1, s'_2, \cdots, s'_{D_l}) \\ &\quad \text{s.t.} \;\; |s'_1| \geq |s'_2| \geq \cdots \geq |s'_{D_l}|\,, \end{aligned} \tag{1}$$

where the operator $|\cdot|$ denotes the absolute value and s'_i's are ordered scores. Suppose pre-defined densities of nonzero elements for N supermask levels are given by $\{k_n\}_{n=1}^{N}$ where $1 > k_1 > \cdots > k_N > 0$. Then for each level n, referred to as "coat n" in Okoshi et al. (2022), the weight supermask is assigned using a step function with a threshold score s_{t_n}:

$$h_{k_n}(s_i) = \begin{cases} 1 & \text{if } |s_i| \geq s_{t_n} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

for each layer. Here, the threshold score for layer l is computed as $s_{t_n} = |s'_{t_n}|$ where the associated index

$$t_n = \lfloor k_n \times D_l \rfloor \tag{3}$$

is used to select the top-$100k_n$% scores $|s_i| \geq s_{t_n}$. If we denote the set of nonzero indices in a supermask with density k by

$$\mathbb{H}_k = \{i \,|\, h_k(s_i) \neq 0\}\,, \tag{4}$$

then the following inclusion relation holds:

$$k_i < k_j \Rightarrow \mathbb{H}_{k_i} \subseteq \mathbb{H}_{k_j}\,. \tag{5}$$

This implies that nonzero indices of a higher level supermask are a subset of those of a lower level supermask since $k_1 > \cdots > k_N$. In our experiments, the first density k_1 is set to be the target proportion of nonzero weights, and the other densities $\{k_n\}_{n=2}^{N}$ are determined by linear method of Okoshi et al. (2022). As typically done in model compression, we do not prune random biases in the model.

Overlay of Supermasks with Learnable Scales: Given supermasks for all levels of densities, we accumulate them to calculate the output of the network. Here, we incorporate scale parameters $\alpha = \{\alpha_n\}_{n=1}^{N}$ to the mask accumulation rather than training a simple overlay of the supermasks as done in Okoshi et al. (2022). Specifically, we use the following masked weights $w = \{w_i\}_{i=1}^{D_l}$:

$$w_i = w_i^{\text{rand}} \sum_{n=1}^{N} \text{clip}_\epsilon(\alpha_n) h_{k_n}(s_i)\,, \tag{6}$$

where $\epsilon > 0$ is a small positive constant. Here, α_1 is set to 1 and $\{\alpha_n\}_{n=2}^{N}$ are trainable parameters. For the stability of training, α_n's are clipped to be in the interval of $[\epsilon, \infty)$ so that the scale of each mask is positive. From this, the output of neuron v in the l-th fully connected layer in the forward pass is, for example, computed by

$$\mathcal{I}_v = \sum_{u \in \mathcal{V}^{(l-1)}} w_{uv}^{\text{rand}} \mathcal{Z}_u \sum_{n=1}^{N} \text{clip}_\epsilon(\alpha_n) h_{k_n}(s_{uv}) + b_v^{\text{rand}} \tag{7}$$

where w_{uv}^{rand} is the randomly initialized weight of edge connecting nodes u and v with score s_{uv}, b_v^{rand} is the random bias at node v, $\mathcal{Z}_u$ is the output of node u, and $\mathcal{V}^{(l-1)}$ is

Table 1. Effectiveness of scale learning in the proposed framework. Random subnetworks are selected to fit Bosphorus (Bospho), Jockey, ReadySteadyGo (Ready) and YachtRide (Yacht) videos with fixed and learnable scale settings. The performance is measured by reconstruction (PSNR) and visual quality (MS-SSIM). Bold indicates the best result, and ↑ denotes higher values are better. The first column (Params) is the number of parameters in each dense model, the number in the second column (Method) is the number of supermasks. Employing multiple supermasks generally yields better performance. Adding scale parameters further improves video compression quality for most combinations of BPP and video sequence.

Params	Method	BPP	PSNR (↑)				MS-SSIM (↑)			
			Bospho	Jockey	Ready	Yacht	Bospho	Jockey	Ready	Yacht
24.4M	1-Fixed	0.020	28.56	22.69	20.69	25.73	0.862	0.761	0.680	0.801
	3-Fixed	0.025	32.75	28.93	23.00	27.40	0.927	0.850	0.773	0.859
	3-Learned	0.025	**32.80**	**29.80**	**23.77**	**27.71**	**0.929**	**0.862**	**0.797**	**0.863**
36.5M	1-Fixed	0.029	31.92	27.40	21.72	26.70	0.913	0.823	0.724	0.838
	3-Fixed	0.037	32.87	30.44	24.17	28.09	0.935	0.875	0.812	0.878
	3-Learned	0.037	**33.61**	**31.16**	**24.81**	**28.38**	**0.940**	**0.886**	**0.832**	**0.882**
59.1M	1-Fixed	0.047	32.29	27.66	22.20	26.78	0.920	0.828	0.743	0.839
	3-Fixed	0.060	34.08	31.67	25.40	28.97	0.945	0.892	0.848	0.898
	3-Learned	0.060	**34.52**	**32.09**	**26.03**	**29.14**	**0.950**	**0.899**	**0.865**	**0.899**
87.7M	1-Fixed	0.070	32.75	28.13	22.74	27.31	0.927	0.836	0.764	0.857
	3-Fixed	0.089	34.52	32.22	26.42	29.73	0.954	0.900	0.872	0.911
	3-Learned	0.089	**35.29**	**33.00**	**26.14**	**30.00**	**0.957**	**0.912**	**0.873**	**0.915**
190.2M	1-Fixed	0.153	33.01	29.81	23.99	27.96	0.933	0.865	0.807	0.873
	3-Fixed	0.192	**36.46**	**33.31**	28.64	**30.92**	**0.967**	**0.918**	0.915	0.928
	3-Learned	0.192	35.63	32.69	**28.85**	30.86	0.960	0.908	**0.918**	**0.929**

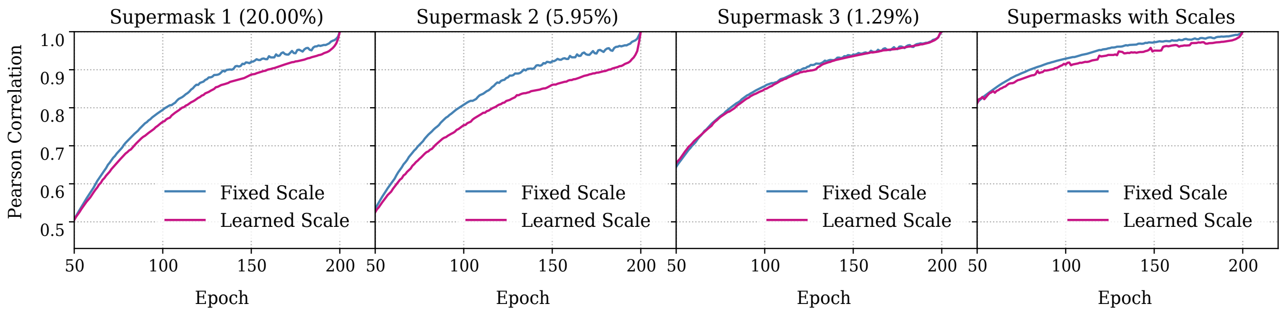


Figure 3. Correlation with final supermask. For each epoch of running the proposed framework with three supermasks, we compute Pearson correlation coefficients between the final supermask (at 200 epoch) and the current supermask (from 50 epochs) and plot them. The first three plots are results for the three levels, and the last one is obtained from the accumulated total mask (scale applied version). For both fixed and learnable scale approaches, the subnetworks are selected to fit YachtRide video using our framework with supermask densities $(k_1, k_2, k_3) = (0.2, 0.0595, 0.0129)$. When considering all plots, it is observed that pruning with learnable scales converges on the final mask more slowly while the speed gap at each level is slightly different.

the set of nodes in $(l-1)$-th layer. When α_n's are all fixed to be ones, (7) is identical to the corresponding output of the algorithm in Okoshi et al. (2022). Also, if $N = 1$, we recover the method of Ramanujan et al. (2020). The scale parameters $\alpha = \{\alpha_n\}_{n=1}^N$ are shared across all layers in the network, so the number of additional trainable parameters is $N-1$ in our framework compared to Okoshi et al. (2022). In the backward path, the mean-squared error loss between the ground-truth videos and the network outputs is computed and only the score s_i's and scale α_n's are updated by back-propagation. Here, the straight-through estimator (Bengio et al., 2013) is used to calculate the gradient of mask function (2) as in Ramanujan et al. (2020) and Okoshi et al. (2022). A pseudo code of training the proposed algorithm is described in Appendix A.1.

3.3. Encoded Model Size

In our framework, it is sufficient to only store the random seed used at train time, final supermasks, and scale parameters for recovering the subnetwork weights (6). Untrained network parameters can be reproduced by a random number generator using the random seed. Because of the hierarchical structure of supermasks (5), we store the location of the nonzero elements of the n-th supermask in the $(n-1)$-th supermask following Okoshi et al. (2022). If the random seed and $N-1$ scale parameters $\{\alpha_n\}_{n=2}^N$ are saved in B bits, then the total number of bits of unary encoded multi-level supermasks with learned scales for the l-th layer with D_l weights is computed by

$$\begin{cases} D_l + B & \text{for } N = 1 \\ \left(D_l + \sum_{n=2}^{N} \lfloor k_{n-1} \times D_l \rfloor\right) + B + (N-1)B & \text{for } N > 1 \end{cases} \tag{8}$$

where the first and second terms are for the weight masks and random seed, and the third term for $N > 1$ corresponds to the bits for the learned scale parameters $\{\alpha_n\}_{n=2}^N$. We note that the proposed algorithm incurs only $(N-1)B$ bits of additional storing costs compared to the method of Okoshi et al. (2022), and we found $N = 3$ works quite well in our experiments.

4. Experimental Evaluation

This section provides experimental evaluation of the proposed video representation framework. We first show the superiority of our prune-at-initialization algorithm with learnable scales under the same network architecture settings in Sections 4.2, 4.3, and 4.4. We also analyze optimal framework setups in Sections 4.5 and 4.6 and perceptual quality of the decoded videos in Section 4.7. We compare our approach with state-of-the-art video compression methods in Section 4.8 and provide encoding time in Section 4.9.

4.1. Implementation Details

Dataset: Following prior video INR methods (e.g., Chen et al., 2021a; Li et al., 2022), we demonstrate the effectiveness of our framework on UVG dataset (Mercat et al., 2020), a widely used benchmark for video compression. UVG dataset consists of 7 video sequences: Beauty, Bosphorus, HoneyBee, Jockey, ReadySteadyGo, ShakeNDry and YachtRide. We extract RGB frames from the original YUV videos with a resolution of 1920×1080 (see Appendix A.2).

Experimental Settings: We adopt NeRV architectures (Chen et al., 2021a) as our dense models (see Appendix A.3 for network specifications of authors' example models). The network weights are randomly initialized using Kaiming Normal distribution (He et al., 2015). Throughout our experiments, we use 3 levels of supermasks with $k_1 = 0.2$, and the other densities $\{k_n\}_{n=2}^3$ are chosen by linear method of Okoshi et al. (2022) unless specified. Further training

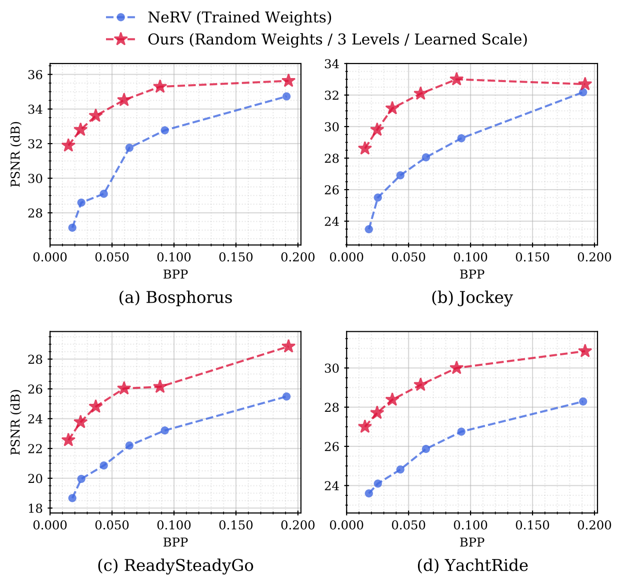


Figure 4. Comparison between proposed prune-at-initialization method and full weight training on Bosphorus, Jockey, ReadySteadyGo and YachtRide sequences. Although our method is based on random weights, the resulting subnetworks significantly outperform fully trained dense models at varying BPPs for all sequences.

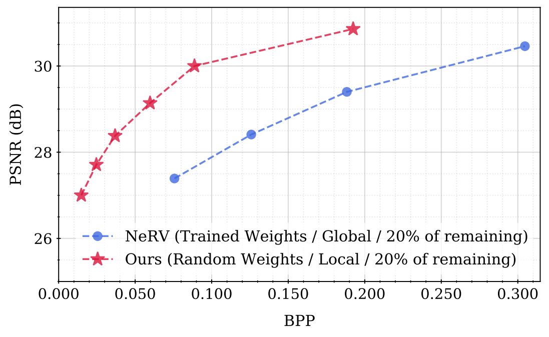


Figure 5. Comparison between the proposed random weight pruning and conventional pruning pipeline (“overfit, prune, re-train”) on implicit video representation models fitted to YachtRide video. The random subnetworks obtained from our layer-wise (local) pruning framework significantly outperform those from the conventional global pruning method at varying BPPs.

details are provided in Appendix A.4.

Metrics: To quantify the reconstruction quality of videos, we use peak signal-to-noise ratio (PSNR) that is a commonly used metric in video compression. For visual quality assessment, decoded images are evaluated by MS-SSIM (Wang et al., 2003). We also adopt bits-per-pixel (BPP) to measure the compression ratio and use the formula of Chen et al. (2021a) for calculating BPP:

$$\frac{\text{Total Number of Bits for Storing Encoded Model}}{\text{Total Number of Pixels}}. \quad (9)$$

The numerator of (9) is obtained from (8) for our models. The quality of video encoding is tested on multiple BPPs (see Table 4 in Appendix A.3 for example values).

4.2. Effectiveness of Scale Learning

As our method is the first work to introduce prune-at-initialization in neural video representation networks, there is currently no existing reference framework based on random weights for comparison. Instead, we compare our method with an alternative approach that directly applies the pruning algorithm of Okoshi et al. (2022) for finding SLTs in classification models where scale α_n for each level in (6) is fixed at one. For both approaches, we use 3 levels of supermasks ($N = 3$) and also report the performance of single supermask case. Specifically, five randomly initialized example networks (ID 1-5 in Table 4) are pruned under three settings: 1-level with fixed scale, 3-levels with fixed scale and 3-levels with learnable scale. Then, we summarize the PSNR values for different BPPs computed on four UVG video sequences in Table 1. It is shown that using multiple supermasks generally yields higher quality of video compression, which is consistent with results reported in Okoshi et al. (2022) for classification networks. In addition, adding scale parameters further improves PSNR for most combinations of BPPs and video sequences with a negligible increase in compression ratios (see Appendix B). We believe that the learned scale setup can achieve higher performance by leveraging per-setting hyperparameter search.

Our intuition is that the proposed method with learnable scales has a wider parameter search space that can fully explore different sparsity patterns, leading the algorithm to find stronger subnetworks. To see this, we compute Pearson correlation coefficients between the final supermasks and supermasks at the end of each training epoch and plot them in Figure 3 as in Tai et al. (2022). For every supermask level, we find that pruning with learnable scales converges to the final mask more slowly while the speed gap at each supermask level is a bit different. The same pattern is observed for the accumulated total supermask with scales. This represents sufficient exploration of diverse sparsity patterns, and as a result the resulting subnetwork with learned scales achieves higher video encoding quality. In this experiment, the random subnetwork selected from the dense model with 14.3M parameters (ID 0 in Table 4) using learned scales attains higher PSNR of 26.72dB on YachtRide video. The corresponding value of the pruned model obtained with constant scale is 26.40dB.

4.3. Comparison with Fully Trained Dense Models

We now compare weight training and our method. We use fully trained *dense* NeRV models (Chen et al., 2021a) as baselines. In fact, BPPs of the example dense models provided by the authors are significantly larger than those for

Table 2. Effect of the number of supermasks N on PSNR and BPP when pruning the model with 59.1M parameters using the proposed framework. Given the first supermask density $k_1 = 0.2$, highest performance is obtained with $N = 3$ on HoneyBee and Jockey.

N	$\{k_n\}_{n=1}^N$	BPP	PSNR (dB)	
			HoneyBee	Jockey
1	0.2	0.0474	36.03	27.66
2	0.2,0.0288	0.0569	37.22	29.67
3	0.2,0.0595,0.0129	0.0597	**38.20**	**32.09**
4	0.2,0.0830,0.0288,0.0083	0.0622	37.90	28.72
5	0.2, 0.1003, 0.0450, 0.0180, 0.0064	0.0647	37.11	28.51

the subnetworks obtained from our method (see Table 4). To ensure fair evaluation, we build dense networks with the same architecture to similarly match the BPPs of our models (see Appendix C for details). Figure 4 demonstrates that our subnetworks with random weights significantly outperform the state-of-the-art trained dense models at various compression ratio settings.

4.4. Comparison with Fully Trained Sparse Models

We consider fully trained *sparse* NeRV models (Chen et al., 2021a) as baselines. The sparse networks are obtained from authors' official implementation of the standard model compression (train, prune, and re-train) using "global" unstructured pruning in `pytorch` library. We use the first 5 example networks (ID 0-4 in Table 4) as dense networks. For fair comparison, the fraction of weights remaining after pruning is set to 20% for both baseline and our approaches. Figure 5 shows the video encoding quality of our pruning method versus the baseline on YachtRide video sequence. Here, we only plot PSNR values of the reference models close to BPP interval of our subnetworks. The subnetworks obtained from our framework perform better than those from the reference approach even though our method is based on layer-wise ("local") pruning of *random* weights. Additional results on a different datasets are presented in Appendix D.

4.5. Effect of Number of Supermasks on Performance

This section evaluates the effect of the number of supermasks on PSNR and compression ratio (BPP) in the proposed framework. Recall that given the first level of density k_1, the other proportions of the remaining weights $\{k_n\}_{n=2}^N$ are determined by linear method of Okoshi et al. (2022). We empirically probe the best N for fixed $k_1 = 0.2$ using the example model with 59.1M parameters (ID 3 in Table 4). Table 2 shows the results for $N \in \{1, 2, 3, 4, 5\}$ on HoneyBee and Jockey videos. While BPP increases monotonically with the number of supermask levels N, the highest performance is obtained with $N = 3$ requiring 4.96% bigger BPP than the one with $N = 2$. We speculate that this is an exploration-exploitation trade-off of having the widest search space at $N = 5$ and smallest search space at $N = 1$.

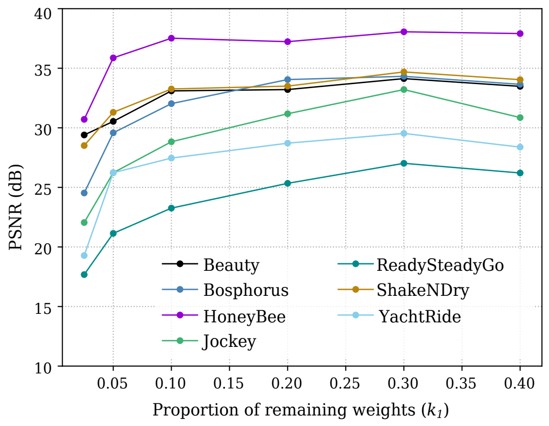


Figure 6. Effect of total weight density on reconstruction performance. For various densities of the non-pruned weights $k_1 \in \{0.025, 0.05, 0.1, 0.2, 0.3, 0.4\}$, we plot PSNR values of the corresponding random subnetworks selected from our framework. With $k_1 \in \{0.2, 0.3, 0.4\}$ we obtain reasonable quality, and the best performance is shown at $k_1 = 0.3$.

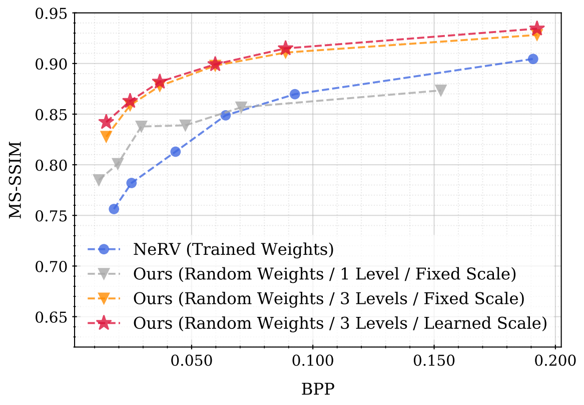


Figure 7. Perceptual quality of decoded YachtRide video. MS-SSIM for variety of models from different video compression frameworks are reported to evaluate visual quality of the decoded images. The proposed pruning method of random weights with learned scale achieves higher MS-SSIM values than weight training as well as the prune-at-initialization algorithm with fixed scale.

4.6. Effect of Total Weight Density on Performance

We investigate the relationship between the proportion of non-pruned weights (i.e., the density of the first supermask k_1) and video encoding quality of the resulting subnetwork in our framework. To see this, we prune the model with 59.1M parameters (ID 3 in Table 4) for varying the values of $k_1 \in \{0.025, 0.05, 0.1, 0.2, 0.3, 0.4\}$. We did not exploit larger densities as a certain level of over-parameterization is required to achieve good performance (e.g., Pensia et al., 2020). Figure 6 shows that descent quality is obtained with $k_1 \in \{0.2, 0.3, 0.4\}$ with a maximum at $k_1 = 0.3$. Aggressive pruning deteriorates the performance as demonstrated in cases of $k_1 \in \{0.025, 0.05, 0.1\}$. The results for $k_1 = 0.4$ suggest that we need to prune sufficiently many weights to lessen the impact of using random parameters.

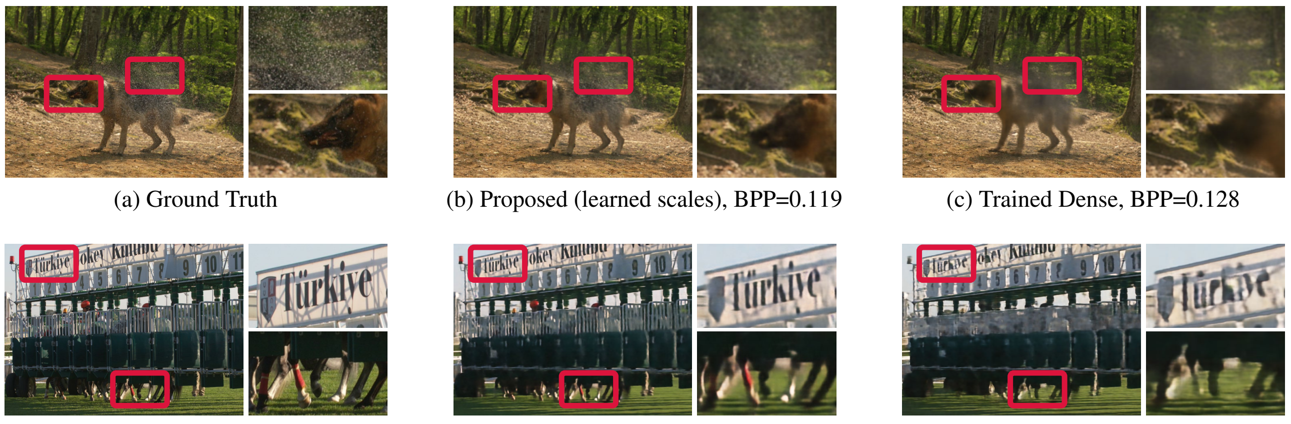

(a) Ground Truth
(b) Proposed (learned scales), BPP=0.119
(c) Trained Dense, BPP=0.128
(d) Ground Truth
(e) Proposed (learned scales), BPP=0.060
(f) Trained Dense, BPP=0.064

Figure 8. Reconstruction results on ShakeNDry and ReadySteadyGo videos. The subnetworks selected by the proposed method with learned scales better encode fine details (e.g., water drops and horse legs).

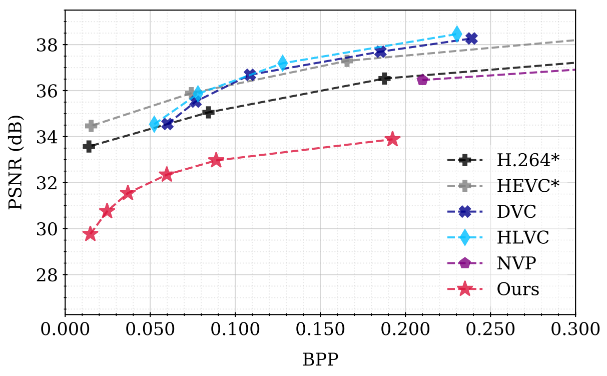


Figure 9. Comparison with state-of-the-art learning-based video compression methods (DVC, HEVC and NVP) and video compression standards (H.264 and HEVC). * indicates that results are taken from DVC. Our random subnetworks show decent performance compared to existing video compression frameworks that consist of carefully optimized complex modules.

4.7. Perceptual Quality

We now demonstrate how the encoded images are perceptually similar to the original images by measuring MS-SSIM. Here, we use three types of baseline methods of Section 4.2 (prune-at-initialization with single fixed scale and 3 fixed scales) and Section 4.3 (training dense models). Figure 7 and Table 1 show that the subnetworks selected from the proposed pruning framework with learned scale generally achieve highest visual quality at various BPPs. In Figure 8, we see that the proposed method of learned scales better captures fine details in dynamic scenes. More visualization results are provided in Appendix E.

4.8. Comparison with Other Compression Methods

We compare compressed videos obtained from the proposed framework and state-of-the-art learning-based video compression methods of DVC (Lu et al., 2019), HLVC (Yang et al., 2020), and NVP (Kim et al., 2022) and video compression standards, H.264 (Wiegand et al., 2003) and HEVC, (Sullivan et al., 2012) on all video sequences of UVG dataset. Following prior works (e.g., Kim et al., 2022), results are averaged over the 7 video sequences. Figure 9 shows that the proposed approach works decently compared to existing top-performing methods even if our subnetworks are based on random weights. Admittedly, there exists a performance gap between ours and the existing video compression algorithms. This is because our core objective is not to construct optimal video compression architecture itself whereas the prior methods consist of carefully designed complex modules such as motion estimation and entropy coding tailored for compression. Also, we believe the proposed prune-at-initialization has room for improvement, such as developing better neural networks for video representation and optimizing initialization methods, which is our future research direction. More video-wise comparison results with video compression standards are illustrated in Appendix F.

4.9. Encoding Time

This section provides encoding time of the proposed method and baseline NeRV models (Chen et al., 2021a). We used a single NVIDIA A100 GPU (80GB) and 4 batches throughout this experiment. For both approaches, 6 example models (ID 0-5) in Table 4 are served as dense models. Our framework is run for 200 epochs under 3-level supermask setup with a total density of $k_1 = 0.2$. NeRV models are trained for 200 epochs and then pruned for 50 epochs using the conventional model compression pipeline to achieve the same density on authors' official implementation. Encoding a video of 600 frames at 1920x1080 resolution takes 67-351 minutes (ID 0-5) using our framework with learnable scale and 64-314 minutes (ID 0-5) with fixed scale. On the other hand, it takes 75-335 minutes (ID 0-5) for compression of NeRV models. Our method is competitive in terms of encoding time compared to weight training.

5. Conclusion

We introduce a new implicit video representation framework that finds SLTs in an image-wise representation model without weight training. Extensive experiments verify that the random subnetworks selected by the proposed algorithm outperform those obtained from existing methods of finding SLTs in classification networks as well as fully trained models with the same encoded model sizes under various video sequences and compression ratio settings. To the best of our knowledge, this is the first work to show the existence of random subnetworks for non-classification tasks that output full HD size videos. We hope our work will stimulate interest in exploring new paradigm for video representations and algorithms for finding SLTs in non-classification networks.

References

Andersson, P., Nilsson, J., Akenine-Möller, T., Oskarsson, M., Åström, K., and Fairchild, M. D. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):15:1–15:23, 2020.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL https://arxiv.org/abs/1308.3432.

Chabra, R., Lenssen, J. E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., and Newcombe, R. A. Deep local shapes: Learning local SDF priors for detailed 3d reconstruction. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

Chen, H., He, B., Wang, H., Ren, Y., Lim, S.-N., and Shrivastava, A. NeRV: Neural representations for videos. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2021a. URL https://github.com/haochen-rye/NeRV.git.

Chen, H., Gwilliam, M. A., He, B., Lim, S.-N., and Shrivastava, A. CNeRV: Content-adaptive neural representation for visual data. In *33rd British Machine Vision Conference (BMVC)*. BMVA Press, 2022.

Chen, Y., Liu, S., and Wang, X. Learning continuous image representation with local implicit image function. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021b.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Diffenderfer, J. and Kailkhura, B. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Training pruned neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W. T., and Funkhouser, T. Learning shape templates with structured implicit functions. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

Genova, K., Cole, F., Sud, A., Sarna, A., and Funkhouser, T. A. Local deep implicit functions for 3d shape. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (GELUs). *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 2021.

Jiang, C. M., Sud, A., Makadia, A., Huang, J., Nießner, M., and Funkhouser, T. A. Local implicit grid representations for 3d scenes. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Kim, S., Yu, S., Lee, J., and Shin, J. Scalable neural video representations with learnable positional features. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

Li, Z., Wang, M., Pi, H., Xu, K., Mei, J., and Liu, Y. E-NeRV: Expedite neural video representation with disentangled spatial-temporal context. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2022.

Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., and Gao, Z. DVC: An end-to-end deep video compression framework. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Malach, E., Yehudai, G., Shalev-Shwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., and Chandraker, M. Modulated periodic activations for generalizable local functional representations. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2021.

Mercat, A., Viitanen, M., and Vanne, J. UVG dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, pp. 297–302, New York, NY, USA, 2020. Association for Computing Machinery.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

Niemeyer, M., Mescheder, L. M., Oechsle, M., and Geiger, A. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Oechsle, M., Peng, S., and Geiger, A. UNISURF: Unifying neural implicit surfaces and radiance fields for multiview reconstruction. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2021.

Okoshi, Y., García-Arias, A. L., Hirose, K., Ando, K., Kawamura, K., Van Chu, T., Motomura, M., and Yu, J. Multicoated supermasks enhance hidden networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

Orseau, L., Hutter, M., and Rivasplata, O. Logarithmic pruning is all you need. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. Convolutional occupancy networks. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., and Zhou, X. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Pensia, A., Rajput, S., Nagle, A., Vishwakarma, H., and Papailiopoulos, D. S. Optimal lottery tickets via subsetsum: Logarithmic over-parameterization is sufficient. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. Real-time single image and video super-resolution using an efficient subpixel convolutional neural network. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Sitzmann, V., Martel, J. N., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Sreenivasan, K., Rajput, S., Sohn, J.-Y., and Papailiopoulos, D. Finding nearly everything within random binary networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2022a.

Sreenivasan, K., Sohn, J.-Y., Yang, L., Grinde, M., Nagle, A., Wang, H., Xing, E., Lee, K., and Papailiopoulos, D. Rare gems: Finding lottery tickets at initialization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Sullivan, G. J., Ohm, J.-R., Han, W.-J., and Wiegand, T. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. doi: 10.1109/TCSVT.2012.2221191.

Tai, K. S., Tian, T., and Lim, S.-N. Spartan: Differentiable sparsity via regularized transportation. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tomar, S. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

Wang, Z., Simoncelli, E., and Bovik, A. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pp. 1398–1402, 2003.

Wiegand, T., Sullivan, G., Bjontegaard, G., and Luthra, A. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003. doi: 10.1109/TCSVT.2003.815165.

Yang, R., Mentzer, F., Van Gool, L., and Timofte, R. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Yeo, S., Jang, Y., Sohn, J.-Y., Han, D., and Yoo, J. Can we find strong lottery tickets in generative models? In *Proceedings of AAAI Conference on Artificial Intelligence*, 2023.

Zhang, Y., van Rozendaal, T., Brehmer, J., Nagel, M., and Cohen, T. Implicit neural video compression. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A. Implementation Details

A.1. Pseudo Code of the Proposed Method

We provide a pseudo code of training our framework with a single batch in Algorithm 1. We note that our networks do not contain batch normalization layers, and all random weight and bias parameters in the networks are not updated at train time. This differs from the previous method of Okoshi et al. (2022) for finding SLTs in classification networks where affine batch normalization layers are used (i.e., scale and bias parameters in batch-norm layers are updated).

Algorithm 1 Multi-level Supermasks with Learnable Scales

```
Input: Video with $T$ frames $\{x_t\}_{t=1}^T$, neural network $f(\cdot; w, b)$ with weight and bias parameters $w$ and $b$, where the weight parameter vector is $D = \sum_l D_l$ dimensional with $D_l$ weights for layer $l$, $N$ supermasks with densities $\{k_n\}_{n=1}^N$, number of iterations $E$, learning rate $\eta$
Output: Supermasks $m = \{m_n\}_{n=1}^N$ with $m_i \in [0,1]^D$ and scale parameters $\alpha = \{\alpha_n\}_{n=1}^N$ with $\alpha_n > 0$
Randomly initialize weight $w_{\text{rand}} \in \mathbb{R}^D$ and bias $b_{\text{rand}}$
Randomly initialize score $s \in [0,1]^D$
for $e = 1$ to $E$ do
  $s' \leftarrow \text{sort}(s)$    /* layer-wise sorting*/
  for $n = 1$ to $N$ do
    $s_{t_n} \leftarrow s'_q$ where $q \leftarrow \lfloor k_n \times D_l \rfloor$    /* for each layer $l$ with $D_l$ weights, top-$\lfloor k_n \times D_l \rfloor$ thresholding*/
    $m_n \leftarrow h_{k_n}(s)$    /* step function $h_{k_n}$ with threshold $s_{t_n}$ is applied to elements of $s$ */
  end for
  $w \leftarrow w_{\text{rand}} \odot \sum_{n=1}^N \text{clip}_\epsilon(\alpha_n) m_n$    /* $\odot$ is an element-wise multiplication operator */
  $\hat{x}_t \leftarrow f(t; w, b_{\text{rand}})$
  loss $\leftarrow$ MSE$(\hat{x}_t, x_t)$
  /* use straight-through estimator for calculating gradients of $h_{k_n}$ */
  $s \leftarrow s - \eta \nabla_s \text{loss}$
  $\alpha \leftarrow \alpha - \eta \nabla_\alpha \text{loss}$
end for
```

A.2. UVG Dataset

Following prior works (e.g., Chen et al., 2021a), we use `ffmpeg` (Tomar, 2006) to extract RGB frames form original YUV videos:

```
ffmpeg -f rawvideo -s 1920x1080 -pix_fmt yuv420p -probesize 32M -framerate 120
       -i <SEQ>_1920x1080_120fps_420_8bit_YUV.yuv
       -sws_flags bilinear+accurate_rnd+full_chroma_int -y <SEQ>/%04d.png
```

where `<SEQ>` is the input video sequence name.

A.3. NeRV

Network Architecture: We adopt NeRV (Chen et al., 2021a), an image-wise representation model, to represent a video $V = \{x_t\}_{t=1}^T$ with a mapping $f_\theta : \mathbb{R} \to \mathbb{R}^{3\times H\times W}$ parameterized by the network parameters θ. We recap its architecture details for completeness. Here, T, H and W are the number of video frames and height and width of the RGB frame, respectively. For each video, the input is a frame index t and the output $f_\theta(t)$ fits to the corresponding image $x_t \in \mathbb{R}^{3\times H\times W}$. The network f_θ first maps the normalized frame index $t \in [0, 1)$ to a positional encoding vector (e.g., Mildenhall et al., 2020; Tancik et al., 2020):

$$\left(\sin(b^0\pi t), \cos(b^0\pi t), \cdots, \sin(b^{l-1}\pi t), \cos(b^{l-1}\pi t)\right) , \tag{10}$$

where b and l are hyperparameters. Then a 2-layer multi-layer perceptron (MLP) takes the positional embedding as input and outputs a small feature map. Finally, the feature map is gradually transformed into the original image size by multiple NeRV blocks that consist of convolutional, PixelShuffle (Shi et al., 2016) and activation layers.

Hyperparameters: For our experiments with UVG dataset (Mercat et al., 2020), hyperparameters b and l in the positional embedding (10) are set to be 1.25 and 80, respectively, and the output channel of the first layer in the MLP is 512. We use the upscale factors $(5, 3, 2, 2, 2)$ in NeRV blocks, and GELU (Hendrycks & Gimpel, 2016) activation as suggested by the authors. NeRV architecture is parameterized by (C_1, C_2) as described in Table 3, so we can obtain models with different sizes by changing these values. Table 3 is directly taken from Appendix A.1 in Chen et al. (2021a).

Table 3. NeRV architecture for UVG dataset with a frame resolution 1920×1080.

Layer	Modules	Upscale Factor	Output Size $(C \times H \times W)$
0	Positional Encoding	-	$160 \times 1 \times 1$
1	MLP and Reshape	-	$C_1 \times 16 \times 9$
2	NeRV block	$5\times$	$C_2 \times 80 \times 45$
3	NeRV block	$3\times$	$C_2/2 \times 240 \times 135$
4	NeRV block	$2\times$	$C_2/4 \times 480 \times 270$
5	NeRV block	$2\times$	$C_2/8 \times 960 \times 540$
6	NeRV block	$2\times$	$C_2/16 \times 1920 \times 1080$
7	Head layer	-	$3 \times 1920 \times 1080$

Compression Ratio: The formula for calculating compression ratios of trained NeRV models is described in its official implementation. Suppose a model with D parameters is trained with B bit precision to fit a video with T frames of an $H \times W$ resolution. If the proportion of remaining weights is k after pruning, then the BPP is computed by

$$\begin{aligned}&\frac{\text{Total Number of Bits for Storing Encoded Model}}{\text{Total Number of Pixels}}\\=&\frac{(\text{Number of Remaining Weights and Biases}) \times (\text{Parameters Bits})}{\text{Total Number of Pixels}}\\=&\frac{(D \times k) \times B}{T \times H \times W},\end{aligned} \tag{11}$$

where $B = 32$ as the model is trained in full precision (FP32) in our experiments.

Example Models: Authors provide 7 example networks with different number of parameters. In our experiments, we use 6 of them with (C_1, C_2) equal to $(48, 384)$, $(64, 512)$, $(128, 512)$, $(128, 768)$, $(128, 1024)$, and $(192, 1536)$. The number of parameters for the six models is summarized in Table 4.

Table 4. Specifications of 6 example NeRV models for UVG experiments. The fourth column (Params) is the number of parameters in each dense model (100% remaining). We summarize BPPs of sparse models after pruning 80% of the original. The sixth column comes from conventional pruning method (overfit, prune, and re-train). The last two columns are BPPs of pruned models obtained from the proposed approach with learnable scale and the total density $k_1 = 0.2$ for 1-supermask and 3-supermask setups. The BPP calculation is based on full precision models (FP32) fitted to a video sequence that consists of 600 frames with a resolution of 1920×1080.

ID	C_1	C_2	Params	BPP			
				NeRV (trained)		Proposed (random)	
				(100% remaining)	(20% remaining)	1-supermask ($k_1 = 0.2$)	3-supermask ($k_1 = 0.2$)
0	48	384	14.6M	0.376	0.076	0.012	0.015
1	64	512	24.4M	0.627	0.125	0.020	0.025
2	128	512	36.5M	0.938	0.188	0.029	0.037
3	128	768	59.1M	1.519	0.304	0.047	0.060
4	128	1024	87.7M	2.256	0.451	0.070	0.089
5	192	1536	190.2M	4.891	0.978	0.153	0.193

A.4. Training Details of the Proposed Framework

We train our framework for 200 epochs using Adam optimizer (Kingma & Ba, 2015) with a cosine learning rate scheduler and 4 batches on a single NVIDIA A100 GPU (80GB). Multiple learning rates ranging from 0.015 to 0.200 are swept over and the best results are reported. Throughout our experiments, networks are trained in full precision (FP32). We do not consider quantization as many elements of positional encoding vector (10) in the first stage of NeRV network (Chen et al., 2021a) vanish in lower precision system.

B. Comparison of Compression Ratios between Fixed and Learned Scales

We summarize compression ratios (BPPs) of the subnetworks obtained from the proposed framework with fixed and learned scales. We use the 6 example models provided in Table 4 as dense networks in our experiments with UVG dataset (Mercat et al., 2020). In order to accurately show BPPs, the number of parameters in each dense model and corresponding BPP are expressed to 6 and 7 decimal places, respectively. As in Table 5, the increase in BPP due to additional scale parameters is negligible.

Table 5. BPPs of pruned models obtained from our framework with fixed and learned scales on UVG experiments. The fourth column (Params) is the number of parameters in each dense model. We summarize BPPs of subnetworks with total density $k_1 = 0.2$ selected by the proposed approach with fixed and learned scale under 3-supermask setup. The BPP calculation is based on full precision models (FP32) fitted to a video sequence that consists of 600 frames with a resolution of 1920×1080. BPPs are expressed up to 7 decimal places to accurately show the amounts of increase introduced by additional scale parameters.

ID	C_1	C_2	Params	BPP 3-supermask ($k_1 = 0.2$) fixed scale	BPP 3-supermask ($k_1 = 0.2$) learned scale
0	48	384	14.630443M	0.01479122	0.01479127
1	64	512	24.364259M	0.02463866	0.02463872
2	128	512	36.464867M	0.03687914	0.03687919
3	128	768	59.052115M	0.05973678	0.05973683
4	128	1024	87.721923M	0.08875198	0.08875203
5	192	1536	190.155427M	0.19242292	0.19242297

C. Experimental Settings for Fully Trained Dense Models

As shown in Table 4, there is no overlap between BPP range of example dense NeRV models (Chen et al., 2021a), $[0.376, 4.891]$, and the corresponding range of our default setup, $[0.015, 0.193]$. For fair comparison, we build new NeRV networks using the modules in Table 3 to similarly match the range of $[0.015, 0.193]$ by changing (C_1, C_2). The constructed architectures are provided in Table 6. We train the constructed dense models for 200 epochs using authors' optimal hyperparameters.

Table 6. Specifications of constructed dense NeRV models for UVG experiments. The fourth column (Params) is the number of parameters in each network. We provide BPP values of the dense models trained in full precision (FP32) to fit a video that consists of 600 frames with a resolution of 1920×1080.

ID	C_1	C_2	Params	BPP NeRV (trained) (100% remaining)
A	6	48	0.7M	0.018
B	8	64	1.0M	0.025
C	16	64	1.7M	0.043
D	16	128	2.5M	0.064
E	24	144	3.6M	0.093
F	48	192	7.4M	0.191

D. Results on Different Datasets

D.1. Results on Big Buck Bunny Video

We consider fully trained sparse NeRV models (Chen et al., 2021a) as baselines and compare them with the our subnetworks on Big Buck Bunny dataset from `scikit-video`. The baseline networks are obtained from the conventional model compression pipeline that consists of overfitting, pruning and finetuning steps. BPPs of example models (NeRV-M and NeRV-L) provided by the authors are significantly larger than our subnetworks, so we build two models, say NeRV-XXS and NeRV-XS, to match the BPPs close to those of our subnetworks (see Table 7). Figure 10 shows that the proposed pruning method of random weights with learned scales achieves higher PSNR than weight training as well as the prune-at-initialization algorithm with fixed scale parameters.

Table 7. Specifications of NeRV models for Big Buck Bunny experiment. NeRV-S, NeRV-M, and NeRV-L are provided by the authors and NeRV-XS and NeRV-XXS are constructed. The second column (Params) is the number of parameters in each dense model. We summarize BPPs of sparse models after pruning 80% of the original using conventional method (overfit, prune, re-train) and the proposed approach with 3 supermasks and the total density $k_1 = 0.2$. The video sequence consists of 132 frames with a resolution of 720×1280, and the BPP calculation is based on full precision models (FP32).

ID	Params	BPP	
		NeRV (trained) (20% remaining)	Proposed (random) 3-supermask ($k_1 = 0.2$)
XXS	1.3M	0.068	-
XS	1.5M	0.080	-
S	3.2M	0.170	0.033
M	6.3M	0.333	0.065
L	12.5M	0.663	0.129

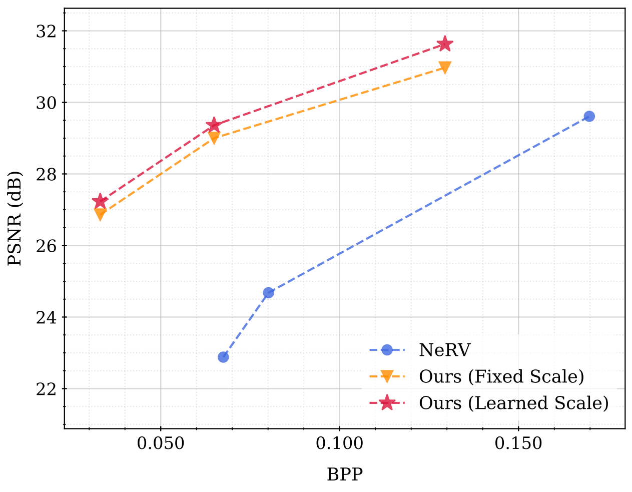


Figure 10. PSNR vs BPP on Big Buck Bunny dataset. The reconstruction quality (PSNR) of decoded images obtained from different implicit video representation frameworks at various compression ratios (BPPs) is reported. The proposed method with learned scales outperforms weight training as well as the prune-at-initialization algorithm with fixed scale parameters.

D.2. Results on Cityscapes Video

We demonstrate the effectiveness of our framework on Cityscapes dataset (Cordts et al., 2016), a widely used benchmark for urban scene understanding. We use one demo video sequence (stuttgart00), which consists of 599 frames with a resolution of 1024×2048. We consider two baseline models named "Trained Sparse" and "Trained Dense". The first one is fully trained sparse models obtained from the conventional model compression pipeline that consists of overfitting, pruning and finetuning steps. The fraction of remaining weights after pruning is set to 20% for the sparse baseline and our framework. The other is fully trained dense networks (without pruning). We use the default setup (3 supermasks and learnable scales) for our random subnetworks. The video encoding quality of each method is tested on multiple BPPs. Figure 11 shows that our method consistently achieves higher reconstruction (PSNR) and visual (MS-SSIM) quality than the baselines under similar compression ratio settings (BPP).

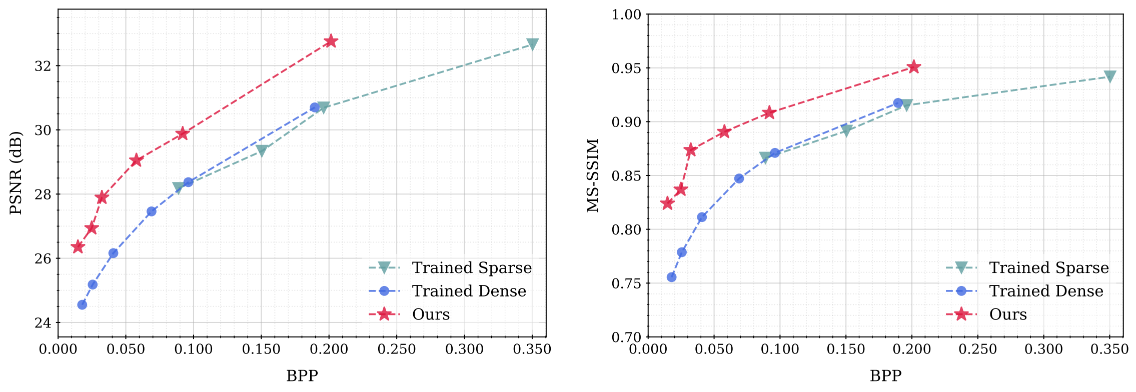


Figure 11. PSNR vs BPP (left) and MS-SSIM vs BPP (right) on Cityscapes dataset. The reconstruction (PSNR) and perceptual (MS-SSIM) quality of decoded images obtained from different implicit video representation frameworks at various compression ratios (BPPs) is reported. The BPP calculation is based on full precision (FP32) for trained models. The proposed method outperforms weight training.

E. More Visualization Results

E.1. Decoded Images

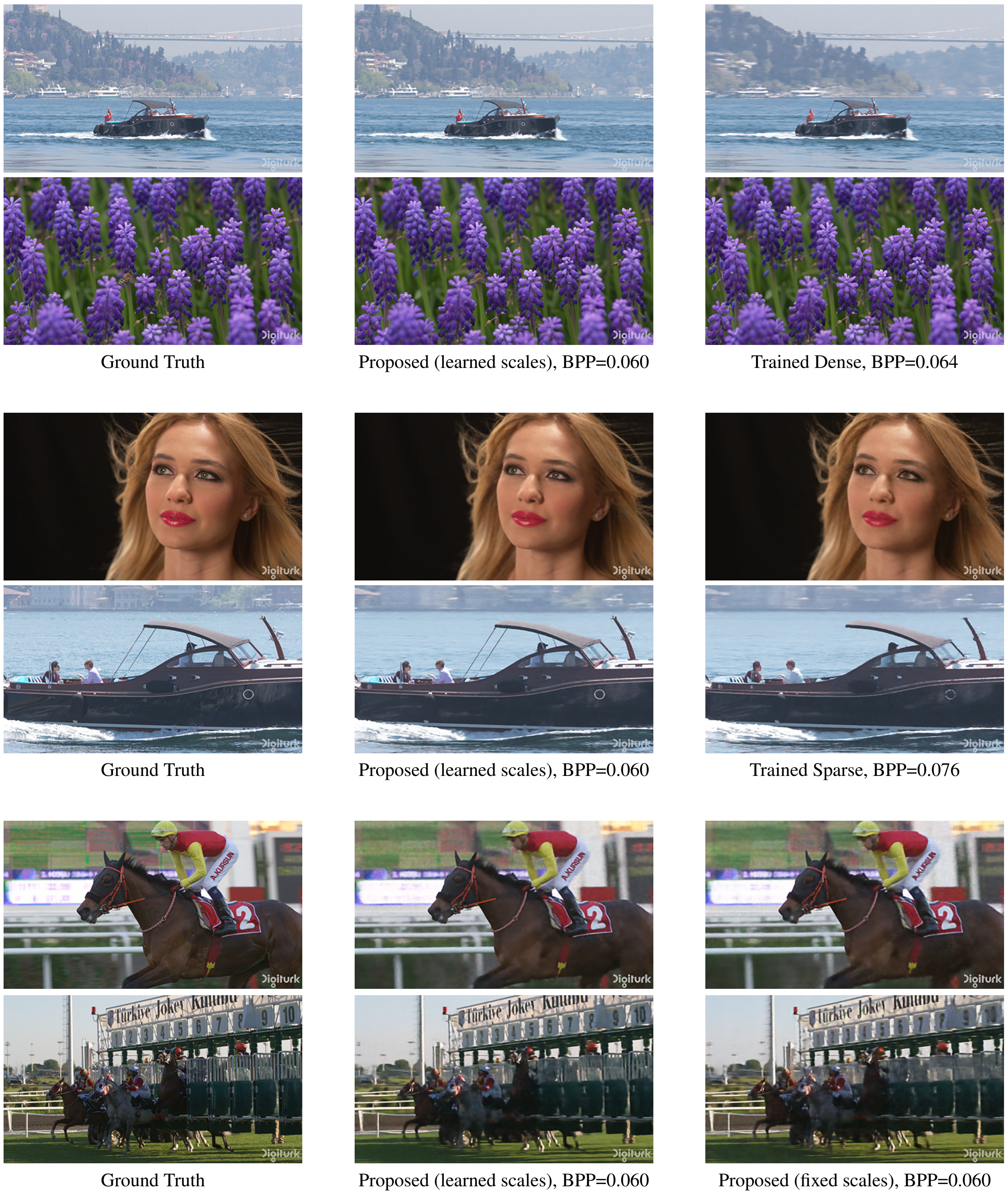

Ground Truth
Proposed (learned scales), BPP=0.060
Trained Dense, BPP=0.064
Ground Truth
Proposed (learned scales), BPP=0.060
Trained Sparse, BPP=0.076
Ground Truth
Proposed (learned scales), BPP=0.060
Proposed (fixed scales), BPP=0.060

Figure 12. More visualization results on Bosphorus, HoneyBee, Beauty, YachtRide, Jockey videos. The proposed method of learned scales encodes fine details better. Please refer to `https://saitmerong.github.io/INRSLT/` for larger images.

E.2. FLIP Visualization

We further provide FLIP maps (Andersson et al., 2020) for the decoded images in Tables 8, 9, and 10. FLIP generates a map of approximating errors recognized by humans when alternating between two images. In FLIP maps, bright color corresponds to large error and dark to small error, and smaller FLIP value is better. We find that at low BPP settings ($\text{BPP} < 0.1$), the proposed method better encodes small objects in large area having repeated patterns compared with three baseline methods. Please refer to `https://saitmerong.github.io/INRSLT/` for larger images.

Table 8. Proposed (learned scales) vs. Trained Dense

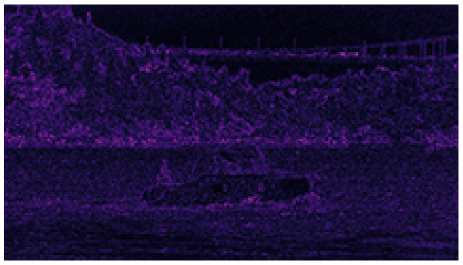
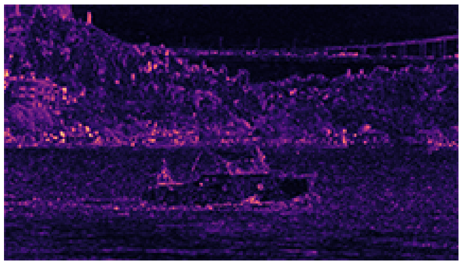
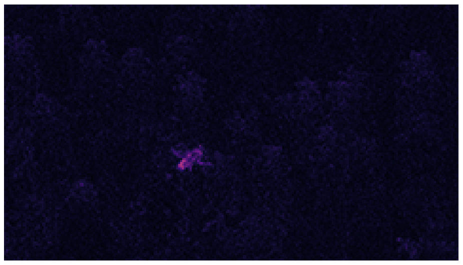
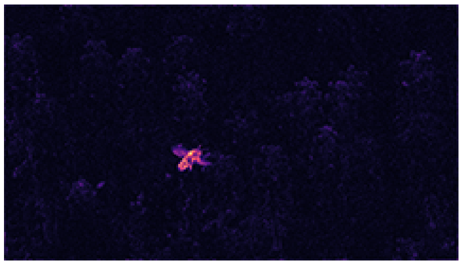
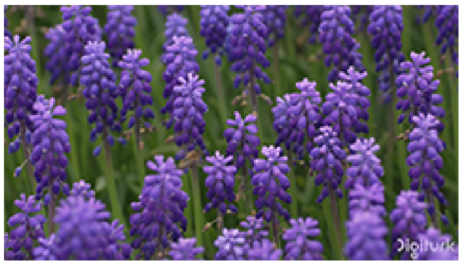
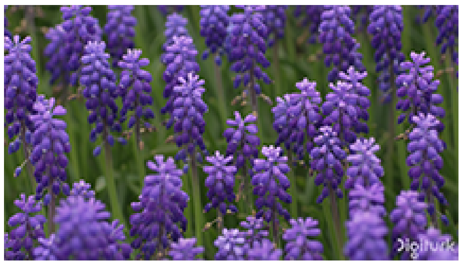
Sequence	Result Type	Proposed (learned scales), BPP=0.060	Trained Dense, BPP=0.064
Bosphorus	FLIP Map	 FLIP=0.1258	 FLIP=0.1613
	Decoded Image		
HoneyBee	FLIP Map	 FLIP=0.0532	 FLIP=0.0534
	Decoded Image		

Table 9. Proposed (learned scales) vs. Trained Sparse

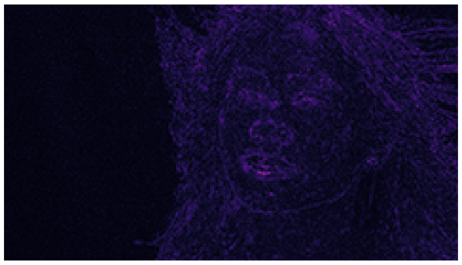
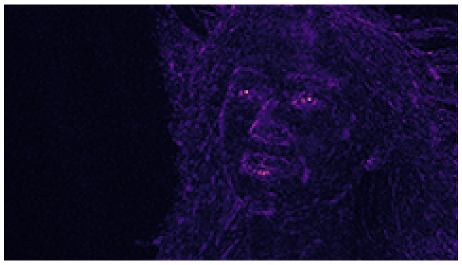
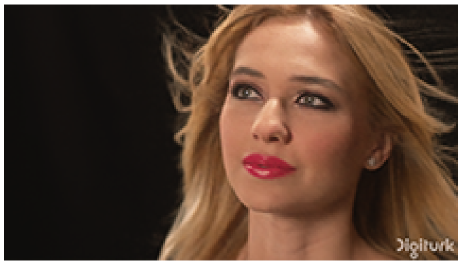
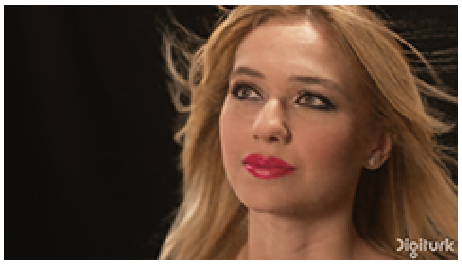
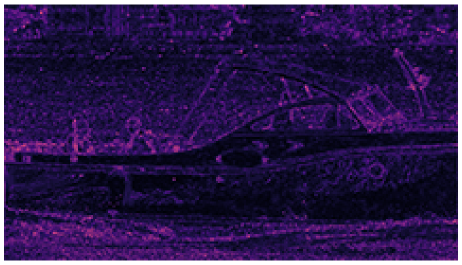
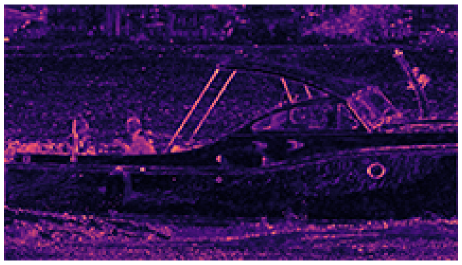
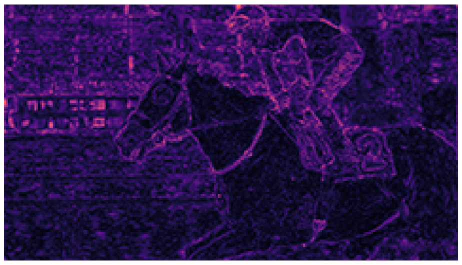
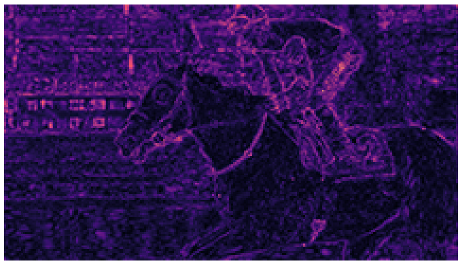
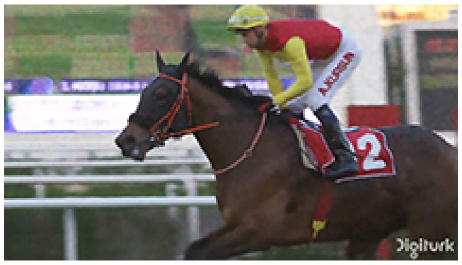
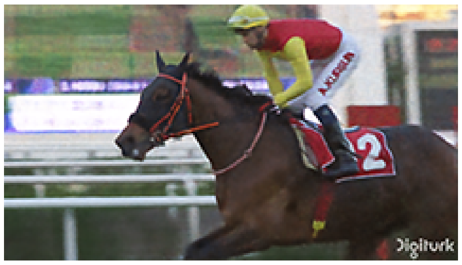
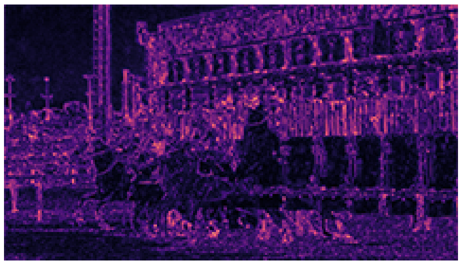
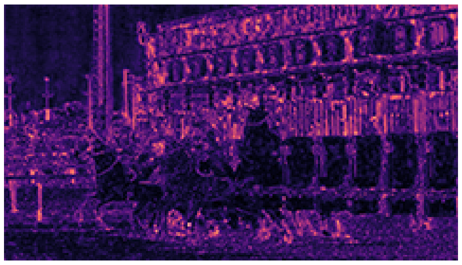
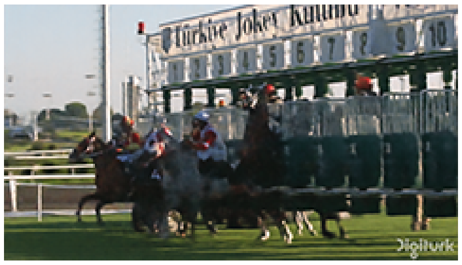
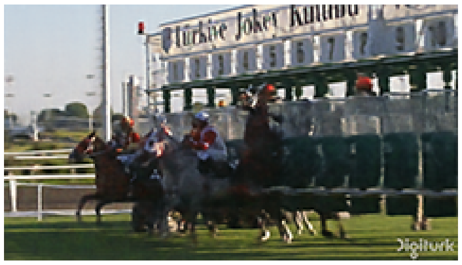
Sequence	Result Type	Proposed (learned scales), BPP=0.060	Trained Sparse, BPP=0.076
Beauty	FLIP Map	 FLIP=0.1258	 FLIP=0.1613
	Decoded Image		
YachtRide	FLIP Map	 FLIP=0.1258	 FLIP=0.1499
	Decoded Image		

Table 10. Proposed (learned scales) vs. Proposed (fixed scales)

Sequence	Result Type	Proposed (learned scales), BPP=0.060	Proposed (fixed scales), BPP=0.060
Jockey	FLIP Map	 FLIP=0.1401	 FLIP=0.1466
	Decoded Image		
Ready Steady Go	FLIP Map	 FLIP=0.1761	 FLIP=0.1932
	Decoded Image		

F. Comparison with Standard Video Codecs

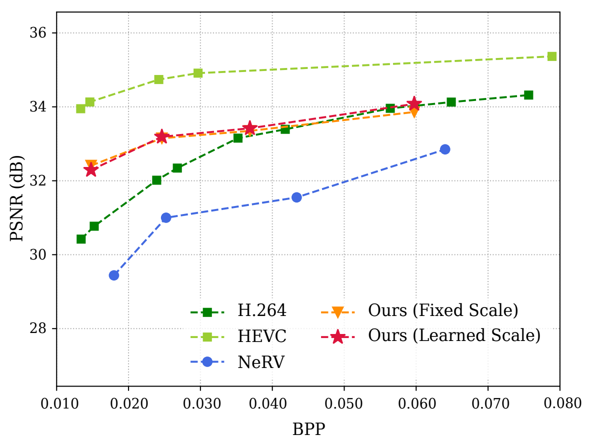
In this section, we provide video-wise comparison results between our method and two video compression standards, H.264 (Wiegand et al., 2003) and HEVC (Sullivan et al., 2012).

Compression Procedure: Following prior works (e.g., Chen et al., 2021a; Kim et al., 2022), `ffmpeg` (Tomar, 2006) commands below are used to compress videos with a medium mode:

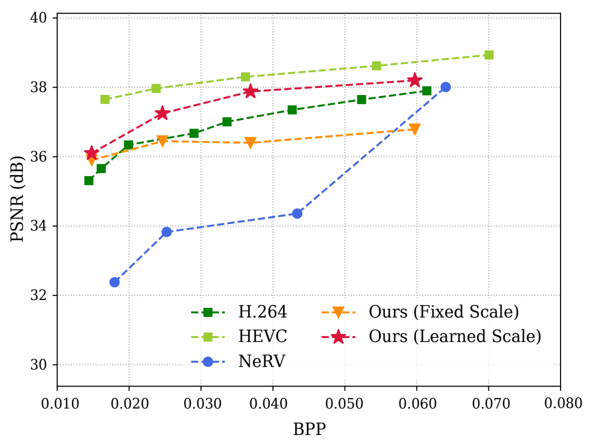
```
ffmpeg -i FILE/f%04d.png -c:v h264 -preset medium -bf 0 -crf CRF FILE.EXT
ffmpeg -i FILE/f%04d.png -c:v hevc -preset medium -x265-params bframes=0 -crf CRF
FILE.EXT
```

where FILE is the file name, CRF is the constant rate factor value, and EXT is the video container format extension.

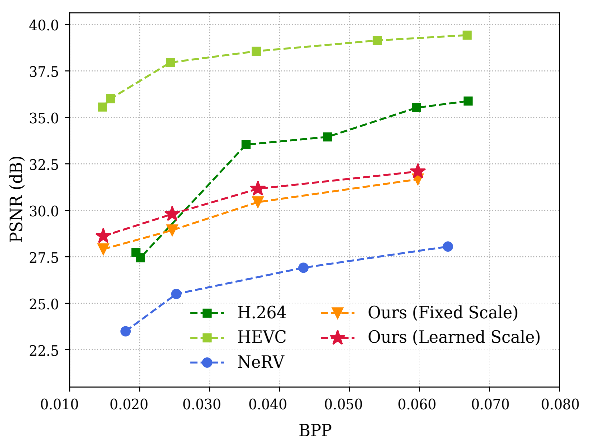
Comparison Results: Figure 13 shows that the proposed framework with learned scales achieves slightly better results than H.264 on low-motion videos (Beauty and HoneyBee). However, H.264 and HEVC capture dynamic motions and high frequency details better as shown in Jockey and ReadySteadyGo sequences. While there exists a general performance gap between carefully optimized video codecs and our method, we believe there are several potential avenues for enhancing the current proposed setup such as designing more specialized INR architectures for video representation and optimizing random weight initialization methods.



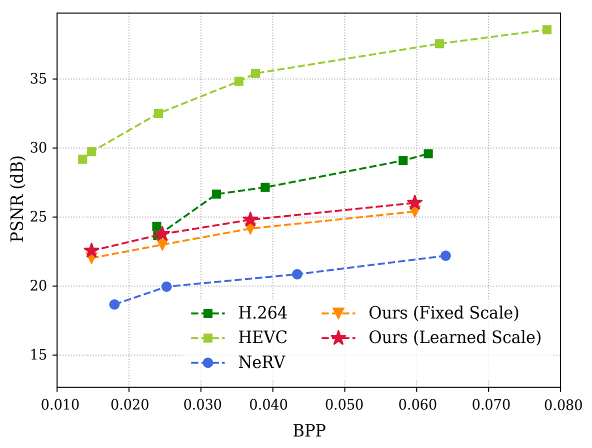
(a) Beauty



(b) HoneyBee



(c) Jockey



(d) ReadySteadyGo

Figure 13. PSNR vs BPP values of the proposed method and video compression standards. The proposed method shows slightly better results than H.264 on low-motion videos (Beauty and HoneyBee) while the legacy codecs significantly outperform INR-based frameworks on high-motion videos (Jockey and ReadySteadyGo).

G. Effect of Choice of Random Seed on Performance

We evaluate how the choice of random seed at initialization affects the performance of our framework. We consider 7 video sequences in UVG dataset (Mercat et al., 2020) and use default experimental settings described in Section 4.1. We prune the first four example models (ID 0-3 in Table 4) using the proposed method with learned scales and total density $k_1 = 0.2$ under 3-supermask setup. In Table 11, we report PSNR values averaged over 5 different random seed runs along with the standard deviation in the parenthesis. In most cases, the standard deviation is less than 0.2. We see that the reconstruction performance is not sensitive to the selection of random seed.

Table 11. Robustness of the proposed framework to the choice of random seed. Random subnetworks are selected to fit Beauty, Bosphorus (Bospho), HoneyBee (HoneyB), Jockey, ReadySteadyGo (Ready), ShakeNDry (ShakeN) and YachtRide (Yacht) videos with learnable scale settings. The PSNR value averaged over 5 different random seed runs are reported along with the standard deviation in the parenthesis. The reconstruction performance is not sensitive to the selection of random seed as the standard deviation is less than 0.2 in most cases.

ID	**Beauty**	**Bospho**	**HoneyB**	**Jockey**	**Ready**	**ShakeN**	**Yacht**
0	32.57 (±0.03)	31.72 (±0.14)	36.36 (±0.17)	28.41 (±0.19)	22.80 (±0.08)	32.23 (±0.15)	26.87 (±0.07)
1	33.18 (±0.06)	32.87 (±0.03)	37.14 (±0.18)	29.68 (±0.19)	23.77 (±0.06)	33.11 (±0.14)	27.69 (±0.05)
2	33.67 (±0.15)	33.51 (±0.18)	37.60 (±0.20)	31.07 (±0.11)	24.84 (±0.13)	33.46 (±0.10)	28.33 (±0.07)
3	33.94 (±0.07)	34.51 (±0.33)	38.19 (±0.16)	32.16 (±0.18)	25.85 (±0.11)	34.18 (±0.20)	29.14 (±0.05)

H. Effect of Model Size on Encoding Time

Finding strong subnetworks in a large dense model requires more iterations because the number of possible subnetworks increases as the number of parameters in the dense model increases. To see this, we provide reconstruction quality (PSNR) of subnetworks obtained from our framework at 50, 100, 150 and 200 epochs on four UVG (Mercat et al., 2020) video sequences (Bosphorus, Jockey, ShakeNDry and YachtRide). We prune three example models (ID 0, 3, and 5 in Table 4) with learnable scales and total density $k_1 = 0.2$ under the 3-supermask setup. In Table 12, the second column (Params) is the number of parameters in each dense model, and Δ is calculated by PSNR at the corresponding epoch minus PSNR at the final epoch (200 epochs). In most cases, more epochs are required to find good subnetworks as the number of parameters in the dense model increases. For example, when compressing YachtRide video with our framework using a small model (ID 0), the performance gap Δ between subnetworks found at 50 and 200 epochs is 0.64 while the corresponding gap using a large model (ID 5) is 3.16.

Table 12. Reconstruction quality at different train epochs. Random subnetworks are selected to fit Bosphorus (Bospho), Jockey, ShakeNDry (ShakeN), and YachtRide (Yacht) videos using the proposed framework. The second column (Params) is the number of parameters in each dense model. Δ is calculated by PSNR at the corresponding epoch minus PSNR at the final epoch (200 epochs). In general, more epochs are required to find good subnetworks as the number of parameters in the dense model increases.

ID	**Params**	**Epochs**	Bospho		Jockey		ShakeN		Yacht	
			PSNR	Δ	PSNR	Δ	PSNR	Δ	PSNR	Δ
0	29.2M	50	30.80	-1.09	27.29	-1.32	21.54	-1.02	26.26	-0.64
		100	31.19	-0.70	27.89	-0.72	22.20	-0.36	26.54	-0.36
		150	31.48	-0.41	28.11	-0.50	22.39	-0.17	26.83	-0.07
		200	31.89		28.61		22.56		26.90	
3	118.1M	50	33.27	-1.25	29.87	-2.22	24.38	-1.65	28.19	-0.95
		100	34.05	-0.47	31.18	-0.91	25.34	-0.69	28.71	-0.43
		150	34.21	-0.31	31.81	-0.28	25.68	-0.35	28.89	-0.25
		200	34.52		32.09		26.03		29.14	
5	380.2M	50	30.30	-5.33	27.05	-5.64	26.58	-2.27	28.26	-3.16
		100	34.39	-1.24	28.65	-4.04	27.81	-1.04	30.04	-1.38
		150	34.41	-1.22	31.76	-0.93	28.34	-0.51	30.37	-1.05
		200	35.63		32.69		28.85		31.42	

I. Pruning Pre-trained Models

We demonstrate the effectiveness of the proposed framework in pruning of pre-trained networks. We use 4 example NeRV (Chen et al., 2021a) networks (ID 1-4 in Table 4) as dense models, and they are trained for 200 epochs using its official implementation to fit YachtRide video sequence of UVG (Mercat et al., 2020) dataset. The pre-trained models are pruned following the model compression pipeline of Chen et al. (2021a) that consists of pruning and re-training (for 50 epochs) such that the proportion of remaining weights is 20%, and we use the resulting sparse models as baselines. We also prune the pre-trained models in the proposed framework such that the total remaining weight density matches to 20% ($k_1 = 0.2$ in our notation) with learnable scales under the 3-supermask setup. We note that pruning in our framework does not involve weight and bias update in the dense models. In Table 13, we report reconstruction (PSNR) and perceptual (MS-SSIM) quality of pre-trained dense models, baseline sparse models, and sparse models obtained from the proposed framework with 50 and 200 pruning epochs. We see that our 50-epoch sparse models show very competitive results compared to the baseline sparse models even though our framework does not re-train network parameters. Moreover, our 200-epoch sparse models almost match the performance of the baseline sparse models. For example, our 50-epoch and 200-epoch sparse models with ID 1 achieve 27.75 and 28.35 PSNR values, respectively, while the corresponding value of the baseline sparse model is 28.41.

Table 13. Effectiveness of the proposed method in pruning pre-trained models. Dense models are trained to fit YachtRide video. Baseline sparse models are obtained from pruning and re-training such that the proportion of remaining weights is 20%. However, our framework does not involve weight and bias update in the dense models. We report encoding quality of the sparse models using PSNR and MS-SSIM. Our 50-epoch sparse models show very competitive results compared to the baseline sparse models even though the proposed framework does not re-train network parameters.

Method	**Weight Remaining**	**Prune Epoch**	**PSNR**				**MS-SSIM**			
			ID 1	**ID 2**	**ID 3**	**ID 4**	**ID 1**	**ID 2**	**ID 3**	**ID 4**
Dense	100%	-	31.87	33.22	35.51	36.76	0.9562	0.9680	0.9815	0.9861
Baseline	20%	50	28.41	29.40	30.46	31.71	0.9042	0.9218	0.9374	0.9508
Ours (50-epoch)	20%	50	27.75	28.35	28.87	29.36	0.8664	0.8847	0.8946	0.9065
Ours (200-epoch)	20%	200	28.35	29.18	30.00	30.80	0.8798	0.8987	0.9134	0.9271