OPTIMIZATION IS PART OF THE CODEC: TRAINING COMPRESSIBLE IMPLICIT NEURAL VIDEO REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Implicit neural representations (INRs) compress a video by fitting a network to it and then pruning and quantizing the network into a bitstream. This pipeline implicitly assumes that improvements in the dense representation translate into improvements after compression. We show that this assumption can fail: as dense fitting improves reconstruction quality, compressed quality eventually declines. The decline stems from a spectral drift of weight energy toward weakly excited input directions, which lowers a sensitivity factor shared by pruning and quantization. A local spectral analysis of the optimizer indicates that weight decay sets where the weights are driven, while the learning rate sets how far. Without weight decay, a larger learning rate deepens the drift, whereas with weight decay it reverses the drift. Building on this, we propose SpecFit (Spectrally-Guided Dense Fitting), a training strategy that pairs moderate weight decay with a large learning rate to reverse the drift. SpecFit leaves the architecture, pruning criterion, and quantizer unchanged, making it a drop-in strategy for NeRV-based video compression. Relative to the standard training of each backbone, it reduces the PSNR BD-rate by 23–28% on average across three NeRV backbones, and it remains effective under a post-training quantization pipeline. Within the standard three-stage pipeline, SpecFit achieves rate–distortion performance competitive with NVRC without NVRC's learned entropy-modeling framework. This simpler decoder achieves a 65.8× speedup in model-decoding time and increases full decoding throughput from 3.08 to 19.75 FPS. Our findings reveal that optimization in video INRs is not merely a procedure for improving dense reconstruction, but a key mechanism that determines how compressible the learned representation ultimately becomes.

1 INTRODUCTION

Efficient video compression remains essential for large-scale video storage and transmission. Whereas conventional and learned codecs encode videos into transform coefficients or latent representations reconstructed by a fixed decoder (Wiegand et al., 2003; Sullivan et al., 2012; Bross et al., 2021; Lu et al., 2019; Hu et al., 2021; Li et al., 2021), implicit neural representations (INRs) encode a signal directly in the parameters of a network fitted to it (Sitzmann et al., 2020; Tancik et al., 2020), so that compressing the signal amounts to compressing the network (Dupont et al., 2021).

Existing video INRs (Chen et al., 2021; 2023; Wang et al., 2026a) typically follow a common pipeline: a network is first densely fitted to reconstruct the video, then compressed by pruning, quantization, and entropy coding. Most existing methods implicitly regard compressibility as a property of the architecture and the subsequent compression stage, while treating dense fitting as a process that should reconstruct the video as accurately as possible. Accordingly, prior work has focused on stronger architectures (Li et al., 2022; He et al., 2023; Lee et al., 2023; Wang et al., 2026b; Zhu et al., 2026b; Zhao et al., 2025; Zhu et al., 2026a; Zhang et al., 2024b) and improved compression-stage procedures (Kwan et al., 2024; Shi et al., 2025; Zhang et al., 2024a; Kwan et al., 2025; Gwilliam et al., 2026; Wu et al., 2025). Under this assumption, a densely fitted model with higher reconstruction quality should provide a better starting point for compression.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, this assumption can fail,, as dense fitting can continue to improve reconstruction quality while increasing the model's vulnerability to compression. As fitting proceeds, pre-compression PSNR continues to rise, whereas PSNR after pruning and quantization declines, widening the gap from about 0.2 dB to more than 5 dB (Fig. 1(a)). This vulnerability is concentrated in a few linear layers, where pruning-induced output perturbations grow relative to the layer output energy. This increase does not arise from pruning more weight energy. Instead, dense fitting shifts the weight energy of these layers toward weakly excited input directions, a phenomenon we term *spectral drift*. The small weights removed by pruning do not follow this shift and therefore couple more strongly to the layer input than the layer as a whole. These observations indicate that compressibility depends not only on the architecture and the compression stage, but also on dense fitting itself.

To characterize this dependence on dense fitting, we examine the *spectral position* of the weights, a sensitivity factor shared by pruning and quantization. Further analysis shows that spectral drift lowers this factor, which becomes fixed once dense fitting ends and cannot be corrected by post-training compression. Addressing spectral drift therefore requires intervention during dense fitting. A local spectral analysis of the optimizer reveals how this drift arises: weights along weakly excited input directions converge last and settle at larger norms. In this analysis, weight decay determines the optimization endpoint, while the learning rate controls how closely training approaches it, amplifying the effect of weight decay at larger learning rates. We therefore propose SpecFit (Spectrally-Guided Dense Fitting), which combines moderate weight decay with a large learning rate to address spectral drift during dense fitting. SpecFit modifies only dense fitting and leaves the architecture, pruning criterion, and quantizer unchanged, making it a drop-in addition to existing NeRV pipelines.

Extensive experiments provide strong empirical evidence for the effectiveness and generality of SpecFit. Built upon MSNeRV (Zhu et al., 2025), SpecFit achieves the best rate–distortion performance among all compared codecs on HEVC Class B, reducing BD-rate by 15.09% (PSNR) and 40.32% (MS-SSIM) relative to VTM (Bross et al., 2021). The gains consistently transfer to other INR backbones, including HiNeRV (Kwan et al., 2023), RNeRV (Zhang et al., 2026), and HANeRV (Tang et al., 2026), and remain effective with the post-training mixed-precision quantization of NeuroQuant (Shi et al., 2025). Compared with NVRC (Kwan et al., 2024) under the same representation architecture, SpecFit reaches comparable PSNR with 27% less encoding time using only the standard three-stage HiNeRV pipeline, while enabling real-time 1080p decoding at 50.4 FPS on three GPUs. Our findings suggest that compression robustness in INRs is largely determined before compression begins, highlighting optimization as a key factor alongside network architecture and compression design. Code and models will be released.

2 Understanding Optimization-Induced Compression Vulnerability

2.1 Problem Setup and INR Compression Pipeline

Preliminaries. A video is denoted as $\mathcal{V} = \{\boldsymbol{x}_t\}_{t=1}^{T_\mathrm{v}}$, where $\boldsymbol{x}_t$ is the t-th frame and T_v is the total number of frames. An implicit neural representation (INR) $f_{\boldsymbol{\theta}}$ encodes the video in the parameters of a neural network, which maps each frame index t to its reconstruction $\hat{\mathbf{x}}_t = f_{\boldsymbol{\theta}}(t)$. Here, $\boldsymbol{\theta} = \{\boldsymbol{W}_l\}_{l=1}^{L}$ collects the weights of all L layers, and $w_{l,i}$ denotes the i-th entry of $\boldsymbol{W}_l$. The parameters are fitted by minimizing

$$\mathcal{L}_{\mathrm{rec}}(\boldsymbol{\theta}) = \frac{1}{T_\mathrm{v}} \sum_{t=1}^{T_\mathrm{v}} d\big(\hat{\boldsymbol{x}}_t, \boldsymbol{x}_t\big), \tag{1}$$

where $d(\cdot,\cdot)$ denotes the reconstruction distortion.

Compression pipeline. HiNeRV (Kwan et al., 2023) compresses $f_{\boldsymbol{\theta}}$ in three stages: (i) *dense fitting*, which optimizes all parameters under Eq. 1, (ii) *pruning followed by sparse fine-tuning*, and (iii) *quantization-aware fine-tuning* (QAT). During pruning, each eligible weight $w_{l,i}$ receives a layer-normalized magnitude score, which determines its binary pruning mask:

$$s_{l,i} = \frac{|w_{l,i}|}{N_l^{\gamma} + \epsilon}, \qquad M_{l,i} = \mathbb{I}(s_{l,i} > \tau_p), \tag{2}$$

where N_l is the number of unpruned weights in layer l, γ controls layer-size normalization, and $\epsilon > 0$ ensures numerical stability. The indicator function $\mathbb{I}(\cdot)$ retains weights whose scores exceed the global threshold τ_p, chosen to achieve the target pruning ratio. Sparse fine-tuning then optimizes the retained parameters under the fixed mask. QAT further adapts these parameters to quantization effects, after which the quantized parameters are entropy coded into the final bitstream.

2.2 Dense Fitting Can Worsen Compressibility

Dense fitting can improve reconstruction quality while making the model more vulnerable to compression. To track this effect, we compress the INR at each checkpoint by pruning 20% of the eligible weights using Eq. 2, followed by 6-bit quantization, without fine-tuning. Under this protocol, pre- and post-compression PSNR diverge as dense fitting proceeds (Fig. 1(a)). After epoch 60, the pre-compression PSNR of standard dense fitting continues to improve, while its post-compression PSNR declines. The gap widens from approximately 0.2 dB at epoch 29 to more than 5 dB at epoch 299. Thus, improving dense reconstruction does not ensure better reconstruction after compression.

This growing compression vulnerability is unevenly distributed across layers (Fig. 1(b)). We report the three most sensitive layers and the least sensitive layer, using Si-Bj-`fc1` to denote the first linear layer in block j of stage i. All three most sensitive layers are `fc1` layers, and compressing each layer causes a larger PSNR drop later in training. At epoch 299, compressing S0-B2-`fc1` alone incurs a PSNR loss of approximately 4.3 dB, approaching the roughly 5 dB loss from compressing the entire model. By contrast, the loss from compressing the stem convolution remains near zero throughout fitting. These differences motivate our subsequent focus on the `fc1` layers.

2.3 Dense Fitting Reshapes Pruning Perturbation

Layer-wise pruning perturbation. Consider an `fc1` layer l with weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$ and input $\boldsymbol{x}_l \in \mathbb{R}^{C_{\text{in}}}$. For a pruning mask $\boldsymbol{M}_l$, the removed weights are

$$\Delta \boldsymbol{W}_l = (\mathbf{1} - \boldsymbol{M}_l) \odot \boldsymbol{W}_l, \tag{3}$$

where $\odot$ is the element-wise product and $\mathbf{1}$ is the all-ones matrix. For the same input, pruning changes the pre-activation output by $-\Delta \boldsymbol{W}_l \boldsymbol{x}_l$, with expected squared norm

$$\mathcal{E}_l = \mathbb{E}\big\|\Delta \boldsymbol{W}_l \boldsymbol{x}_l\big\|_2^2 = \mathbb{E}\big[\mathrm{Tr}\big(\Delta \boldsymbol{W}_l \boldsymbol{x}_l \boldsymbol{x}_l^\top \Delta \boldsymbol{W}_l^\top\big)\big] = \mathrm{Tr}\big(\Delta \boldsymbol{W}_l \boldsymbol{\Sigma}_{x,l} \Delta \boldsymbol{W}_l^\top\big)\,, \tag{4}$$

where $\boldsymbol{\Sigma}_{x,l} = \mathbb{E}[\boldsymbol{x}_l \boldsymbol{x}_l^\top]$ is the second-moment matrix of the layer input. $\mathcal{E}_l$ is the layer-wise reconstruction objective of L-OBS, OBC, and SparseGPT (Dong et al., 2017; Frantar & Alistarh, 2022; 2023) without weight compensation. Since the layer output energy $\mathbb{E}\|\boldsymbol{W}_l \boldsymbol{x}_l\|_2^2 = \mathrm{Tr}(\boldsymbol{W}_l \boldsymbol{\Sigma}_{x,l} \boldsymbol{W}_l^\top)$ also changes during fitting, we normalize $\mathcal{E}_l$ by it:

$$\phi_l = \frac{\mathcal{E}_l}{\mathbb{E}\big\|\boldsymbol{W}_l \boldsymbol{x}_l\big\|_2^2} = \frac{\mathrm{Tr}\big(\Delta \boldsymbol{W}_l \boldsymbol{\Sigma}_{x,l} \Delta \boldsymbol{W}_l^\top\big)}{\mathrm{Tr}\big(\boldsymbol{W}_l \boldsymbol{\Sigma}_{x,l} \boldsymbol{W}_l^\top\big)}. \tag{5}$$

Thus, $\sqrt{\phi_l}$ measures the root-mean-square pruning perturbation relative to the root-mean-square linear output. Under the same 20% pruning setting as Sec. 2.2, with masks selected using Eq. 2, ϕ_l increases throughout dense fitting in the evaluated `fc1` layers (Fig. 1(c)). Pruning therefore induces an increasingly large perturbation relative to the output energy of these layers.

Weight-energy and input-coupling factors. A larger ϕ_l can arise from two sources: pruning may remove a larger fraction of weight energy, or it may remove weights that act on higher-energy input directions. To separate these effects, let $\boldsymbol{\Sigma}_{x,l} = \boldsymbol{V}_l \,\mathrm{diag}(e_1, \ldots, e_{C_{\text{in}}}) \boldsymbol{V}_l^\top = \sum_{j=1}^{C_{\text{in}}} e_j \boldsymbol{v}_j \boldsymbol{v}_j^\top$ be the eigendecomposition of the input second-moment matrix, where e_j is the input energy along the eigenvector $\boldsymbol{v}_j$, and let $\bar{e} = \mathrm{Tr}(\boldsymbol{\Sigma}_{x,l})/C_{\text{in}}$ denote the mean input energy. We omit the layer index on e_j, $\boldsymbol{v}_j$, $\bar{e}$, and the energy shares $\pi_j(\cdot)$ defined below for brevity. For any nonzero matrix $\boldsymbol{A}$, we define the normalized spectral factor

$$\kappa_l(\boldsymbol{A}) = \frac{C_{\text{in}} \,\mathrm{Tr}(\boldsymbol{A} \boldsymbol{\Sigma}_{x,l} \boldsymbol{A}^\top)}{\|\boldsymbol{A}\|_F^2 \,\mathrm{Tr}(\boldsymbol{\Sigma}_{x,l})} = \frac{1}{\bar{e}} \frac{\sum_j e_j \|\boldsymbol{A} \boldsymbol{v}_j\|_2^2}{\sum_j \|\boldsymbol{A} \boldsymbol{v}_j\|_2^2} = \frac{1}{\bar{e}} \sum_j \pi_j(\boldsymbol{A})\, e_j, \quad \pi_j(\boldsymbol{A}) = \frac{\|\boldsymbol{A} \boldsymbol{v}_j\|_2^2}{\|\boldsymbol{A}\|_F^2}. \tag{6}$$

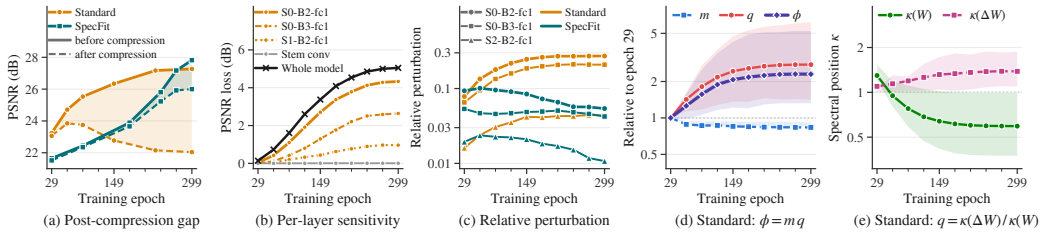


Figure 1: **Dense optimization progressively induces compression vulnerability.** (a) Growing pre-/post-compression PSNR gap. (b) Increasing layer-wise pruning sensitivity. (c) Increasing relative pruning perturbation in representative `fc1` layers. (d) Exact decomposition $\phi = mq$: decreasing weight-energy factor m but increasing spectral factor q. (e) Spectral factors underlying q: decreasing $\kappa(\boldsymbol{W})$ and comparatively stable $\kappa(\Delta\boldsymbol{W})$.

The factor $\kappa_l(\boldsymbol{A})$ measures the output energy induced per unit weight energy under the input distribution of layer l, relative to the mean input energy $\bar{e}$. Equivalently, it is a weighted average of the input energies e_j, where the weight $\pi_j(\boldsymbol{A})$ is the fraction of the energy of $\boldsymbol{A}$ along the eigenvector $\boldsymbol{v}_j$, so that $\sum_j \pi_j(\boldsymbol{A}) = 1$. Hence $\kappa_l(\boldsymbol{A}) = 1$ when $\boldsymbol{A}$ spreads its energy uniformly across eigendirections, and $\kappa_l(\boldsymbol{A})$ exceeds 1 when $\boldsymbol{A}$ concentrates on higher-energy input directions and falls below 1 when it concentrates on lower-energy ones.

Rearranging Eq. 6 expresses the induced output energy of any $\boldsymbol{A}$ as the product of its weight energy and its spectral factor, $\mathrm{Tr}\big(\boldsymbol{A}\boldsymbol{\Sigma}_{x,l}\boldsymbol{A}^\top\big) = \bar{e}\,\|\boldsymbol{A}\|_F^2\,\kappa_l(\boldsymbol{A})$. Applying this to both the numerator ($\boldsymbol{A} = \Delta\boldsymbol{W}_l$) and the denominator ($\boldsymbol{A} = \boldsymbol{W}_l$) of Eq. 5, the factor $\bar{e}$ cancels and yields the exact factorization

$$\phi_l = \frac{\mathrm{Tr}\big(\Delta\boldsymbol{W}_l\boldsymbol{\Sigma}_{x,l}\Delta\boldsymbol{W}_l^\top\big)}{\mathrm{Tr}\big(\boldsymbol{W}_l\boldsymbol{\Sigma}_{x,l}\boldsymbol{W}_l^\top\big)} = \frac{\bar{e}\,\|\Delta\boldsymbol{W}_l\|_F^2\,\kappa_l(\Delta\boldsymbol{W}_l)}{\bar{e}\,\|\boldsymbol{W}_l\|_F^2\,\kappa_l(\boldsymbol{W}_l)} = \underbrace{\frac{\|\Delta\boldsymbol{W}_l\|_F^2}{\|\boldsymbol{W}_l\|_F^2}}_{m_l} \cdot \underbrace{\frac{\kappa_l(\Delta\boldsymbol{W}_l)}{\kappa_l(\boldsymbol{W}_l)}}_{q_l}, \tag{7}$$

where m_l is the fraction of weight energy removed by pruning and q_l is the input coupling of the removed weights relative to the full layer. Within each layer, the criterion in Eq. 2 minimizes m_l for a given number of removed weights but ignores q_l.

We track this decomposition over ten checkpoints spaced along dense fitting. As shown in Fig. 1(d), in all examined `fc1` layers ϕ_l increases while m_l decreases: the pruned weights carry a shrinking share of the weight energy, yet cause a growing output perturbation. Since m_l decreases while $\phi_l = m_l q_l$ increases, the growth of ϕ_l must arise from the increase in q_l: relative to the layer as a whole, the removed weights induce more output energy per unit weight energy. In other words, weight magnitude becomes an increasingly unreliable proxy for output contribution as fitting proceeds.

Fig. 1(e) resolves q_l into its two spectral factors. During fitting, $\kappa_l(\boldsymbol{W}_l)$ drops from above to below 1, whereas $\kappa_l(\Delta\boldsymbol{W}_l)$ increases only slightly. The two curves thus cross and separate, and the growth of q_l stems mainly from the decline of $\kappa_l(\boldsymbol{W}_l)$. This decline means that the layer shifts an increasing share of its weight energy toward weakly excited input directions, and a value below 1 means that this share exceeds that of a uniform allocation. We refer to this progressive shift as *spectral drift*.

3 SpecFit: Spectrally-Guided Dense Fitting

Sec. 2.3 showed that dense fitting induces spectral drift, a decline in $\kappa(\boldsymbol{W})$ that raises the relative pruning perturbation ϕ. This section explains why the fix belongs in dense fitting, how the optimizer moves $\kappa(\boldsymbol{W})$, and what SpecFit changes. We drop the layer index l.

3.1 Why Act during Dense Fitting

Pruning and quantization share a dependence on the spectral position of the dense weights. We write ϕ^{P} (P for pruning) for the relative pruning perturbation ϕ of Eq. 5, and ϕ^{Q} (Q for quantization) for its counterpart obtained by replacing $\Delta\boldsymbol{W}$ with the rounding error $\boldsymbol{N}$ and averaging over $\boldsymbol{N}$. For

quantization, we model $\boldsymbol{N}$ as independent of the input, with zero-mean entries that are uncorrelated and share a common variance within each row. Then

$$\phi^{\mathrm{P}} = m\,\frac{\kappa(\Delta \boldsymbol{W})}{\kappa(\boldsymbol{W})}, \qquad \phi^{\mathrm{Q}} = \frac{\mathbb{E}_{\boldsymbol{N}}\,\mathrm{Tr}\big(\boldsymbol{N}\boldsymbol{\Sigma}_x\boldsymbol{N}^\top\big)}{\mathrm{Tr}(\boldsymbol{W}\boldsymbol{\Sigma}_x\boldsymbol{W}^\top)} = \frac{\nu}{\kappa(\boldsymbol{W})}, \tag{8}$$

where ϕ^{P} is factorized as in Eq. 7, the last equality follows from the noise model (Appendix A), and $\nu = \mathbb{E}_{\boldsymbol{N}}\|\boldsymbol{N}\|_F^2/\|\boldsymbol{W}\|_F^2$ is the relative energy of the rounding error. Thus, both perturbations contain the shared sensitivity factor $1/\kappa(\boldsymbol{W})$. Unlike the perturbation-specific terms, this factor is already determined by the dense solution and cannot be altered by a post-training compression rule that keeps $\boldsymbol{W}$ fixed. This motivates shaping $\kappa(\boldsymbol{W})$ during dense fitting.

3.2 How the Optimizer Moves $\kappa(\boldsymbol{W})$

Let $\boldsymbol{\Sigma}_x = \sum_j e_j \boldsymbol{v}_j \boldsymbol{v}_j^\top$ and $\widetilde{\boldsymbol{w}}_j = \boldsymbol{W}\boldsymbol{v}_j$. Specializing Eq. 6 to $\boldsymbol{A} = \boldsymbol{W}$ and writing $\pi_j \equiv \pi_j(\boldsymbol{W})$, the spectral position is a weighted average of the eigenvalues,

$$\kappa(\boldsymbol{W}) = \frac{1}{\bar{e}}\sum_j \pi_j\, e_j, \qquad \pi_j = \frac{\|\widetilde{\boldsymbol{w}}_j\|_2^2}{\|\boldsymbol{W}\|_{\mathrm{F}}^2}, \tag{9}$$

where π_j is the share of weight energy in mode j and $\bar{e}$ is the mean eigenvalue. $\kappa(\boldsymbol{W})$ is high when the weights sit on strong modes, low when they sit on weak modes.

We model the layer as minimizing $\frac{1}{2}\mathbb{E}\|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{x}\|^2$ for a target output $\boldsymbol{y}$. This loss decouples across modes. Let η_t be the learning rate at step t and λ the weight-decay coefficient of AdamW (Loshchilov & Hutter, 2019), whose decay term is $\eta_t\lambda\boldsymbol{W}$. We approximate AdamW by dropping momentum and replacing its elementwise preconditioner $\sqrt{\hat{v}_t}$, the second-moment estimate, with a fixed scalar $\bar{p} > 0$. In the small-step limit,

$$\widetilde{\boldsymbol{w}}_j(\widetilde{S}) - \widetilde{\boldsymbol{w}}_j^\star(\alpha) = e^{-(e_j+\alpha)\widetilde{S}}\big(\widetilde{\boldsymbol{w}}_j(0) - \widetilde{\boldsymbol{w}}_j^\star(\alpha)\big), \qquad \widetilde{\boldsymbol{w}}_j^\star(\alpha) = \frac{\boldsymbol{c}_j}{e_j+\alpha}, \tag{10}$$

where $\boldsymbol{c}_j = \boldsymbol{C}\boldsymbol{v}_j$ with $\boldsymbol{C} = \mathbb{E}[\boldsymbol{y}\boldsymbol{x}^\top]$, $\alpha = \lambda\bar{p}$, and $\widetilde{S} = \sum_{t<T}\eta_t/\bar{p}$ after T steps. Each mode relaxes toward its target at rate $e_j + \alpha$.

Weight decay raises the endpoint. As $\widetilde{S}$ grows, $\boldsymbol{W}$ approaches the ridge solution $\boldsymbol{W}^\star(\alpha) = \boldsymbol{C}(\boldsymbol{\Sigma}_x + \alpha\boldsymbol{I})^{-1}$. By Eq. 10, its energy shares $\pi_j^\star(\alpha) \equiv \pi_j\big(\boldsymbol{W}^\star(\alpha)\big)$ satisfy

$$\pi_j^\star(\alpha) \;\propto\; \frac{\|\boldsymbol{c}_j\|_2^2}{(e_j+\alpha)^2}. \tag{11}$$

Without decay, the denominator is e_j^2, so weak modes take a large share and the endpoint $\kappa^\star(0)$ is low. This is the drift of Sec. 2.3. Decay adds α to every denominator. The relative change is largest for small e_j, so decay moves energy from weak modes to strong ones. The endpoint $\kappa^\star(\alpha) = \kappa\big(\boldsymbol{W}^\star(\alpha)\big)$ therefore rises with α.

Learning rate sets how far training gets. The learning rate enters Eq. 10 only through $\widetilde{S}$. It leaves the target $\widetilde{\boldsymbol{w}}_j^\star(\alpha)$ unchanged and shrinks the residual of mode j by $e^{-(e_j+\alpha)\widetilde{S}}$, so a larger learning rate brings mode closer to its target within the same steps. Whether this helps depends on where the endpoint lies. Without decay, strong modes settle first, and the progress fills weak modes toward their large targets $\boldsymbol{c}_j/e_j$. Since $\kappa^\star(0)$ is low, a larger learning rate carries $\boldsymbol{W}$ further along the drift. With decay, the same progress carries $\boldsymbol{W}$ toward $\boldsymbol{W}^\star(\alpha)$ and $\kappa(\boldsymbol{W})$ toward the raised endpoint $\kappa^\star(\alpha)$. The learning rate therefore amplifies the direction set by decay but does not choose it.

3.3 What SpecFit Changes

Based on the preceding analysis, we propose SpecFit, a training strategy that combines moderate weight decay with a large learning rate. Specifically, moderate weight decay raises the endpoint $\kappa^\star(\alpha)$, while the associated cost remains small for modes with $e_j \gg \alpha$. A large learning rate advances each layer further toward this raised endpoint within the same number of training steps. The architecture, pruning criterion, and quantizer retain their standard settings.

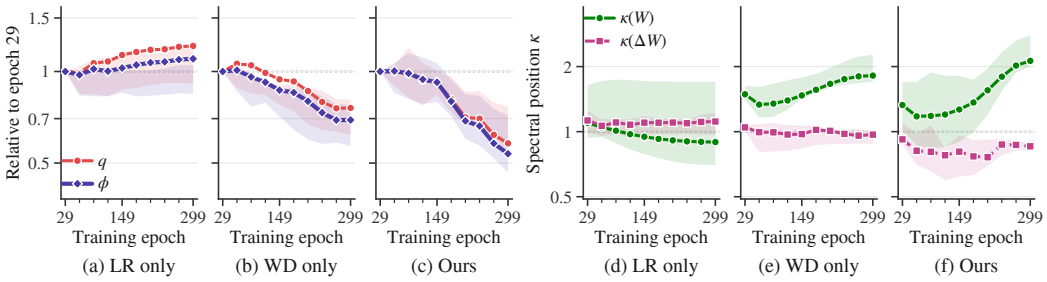


Figure 2: **Learning rate and weight decay reshape the spectral characteristics of pruning perturbations.** Under 20% magnitude pruning of each `fc1` layer, (a–c) show the evolution of q and ϕ relative to epoch 29, and (d–f) the corresponding spectral positions $\kappa(\boldsymbol{W})$ and $\kappa(\Delta\boldsymbol{W})$.

Table 1: Rate–distortion comparison on MCL-JCV and HEVC-B. BD-rate (%) is computed with VTM as the anchor (lower is better). Full results are provided in Table 13 (Appendix F).

Dataset	Metric	VTM	DCVC-RT	HiNeRV	NVRC	MSNeRV	Ours
MCL-JCV	PSNR	**0.00**	3.55	73.19	17.66	15.55	7.33
	MS-SSIM	0.00	-0.08	44.00	-13.48	-8.21	**-13.55**
HEVC-B	PSNR	0.00	42.25	69.69	-8.35	-4.25	**-15.09**
	MS-SSIM	0.00	23.29	1.86	-40.01	-29.98	**-40.32**

Mechanistic verification. To isolate the two components of SpecFit, Fig. 2 compares three runs that depart from the standard recipe in the learning rate, the weight decay, or both. LR only uses the large learning rate of SpecFit with no weight decay, as in the standard recipe. WD only uses the weight decay of SpecFit with the standard learning rate. SpecFit (Ours) combines both. With LR only, both q and ϕ increase, as $\kappa(\boldsymbol{W})$ decreases and $\kappa(\Delta\boldsymbol{W})$ remains nearly unchanged. Adding weight decay reverses this behavior: in both WD only and SpecFit, q and ϕ decrease, while $\kappa(\boldsymbol{W})$ rises after an initial dip. Under the same weight decay, SpecFit's larger learning rate drives a stronger increase in $\kappa(\boldsymbol{W})$: its value at epoch 299 is about $1.60\times$ that at epoch 29, compared with $1.22\times$ for WD only. Correspondingly, the endpoint ratios of q and ϕ are 0.58 and 0.54 for SpecFit, versus 0.76 and 0.69 for WD only. Because $\kappa(\Delta\boldsymbol{W})$ changes similarly in the two runs, the larger drop in q is primarily associated with the stronger rise in $\kappa(\boldsymbol{W})$.

4 Experiments

4.1 Experimental Setup

Datasets. The evaluation of our method is based on commonly used datasets: UVG (Mercat et al., 2020), HEVC Class B , HEVC Class E (Bossen, 2013), Surveillance (Zhao et al., 2021), SCC (Xu & Liu, 2021), and MCL-JCV (Wang et al., 2016).

Evaluation Metrics. Reconstruction quality is evaluated using peak signal-to-noise ratio (PSNR) and multi-scale structural similarity (MS-SSIM) (Wang et al., 2003). The coding rate is measured in bits per pixel (bpp) from the actual entropy-coded bitstream. Overall rate–distortion performance is further assessed using Bjøntegaard delta rate (BD-rate) (Bjontegaard, 2001), where a lower value indicates better compression efficiency.

4.2 Main Results

Overall Rate–Distortion Performance. We build our method upon MSNeRV (Zhu et al., 2025) and compare it with representative video codecs. Using VVC VTM (Bross et al., 2021) as the anchor, we compare against the learned video codec DCVC-RT (Jia et al., 2025) and the INR-based codecs HiNeRV (Kwan et al., 2023), MSNeRV, and NVRC (Kwan et al., 2024). Table 1 reports BD-rate on MCL-JCV and HEVC-B, and Fig. 3(a) shows the rate–distortion curves on HEVC-B. On HEVC-

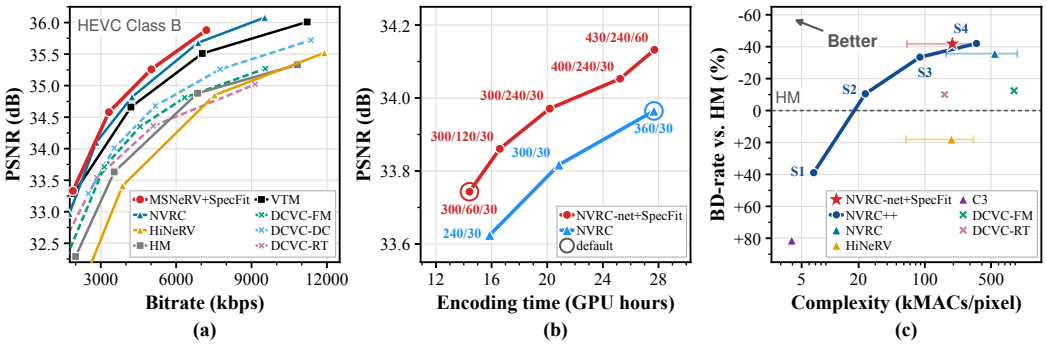


Figure 3: Rate–distortion performance, encoding cost, and decoding efficiency. (a) Rate–distortion comparison (PSNR) on the HEVC Class B dataset. (b) Encoding time–quality comparison with NVRC at the same bitrate. Labels give the number of training epochs per stage. (c) BD-rate (PSNR) relative to HM as a function of decoding complexity, measured in kMACs/pixel.

Table 2: Generalization across different INR backbones. BD-rate (%) is computed relative to the original optimization configuration of each backbone (lower is better). Avg. denotes the average over all six datasets (UVG, HEVC-B, HEVC-E, Surveillance, SCC, and MCL-JCV). Per-dataset results are provided in Table 14 (Appendix F).

Backbone	UVG		HEVC-B		MCL-JCV		Avg.	
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
HiNeRV	-20.20	-19.49	-34.06	-32.76	-26.73	-24.32	-26.83	-24.89
RNeRV	-25.43	-24.42	-34.41	-33.33	-25.10	-24.46	-27.83	-26.44
HANeRV	-18.81	-18.23	-26.23	-24.19	-23.12	-22.99	-23.45	-22.74
NeuroQuant	-20.21	-19.34	-33.11	-32.72	-23.91	-23.11	-26.66	-25.69

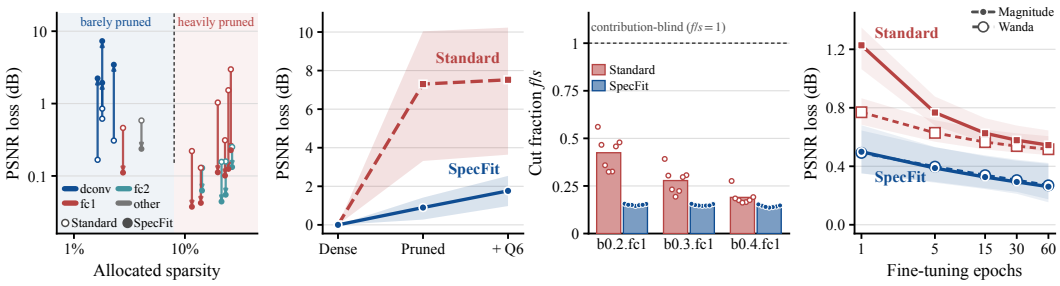
B, our method achieves the best rate–distortion performance among all compared codecs, reducing BD-rate by 15.09% (PSNR) and 40.32% (MS-SSIM) relative to VTM. On MCL-JCV, our method achieves the best MS-SSIM-based BD-rate, while under PSNR, VTM and DCVC-RT (Jia et al., 2025) remain more efficient. Nevertheless, our method outperforms all INR-based codecs.

Efficiency against state-of-the-art neural codecs. Under the same representation architecture, our $300/240/30$ configuration achieves PSNR comparable to that of NVRC (Kwan et al., 2024) while requiring 27% less encoding time (Fig. 3(b)). Fig. 3(c) shows that our method achieves better coding efficiency than NVRC, and the DCVC variants (Li et al., 2023; 2024; Jia et al., 2025), while requiring lower decoding complexity than high-quality NVRC++ at comparable performance. Although C3 (Kim et al., 2024) and the low-complexity NVRC++ (Kwan et al., 2026) configurations achieve lower decoding complexity, this comes at the cost of substantially lower compression quality. Compared with NVRC, our method achieves a 65.8× speedup in model decoding and improves full decoding throughput by 6.4× on a single RTX A6000, increasing to 14.0× with three GPUs (Table 3), demonstrating its potential for real-time decoding.

Generalization across NeRV Architectures. Table 2 shows consistent improvements across datasets, with average PSNR-based BD-rate reductions of 26.83%, 27.83%, and 23.45% for HiNeRV (Kwan et al., 2023), RNeRV (Zhang et al., 2026), and HANeRV (Tang et al., 2026), respectively. Beyond architectural generalization, we further evaluate our approach within the NeuroQuant compression framework (Shi et al., 2025), using HiNeRV as the backbone. Our approach remains effective under this substantially different compression pipeline.

4.3 Ablation Studies

Learning rate and weight decay act jointly. Table 5 varies the optimizer settings while keeping the compression pipeline fixed. Increasing the learning rate alone slightly improves the Full model, but increases the Q6 compression loss from 0.48 to 0.72 dB. Weight decay alone improves recon-



(a) Sensitivity redistribution (b) Compression, no fine-tuning (c) Mask selectivity (d) Criterion robustness

Figure 4: Analysis of compression robustness. (a) Layer-wise sensitivity redistribution under the global pruning mask, with arrows indicating the shift from conventional optimization to ours. (b) Reconstruction quality after pruning and quantization without fine-tuning. (c) Cut fraction normalized by layer sparsity, f/s, for `fc1` layers (dots: individual sequences, bars: mean). (d) Impact of pruning criteria under different Fine-tuning epochs.

Table 3: **Decoding efficiency** on BasketballDrive (1080p), RTX A6000, FP16, averaged over 3 runs.

Method	GPU	Model dec. (s)	Frame recon. (s)	FPS
NVRC	1	127.59	34.44	3.08
	3	127.48	11.49	3.60
NVRC-net +SpecFit	1	1.94	23.38	19.75
	3	2.13	7.79	50.39

Table 4: **Stage-specific weight decay** on UVG sequences. Results are average PSNR (dB).

Decay stage	$\lambda = 0.05$		$\lambda = 0.1$	
	Pruned	Q6	Pruned	Q6
All	32.65	32.42	32.54	32.26
No QAT	32.65	32.44	32.54	32.31
Dense only	**32.70**	**32.50**	**32.66**	**32.48**

struction quality, yet does not reduce the relative loss after compression. Their combination changes this pattern: it achieves the highest Q6 PSNR and reduces the compression loss to 0.08 dB.

The effect is not explained by decay budget alone. Under the same decay budget, $\eta\lambda = 5\times10^{-4}$, different allocations between the two hyperparameters still produce different reconstruction quality. The balanced configuration achieves the best Full and Q6 PSNR, indicating that the gain depends on how optimization pressure and regularization are allocated, not only on their product.

Weight decay is useful primarily during dense fitting. As shown in Table 4, applying weight decay only during dense fitting gives the highest Pruned and Q6 PSNR for both tested decay strengths. Keeping weight decay beyond dense fitting consistently degrades the final compressed reconstruction, confirming that its role is to shape the dense parameters before compression rather than to regularize compression-stage fine-tuning.

4.4 Tracing the Origin of Pruning Robustness

Robustness concentrates in heavily pruned layers. Fig. 4(a) relates each layer's allocated sparsity to its sensitivity under a fixed 20% layer-wise pruning ratio. Under SpecFit, several lightly pruned `dconv` layers exhibit increased sensitivity, whereas heavily pruned layers, particularly `fc1`, show substantially reduced PSNR loss. Thus, SpecFit does not uniformly improve layer-wise robustness. Its gains are concentrated in the layers that receive high sparsity during global pruning.

Compression robustness is shaped during dense fitting. We directly prune and quantize the dense models without compression-stage fine-tuning. The robustness gap already appears after pruning and persists after quantization: the final PSNR loss is approximately 7.5 dB for Standard, versus only 1.82 dB for SpecFit (Fig. 4(b)). Thus, the gain originates from the parameter state formed during dense fitting, rather than from subsequent compression-stage fine-tuning.

Robustness persists under stronger compression. Table 6 varies pruning strength and quantization precision independently. Our optimization improves PSNR in every setting. Across pruning ratios from 15% to 55%, the gain ranges from 0.96 to 1.20 dB. Across quantization levels, it stays between 0.96 and 1.05 dB from Q8 to Q4 and narrows to 0.55 dB at Q3, where both methods degrade sharply.

Table 5: Optimizer ablations on HiNeRV for BasketballDrive. Full denotes the dense model before compression, Pruned denotes the model after pruning and sparse fine-tuning, and Q6 denotes the final model after 6-bit quantization-aware fine-tuning. Left: component ablation. Right: matched decay-budget configurations, where $\eta\lambda = 5 \times 10^{-4}$ and the normalized learning-rate schedule is shared across configurations. Parentheses report PSNR changes relative to the full model.

Component ablation						**Matched decay budget**				
Setting	η	λ	Full	Pruned	Q6	η	λ	Full	Pruned	Q6
Baseline	2×10^{-3}	0	27.27	27.13 (−0.14)	26.79 (−0.48)	2×10^{-3}	0.25	27.15	27.43 (+0.28)	27.26 (+0.11)
LR only	1×10^{-2}	0	27.37	27.16 (−0.21)	26.65 (−0.72)	5×10^{-3}	0.10	27.62	**27.95** (+0.33)	27.61 (−0.01)
WD only	2×10^{-3}	0.05	27.68	27.38 (−0.30)	27.13 (−0.55)	1×10^{-2}	0.05	**27.83**	**27.95** (+0.12)	**27.75** (−0.08)
Ours	1×10^{-2}	0.05	**27.83**	**27.95** (+0.12)	**27.75** (−0.08)	2×10^{-2}	0.025	27.77	27.88 (+0.11)	27.65 (−0.12)

Table 6: Robustness to pruning and quantization. PSNR (dB) is reported.

	Pruning Ratio @ Q6					**Quantization @ 15% Pruning**					
Method	15%	25%	35%	45%	55%	Q8	Q7	Q6	Q5	Q4	Q3
Standard	30.85	30.51	30.00	29.28	28.37	31.11	31.06	30.85	29.98	27.62	23.82
SpecFit	31.86	31.64	31.20	30.47	29.33	32.10	32.05	31.86	31.03	28.58	24.37

The improvement therefore extends well beyond the default compression setting and largely persists under substantially stronger pruning and quantization.

Magnitude becomes a sharper proxy for contribution. Following XINC (Padmanabhan et al., 2024), we measure the cut fraction f, the share of each token's contribution mass $\sum_{i,j} |w_{ij} h_{tj}|$ removed by the pruning mask, averaged over tokens (Appendix D.3), and normalize it by the sparsity s. As shown in Fig. 4(c), SpecFit yields a lower f/s than Standard in every layer, with values nearly equal across layers and tightly clustered, whereas Standard's vary across layers and are more dispersed. The gain thus reflects which weights are removed rather than how many.

Robustness persists across pruning criteria. We replace magnitude pruning with Wanda (Sun et al., 2024), which accounts for input-channel strength by ranking each weight w_{ij}, connecting input channel j to output channel i, by $|w_{ij}| \, \|[\boldsymbol{X}_l]_{:,j}\|_2$. Here the rows of $\boldsymbol{X}_l \in \mathbb{R}^{N_{\text{tok}} \times C_{\text{in}}}$ are the layer inputs $\boldsymbol{x}_l^\top$ collected over the tokens of the fitted video, so that $[\boldsymbol{X}_l]_{:,j}$ gathers the activations of input channel j. Wanda reduces the initial pruning loss of the standard optimization, but Standard+Wanda still incurs a larger PSNR drop than SpecFit+Magnitude throughout fine-tuning (Fig. 4(d)). Meanwhile, the difference between Wanda and magnitude pruning under our optimization quickly narrows. Activation-aware ranking therefore improves the initial pruning mask, while the larger robustness gain is already established during dense fitting.

5 Conclusion

We show that standard INR compression methods rely on an assumption that may not always hold: a better-fitted dense model is not necessarily a better model to compress. We trace this vulnerability to spectral drift during dense fitting, which shifts weight energy toward weakly excited input directions and lowers a sensitivity factor shared by pruning and quantization. Our optimizer analysis shows that weight decay sets where the weights are driven and the learning rate how far, so with weight decay, a larger learning rate reverses rather than deepens the drift. SpecFit therefore pairs moderate weight decay with a large learning rate, leaving the architecture unchanged. It reduces BD-rate by 23–28% on average across three NeRV backbones and rivals the rate–distortion performance of NVRC without a learned entropy model, while decoding the model $65.8\times$ faster. More broadly, our findings suggest that compression robustness in INRs is largely determined before compression begins, and that optimization should be treated as part of codec design: it not only minimizes reconstruction error, but also shapes the representation that the codec must ultimately compress.

AI USE STATEMENT

In this work, we have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims or hypotheses, designing the research methodology or experiments, generating experimental results, or interpreting the results. Other tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools for English language polishing, sentence restructuring, preparing tables and captions, literature search, and assisting with the creation of plots from experimental data.

We have reviewed all AI-assisted work. All references identified with the assistance of generative AI were independently verified by the authors against the original sources. AI-assisted tables, captions, and plots were checked against the corresponding experimental data, and all AI-assisted text was reviewed and revised by the authors to ensure technical correctness and consistency with the intended scientific meaning.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Gisle Bjontegaard. Calculation of average PSNR differences between RD-curves. Technical Report VCEG-M33, ITU-T Video Coding Experts Group (VCEG), 2001.

Frank Bossen. Common test conditions and software reference configurations. Technical Report JCTVC-L1100, Joint Collaborative Team on Video Coding (JCT-VC), 2013.

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021. doi: 10.1109/TCSVT.2021.3101953.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. NeRV: Neural representations for videos. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. HNeRV: A hybrid neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10270–10279, 2023.

Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in neural information processing systems*, 30, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. COIN: COmpression with implicit neural representations. In *ICLR Workshop on Neural Compression: From Information Theory to Applications*, 2021.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Matthew Gwilliam, Roy Zhang, Namitha Padmanabhan, Hongyang Du, and Abhinav Shrivastava. How to design and train your implicit neural representation for video compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 729–739, 2026.

Bo He, Xitong Yang, Hanyu Wang, Zuxuan Wu, Hao Chen, Shuaiyi Huang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Towards scalable neural representation for diverse videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6132–6142, 2023.

Zhihao Hu, Guo Lu, and Dong Xu. FVC: A new framework towards deep video compression in feature space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1502–1511, 2021.

Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards practical real-time neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12543–12552, 2025.

Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. C3: High-performance and low-complexity neural compression from a single image or video. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9347–9358. IEEE, 2024.

Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. HiNeRV: Video compression with hierarchical encoding-based neural representation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. NVRC: Neural video representation compression. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4210.

Ho Man Kwan, Tianhao Peng, Ge Gao, Fan Zhang, Mike Nilsson, Andrew Gower, and David Bull. Ultra-lightweight neural video representation compression. In *2025 Picture Coding Symposium (PCS)*, pp. 1–5. IEEE, 2025.

Ho Man Kwan, Tianhao Peng, Fan Zhang, Mike Nilsson, Andrew Gower, and David Bull. Enhanced neural video representation compression across extreme complexity and quality scales. In *European Conference on Computer Vision*, pp. 416–434. Springer, 2026.

Joo Chan Lee, Daniel Rho, Jong Hwan Ko, and Eunbyung Park. FFNeRV: Flow-guided frame-wise neural representations for videos. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 7859–7870, 2023. doi: 10.1145/3581783.3612444.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22616–22626. IEEE, 2023.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with feature modulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26099–26108, 2024.

Zizhang Li, Mengmeng Wang, Huaijin Pi, Kechun Xu, Jianbiao Mei, and Yong Liu. E-NeRV: Expedite neural video representation with disentangled spatial-temporal context. In *European Conference on Computer Vision*, pp. 267–284. Springer, 2022. doi: 10.1007/978-3-031-19833-5_16.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11006–11015, 2019.

Alexandre Mercat, Marko Viitanen, and Jarno Vanne. UVG dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, pp. 297–302, 2020. doi: 10.1145/3339825.3394937.

Namitha Padmanabhan, Matthew Gwilliam, Pulkit Kumar, Shishira R Maiya, Max Ehrlich, and Abhinav Shrivastava. Explaining the implicit neural canvas: Connecting pixels to neurons by tracing their contributions. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10957–10967. IEEE, 2024.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Junqi Shi, Zhujia Chen, Hanfei Li, Qi Zhao, Ming Lu, Tong Chen, and Zhan Ma. On quantizing neural representation for variable-rate video coding. In *International Conference on Learning Representations*, 2025.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 7462–7473, 2020.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. pmlr, 2013.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 7537–7547, 2020.

Lv Tang, Jun Zhu, Xinfeng Zhang, Li Zhang, Siwei Ma, and Qingming Huang. HANeRV: Hierarchically adaptive neural representation for video compression. *IEEE Transactions on Image Processing*, 35:5773–5788, 2026. doi: 10.1109/TIP.2026.3696103.

Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C.-C. Jay Kuo. MCL-JCV: A JND-based h.264/AVC video quality assessment dataset. In *IEEE International Conference on Image Processing (ICIP)*, pp. 1509–1513, 2016. doi: 10.1109/ICIP.2016.7532610.

Jia Wang, Xinfeng Zhang, Gai Zhang, Jun Zhu, Lv Tang, and Li Zhang. UAR-NVC: A unified autoregressive framework for memory-efficient neural video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 36(2):1681–1695, 2026a. doi: 10.1109/TCSVT.2025.3603282.

Jia Wang, Jun Zhu, and Xinfeng Zhang. SRNeRV: A scale-wise recursive framework for neural video representation. In *2026 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1546–1550, 2026b. doi: 10.1109/ISCAS66217.2026.11562533.

Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers*, volume 2, pp. 1398–1402, 2003.

Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. 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.

Gong-Yao Wu, Cih-Wei Wong, and Hsu-Feng Hsiao. Spatiotemporally modulated dual-mlp architecture for neural video representation and compression. In *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2025.

Dongpo Xu, Shengdong Zhang, Huisheng Zhang, and Danilo P Mandic. Convergence of the rmsprop deep learning method with penalty for nonconvex optimization. *Neural Networks*, 139: 17–23, 2021.

Xiaozhong Xu and Shan Liu. Overview of screen content coding in recently developed video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(2):839–852, 2021.

Gai Zhang, Lv Tang, and Xinfeng Zhang. Vqnerv: Vector quantization neural representation for video compression. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2024a.

Gai Zhang, Xinfeng Zhang, Lv Tang, Jun Zhu, Yue Li, Kai Zhang, and Li Zhang. RNeRV: Recurrent neural representation for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2026.

Xinjie Zhang, Ren Yang, Dailan He, Xingtong Ge, Tongda Xu, Yan Wang, Hongwei Qin, and Jun Zhang. Boosting neural representations for videos with a conditional decoder. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2556–2566. IEEE, 2024b.

Jiancheng Zhao, Yifan Zhan, Qingtian Zhu, Mingze Ma, Muyao Niu, Zunian Wan, Xiang Ji, and Yinqiang Zheng. Tree-NeRV: Efficient non-uniform sampling for neural video representation via tree-structured feature grids. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15076–15085. IEEE, 2025.

Lei Zhao, Shiqi Wang, Shanshe Wang, Yan Ye, Siwei Ma, and Wen Gao. Enhanced surveillance video compression with dual reference frames generation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1592–1606, 2021.

Jun Zhu, Xinfeng Zhang, Lv Tang, and JunHao Jiang. MSNeRV: Neural video representation with multi-scale feature fusion. *arXiv preprint arXiv:2506.15276*, 2025.

Jun Zhu, Xinfeng Zhang, Lv Tang, Junhao Jiang, Gai Zhang, and Jia Wang. Frequency-aware Neural Representation for Videos. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2026a.

Jun Zhu, Xinfeng Zhang, Lv Tang, Junhao Jiang, Gai Zhang, and Jia Wang. Video Compression with Hierarchical Temporal Neural Representation. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2026b.

A Quantization Perturbation

Let $\boldsymbol{N} = \hat{\boldsymbol{W}} - \boldsymbol{W}$ denote the quantization error. Writing $\boldsymbol{n}_r^\top$ for its r-th row, we assume the standard additive-noise model

$$\mathbb{E}_{\boldsymbol{N}}[\boldsymbol{n}_r \boldsymbol{n}_r^\top] = \sigma_r^2 \boldsymbol{I}.$$

Then

$$\mathbb{E}_{\boldsymbol{N}} \operatorname{Tr}(\boldsymbol{N}\boldsymbol{\Sigma}_x\boldsymbol{N}^\top) = \sum_r \operatorname{Tr}\big(\boldsymbol{\Sigma}_x \mathbb{E}[\boldsymbol{n}_r\boldsymbol{n}_r^\top]\big) = \operatorname{Tr}(\boldsymbol{\Sigma}_x)\sum_r \sigma_r^2 = \bar{e}\,\mathbb{E}_{\boldsymbol{N}}\|\boldsymbol{N}\|_F^2, \tag{12}$$

where $\bar{e} = \operatorname{Tr}(\boldsymbol{\Sigma}_x)/C_{\text{in}}$. Using

$$\operatorname{Tr}(\boldsymbol{W}\boldsymbol{\Sigma}_x\boldsymbol{W}^\top) = \bar{e}\,\|\boldsymbol{W}\|_F^2 \kappa(\boldsymbol{W}),$$

we obtain

$$\phi^{\mathrm{Q}} = \frac{\mathbb{E}_{\boldsymbol{N}} \operatorname{Tr}(\boldsymbol{N}\boldsymbol{\Sigma}_x\boldsymbol{N}^\top)}{\operatorname{Tr}(\boldsymbol{W}\boldsymbol{\Sigma}_x\boldsymbol{W}^\top)} = \frac{\nu}{\kappa(\boldsymbol{W})}, \qquad \nu = \frac{\mathbb{E}_{\boldsymbol{N}}\|\boldsymbol{N}\|_F^2}{\|\boldsymbol{W}\|_F^2}.$$

Thus, under the additive-noise approximation, quantization shares the same spectral sensitivity factor $1/\kappa(\boldsymbol{W})$ as pruning.

B Direct View of Spectral Weight Redistribution

The perturbation drift is consistent across layers. The opposing perturbation trends extend to all ten analyzed `fc1` tensors. Fig. 5(a) compares the relative pruning perturbation at the beginning and end of dense fitting, extending the representative trajectories in Fig. 1(c). All tensors lie above the identity line under standard optimization and below it under SpecFit. Thus, the increase under standard optimization and the decrease under SpecFit are consistent across the analyzed tensors.

SpecFit allocates more weight energy to strongly excited directions. The scalar $\kappa(\boldsymbol{W})$ in Eq. 6 summarizes weight-energy allocation relative to the input spectrum. We follow the notation in the main text and omit the layer index for brevity. To examine this allocation directly, consider the eigendecomposition

$$\boldsymbol{\Sigma}_x = \boldsymbol{V}\,\mathrm{diag}(e_1, \ldots, e_{C_{\mathrm{in}}})\boldsymbol{V}^\top = \sum_{j=1}^{C_{\mathrm{in}}} e_j \boldsymbol{v}_j \boldsymbol{v}_j^\top .$$

As in Eq. 9, define

$$\pi_j = p_j(\boldsymbol{W}) = \frac{\|\boldsymbol{W}\boldsymbol{v}_j\|_2^2}{\|\boldsymbol{W}\|_F^2}, \qquad \sum_{j=1}^{C_{\mathrm{in}}} \pi_j = 1, \tag{13}$$

where π_j is the fraction of weight energy allocated to input eigendirection $\boldsymbol{v}_j$. We normalize each eigenvalue by the mean input eigenvalue, $\bar{e} = \mathrm{Tr}(\boldsymbol{\Sigma}_x)/C_{\mathrm{in}}$, and express weight-energy allocation as $C_{\mathrm{in}}\pi_j$. Under uniform allocation, each direction receives the same weight energy and $C_{\mathrm{in}}\pi_j = 1$. These quantities recover $\kappa(\boldsymbol{W})$ as

$$\kappa(\boldsymbol{W}) = \sum_{j=1}^{C_{\mathrm{in}}} \pi_j \frac{e_j}{\bar{e}} = \frac{1}{C_{\mathrm{in}}} \sum_{j=1}^{C_{\mathrm{in}}} \left(\frac{e_j}{\bar{e}}\right) (C_{\mathrm{in}}\pi_j) . \tag{14}$$

Fig. 5(b) therefore reveals the allocation across input eigendirections underlying $\kappa(\boldsymbol{W})$.

At the end of dense fitting, standard optimization yields approximately uniform allocation over most of the input spectrum: directions with substantially different input excitation receive similar weight energy. SpecFit instead allocates less weight energy to weakly excited directions and progressively more to strongly excited ones. This allocation pattern is consistent with the larger $\kappa(\boldsymbol{W})$ observed under SpecFit. The extreme spectral bins contain few modes and are included for completeness; the interpretation rests on the broad trend across the spectrum rather than on individual endpoint bins.

The allocation patterns diverge during dense fitting. To track how the allocation changes during training, let $\mathcal{S}_{0.1}$ denote the set of mode indices corresponding to the smallest eigenvalues e_j whose sum accounts for 10% of $\mathrm{Tr}(\boldsymbol{\Sigma}_x)$. Their normalized mean weight-energy allocation is

$$R_{\mathrm{low}} = \frac{\sum_{j\in\mathcal{S}_{0.1}} \pi_j}{|\mathcal{S}_{0.1}|/C_{\mathrm{in}}} = \frac{1}{|\mathcal{S}_{0.1}|} \sum_{j\in\mathcal{S}_{0.1}} C_{\mathrm{in}}\pi_j . \tag{15}$$

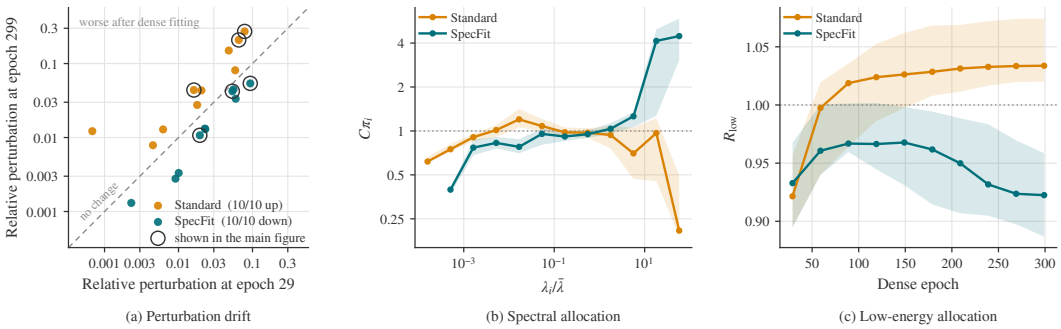
Here, $R_{\mathrm{low}} = 1$ denotes uniform per-mode allocation, while values above or below one indicate over-allocation or under-allocation to weakly excited modes, respectively.

Fig. 5(c) shows that both optimizers begin near the uniform reference at epoch 29. Under standard optimization, R_{low} rises from 0.922 to 1.034, eventually exceeding the uniform reference. Under SpecFit, it remains below one and changes only slightly overall, from 0.933 to 0.922, with a downward trend later in fitting. The two allocation patterns thus diverge during dense fitting from similar initial values. Together with Fig. 5(b), these trajectories support the mechanism described in Sec. 3: standard optimization shifts weight-energy allocation toward weakly excited directions, whereas SpecFit limits this drift and favors directions more strongly excited by the fitted inputs.

C Detailed Experimental Setup

C.1 Datasets and Preprocessing

We evaluate our method on six widely used video benchmarks: UVG (Mercat et al., 2020), HEVC Class B, Class E (Bossen, 2013), Surveillance (Zhao et al., 2021), SCC (Xu & Liu, 2021), and



(a) Perturbation drift (b) Spectral allocation (c) Low-energy allocation

Figure 5: **Additional evidence of optimization-induced spectral drift.** (a) Relative pruning perturbation before and after dense fitting for all analyzed `fc1` tensors. Circled points denote the layers shown in Fig. 1(c). (b) Final weight-energy allocation $C_{\text{in}}\pi_j$ versus normalized input eigenvalue $e_j/\bar{e}$. (c) Normalized allocation R_{low} to the low-energy subspace during dense fitting. In panel (b), the axis labels use $\lambda_i/\bar{\lambda}$ and $C\pi_i$, which correspond to $e_j/\bar{e}$ and $C_{\text{in}}\pi_j$ in the text, respectively. i and j both index input eigendirections.

Table 7: Summary of the datasets used in our experiments.

Dataset	Sequences	Resolution
UVG	7	1920×1080
HEVC Class B	5	1920×1080
HEVC Class E	3	1280×720
Surveillance	7	3×1080p, 4×720p
SCC	7	1920×1080
MCL-JCV	24	1920×1080
Total	53	–

MCL-JCV (Wang et al., 2016). Table 7 summarizes the number of sequences and spatial resolutions of each dataset. Overall, the evaluation covers 53 sequences with diverse spatial and temporal characteristics, including natural videos, videoconferencing, surveillance footage, and screen content. Following the standard per-video fitting protocol of neural video representations, each model is optimized independently on a single target video. All compared methods use exactly the same input sequences and preprocessing protocol for each video.

C.2 Benchmark Codec Configurations

VTM configuration. We use H.266/VVC (Bross et al., 2021) with the VTM-24.0 reference software under the Random Access (RA) configuration, with an intra period of 32 and a GOP size of 16. QP values are selected separately for each dataset to ensure sufficient overlap with the INR rate–distortion curves, and BD-rate is computed over the common overlapping rate range.

VTM bitstreams are decoded to YUV frames and then converted to 8-bit RGB images before evaluation. Consistent with the INR- and learning-based codecs, we compute both PSNR and MS-SSIM on the reconstructed RGB images using the same evaluation implementation. Thus, all codecs are evaluated in the same RGB domain, irrespective of their internal coding color space.

C.3 Implementation and Evaluation Details

Optimization settings. SpecFit modifies only the optimization procedure during model fitting. Unless otherwise specified, we use AdamW(Loshchilov & Hutter, 2019) for optimization, with learning rates of 1×10^{-2}, 1×10^{-3}, and 1×10^{-4} for dense fitting, pruning fine-tuning, and quantization-aware training (QAT), respectively. Weight decay is applied only during dense fitting

with $\lambda = 0.05$ and is disabled during pruning and QAT. The remaining training schedule follows the corresponding backbone configuration.

Backbone-specific configurations. For each backbone, we retain its original network architecture, model configuration, and compression pipeline. In particular, the parameter budget, pruning procedure, quantization scheme, and entropy-coding configuration follow the original setting of each backbone. The baseline and SpecFit therefore differ only in the optimization strategy, without introducing additional model parameters or modifying the underlying compression method.

Training details. We retain the original training schedule of each backbone without increasing the number of training epochs. HiNeRV-style backbones follow the default three-stage schedule of 300/60/30 epochs for dense fitting, pruning fine-tuning, and QAT, respectively, whereas NVRC follows its original 360/30-epoch schedule for soft-rounding training and Quant-Noise fine-tuning. Unless otherwise specified, all models are trained on NVIDIA A40 GPUs.

D Further Analysis of Compression Robustness

D.1 Robustness Persists Through Pruning and QAT

We further ask whether the robustness obtained during dense fitting is retained through the later compression stages. Fig. 6 tracks reconstruction PSNR during pruning fine-tuning and quantization-aware training (QAT), normalized by each model's PSNR at the end of dense fitting.

SpecFit is less sensitive to the pruning perturbation across all three HEVC Class B sequences. Immediately after pruning, its worst PSNR degradation is 1.25 dB on BasketballDrive, 1.00 dB on Kimono1, and 0.78 dB on ParkScene, compared with 2.37, 2.38, and 1.87 dB under standard optimization. Because this gap appears before compression-stage fine-tuning has time to compensate for the perturbation, the result suggests that the relevant robustness is already formed during dense fitting rather than created by sparse fine-tuning.

The same ordering remains after QAT. At the end of compression, SpecFit's PSNR deficits relative to its dense checkpoints are 0.03, 0.22, and 0.15 dB, whereas the corresponding deficits for standard optimization are 0.29, 0.49, and 0.37 dB. Thus, compression-stage adaptation recovers part of the pruning and quantization loss for both models, but it does not eliminate the robustness difference established during dense fitting. We next localize where this difference arises at the layer level.

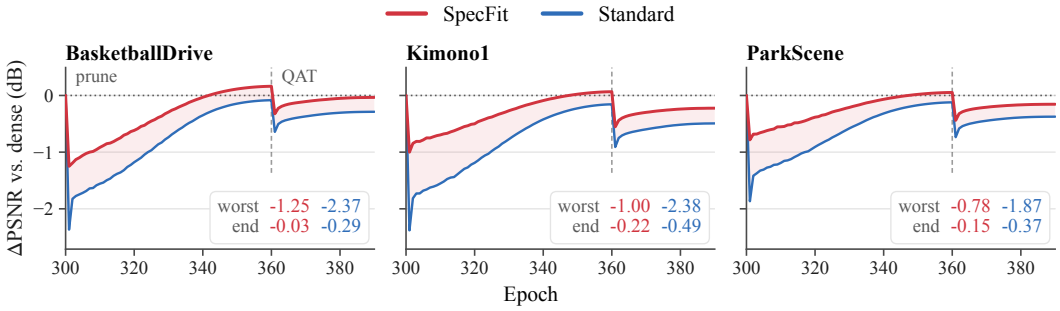


Figure 6: **Compression robustness persists through pruning and QAT.** PSNR relative to the end of dense fitting on three HEVC Class B sequences. SpecFit undergoes smaller degradation during both compression stages.

D.2 Pruning Sensitivity Concentrates in `fc1` Layers

Table 8 isolates the loss of each layer under the allocated mask. Under Standard training, degradation is dominated by a few early-stage `fc1` layers, which account for 87% of the summed loss, while most of the remaining layers are nearly insensitive. SpecFit's improvement comes almost entirely from these layers: at matched sparsity, S0-B2-fc1 drops from 4.33 to 0.53 dB, so the gain reflects a change in the weights rather than a more lenient allocation. We therefore focus the main-text analysis on `fc1`, where the Standard model fails under pruning and SpecFit takes effect.

Table 8: **Per-layer PSNR loss under the allocated mask.** Loss incurred by each decoder layer when it alone is pruned at the sparsity s that the global mask allocated to it.

Layer	s (%)		PSNR loss (dB)	
	Standard	SpecFit	Standard	SpecFit
S0-B2-fc1	25.9	26.0	4.33	0.53
S0-B3-fc1	24.6	22.9	2.64	0.19
S1-B2-fc1	19.8	17.8	0.96	0.08
S0-B2-fc2	26.6	30.3	0.59	0.43
S0-B4-fc1	23.0	21.5	0.47	0.12
S0-B3-fc2	23.3	25.1	0.25	0.11
S0-B4-fc2	21.4	18.9	0.19	0.04
Other layers	–	–	0.20	0.14
All layers	20.8	20.8	9.64	1.65

D.3 Token-Level Contribution Analysis

We next ask whether the improved pruning robustness can be explained at the level of the contributions removed by the mask. Following the contribution analysis of XINC (Padmanabhan et al., 2024), we measure how much of the pre-activation contribution to each token is cut by pruning. For an `fc1` layer, weight w_{ij} contributes

$$c_{ij}(t) = w_{ij} h_{tj}$$

to token t. Given a binary keep mask $M_{ij} \in \{0, 1\}$, we define the *token-level cut fraction*

$$f(t) = \frac{\sum_{i,j}(1 - M_{ij})\,|c_{ij}(t)|}{\sum_{i,j}|c_{ij}(t)|}, \tag{16}$$

and write $\bar{f}$ for its mean over all tokens in a frame. A lower $\bar{f}$ means that pruning removes a smaller share of the layer's contribution mass.

Exact computation. Directly materializing all $c_{ij}(t)$ values requires several gigabytes per frame. The pointwise structure of `fc1` makes the statistic computable exactly without this tensor. Specifically,

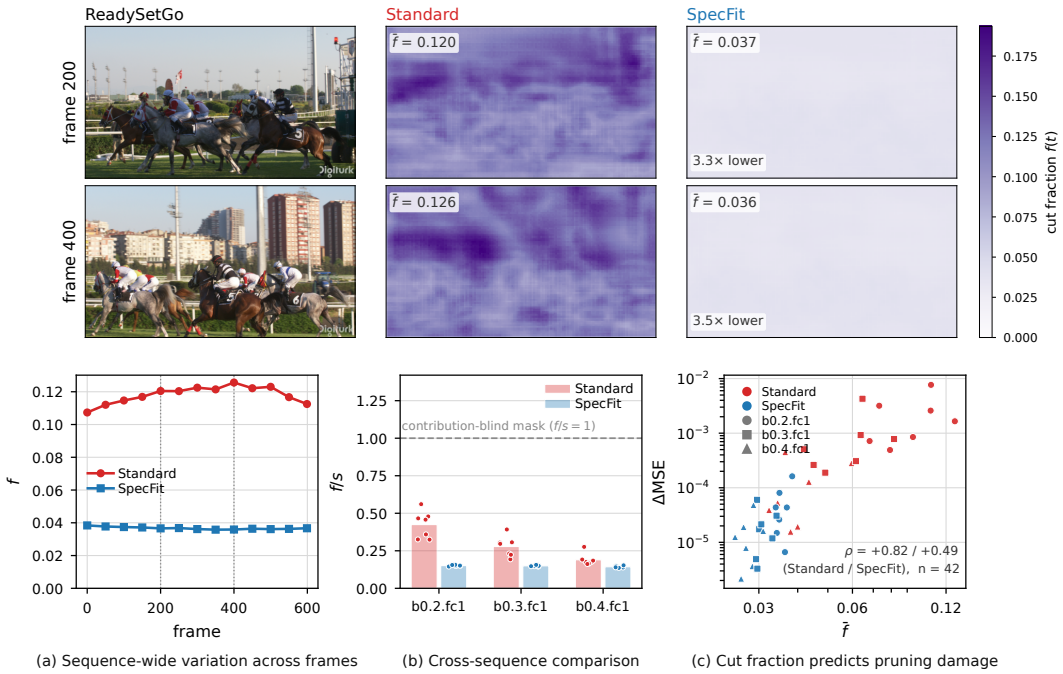
$$f(t) = \frac{\sum_j |h_{tj}| \sum_i (1 - M_{ij})|w_{ij}|}{\sum_j |h_{tj}| \sum_i |w_{ij}|}. \tag{17}$$

Thus, after precomputing the kept and removed channel weights for a layer, evaluating all tokens in a frame requires only two matrix–vector products.

Setup. We compare the dense baseline and our dense checkpoint at the end of fitting, then apply the HiNeRV global pruning mask at a 15% pruning ratio without fine-tuning.

Results. If a mask removed contribution mass independently of activations, we would expect $\bar{f}/s = 1$ at sparsity s. Both models fall below this reference, but our model consistently removes less contribution mass. The reduction is stable across layers and sequences, including layers assigned higher sparsity by the global mask (Fig. 7(b)). On ReadySetGo, our optimization reduces $\bar{f}$ by roughly $3\times$ on every frame for `b0.2.fc1` (Fig. 7, top and (a)).

The token-level pattern does not directly localize reconstruction error: within a frame, $f(t)$ correlates only weakly with per-pixel error, likely because later layers spatially mix the perturbation. At the layer level, however, the statistic is predictive. Across the 42 layer–sequence cells, $\bar{f}$ tracks the MSE increase caused by pruning each layer in isolation (Fig. 7(c)). Within the baseline, the Spearman correlation is 0.82. This supports the interpretation that our optimization improves compression robustness by moving pruning mass away from contributions that are active on the fitted video.



(a) Sequence-wide variation across frames (b) Cross-sequence comparison (c) Cut fraction predicts pruning damage

Figure 7: **Token-level cut fraction.** Top: spatial maps of $f(t)$ for `b0.2.fc1` on frames 200 and 400 of ReadySetGo, using a shared colour scale. (a) Frame-wise $\bar{f}$ on ReadySetGo; dotted lines mark the two frames shown above. (b) Normalized cut fraction $\bar{f}/s$ on seven UVG sequences (bars: mean, points: sequences). The dashed line is the contribution-blind reference. (c) Layer-level $\bar{f}$ versus the MSE increase from pruning that layer alone across 42 layer–sequence cells.

E Optimization Ablations Support Dense-Stage Weight Decay

E.1 Learning-Rate Retuning Is Not Sufficient

A possible explanation is that the baseline underperforms because its learning rates are not tuned for the pruned and quantized model. We therefore sweep the main-stage and pruning-stage learning-rate policies on UVG (Fig. 8). Even with a per-sequence oracle choice, Q6 PSNR improves by only 0.12 dB on average. A single shared policy already captures 0.10 dB of this gain. At a fixed main-stage learning rate, changing the pruning-stage multiplier alters PSNR by at most 0.09 dB.

These results rule out standard learning-rate retuning as the main source of our improvement. The available gain from retuning is small, and most of it can be obtained by one shared policy rather than sequence-specific choices.

E.2 Weight-Decay Strength Has a Non-Monotonic Effect

We next vary the weight-decay coefficient while keeping the rest of the training configuration fixed. Table 9 shows that increasing λ from 10^{-5} to 5×10^{-2} improves reconstruction quality throughout the pipeline. Q6 PSNR rises from 31.522 to 32.544 dB, while Q6 BPP remains nearly unchanged.

The trend is not monotonic. Increasing λ further to 10^{-1} produces the lowest BPP, but reduces PSNR at the dense, pruned, and Q6 stages. Thus, the benefit is not explained by uniformly shrinking the parameters more strongly. Instead, an intermediate decay strength gives a more favorable parameter trajectory for subsequent pruning and quantization. We use $\lambda = 5 \times 10^{-2}$ in the remaining experiments because it gives the highest reconstruction quality at a similar bitrate.

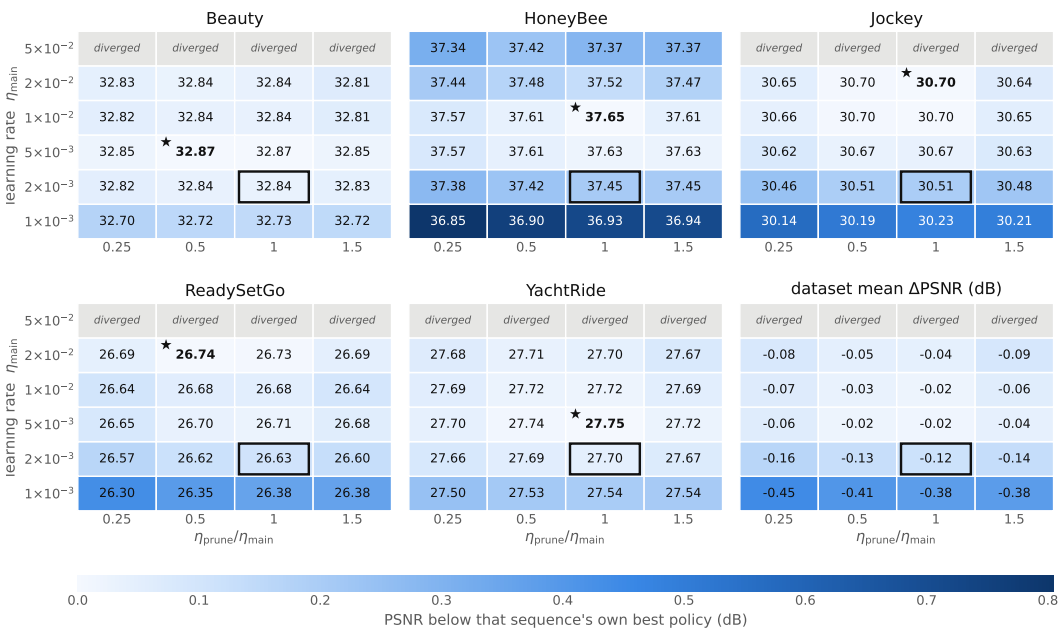


Figure 8: **Learning-rate retuning yields only marginal gains over the baseline.** Q6 PSNR across main-stage and pruning-stage learning-rate policies on UVG. Boxes mark the baseline policy, and stars mark the per-sequence optima.

Table 9: Sensitivity to the weight-decay coefficient. Lower BPP and higher PSNR are better.

λ	Dense PSNR	Pruned PSNR	Q6 BPP	Q6 PSNR
1×10^{-5}	32.236	32.084	0.022775	31.522
1×10^{-4}	32.237	32.090	0.022800	31.553
1×10^{-3}	32.279	32.146	0.022825	31.644
3×10^{-3}	32.330	32.233	0.022800	31.799
1×10^{-2}	32.475	32.439	0.022900	32.140
5×10^{-2}	**32.721**	**32.771**	0.022800	**32.544**
1×10^{-1}	32.555	32.689	**0.022625**	32.400

E.3 Dense-Stage Decay Accounts for Most of the Gain

SpecFit applies weight decay only during dense fitting. We now test whether applying it during later compression stages is also useful. Table 10 varies both the decay strength and the stages where decay is active. The dense stage accounts for most of the gain. Increasing λ from 10^{-3} to 0.05 improves Q6 PSNR by about 1.0 dB, while increasing it to 0.1 reduces quality. Applying decay after dense fitting does not improve the final model. For larger λ, decay during the compression stages lowers both pruned and Q6 PSNR. The degradation mainly comes from the pruning stage: removing decay from QAT alone recovers little, whereas removing it from pruning recovers nearly all of the lost PSNR. Once the mask is fixed, the remaining stages primarily recover reconstruction quality, so additional decay adds bias without improving pruning robustness.

E.4 Optimizer and Regularization Ablations

Alternative optimizers. Changing the base optimizer alone does not reproduce the compression gains from the proposed dense-stage weight decay. We compare AdamW(Loshchilov & Hutter, 2019), Adagrad(Duchi et al., 2011), RMSProp(Xu et al., 2021), and Momentum(Polyak, 1964; Sutskever et al., 2013) with weight decay disabled, tuning the learning rate separately for each

Table 10: **Weight decay applied at different training stages**. *All*: decay in every stage. *No QAT*: decay removed from QAT. *Dense only*: decay removed from both pruning and QAT.

		Pruned			Q6		
λ	**Main**	All	No QAT	Dense only	All	No QAT	Dense only
1×10^{-3}	32.13	32.00	32.00	32.00	31.50	31.50	31.50
3×10^{-3}	32.19	32.10	32.10	32.10	31.66	31.66	31.66
1×10^{-2}	32.34	32.31	32.31	32.31	32.01	32.01	32.01
5×10^{-2}	32.59	32.65	32.65	**32.70**	32.42	32.44	**32.50**
1×10^{-1}	32.40	32.54	32.54	32.66	32.26	32.31	32.48

Table 11: **Learning-rate sweep for alternative optimizers** on BasketballDrive ($\lambda = 0$).

AdamW			**Momentum**			**Adagrad**			**RMSProp**		
LR	Full	Q6	LR	Full	Q6	LR	Full	Q6	LR	Full	Q6
3×10^{-3}	27.31	26.58	3×10^{-2}	24.37	24.03	3×10^{-2}	26.18	25.80	1×10^{-4}	26.77	25.91
5×10^{-3}	27.35	26.60	1×10^{-1}	25.64	25.10	1×10^{-1}	26.78	26.34	3×10^{-4}	26.94	26.12
1×10^{-2}	**27.38**	**26.62**	3×10^{-1}	26.42	25.85	3×10^{-1}	26.94	26.38	1×10^{-3}	26.98	26.13
2×10^{-2}	*diverged*		5×10^{-1}	25.98	25.57	5×10^{-1}	26.81	26.22	1.5×10^{-3}	*diverged*	
3×10^{-2}	*diverged*		1	26.24	25.79	1	15.41	23.15	3×10^{-3}	*diverged*	

optimizer. AdamW achieves the highest Q6 PSNR at 26.62 dB, followed by Adagrad, RMSProp, and Momentum at 26.38, 26.13, and 25.85 dB, respectively (Table 11). The learning-rate sweeps span both underfitting and unstable regimes, suggesting that this ranking is not an artifact of a single learning-rate choice. Adding dense-stage weight decay to AdamW further increases Q6 PSNR from 26.62 to 27.75 dB, exceeding the best result obtained by any of the tested optimizers without weight decay.

Effect of decoupled weight decay. Table 12 shows that decoupled weight decay is substantially more effective than coupled ℓ_2 regularization. AdamW with $\lambda = 0.05$ improves Q6 PSNR from 26.62 to 27.75 dB, whereas Adam+ℓ_2 progressively degrades both dense and compressed reconstruction as λ increases. This supports the use of decoupled weight decay during dense fitting.

F Extended Quantitative Results

F.1 Rate–Distortion

Table 13 compares rate–distortion performance on MCL-JCV and HEVC-B, with BD-rates calculated against VTM. The results show that the proposed optimization improves the MSNeRV backbone on both datasets in terms of PSNR and MS-SSIM. On MCL-JCV, it reduces the PSNR BD-rate from 15.55% to 7.33% and the MS-SSIM BD-rate from -8.21% to -13.55%.

Larger improvements over MSNeRV are observed on HEVC-B, where our method achieves PSNR and MS-SSIM BD-rates of -15.09% and -40.32%, respectively, compared with -4.25% and -29.98% for MSNeRV. Our method also outperforms the other INR-based codecs on HEVC-B, including NVRC, which achieves PSNR and MS-SSIM BD-rates of -8.35% and -40.01%, respectively. These results demonstrate that the proposed optimization improves the coding efficiency of the MSNeRV backbone without modifying its architecture.

F.2 Generalization Across Backbones and Datasets

Table 14 examines whether SpecFit remains effective beyond a single INR backbone or dataset. The evaluation covers three INR backbones and an additional NeuroQuant pipeline. For each backbone–dataset pair, BD-rate is computed relative to the same backbone trained with its original optimization configuration, thereby isolating the effect of SpecFit while keeping the representation architecture and compression pipeline unchanged.

Table 12: **AdamW vs. Adam+ℓ_2** on BasketballDrive.

Optimizer	λ	Full	Q6
AdamW	0	27.38	26.62
	5×10^{-2}	**27.83**	**27.75**
Adam+ℓ_2	2×10^{-6}	26.63	26.93
	5×10^{-6}	25.16	26.03
	1×10^{-5}	23.70	25.18
	2×10^{-5}	22.66	24.48

Table 13: Full rate–distortion on MCL-JCV and HEVC-B. BD-rate (%) is computed with VTM as the anchor (lower is better).

		Traditional		Conditional NVC			INR-based			
Dataset	Metric	HM	VTM	DCVC-DC	DCVC-FM	DCVC-RT	HiNeRV	NVRC	MSNeRV	Ours
MCL-JCV	PSNR	46.06	0.00	**−2.30**	1.29	3.55	73.19	17.66	15.55	7.33
	MS-SSIM	52.33	0.00	−5.80	5.77	−0.08	44.00	−13.48	−8.21	**−13.55**
HEVC-B	PSNR	48.42	0.00	26.03	34.40	42.25	69.69	−8.35	−4.25	**−15.09**
	MS-SSIM	71.51	0.00	30.92	35.11	23.29	1.86	−40.01	−29.98	**−40.32**

SpecFit reduces BD-rate for every tested backbone–dataset pair under both PSNR and MS-SSIM. The average gains are comparable across the four backbones, with all PSNR and MS-SSIM reductions lying in the low-to-mid 20% range or better. The trend also persists across datasets with different video characteristics. Although the magnitude varies across datasets and backbones, all entries in Table 14 remain negative, including the smaller gains on UVG and the larger gains on HEVC-B.

Fig. 9 further shows representative rate–distortion curves on ParkScene and Kimono1. Across all four backbones, SpecFit consistently shifts the curves toward higher reconstruction quality over the evaluated bitrate range under both quality metrics. The improvement is visible across multiple operating points rather than being driven by an isolated rate point, providing curve-level evidence for the backbone-level gains summarized by BD-rate.

Taken together, these results show that the improvement is not specific to a particular INR architecture or dataset. Since each comparison uses the same backbone, model configuration, and compression pipeline, the observed gains are attributable to the change in optimization strategy rather than to architectural modifications.

F.3 Performance Across Model Scales

Table 15 examines whether the proposed optimization strategy remains effective as model capacity changes. We evaluate HiNeRV and RNeRV under six closely matched parameter budgets, spanning compact and larger INR models. For each backbone and parameter budget, the baseline and Ours share the same architecture and parameter count, differing only in the optimization strategy. This controlled comparison isolates the effect of optimization from the effect of model size.

The proposed strategy improves the average PSNR at every tested scale for both backbones. The gains appear already in the smallest models, where the parameter budget is most constrained, and remain present as the models become larger. This trend indicates that the improvement does not depend on adding capacity or operating only in an over-parameterized regime. Instead, the optimization strategy helps a fixed architecture use its available parameters more effectively.

The per-sequence results show the same overall pattern. Most sequences improve consistently across scales and backbones, with only a negligible per-sequence deviation for the smallest RNeRV model on Bulong. This isolated case does not change the average trend. Overall, the results support that the proposed optimization improves parameter efficiency across model scales and generalizes across different INR backbones.

Table 14: Full per-dataset results of generalization across different INR backbones. Each entry reports BD-rate (%) as PSNR/MS-SSIM, computed relative to the original optimization configuration of each backbone (lower is better). Avg. denotes the average over all six datasets.

Backbone	UVG	HEVC-B	HEVC-E	Surv.	SCC	MCL-JCV	Avg.
HiNeRV	-20.20/-19.49	-34.06/-32.76	-25.74/-26.06	-28.65/-24.05	-25.62/-22.68	-26.73/-24.32	-26.83/-24.89
RNeRV	-25.43/-24.42	-34.41/-33.33	-26.44/-24.19	-29.40/-26.84	-26.22/-25.41	-25.10/-24.46	-27.83/-26.44
HANeRV	-18.81/-18.23	-26.23/-24.19	-22.56/-22.31	-24.23/-23.89	-25.76/-24.81	-23.12/-22.99	-23.45/-22.74
NeuroQuant	-20.21/-19.34	-33.11/-32.72	-26.32/-25.19	-29.39/-28.57	-26.99/-25.23	-23.91/-23.11	-26.66/-25.69

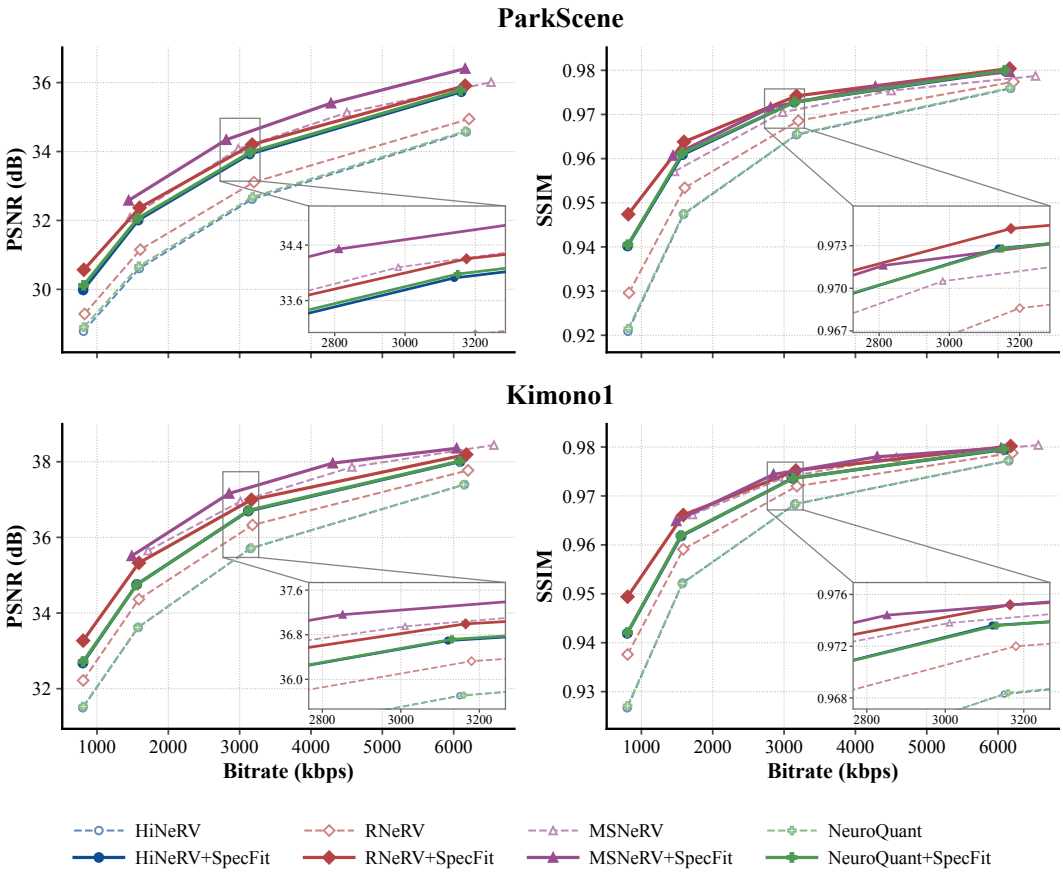


Figure 9: Rate–distortion performance on ParkScene (top) and Kimono1 (bottom), in PSNR (left) and SSIM (right). For each backbone—HiNeRV, RNeRV, MSNeRV and NeuroQuant—the dashed curve with hollow markers is the original model and the solid curve with filled markers is the same model equipped with our method. The insets magnify the mid-rate region, where the curves overlap.

G Further Comparison with DCVC-based Codecs

Performance on specialized video content. We further compare INR-based compression with DCVC-based codecs on video content beyond conventional natural-video benchmarks, including surveillance and screen-content sequences. As shown in Fig. 10, our method exhibits a clear advantage on these specialized contents.

The advantage arises from two complementary properties of sequence-specific neural representations. First, surveillance videos typically contain slowly varying backgrounds and infrequent scene transitions, resulting in strong redundancy over long temporal ranges. Such content is well suited to an INR that represents the entire sequence with a shared set of parameters.

Table 15: Per-sequence PSNR (dB) comparison on HiNeRV and RNeRV across six closely matched parameter budgets. Standard and SpecFit use the same architecture and parameter count within each backbone, differing only by the optimization strategy. HSW and WMHD denote Hongsilouwest and Weiminghudong, respectively.

Backbone	Optimization	Params	Bulong	Night	Outdoor	HSW-0222	HSW-0246	HSW-0319	WMHD	Avg.
HiNeRV	Standard	0.55M	19.150	23.854	21.866	28.079	27.081	27.696	30.556	25.469
	SpecFit		20.126	24.522	22.864	29.021	28.020	28.638	31.720	**26.416**
	Standard	0.86M	21.185	25.602	24.347	30.678	29.224	29.882	32.919	27.691
	SpecFit		21.546	26.167	25.449	31.642	30.351	30.875	34.099	**28.590**
	Standard	1.56M	22.977	27.201	26.913	33.406	31.464	32.410	35.144	29.931
	SpecFit		23.911	27.935	28.040	34.479	32.466	33.436	36.394	**30.952**
	Standard	3.15M	25.417	29.179	29.533	36.382	34.293	35.520	37.738	32.580
	SpecFit		26.741	30.174	30.705	37.492	35.475	36.944	38.793	**33.760**
	Standard	6.42M	27.510	31.064	32.269	39.000	36.853	38.325	39.729	34.964
	SpecFit		29.034	32.012	33.390	40.385	38.250	39.600	40.857	**36.218**
	Standard	12.67M	29.999	32.937	34.662	41.808	39.686	41.148	41.990	37.461
	SpecFit		31.308	33.671	35.519	42.924	40.828	41.916	42.579	**38.392**
RNeRV	Standard	0.56M	19.953	24.054	21.628	28.505	27.479	28.030	30.918	25.795
	SpecFit		19.906	24.584	22.716	29.344	28.232	28.935	31.751	**26.495**
	Standard	0.87M	21.493	25.778	24.326	30.980	29.570	30.248	33.165	27.937
	SpecFit		22.084	26.297	25.329	31.875	30.400	31.188	34.054	**28.747**
	Standard	1.58M	23.263	27.300	26.795	33.548	31.700	32.590	35.723	30.131
	SpecFit		24.256	28.031	28.080	34.667	32.837	34.212	36.668	**31.250**
	Standard	3.18M	25.653	29.165	29.501	36.640	34.658	36.003	37.943	32.795
	SpecFit		26.569	30.009	30.746	37.780	35.962	37.484	38.959	**33.930**
	Standard	6.46M	27.732	30.866	32.012	39.053	37.088	38.470	39.912	35.019
	SpecFit		28.860	31.884	33.438	40.836	38.436	40.058	40.934	**36.349**
	Standard	12.73M	30.039	32.751	34.421	41.814	39.797	41.178	42.056	37.436
	SpecFit		31.046	33.567	35.641	43.076	40.990	42.034	43.053	**38.487**

Second, the advantage becomes particularly pronounced on screen-content videos. Unlike pretrained learned codecs, whose performance depends on the video distribution seen during training, INR-based codecs optimize the representation directly for each target sequence. They are therefore less dependent on a fixed training distribution, which is beneficial when the test content differs substantially from natural videos. This distinction is evident on SCC, where text, icons, sharp boundaries, and large homogeneous regions lead to markedly different visual statistics from conventional camera-captured content. Our method consequently maintains strong rate–distortion performance, whereas the performance of the DCVC variants degrades substantially.

Frame-wise quality stability. We further compare the reconstruction quality over individual frames at similar bitrates. Fig. 11 reports the per-frame PSNR and MS-SSIM curves of our method and DCVC-RT on BasketballDrive and Bosphorus. Although the two methods operate at comparable bitrates, DCVC-RT exhibits pronounced frame-to-frame fluctuations, whereas our reconstruction quality varies much more smoothly over time.

To quantify this behavior, we report the mean absolute quality difference between adjacent frames,

$$\Delta_{\mathrm{adj}} = \frac{1}{T-1} \sum_{t=1}^{T-1} \left| q_{t+1} - q_t \right|, \tag{18}$$

where q_t denotes the PSNR or MS-SSIM of frame t. Lower Δ_{adj} indicates a more stable frame-wise quality trajectory.

On BasketballDrive, our method reduces the adjacent-frame PSNR variation from 0.32 dB to 0.07 dB ($4.7\times$ lower), and the MS-SSIM variation from 1.88×10^{-3} to 0.22×10^{-3} ($8.7\times$ lower). The difference is even more pronounced on Bosphorus, where the corresponding variations are reduced from 0.35 dB to 0.03 dB ($12.0\times$ lower) and from 1.49×10^{-3} to 0.06×10^{-3} ($23.2\times$ lower), respectively.

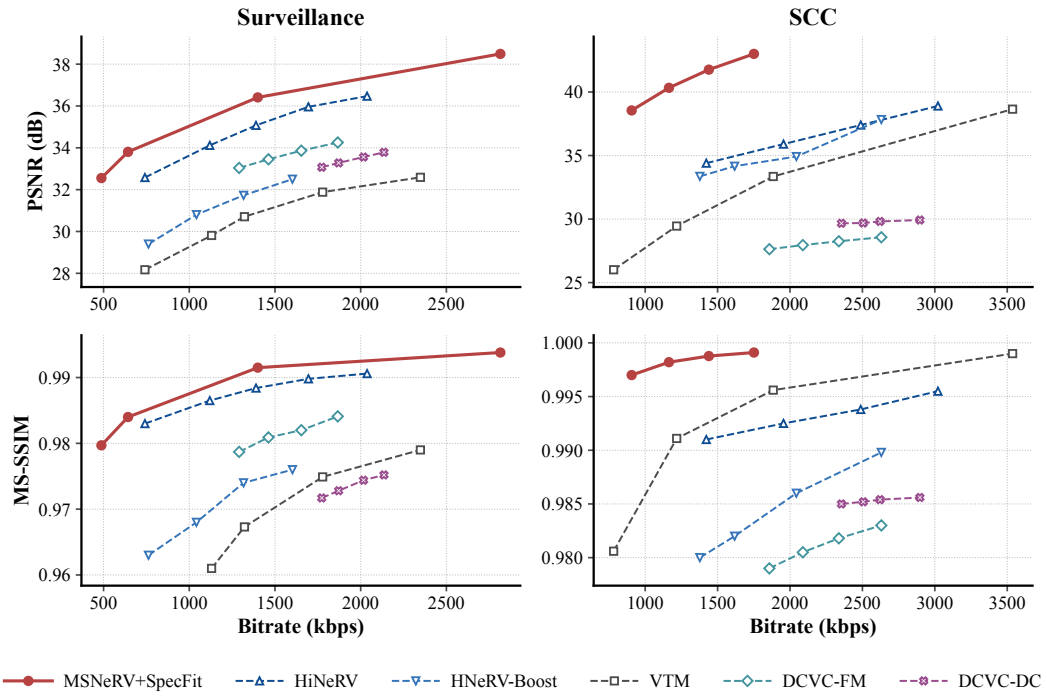


Figure 10: Rate–distortion comparison with DCVC-based codecs on the Surveillance and SCC datasets. PSNR and MS-SSIM results are reported for both datasets.

H EXTENDED VISUAL COMPARISON.

Figures 12 and 13 show additional qualitative results on HEVC Class B and UVG. Compared with HiNeRV and NVRC at similar bitrates, Ours retains sharper boundaries and finer local textures in the highlighted regions. The improvement remains visible across both motion-intensive and texture-rich sequences.

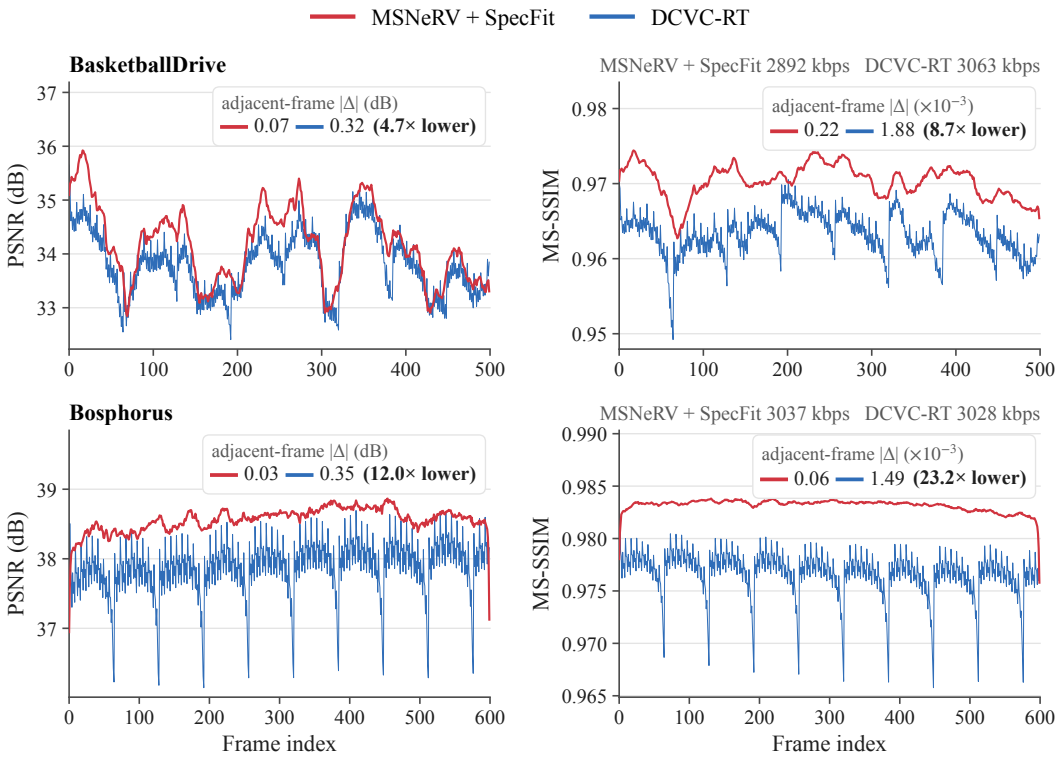


Figure 11: Per-frame distortion results on the UVG (*Bosphorus*) and HEVC ClassB (*BasketballDrive*).



Figure 12: Extended visual comparisons on the HEVC ClassB dataset (*BasketballDrive*).

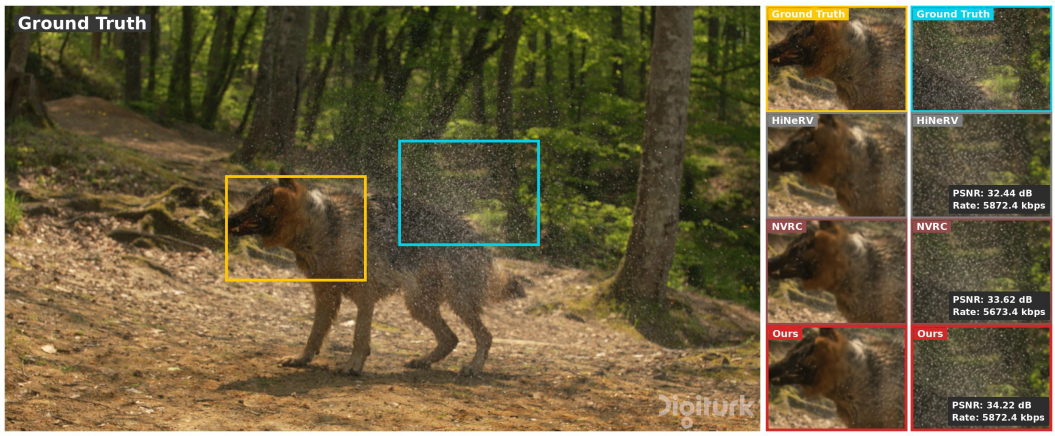


Figure 13: Extended visual comparisons on the UVG dataset (*ShakeNDry*).