Neural-Centric Video Processing Pipeline for Unified Multi-Task Inference

Seyeon Lee[1] Juncheol Ye[1] Jaehong Kim[2] Dongsu Han[1]
[1]Korea Advanced Institute of Science and Technology [2]Inha University
{thisisseyeon, juncheol, dhan.ee}@kaist.ac.kr jaehong.kim@inha.ac.kr

Abstract

Videos are increasingly used as inputs to machine learning systems, where repeated decoding and processing across diverse downstream tasks dominate computational cost. However, existing video pipelines remain inefficient. Traditional codecs such as H.264 and H.265 are optimized for human perception and require full pixel decoding for every query, compressed-domain methods are tied to specific codec structures with limited flexibility, and machine-oriented video coding approaches often rely on task-specific encoders and separate representations without supporting human visualization. We propose Neural Video Pipeline (NVP), a framework that leverages implicit neural representations to directly extract task-specific features from intermediate layers, eliminating pixel reconstruction overhead. NVP introduces lightweight micro adapters that map these features into the representation space of downstream models, bypassing both decoding and early-stage feature extraction. Across four representative tasks—image classification, object detection, action recognition, and segmentation—NVP reduces latency by up to 89.5% and inference FLOPs by up to 29.9%, while supporting multiple tasks using a single unified representation.

1. Introduction

Recent advances in video understanding have elevated downstream tasks such as action recognition [3, 14], object detection [2], and anomaly detection [35] into core components of end-to-end video systems. In applications including video-on-demand, surveillance footage analysis [23], and content moderation [13], the same content is accessed repeatedly by heterogeneous consumers over an extended lifetime. These deployments follow a Write-Once-Read-Many (WORM) pattern: a video is encoded once at a central server and repeatedly queried across diverse tasks, models, temporal ranges, and resolutions. Under these workloads, the dominant cost is repeated decoding and inference—not one-time encoding. Moreover, because query combinations vary widely, precomputing all results is impractical due to combinatorial storage overhead.

However, current video processing pipelines are not designed with this repeated, diverse-task scenario in mind. Current video processing follows a well-established paradigm: raw frames are encoded using traditional codecs (H.264 [45], HEVC [43]) for storage, then decoded on-demand for human viewing or AI inference. This approach has been adequate when humans were the primary consumers of video content. Traditional codecs are designed to maximize compression while optimizing perceptual quality metrics (PSNR, SSIM) for human viewers.

This paradigm becomes inefficient when neural networks consume video. For each inference query, the traditional pipeline requires three sequential stages: (1) *decoding* compressed video to pixels, (2) *preprocessing* pixels through resizing, normalization, and format conversion, and (3) *feature extraction* through early layers of the neural network backbone. These codec transformations introduce unnecessary overhead for machine vision. More critically, when multiple downstream tasks run on the same video, each task requires this decode→preprocess→extract pipeline, leading to severe computational redundancy [49].

Two research directions have emerged to reduce this overhead. *Compressed Domain Inference* (CDI) [1, 28, 38] performs inference directly on compressed bitstream features (motion vectors, residuals), bypassing decoding. However, CDI is tightly coupled to specific codec structures, requiring task-specific redesign. *Video Coding for Machines* (VCM) [7, 8, 10, 12, 16, 17, 22, 32, 34, 40] pursues a complementary goal: replacing pixel encoding entirely with learned compression of task-optimized neural features, improving bitrate-accuracy trade-offs. However, VCM methods require custom-designed encoders for each application domain [7, 17, 22, 33, 39] and force training from scratch, precluding the reuse of pretrained RGB-based models [9, 10, 15, 32, 33, 48]; most target static images [8, 9, 15, 48]; and rarely support human visualization [8, 15, 33]. Critically, neither CDI nor VCM provides a unified representation serving arbitrary downstream tasks without re-encoding.

We identify an opportunity in INR to transcend this dichotomy. INRs are inherently composed of CNN layers

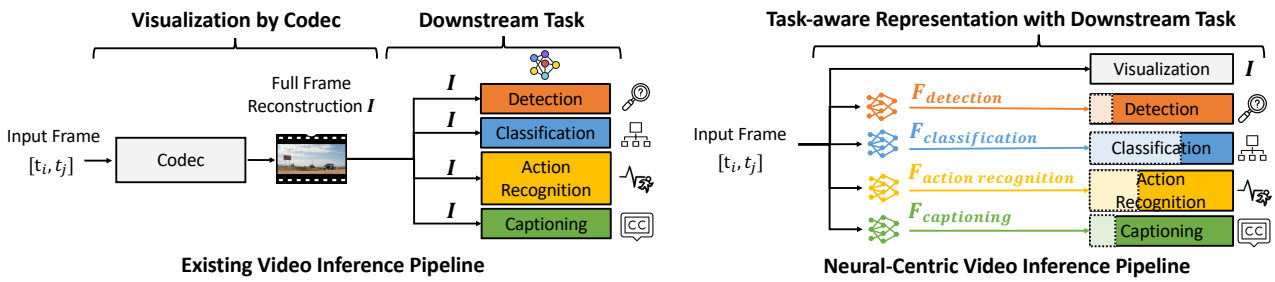

Figure 1. Comparison of traditional pixel-based and neural-centric video inference pipelines. Traditional approaches (left) require full decode-preprocess-extract stages for each task, while NVP (right) directly extracts task-specific features from neural representations.

that hierarchically generate features. The key observation is that these intermediate features already reside in neural feature space—as CNN activations rather than pixels. This shared representation space enables INR features to interface directly with downstream neural networks, eliminating the need for pixel decoding.

From this observation, we propose encoding videos directly as INRs. In this paradigm, each video becomes a continuous function that can emit hierarchical features at any desired abstraction level, directly providing task-specific representations to downstream models without intermediate pixel reconstruction (Figure 1).

Based on this principle, we introduce the Neural-Centric Video Pipeline (NVP), a unified framework that brings the entire video processing pipeline—from encoding to inference—into the neural domain. Within each INR, NVP identifies task-specific layers that contain minimal sufficient information and learns a Micro Adapter to bridge these layers to corresponding points in pretrained backbones. At inference time, decoding halts at the task-selected layer, and the adapter directly feeds the resulting feature map into the downstream model—bypassing full pixel reconstruction and redundant early-stage processing. Moreover, the INR itself is jointly optimized with downstream task objectives, enhancing task accuracy under a fixed bitrate budget. Through this integration, NVP realizes a fully neural video pipeline tailored for WORM workloads, unifying compression, storage, and inference while eliminating redundant computation across tasks.

We comprehensively evaluate NVP across four representative video analysis tasks—classification [11, 20, 36], object detection [2], action recognition [3, 14], and captioning [30]—against conventional H.264-based pipelines and other neural-based baselines. Compared to traditional CPU/GPU decoding pipelines, NVP achieves substantial latency reductions of up to 89.46% (CPU) and 62.76% (GPU) across tasks, while maintaining comparable or improved accuracy. Notably, at lower bitrate budgets, NVP achieves 7.52% higher accuracy (CLIP-RN50 at 0.025 bpp) and 10.49% higher accuracy (SlowFast at 0.02 bpp) compared to H.264, demonstrating superior rate-accuracy trade-offs. Against NeRV-based implicit neural representation baselines, NVP achieves up to 73.39% faster end-to-end inference and 89.40% lower compute cost through its partial decoding and task-aware adapter design. Additionally, these efficiency gains are achieved without requiring task-specific redesign or complex re-training: NVP can be directly integrated into existing pipelines, with adapters and INR representations trained once and reused seamlessly across tasks.

2. Related Work

Video Coding for Machines. Traditional video codecs (H.264 [45], HEVC [43]) optimize for human perceptual quality (PSNR, SSIM), introducing unnecessary overhead for machine vision. They fully reconstruct RGB frames, while neural networks discard more than 50% of pixel-level detail during feature extraction [41]. The MPEG Video Coding for Machines (VCM) [12, 16, 34] initiative addresses this by optimizing directly for task performance through feature compression [7, 8, 22] or task-specific codecs [10, 17, 32, 40].

Recent work also explores feedforward neural encoders (e.g., VQ-VAE-based compression, Compressed Vision [10], Omnipotent [15]) that learn compact features without explicit pixel reconstruction.

However, these methods still have key limitations: they often require task-specific encoders or pipelines [7, 17], retrain downstream models from scratch to consume compressed features [32, 40], limiting reuse of pretrained RGB backbones, are frequently restricted to image-centric settings [10, 29], and rarely support human visualization [32]. As a result, they do not provide a unified representation for flexible multi-task inference.

Compressed Domain Inference (CDI). CDI methods operate on codec byproducts (e.g., motion vectors, residuals) without full decoding. CoViAR [1] and DMC-Net [40] achieve significant speedups for video understanding tasks.

While practical to deploy, CDI is constrained by codec design. Representations are optimized for reconstruction, not semantics, and task-specific models limit flexibility. Fixed codecs also prevent end-to-end optimization with

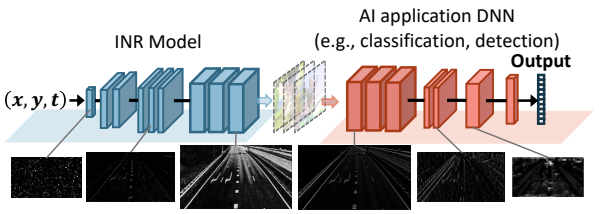

Figure 2. Abstraction levels between NeRV decoding and YOLO inference are reversed.

downstream models.

Implicit Neural Representations for Video. INRs have recently emerged as an alternative to traditional codecs, improving compression [27], quality [25], scalability [19], and speed [5, 31]. Recent work such as C3 [24] demonstrates effective INR-based video compression. Prior work primarily treats INRs as codec replacements for pixel reconstruction. In contrast, we leverage INRs as neural-native representations that directly interface with downstream models, enabling unified support for visualization and multi-task inference.

3. Design Overview

Goal and Benefits. Our goal is to optimize the entire video processing pipeline for multiple downstream tasks by directly extracting task-specific features from a single neural substrate. This design eliminates redundant computation by directly generating task-specific features from the neural representation, bypassing full pixel reconstruction and feature extraction.

This property yields three benefits: (1) reduced processing latency and computation cost for downstream tasks, (2) a more compact encoded representation for the same task accuracy, since videos are represented directly as task-optimized features, and (3) operating entirely in the neural domain enables further joint end-to-end optimization.

Key Insight. We realize this goal by encoding videos as INRs. As illustrated in Figure 2, NeRV [4] decodes frames in a coarse-to-fine manner: early layers capture coarse global semantics while deeper layers refine fine-grained spatial details, mirroring the reverse of how downstream vision models form increasingly abstract representations. This mirrored hierarchy reveals that NeRV's intermediate features already constitute a rich semantic representation at multiple abstraction levels, aligned with what downstream tasks require. Consequently, these features can be consumed directly without full pixel reconstruction or additional feature-extraction stages.

Design Principles. To make the neural-centric pipeline practical, we design it with the following principles in mind:

- **Multi-Task Efficiency.** The pipeline should extract task-specific representations from a single shared video entity, minimizing storage overhead while enabling efficient concurrent inference across multiple tasks.
- **Minimal generation cost.** During inference, the pipeline should produce only the features relevant to each downstream task, bypassing the wasteful path of full pixel-level reconstruction and thereby minimizing computational overhead and generation latency.
- **Pretrained Model Compatibility.** The pipeline must produce representations inherently compatible with these models, avoiding resource-intensive retraining or fine-tuning of large backbone networks.

4. Neural-Centric Video Pipeline

We propose the Neural-Centric Video Pipeline (NVP), a framework that efficiently delivers task-ready representations for multiple downstream tasks through a two-phase approach: encoding and inference (Figure 3).

During the encoding phase, raw videos are encoded into INR (e.g. NeRV) through two key operations: (1) *Representation Selection*, which identifies optimal pairings between INR's intermediate layers and backbone entry points using CKA-based pruning, and (2) *Adapter Tuning*, which trains lightweight *Micro Adapters* bridging INR's feature space to each backbone's expected representations. At inference time, NVP decodes only up to the required intermediate layer, passes features through the corresponding adapter, and injects them into the appropriate backbone layer. When multiple tasks query the same video, NVP reuses shared computations across overlapping INR hierarchy portions, minimizing redundant decoding.

4.1. Encoding Phase

During the encoding phase, NVP prepares the foundation for efficient task-specific representation generation. Given an INR-encoded video (either fully converged or having reached a target PSNR threshold) and a set of downstream task models, the NVP produces three important outputs for each task: (1) the optimal intermediate layer within the INR hierarchy to extract features from, (2) the corresponding injection point in the task's backbone network, and (3) a trained adapter that maps between these two representation spaces. This phase essentially constructs the task-specific pathways that will enable efficient selective decoding during inference.

Representation Selection. Determining which INR features to extract and where to inject them into task-specific backbones is non-trivial. Different tasks require varying levels of visual abstraction—from coarse semantic features for classification to fine spatial details for detection—while each video exhibits unique spatial and temporal characteristics. Exhaustively searching all possible INR-layer to backbone-layer combinations would require training and evaluating hundreds of adapters per task, making deployment prohibitively expensive.

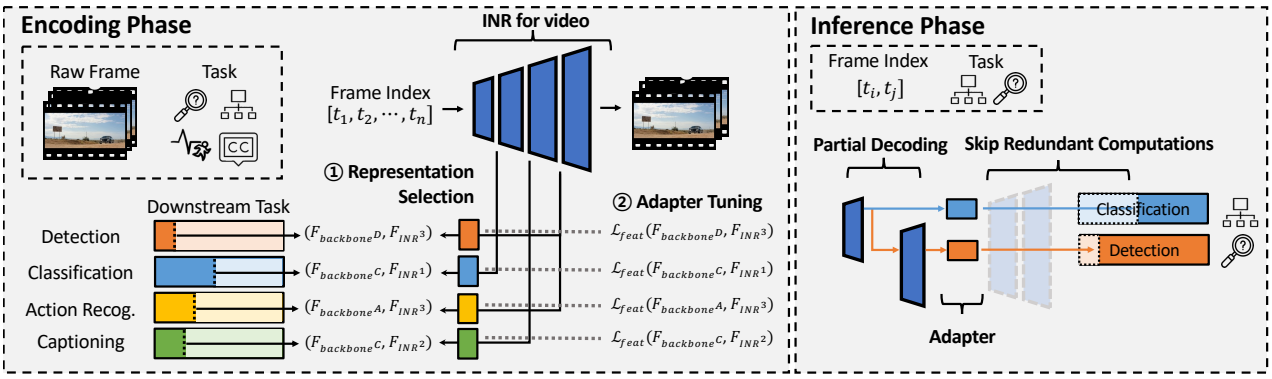

Figure 3. Overview of the Neural-Centric Video Pipeline (NVP).

To efficiently navigate this large search space, NVP employs Centered Kernel Alignment (CKA) [26], a metric that quantifies representational similarity between neural network activations. CKA measures alignment in kernel space rather than raw feature distance, providing robustness to scale variations and capturing structural correspondence between representations. For each potential pairing, we compute the CKA similarity between INR intermediate features F_{INR} and backbone features F_{backbone} as:

$$\text{CKA}(F_{\text{INR}}, F_{\text{backbone}}) = \frac{\| F_{\text{backbone}}^{\top} F_{\text{INR}} \|_F^2}{\| F_{\text{backbone}}^{\top} F_{\text{backbone}} \|_F \| F_{\text{INR}}^{\top} F_{\text{INR}} \|_F} \tag{1}$$

where $|\cdot|_F$ denotes the Frobenius norm. This CKA metric focuses on structural similarity rather than raw distance, making it robust to scale changes and partial overlap in features. By ranking all INR-backbone pairings by CKA similarity, NVP selects the top k alignments for each task (e.g., top 10%), pruning the vast number of potential candidates (often in the hundreds) down to a small, promising subset.

Adapter Tuning. Even when two feature sets appear similar in their representation spaces, a numerical feature-to-feature mapping is required to align representations. NVP introduces lightweight Micro Adapters—simple 1×1 convolutional layers designed for minimal computational overhead and rapid convergence. Unlike complex feature transfer methods that require tuning kernel sizes, strides, or pooling parameters, our approach decouples spatial and channel transformations: INR's intermediate features are first spatially resized via bilinear interpolation to match the target backbone's spatial dimensions, then Micro Adapters perform channel-wise transformation to produce the exact feature distribution expected by the downstream model. When alignment requires deeper transformation, it stacks additional convolutions to increase mapping capacity while maintaining architectural simplicity. Despite their critical role in bridging representation spaces, these adapters introduce negligible overhead, adding only 0.02-0.96% parameters relative to the backbone models they serve. This design decouples video representation from task-specific modeling, enabling new tasks to be added by training only lightweight adapters without retraining the encoded video.

To train these adapters effectively, NVP employs a composite loss function that balances multiple objectives for robust feature alignment:

$$\mathcal{L}_{\text{feat}}(\hat{F}, F) = \alpha \underbrace{\|\hat{F} - F\|_2^2}_{\text{MSE}} + \beta \underbrace{\left(1 - \frac{\hat{F} \cdot F}{\|\hat{F}\| \|F\|}\right)}_{\text{Cosine distance}} + \gamma \underbrace{\text{SmoothL1}(\hat{F}, F)}_{\text{Huber loss}} \tag{2}$$

where $\hat{F}$ represents the INR intermediate features and F denotes the target backbone features, with hyperparameters α, β, and γ weighting each term's contribution. This multi-objective formulation serves distinct purposes: The MSE term ensures accurate numerical reconstruction, the cosine similarity encourages alignment in orientation of features within the latent space, and the Huber loss robustly handles outliers and mismatches between INR and backbone representations. The MSE provides the primary learning signal while the cosine and Huber terms act as regularizers, enabling rapid convergence and stable adapter training.

Joint Optimization with Encoding. Since NVP represents videos as neural networks, gradients can flow through the entire pipeline from task predictions back to the INR weights. This end-to-end differentiability enables task-aware video encoding—the representation itself is shaped by downstream objectives during training, improving task performance under fixed bitrate constraints.

The joint optimization combines reconstruction quality with task-specific objectives:

$$\mathcal{L}_{\text{multi}} = \lambda_{\text{recon}} \mathcal{L}_{\text{recon}} + \sum_{i=1}^{T} w_i \left(\lambda_{\text{task}}^{(i)} \mathcal{L}_{\text{task}}^{(i)} + \lambda_{\text{feat}}^{(i)} \mathcal{L}_{\text{feat}}^{(i)} \right) \tag{3}$$

where $\mathcal{L}_{\text{recon}}$ preserves visual fidelity for human viewing, $\mathcal{L}_{\text{task}}$ optimizes task-specific objectives (e.g., cross-entropy for classification, bounding box regression for detection), and $\mathcal{L}_{\text{feat}}$ is the feature transfer loss. We set $\lambda_{\text{recon}} = 1.0$, $\lambda_{\text{task}} = 0.5$, $\lambda_{\text{feat}} = 0.2$ to prioritize reconstruction while allowing task gradients to shape features. Task weights w_i balance gradient contributions accross heterogeneous objectives.

4.2. Efficient Multi-task Inference

Upon receiving a task request, NVP leverages the layer routing information from the encoding phase—the selected intermediate layer, backbone injection point, and trained adapter—to execute partial decoding. NVP decodes only up to the identified intermediate layer, applies the corresponding Micro Adapter for feature transformation, and directly injects the adapted features into the target backbone. This selective layer routing eliminates the computational overhead of both pixel reconstruction and early backbone layers, reducing inference cost to only the essential operations required for the specific task.

Multi-Task Serving. NVP's architectural design naturally supports efficient concurrent task execution through hierarchical computation sharing. Since all tasks operate on a unified INR representation with progressively decoded features, intermediate computations can be shared across tasks with overlapping layer requirements. When multiple tasks request features from the same video, NVP computes shared intermediate layers only once: tasks requiring deeper features reuse the outputs from shallower layers computed for other tasks, extending the decoding only for their additional requirements. This computation sharing is inherently enabled by the unified representation—in contrast, systems using task-specific encodings must redundantly process each task independently, missing these sharing opportunities.

5. Experiments

Tasks, Models, and Datasets. We evaluate NVP across four tasks: classification, object detection, action recognition, and video captioning. For classification and object detection, we sample 3,000 clips from ImageNet VID 2015 [37]; for action recognition, 3,000 clips from UCF101 [42]; and for captioning, 100 videos from MSR-VTT [47]. We employ standard pretrained models without fine-tuning to isolate representation effectiveness. Specifically, we use ResNet-50 [20], CLIP-RN50 [36], and ViT-B/16 [11] for classification; SlowFast [14] and I3D [3] for action recognition; DETR [2] for object detection; and BLIP [30] for captioning.

INR Architectures. Our implicit neural representation follows the NeRV [4] and HNeRV [6] frameworks. The embedding dimension and channel counts determine encoding capacity. We conducted preliminary experiments across various resolutions and frame counts to identify architectures achieving PSNR $\geq$ 33 dB, then assigned appropriate architectures to each video based on its resolution and frame count. The resulting INR models consist of 3–6 hierarchical decoding blocks, each comprising convolutional layers followed by pixel-shuffling operations to progressively increase spatial resolution. Detailed architecture specifications are provided in the supplementary material.

Evaluation Metrics. For computational efficiency, we measure inference latency (milliseconds) and computational complexity (FLOPs), following standard convention (1 MAC = 1 FLOP). Total cost includes representation preparation (decoding, preprocessing, GPU transfer) and downstream model inference. For task performance, we use: (1) top-1 accuracy for classification and action recognition, (2) mean average precision (mAP) for object detection, and (3) CLIP-score [21] for video captioning. We use ground-truth annotations for object detection and video captioning. For classification, we generate pseudo ground-truth labels using predictions from large pretrained models in the same backbone family (ResNet/ViT pretrained on ImageNet-21K), while ImageNet labels serve as ground truth for CLIP-based models.

All experiments are conducted on a workstation equipped with an Intel Xeon Silver 4210R CPU (10 cores, 2.4 GHz, 20 threads) and a single NVIDIA GeForce RTX 4090 GPU (24GB VRAM). Video encoding and decoding operations use FFmpeg 6.0 with NVENC/NVDEC support for GPU-accelerated codec operations. Further detail on the encoding and decoding are provided in the supplement.

Training Details. We consider two training modes: adapter-only with frozen INR, and joint optimization of INR and adapters. For frozen training, we use 30 epochs with early stopping and learning rate 1×10^{-4}. Joint training performs 100-epoch INR warm-up followed by 100-epoch joint optimization, with learning rates 1×10^{-5} (INR) and 1×10^{-4} (adapters). Both modes use batch size 4, loss weights $(\lambda_{\text{recon}}, \lambda_{\text{task}}, \lambda_{\text{feat}}) = (1.0, 0.5, 0.2)$, and feature transfer weights $(\alpha, \beta, \gamma) = (1.0, 0.2, 0.1)$. We select top-K=10 layer pairs via CKA-based pruning. At inference, the INR decodes to the maximum required depth, sharing features across multiple tasks. Full details are provided in the supplement.

5.1. Comparison with traditional pipeline

Baselines. We evaluate the computational efficiency of our pipeline against three baseline strategies: H.264 decoding (CPU and GPU) and full INR (NeRV, HNeRV) decoding, all with complete RGB reconstruction before inference. Table 1 demonstrates how NVP with trained adapters maintains baseline model performance while achieving substantial gains in both latency and computational cost. Here, input process encompasses all operations from initial codec decoding through image loading, resizing, and GPU transfer until deep learning model inference begins.

Processing Time (Latency). Our pipeline achieves up to 89.46% reduction in total inference latency (for CLIP-RN50) compared to H.264-based approaches, demonstrating significant speedup. This improvement stems from three key factors.

Model	Runtime Path	Processing Time (ms)			Compute (GFLOPs)			Task Accuracy
		Input Process	Inference	Total	Input Process	Inference	Total	
Classification								
ResNet50	CPU/GPU Codec	53.76/10.35	6.82	60.59/17.17	-	4.134	-	74.59%
	NeRV/HNeRV	8.72/6.28	6.63	15.35/15.00	105.03/46.85	4.134	109.16/50.98	73.68%
	NVP(NeRV)	5.31	4.57	9.88 (83.69%↓/35.64%↓)	23.84	2.987	26.827 (75.42%↓)	76.73%
	NVP(HNeRV)	3.14	4.67	7.81 (87.11%↓/47.93%↓)	2.21	2.898	5.108 (89.98%↓)	76.74%
CLIP-RN50	CPU/GPU Codec	57.04/10.32	8.13	65.17/18.45	-	6.126	-	89.65%
	NeRV/HNeRV	8.73/6.28	8.13	16.86/14.41	105.03/46.85	6.126	111.16/52.98	87.13%
	NVP(NeRV)	2.79	5.86	8.65 (86.73%↓/48.70%↓)	14.82	5.283	20.103 (81.91%↓)	90.92%
	NVP(HNeRV)	1.05	5.82	6.87 (89.46%↓/52.32%↓)	1.78	5.293	7.073 (86.64%↓)	90.86%
ViT	CPU/GPU Codec	53.07/10.32	9.74	62.81/20.06	-	11.286	-	83.05%
	NeRV/HNeRV	8.73/6.28	9.74	18.47/14.41	105.03/46.85	11.286	116.32/58.14	79.15%
	NVP(NeRV)	5.33	8.76	14.09 (77.57%↓/23.71%↓)	36.12	11.054	47.174 (59.44%↓)	81.57%
	NVP(HNeRV)	3.46	8.76	12.22 (80.54%↓/15.20%↓)	6.58	11.054	17.634 (69.67%↓)	81.57%
Action Recognition								
SlowFast (32f)	CPU/GPU Codec	28.57/27.76	16.54	45.11/44.30	-	50.84	-	62.0%
	NeRV/HNeRV	113.54/70.12	16.54	130.08/86.66	1618.15/460.84	50.84	1668.99/511.8	62.0%
	NVP(NeRV)	20.04	14.57	34.61 (23.28%↓/73.39%↓)	128.39	48.54	176.93 (89.40%↓)	61.72%
	NVP(HNeRV)	18.82	14.57	33.39 (57.68%↓/77.97%↓)	95.27	48.37	143.64 (71.93%↓)	61.76%
I3D (32f)	CPU/GPU Codec	30.57/27.73	7.34	37.91/35.08	-	114.72	-	62.0%
	NeRV/HNeRV	113.54/70.12	7.34	120.88/77.46	1618.15/460.84	114.72	1732.87/575.56	62.0%
	NVP(NeRV)	18.24	7.32	25.56 (32.58%↓/78.86%↓)	113.34	112.27	225.61 (86.98%↓)	59.98%
	NVP(HNeRV)	18.84	7.32	26.16 (30.99%↓/66.23%↓)	95.37	112.27	207.64 (63.92%↓)	60.35%
Object Detection								
DETR	CPU/GPU Codec	76.40/9.73	32.32	108.72/42.05	-	84.69	-	0.4436
	NeRV/HNeRV	8.72/6.28	32.32	42.04/38.6	105.03/46.85	84.69	189.72/131.54	0.4298
	NVP(NeRV)	5.11	31.02	36.13 (66.77%↓/14.06%↓)	37.02	84.45	121.47 (35.97%↓)	0.4385
	NVP(HNeRV)	3.97	31.02	34.99 (67.82%↓/9.35%↓)	13.21	84.45	97.66 (25.76%↓)	0.4395
Captioning								
BLIP (8f, 20tok)	CPU/GPU Codec	13.23/3.12	213.70	226.93/216.82	-	183.44	-	0.2952
	NeRV/HNeRV	44.85/26.21	213.70	258.55/239.91	404.51/115.21	183.44	587.95/298.65	0.2950
	NVP (NeRV)	6.38	213.53	219.91 (3.09%↓/14.95%↓)	34.28	183.44	217.72 (62.97%↓)	0.2947
	NVP(HNeRV)	5.92	213.53	189.36 (16.56%↓/21.07%↓)	32.12	183.44	215.56 (27.82%↓)	0.2949

Table 1. Comparison of decoding-inference pipelines across models and tasks. Red indicates improvement over CPU-based pipeline (H.264 decoding + preprocessing); Blue indicates improvement over INR pipeline. Task categories are denoted as section headers. For object detection and video captioning, mAP and CLIP-score similarity are reported, respectively.

BPP	0.025	0.05	0.075
H.264	82.60%	89.97%	90.78%
NVP(Ours)	**90.12%**	**91.30%**	**91.02%**

(a) CLIP-RN50 (Top-1 Accuracy)

BPP	0.02	0.08	0.14
H.264	52.89%	60.06%	**64.62%**
NVP(Ours)	**63.38%**	**61.73%**	63.55%

(b) SlowFast (Top-1 Accuracy)

BPP	0.05	0.15	0.25
H.264	0.4005	**0.4387**	0.3888
NVP(Ours)	**0.4283**	0.4323	**0.4398**

(c) DETR (mAP50)

Table 2. BPP (Bits Per Pixel) vs. task performance. **Bold** indicates the higher accuracy at each BPP level.

First, decoding cost itself is substantially reduced. While traditional codecs require decoding from keyframes (I-frames), INR directly decodes target frames on-demand. This effect is amplified when sampling frames rather than decoding all consecutive frames. Additionally, INR inherently offers fast decoding speed, and we further accelerate this through partial decoding that generates only necessary intermediate representations. Second, we eliminate preprocessing overhead entirely. Traditional pipelines require image loading, resizing, normalization, and other preprocessing steps, whereas adapters in NVP handle these operations implicitly. As shown in Figure 4, among preprocessing operations, loading and resizing consume the majority of time, with CPU-based approaches incurring additional overhead from transformation operations. NVP completely bypasses these stages, contributing significantly to the achieved speedup. Third, we skip early layers of the downstream task model. By directly injecting task-specific features into intermediate layers, we avoid redundant computation in the backbone's initial processing stages. This layer-skipping effect is particularly pronounced for CNN-based models, where early convolutional layers account for substantial computational cost.

Computational Complexity (FLOPs). We compare computational complexity in FLOPs against the full-frame INR baseline. Since traditional codecs are not neural-based, di-

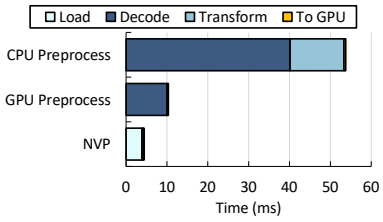

Figure 4. Preprocessing time comparison.

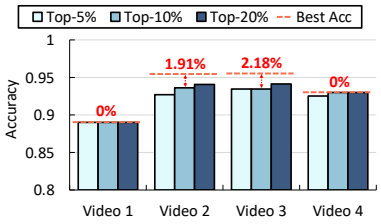

Figure 5. Accuracy of CKA-based pruning.

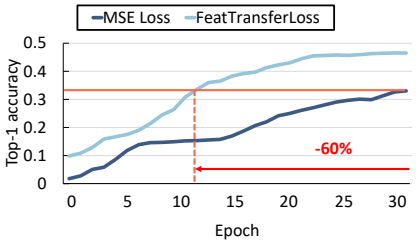

Figure 6. Convergence of FeatTransfer loss vs. MSE loss.

rect FLOP comparison is inappropriate; comparing with Full INR allows us to precisely quantify how NVP reduces computation at each pipeline stage.

INR decoding blocks exhibit exponentially increasing computational complexity with resolution. By selectively omitting later decoding stages—which are computationally heaviest—our approach achieves substantial FLOP reduction of up to 89.98% in total (for ResNet-50). While the majority of savings come from efficient partial decoding, backbone layer skipping through adapters additionally reduces inference model FLOPs by up to 29.9%, further decreasing end-to-end latency.

Task Accuracy. Our neural-centric pipeline maintains task accuracy comparable to traditional baselines. In some cases, the continuous and stable characteristics of neural representations even yield higher classification accuracy than image-based inputs. For instance, using ResNet50, our pipeline achieves 76.74% accuracy, surpassing both H.264 (74.59%). Notably, this improvement is achieved without explicit supervision on task labels—the continuous nature of neural representations naturally provides more stable outputs. Unlike frame-based approaches where the same object may produce inconsistent predictions within a video, our approach demonstrates robustness against temporal inconsistency. When comparing per-frame consistency across an entire video, our method achieves 4.8% higher accuracy, exhibiting superior temporal stability. Even within the same NeRV-based decoding framework, our task-specific intermediate representations outperform pixel-level decoding.

Compression Efficiency. Beyond inference efficiency, compression performance is equally important for storage and transmission. Table 2 presents accuracy versus bitrate per pixel (bpp) across different tasks. NVP operates at remarkably low bitrates—most configurations stay below 0.5 bpp—while maintaining comparable or superior accuracy to H.264 at much higher bitrates. Notably, the compression rate is determined by which INR layer is used for partial decoding, and this choice does not affect task accuracy. This decoupling allows NVP to flexibly adapt compression levels based on bandwidth or storage constraints without sacrificing downstream task performance.

Joint Training Benefits. A key advantage of NVP is that the video representation itself is a neural network, enabling true end-to-end optimization with downstream tasks. To demonstrate this, we jointly trained the INR with task-specific loss functions from ResNet50 and DETR. The jointly optimized models achieve 83.27% (ResNet50) and 0.4438 mAP50 (DETR), surpassing adapter-only training results. This demonstrates that co-optimizing the video representation with task objectives yields higher accuracy, with potential for further gains when training with multiple tasks simultaneously on a shared representation.

Method	FPS	# Params	Domain	Visualize
CoVIAR [1]	172	58.351M	Action recognition	–
MMNet [44]	10 - 41	18.882M	Detection	–
DeepSVC [32]	10	59.3M	Both	O
NVP (Ours)	833/564	0.4236M/0.4827M	Both	O

Table 3. Comparison with machine-centric approaches. NVP results are reported for action recognition and object detection tasks, respectively.

Feature	Omnipotent	Comp.Vision	NVP
Visualization	X	O	O
Encoding Time	—	Very Short	Long (one-time)
Decoding Speed	—	Fast	Very Fast
Feature Extractor	Own Encoder	VQ-VAE	NeRV-variants
Task Scalability	Train from Scratch		Adapter Only
End-to-End Opt.	O	X	O

Table 4. Comparison with alternative approaches.

5.2. Comparison with Machine-Centric Approaches

We compare against CDI methods [1, 44] and VCM approaches [15, 32, 46].

Single-Task Performance. Table 3 compares NVP with machine-centric approaches. DeepSVC only extracts features at 10 FPS—significantly slower than NVP. Leveraging INR's compact representation, NVP requires far fewer parameters (0.42M/0.48M vs 59.3M for DeepSVC). Prior methods target specific domains (action recognition or detection), whereas NVP supports diverse tasks. In terms of compression efficiency, DeepSVC achieves 95.95% of baseline performance at 0.0316 bpp, while NVP surpasses baseline accuracy at 0.02 bpp.

Multi-Task Scenarios. The advantages of NVP become more pronounced in multi-task settings. CDI and VCM methods require separate models per task, accumulating parameters linearly with the number of tasks. In contrast, NVP uses a single shared representation, with total parameters determined by the largest task require-

Model	# Params (M)	
	Model	Adapter
ResNet-50	25.56	0.0039 (0.02%)
CLIP-RN50	23.53	0.0092 (0.04%)
ViT-B/16	58.07	0.0875 (0.15%)
SlowFast-R50	34.57	0.335 (0.96%)
I3D-R50	28.04	0.246 (0.88%)
DETR-R50	41.50	0.1342 (0.32%)
BLIP	247.41	0.483 (0.20%)

Table 5. Adapter parameters vs. full model parameters.

ment—supporting both action recognition and detection requires only 0.4827M parameters, regardless of task count.

Adaptation and Scalability. Adding new tasks reveals another critical advantage. CDI methods must redesign or retrain models to process codec-extracted features. Among VCM approaches, feedforward methods such as Omnipotent and Compressed Vision offer fast encoding but require task-specific backbone modifications and full retraining for new tasks—for example, adapting Compressed Vision requires retraining S3D on Kinetics-600, typically taking several days on multiple GPUs. As summarized in Table 4, neither supports end-to-end optimization, and Omnipotent does not support visualization. In contrast, NVP adds new tasks by training lightweight adapters in under 5 minutes, without modifying the stored video representation.

5.3. Ablation Study

Efficiency of Micro-Adapters. Table 5 summarizes the parameter counts for micro-adapters across various backbone models. Specifically, it reports the average number of adapter parameters required for each backbone architecture, highlighting their relative size compared to the full models.

The lightweight micro-adapters introduce minimal parameter overhead, with the largest adapter (BLIP) accounting for only 0.19% of the total model parameters. For smaller backbone models, adapters are even more compact. For instance, ResNet-50 adapters use as little as 0.02% of the total parameters. The slightly larger adapter size in I3D arises from its inherent Conv3D architecture, but even in this case, the overhead remains under 1%. Such minimal additions have negligible impact on overall model size and computational load, highlighting the practicality and efficiency of our representation integration strategy.

Validation of CKA-based Pruning. We evaluate the validity of our CKA-based pruning strategy by comparing it with exhaustive search results on action recognition (I3D model). Figure 5 shows that pruning adapters based on top-CKA scores achieves optimal accuracy or remains within a 2% margin, despite training only the top 5%, 10%, or 20% of adapter candidates. Considering this selective approach reduces computational cost by 68.8% compared to exhaustive search, our method efficiently identifies near-optimal representations.

Impact of Feature-Transfer Loss Function. We compare our feature-transfer loss against a baseline that uses only MSE for adapter training. As shown in Figure 6, adapters trained with feature-transfer loss converge 60% faster on average and consistently reach higher top-1 pseudo-label accuracy. MSE can lead to unstable convergence and slower alignment, particularly when feature channels differ substantially in scale. However, the cosine and Huber components encourage channel-wise orientation alignment, yielding a balanced and semantically meaningful adaptation. Consequently, feature-transfer loss delivers faster convergence and higher accuracy compared to using MSE alone.

6. Discussion & Future Work

Limitations and Future Directions. While our straightforward feature-to-feature mapping simplifies task adaptation, it can lead to reduced accuracy in pixel-sensitive tasks like object detection; incorporating task-specific losses such as bounding-box regression into adapter training is a natural extension. Additionally, the NeRV-based encoder may not be optimal for transformer-based backbones, and future backbone-aware INR architectures could further extend NVP's effectiveness.

Encoding Cost Amortization. A practical concern with INR-based pipelines is the high one-time encoding cost: fitting NeRV and HNeRV to a 10-second 1080p video requires approximately 30 and 12 minutes, respectively. However, in WORM scenarios common to VOD and large-scale video analytics, this cost is amortized over repeated inference. Taking SlowFast as an example, break-even is reached at roughly 150K and 60K queries for NeRV and HNeRV—equivalent to only 6–15 queries per user assuming 10K users—and in multi-task settings, query counts accumulate even faster. More importantly, INR encoding speed is improving rapidly: E-NeRV, HNeRV, and MetaNeRV [18] achieve 8×, 16×, and 30× speedups over NeRV, and as NVP is applicable to NeRV-variant, the amortization point will continue shrinking as the field advances.

7. Conclusion

We propose NVP, a neural-centric video pipeline that optimizes multi-task video inference by storing and processing videos entirely in neural feature space. NVP leverages hierarchical INR features with lightweight adapters to directly feed task-specific representations to downstream models, eliminating the decode-preprocess-extract overhead of traditional pipelines. Our system achieves substantial reductions in latency and computational cost while maintaining comparable accuracy across diverse video tasks, demonstrating the practicality of neural-native video representations for real-world multi-task systems.

Acknowledgements. We appreciate anonymous reviewers for providing constructive feedback and suggestions. This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (Ministry of Science and ICT) [No. RS-2024-00406715 and No. RS-2024-00340099].

References

[1] Haoyuan Cao, Shining Yu, and Jiashi Feng. Compressed video action recognition with refined motion vector, 2019. 1, 2, 7

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 5

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 5

[4] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos, 2021. 3, 5

[5] Hao Chen, Matt Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. Hnerv: A hybrid neural representation for videos, 2023. 3

[6] Hao Chen, Matt Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. Hnerv: A hybrid neural representation for videos. *arXiv preprint arXiv:2304.02633*, 2023. 5

[7] Hyomin Choi and Ivan V. Bajic. Deep feature compression for collaborative object detection, 2018. 1, 2

[8] Hyomin Choi and Ivan V. Bajic. Near-lossless deep feature compression for collaborative intelligence, 2018. 1, 2

[9] Hyomin Choi and Ivan V. Bajic. Scalable image coding for humans and machines. *IEEE Transactions on Image Processing*, 31:2739–2754, 2022. 1

[10] Felipe Codevilla, Jean Gabriel Simard, Ross Goroshin, and Chris Pal. Learned image compression for machine perception, 2021. 1, 2

[11] Alexey Dosovitskiy and .... An image is worth 16×16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2, 5

[12] Ling-Yu Duan, Vijay Chandrasekhar, Shiqi Wang, Yihang Lou, Jie Lin, Yan Bai, Tiejun Huang, Alex Kot, and Wen Gao. Compact descriptors for video analysis: the emerging mpeg standard. *IEEE MultiMedia*, PP, 2017. 1, 2

[13] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 1

[14] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2019. 1, 2, 5

[15] Ruoyu Feng, Xin Jin, Zongyu Guo, Runsen Feng, Yixin Gao, Tianyu He, Zhizheng Zhang, Simeng Sun, and Zhibo Chen. Image coding for machines with omnipotent feature learning, 2022. 1, 2, 7

[16] Wen Gao, Shan Liu, Xiaozhong Xu, Manouchehr Rafie, Yuan Zhang, and Igor Curcio. Recent standard development activities on video coding for machines, 2021. 1, 2

[17] Xingtong Ge, Jixiang Luo, Xinjie Zhang, Tongda Xu, Guo Lu, Dailan He, Jing Geng, Yan Wang, Jun Zhang, and Hongwei Qin. Task-aware encoder control for deep video compression, 2024. 1, 2

[18] Jialong Guo, Ke liu, Jiangchao Yao, Zhihua Wang, Jiajun Bu, and Haishuai Wang. Metanerv: Meta neural representations for videos with spatial-temporal guidance, 2025. 8

[19] 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, 2023. 3

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 5

[21] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning, 2022. 5

[22] Ademola Ikusan and Rui Dai. Deep feature compression with rate-distortion optimization for networked camera systems. In *Proceedings of the 14th ACM Multimedia Systems Conference*, page 86–96, New York, NY, USA, 2023. Association for Computing Machinery. 1, 2

[23] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: optimizing neural network queries over video at scale. *Proc. VLDB Endow.*, 10(11):1586–1597, 2017. 1

[24] 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, 2023. 3

[25] Jina Kim, Jihoo Lee, and Je-Won Kang. *SNeRV: Spectra-Preserving Neural Representation for Video*, page 332–348. Springer Nature Switzerland, 2024. 3

[26] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019. 4

[27] Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. Hinerv: Video compression with hierarchical encoding-based neural representation. In *NeurIPS*, 2023. 3

[28] Bing Li, Jiaxin Chen, Dongming Zhang, Xiuguo Bao, and Di Huang. Representation learning for compressed video action recognition via attentive cross-modal interaction with motion enhancement, 2022. 1

[29] Han Li, Shaohui Li, Shuangrui Ding, Wenrui Dai, Maida Cao, Chenglin Li, Junni Zou, and Hongkai Xiong. Image compression for machine and human vision with spatial-frequency adaptation, 2024. 2

[30] Junnan Li, Dongxu Li, Yue Shan, Xiujun Yin, Pengchuan Wang, Lei Zhang, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022. 2, 5

[31] 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 *ECCV*, 2022. 3

[32] Hongbin Lin, Bolin Chen, Zhichen Zhang, Jielian Lin, Xu Wang, and Tiesong Zhao. Deepsvc: Deep scalable video

coding for both machine and human vision. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 9205–9214, New York, NY, USA, 2023. Association for Computing Machinery. 1, 2, 7

[33] Jinming Liu, Heming Sun, and Jiro Katto. Improving multiple machine vision tasks in the compressed domain. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 331–337, 2022. 1

[34] MPEG. Mpeg-nnr (neural network representation). https://www.mpeg.org/standards/Explorations/34/, 2024. Accessed: 2025-11-11. 1, 2

[35] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 54(2):1–38, 2021. 1

[36] Alec Radford and .... Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2, 5

[37] Olga Russakovsky and .... Imagenet large scale visual recognition challenge. In *IJCV*, 2015. 5

[38] Yaojie Shen, Xin Gu, Kai Xu, Heng Fan, Longyin Wen, and Libo Zhang. Accurate and fast compressed video captioning, 2024. 1

[39] Xihua Sheng, Li Li, Dong Liu, and Houqiang Li. Vnvc: A versatile neural video coding framework for efficient human-machine vision, 2023. 1

[40] Zheng Shou, Xudong Lin, Yannis Kalantidis, Laura Sevilla-Lara, Marcus Rohrbach, Shih-Fu Chang, and Zhicheng Yan. Dmc-net: Generating discriminative motion cues for fast compressed video action recognition, 2019. 1, 2

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[42] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 5

[43] 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. 1, 2

[44] Shiyao Wang, Hongchao Lu, and Zhidong Deng. Fast object detection in compressed video, 2019. 7

[45] 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. 1, 2

[46] Olivia Wiles, Joao Carreira, Iain Barr, Andrew Zisserman, and Mateusz Malinowski. Compressed vision for efficient video understanding, 2022. 7

[47] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5288–5296, 2016. 5

[48] Ning Yan, Changsheng Gao, Dong Liu, Houqiang Li, and Li Li. Sssic: Semantics-to-signal scalable image coding with learned structural representations. *IEEE Transactions on Image Processing*, PP:1–1, 2021. 1

[49] Juncheol Ye, Seungkook Lee, Hwijoon Lim, Jihyuk Lee, Uitaek Hong, Youngjin Kwon, and Dongsu Han. Sand: A new programming abstraction for video-based deep learning. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, page 589–605, New York, NY, USA, 2025. Association for Computing Machinery. 1