EV-JEPA: ELASTIC V-JEPA FOR ANY DEVICE AND CROSS-DEVICE INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision models like V-JEPA are increasingly deployed across a heterogeneous edge–cloud continuum spanning smart glasses, robots, and servers. Yet, they are trained as rigid, fixed-capacity networks bound to a single device. Prior work addresses one axis at a time: slimmable and NAS-based methods provide width elasticity but assume single-device inference, while split-computing methods enable cross-device inference only at fixed partitions and static capacities. We introduce Elastic V-JEPA (EV-JEPA), a model that is width-reconfigurable and natively partitionable across heterogeneous devices. At its core is Transition-Aware Triplet Training: at every step, a pair of widths is trained both standalone and as the endpoints of a cross-device hybrid configuration that transitions between them at a split layer, so the same weights serve single-width and cross-width execution. Across SSv2, K400, and EK100, EV-JEPA stays within 1 point of independently trained baselines and improves over prior elastic training by up to 11 points. Hybrid execution recovers to within 2 points of the higher-capacity standalone model under many-to-one, any-to-any, and variable-split configurations. The same principle extends to an elastic token compressor that trades on-device computation, latency, and accuracy. The pretrained models and code will be released upon acceptance.

1 INTRODUCTION

Vision foundation models (VFMs) like V-JEPA [2] are a cornerstone of modern computer vision and AI, powering applications that range from classification [1; 2] to action anticipation and planning [1; 11]. Traditionally, these models are designed to run on powerful accelerators. However, the rise of wearable smart glasses [14] and robotics [32] is reshaping where such models must operate. Rather than relying on a centralized cloud, the deployment infrastructure is evolving into a distributed edge-cloud continuum [44; 13]. A heterogeneous, multi-tier architecture where edge devices with different compute budgets and remote servers cooperate to serve inference. Conventionally, such cooperation has involved edge devices streaming raw video to the cloud, which is at odds with the need for smart glasses and robots to assist in real time. Streaming is power-hungry and slow due to low-bandwidth wireless links the edge relies on [12], making transmission a primary bottleneck. These constraints motivate research on efficient VFMs [27; 21; 33; 26], promoting on-device compute. Yet, such approaches share two key limitations: i) they train a separate model per compute budget, and ii) they treat each device in isolation, so an edge device is capped at its own budget and the collaborative compute still relies on video streaming.

To address i), one line of work makes a single network serve multiple compute budgets (elastic networks), whether by searching sub-architectures from a supernetwork [4] or through variable widths [10; 42; 20]. These methods differ in how far elasticity reaches, from a sublayer's width [10] to the hidden dimension itself [42], but not in what they produce: a set of self-contained models (one per compute budget) whose representations are not cross-width compatible. They are intended to serve different individual devices, but not to split a singl inference across them. A separate line of work addresses ii) by splitting the network's workload across multiple heterogeneous compute nodes [12]. While effective for exploiting distributed hardware, they assume a fixed partition and a static per-node capacity, forfeiting standalone execution and treating distributed inference as a requirement rather than an option. As a result, existing approaches provide either width elasticity or collaboration, but not both. The two failures interlock as elastic methods cannot carry a representation across widths, and split methods can serve only rigid architectures. This raises a fundamental question: can we train

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.

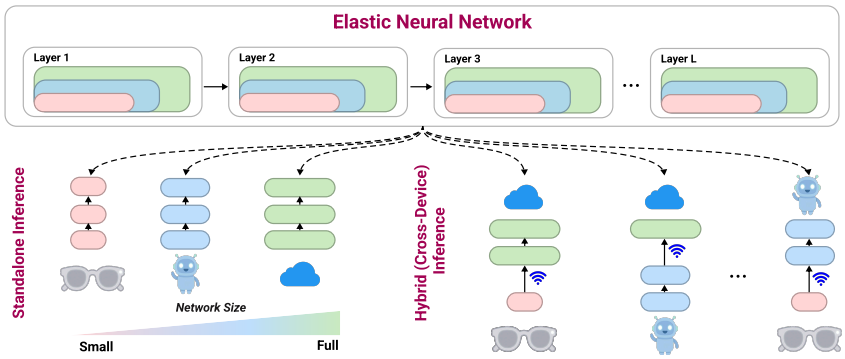


Figure 1: **One elastic model for the device continuum.** We propose a single *elastic network* that is trained once and deployed across the entire continuum: (1) its width is reconfigurable at inference time, allowing the same parameters to serve any compute budget ***(Standalone Inference)***; and (2) its computation can be *split* between devices, with early layers executed on-device and remaining layers on a stronger device, communicating only compact intermediate representations ***(Hybrid Inference)***.

neural networks that are both elastic and collaborative, and be able to run fully on a single device and seamlessly split their computation across multiple ones when needed?

To answer this question, we introduce a novel approach that achieves both width elasticity and collaborative execution instantiated on V-JEPA [2] model named as **Elastic V-JEPA (EV-JEPA)** (Fig. 1). EV-JEPA builds on a vision transformer backbone and supports full-width elasticity across all layer components (Fig. 1 **Standalone Inference**). In addition, beyond standalone width elasticity, we depart from previous works by introducing a *hybrid* (mixed-width) execution mode in which early layers run at a smaller width and the remainder at a larger one, transitioning at a variable split layer. This allows a small device to initiate inference and a more capable one to complete it, thereby improving inference accuracy by streaming tokens rather than videos. However, it imposes a requirement that no elastic method has had to satisfy: representations that are cross-width compatible.

At the core of our approach is a novel training paradigm we term Transition-aware Triplet Training (T3). Unlike prior elastic approaches that optimize a randomly selected width [10] or rigid *{smallest, random, largest}* triplet [38; 39], often resulting in performance degradation at intermediate capacities, T3 continuously samples a *{smaller, hybrid, larger}* triplet. The same width pair is therefore optimized simultaneously as two standalone configurations, and as the endpoints of the hybrid path that connects them, so cross-width compatibility is trained directly as a native capability. Additionally, T3 builds on the same training frameworks as the original V-JEPA with one additional twist: the self-evolving EMA teacher is replaced with a fixed pretrained V-JEPA 2 [1] encoder to provide a stable target space across all widths. This means EV-JEPA is trained with a dense predictive objective where the student must estimate teacher representations per token, an objective significantly stricter than agreeing on a global representation (i.e., classification) [38; 10; 42], making elasticity even harder. To our knowledge, this is the first elastic framework proposed on a densely supervised model. Finally, to close the gap between a network that can be split in principle and one that runs efficiently over a device continuum connected through low-bandwidth links, we also experiment coupling hybrid execution with an elastic, controllable token-compression layer to ease the cross-device communication.

We evaluate our framework on Something-Something V2 (SSv2) [17], Kinetics-400 (K400) [23], as well as Epic-Kitchens 100 (EK100) [9] for action anticipation. Our empirical results show that EV-JEPA trained with T3 outperforms other elastic training regimes.In summary, our contributions are threefold. First, we introduce **EV-JEPA**, an elastic VFM whose width is reconfigurable across all layer components and that natively supports hybrid execution. Second, we propose **T3** which optimizes a width pair simultaneously as two standalone configurations and as the endpoints of the hybrid path connecting them, making cross-width compatibility a trained property. Third, we show on multiple benchmarks that EV-JEPA stays within 1 point of independently trained baselines while improving over prior elastic training by up to 11 points resolving the intermediate subnetwork degradation [42; 38]., and that hybrid execution recovers to within 1.1 points of the highest-capacity. We then show that elasticity extends to token compression, reducing intermediate communication.

2 Related Works

Elastic Networks. OFA [4] and BigNAS [40] train CNN supernets treating width, depth, kernel size, and resolution as elastic dimensions, from which sub-architectures are extracted post hoc, with later work refining training and search [18; 35] and a related line coupling weight sharing with NAS-style search over transformers [7; 8; 16]. These methods frame elasticity as a search problem whose space grows combinatorially as the gap between the smallest and largest configurations widens. Slimmable approaches instead trade search for a fixed set of nested widths sharing weights. Slimmable networks and their universal extension [39; 38] jointly train fixed widths with switchable batch normalization, the sandwich rule, and in-place distillation later carried to transformers by DynaBERT [20] and to ViTs [19; 42] up to MatFormer [10], which applies the matryoshka principle [24] to FFN blocks. However, these methods share two limitations: (1) degraded performance at intermediate widths [5], and (2) they target isolated hardware. We propose EV-JEPA, a single model whose width is reconfigurable across all layer components and that natively supports mixed-width execution, splitting compute from a smaller to a larger device and addressing both limitations.

Split Computing. Neurosurgeon [22] proposed layer-granularity cross-device splitting with a runtime-selected partition point. SPINN [25] added early exits and co-optimized both policies at runtime. SplitNets [12] targeted head-mounted systems, designing architectures for distributed computation across sensors and edge servers. These methods demonstrate the value of collaborative execution but assume fixed distributed inference architectures. We instead propose a single model that is both elastic and natively partitionable, trained end-to-end to support standalone and collaborative inference.

3 Method

3.1 Problem Formulation

We consider the setting of *elastic & distributed inference*, where a model operates across multiple compute capacities and optionally distributes computation across devices. Let f_θ denote a model parameterized by θ. Instead of a fixed architecture, we assume the model supports a set of *width configurations* $w \in \mathcal{W}$ where $\mathcal{W}$ is a predefined set of supported widths corresponding to different compute budgets (e.g., $\mathcal{W} = \{192, 384, \ldots, 1408\}$). For a given width w, we denote the corresponding subnetwork as:

$$f_\theta^{(w)}(x), \tag{1}$$

which represents the model executed entirely at width w. Elastic inference requires that a single set of parameters θ performs well across all configurations $w \in \mathcal{W}$. In distributed deployments, inference may also be *split across devices*. A resource-constrained device may execute an initial portion of the model using width w_s, after which the compute is transferred to a stronger device that continues computation using a larger width w_l, with $w_s < w_l$. We therefore define a *hybrid configuration*:

$$f_\theta^{(w_s, w_l, k)}(x), \tag{2}$$

where the first k layers operate with width w_s and the remaining $L - k$ layers operate with width w_l. Our goal is therefore twofold. First, the model should achieve strong performance across all widths $f_\theta^{(w)}$, enabling deployment on devices with different compute budgets. Second, the model should support *distributed execution*, where computation can be distributed across devices.

3.2 Elastic Transformer Architecture

We design a transformer architecture whose hidden dimension can vary across a predefined set of widths $\mathcal{W}$. Each width configuration corresponds to a subnetwork obtained by selecting the first w channels of the model parameters. Concretely, following slimmable networks [39] and Matformer [10], the parameters of the largest model with width $w_{\max}$ are shared across all configurations through weight slicing. Consider a transformer layer with attention projections $W_Q, W_K, W_V \in \mathbb{R}^{w_{\max} \times w_{\max}}$. As shown in Fig. 2 (a), a smaller configuration with $w < w_{\max}$ uses the sliced parameters:

$$W_Q^{(w)} = W_Q[1{:}w, 1{:}w], \quad W_K^{(w)} = W_K[1{:}w, 1{:}w], \quad W_V^{(w)} = W_V[1{:}w, 1{:}w]. \tag{3}$$

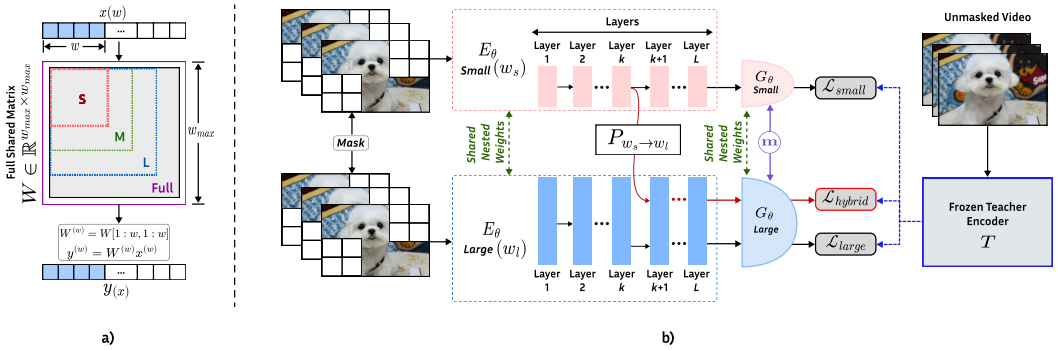


Figure 2: **Elastic architecture and Transition-Aware Triplet Training (T3). (a)** Weight slicing. All width configurations share a single set of parameters: for a layer weight $W \in \mathbb{R}^{w_{\max} \times w_{\max}}$, a width-$w$ subnetwork uses the top-left slice $W^{(w)} = W[1{:}w, 1{:}w]$. **(b)** T3 training. At each iteration we sample a width pair (w_s, w_l) and a split layer k, and jointly optimize three paths over the *same* masked input: a **small** standalone encoder $E_\theta(w_s)$, a **large** standalone encoder $E_\theta(w_l)$, and a hybrid path that runs the first k layers at w_s and transitions to w_l at layer $k+1$ via $P_{w_s \to w_l}$. Each encoder feeds a predictor G_θ together with a learned mask token $\mathbf{m}$; the three paths produc $\mathcal{L}_{\text{small}}$, $\mathcal{L}_{\text{hybrid}}$, and $\mathcal{L}_{\text{large}}$, all regressed against targets from a frozen V-JEPA teacher T on the unmasked video.

The same principle applies to the feed-forward network. Given the full FFN parameters $W_1 \in \mathbb{R}^{r w_{\max} \times w_{\max}}$ and $W_2 \in \mathbb{R}^{w_{\max} \times r w_{\max}}$, with r being the upscaling factor, a width-w uses the slices:

$$W_1^{(w)} = W_1[1{:}rw, 1{:}w], \quad W_2^{(w)} = W_2[1{:}w, 1{:}rw]. \tag{4}$$

This nested parameterization allows all subnetworks to reuse a single set of weights while maintaining architectural compatibility across widths. The supported $\mathcal{W}$ form a discrete set due to the constraint that the hidden dimension must remain divisible by the attention head size.

For hybrid configuration, the architecture supports width transitions between layers. When compute transitions from width w_s to a larger width w_l at layer k, the intermediate representation $h_k^{(w_s)} \in \mathbb{R}^{w_s}$ is expanded to $\mathbb{R}^{w_l}$ through a learned linear projection P. The operator is itself elastic: we maintain a single matrix $P \in \mathbb{R}^{w_{\max} \times w_{\max}}$, and for any transition (w_s, w_l) we use the slice:

$$P_{w_s \to w_l} = P[1{:}w_l,\ 1{:}w_s] \in \mathbb{R}^{w_l \times w_s} \tag{5}$$

so that $h_k^{(w_l)} = P_{w_s \to w_l}(h^{(w_s)})$. This mirrors the slicing scheme of the transformer blocks where a single set of projection parameters is shared across all (w_s, w_l) pairs. The same P is further reused across all possible split layers $k \in \{2, \ldots, L-1\}$, making the operator transition-layer invariant.

3.3 Transition-Aware Triplet Training

Our architecture introduces *hybrid execution*, where computation transitions from width w_s to a larger width w_l at a split layer k[1]. This capability is orthogonal to standalone elasticity. Our objective is therefore to train a single model that performs well both at any standalone width $w \in \mathcal{W}$ and across hybrid configurations (w_s, w_l, k). We formalize this by extending the configuration space. Traditional elastic training optimizes standalone subnetworks sampled from the width set $\mathcal{W}$:

$$\min_\theta \ \mathbb{E}_{w \sim p(\mathcal{W})} \ \mathcal{L}\left(f_\theta^{(w)}\right). \tag{6}$$

In contrast, our formulation expands the configuration space to include hybrid executions,

$$\mathcal{C} = \mathcal{W} \cup \{(w_s, w_l, k) : w_s, w_l \in \mathcal{W},\ w_s < w_l,\ k \in \{2, \ldots, L-1\}\},$$

and optimizes configurations drawn from this joint space:

$$\min_\theta \ \mathbb{E}_{c \sim p(\mathcal{C})} \ \mathcal{L}\left(f_\theta^{(c)}\right). \tag{7}$$

[1]This is different from mix&match in Matformer[10] which uses fixed hidden dimensions in all blocks.

In this formulation, hybrid execution $f_\theta^{(w_s,w_l,k)}$ is treated as a first-class configuration rather than as an emergent property of training. This naturally allows existing sampling strategies to be extended to the hybrid setting. We consider two such extensions as baselines:

- **Distributional sampling (*Matformer-style*)** [10]: a configuration $c \sim p(\mathcal{C})$ is sampled at each iteration and is optimized independently.
- **Sandwich rule (*US-Net-style*)** [38]: the smallest and largest widths are optimized together with a random configuration $c_{\text{rand}} \sim p(\mathcal{C})$ per iteration. This provides strong supervision at the boundaries while intermediate widths compete for the same sampling slot.

As a more suitable strategy for the joint standalone and hybrid objective, we propose **Transition-Aware Triplet Training (T3)**. At each iteration, we sample:

$$w_s, w_l \sim \mathcal{W}, \quad w_s < w_l, \qquad k \sim \{2, \dots, L-1\},$$

and jointly optimize the triplet: $\mathcal{T}_{T3} = \{f_\theta^{(w_s)},\ f_\theta^{(w_s,w_l,k)},\ f_\theta^{(w_l)}\}$, which mirrors the three deployment modes: standalone small, hybrid execution, and standalone large. T3 differs from the baselines in one key respect: the same width pair (w_s, w_l) appears simultaneously as standalone and as the endpoints of a hybrid path within the same iteration. This couples the three configurations through **shared gradients**, directly optimizing both standalone performance and cross-width compatibility.

3.4 Training

We train the elastic model under a V-JEPA-style masked latent prediction objective [1; 2]. V-JEPA follows an encoder–predictor regime in which a student model observes a masked video and predicts latent representations for the masked regions from the visible context using the predictor and learned mask tokens $\mathbf{m}$. In our setting, we retain this predictive training structure but replace the original teacher mechanism with distillation from a frozen pretrained V-JEPA model. Concretely, the student receives a masked input video, predicts latent embeddings for the masked tokens, and is optimized to match target representations produced by the frozen teacher on the corresponding full video.

An input video v is split into spatio-temporal tokens $X = \{x_1, \dots, x_N\}$ and masked with the multi-block strategy of V-JEPA [2]: we sample spatially contiguous blocks and extend them across the full temporal extent of the clip, producing tubelet masks whose union defines the masked index set $\mathcal{M} \subset \{1, \dots, N\}$ and whose complement $\mathcal{V}$ indexes the visible tokens $X_\mathcal{V} = \{x_i\}_{i \in \mathcal{V}}$. These are passed to the student encoder E_θ, while the predictor G_θ predicts latent representations for $i \in \mathcal{M}$. The same $(X, \mathcal{M})$ pair is reused across all configurations in a T3 triplet, so standalone and hybrid paths face an identical predictive task.

Student. The student is our parametrized learnable model f_θ, a composition of our learnable modules including the encoder E_θ and the predictor G_θ, both elastic transformers according to the weight-sharing policy defined in Section 3.2. A sampled configuration $c \in \mathcal{C}$ determines how the encoder is executed. A standalone configuration is specified by a single width $w \in \mathcal{W}$. A hybrid configuration is specified by a tuple (w_s, w_l, k), where the first k encoder layers run at width w_s and the remaining $L - k$ layers run at width w_l. In both cases, we write $w_E(c)$ for the encoder output width, where $w_E(c) = w$ for standalone configurations and $w_E(c) = w_l$ for hybrid configurations. Then, the predictor's elastic width is derived from the $w_E(c)$ by the following rule:

$$w_G(c) = \min\left(w_G^{\max},\ \tfrac{1}{2}\, w_E(c)\right), \qquad w_G^{\max} = 384, \tag{8}$$

so the predictor tracks half the encoder width up to a cap $w_G^{\max}$. The cap $w_G^{\max} = 384$ matches the predictor width of the original V-JEPA [1; 2] while the half-width rule scales the predictor *below* this reference for smaller encoders, deliberately limiting predictor capacity so that the encoder is forced to carry the representational load. Once the widths are defined, for a sampled configuration $c \in \mathcal{C}$, the student prediction for a masked token $i \in M$ is:

$$z_{S,i}^{(c)} = \left[f_\theta^{(c)}(X_\mathcal{V})\right]_i = \left[G_\theta^{(w_G(c))}\left(W_{\text{proj}}^{(w_E(c)\to w_G(c))}\ \left(E_\theta^{(w_E(c))}(X_\mathcal{V})\right),\ \mathbf{m}\right)\right]_i, \tag{9}$$

where $W_{\text{proj}}^{(w_E(c)\to w_G(c))}$ is an elastic linear projection that maps the encoder output to $w_G(c)$, and $\mathbf{m} \in \mathbb{R}^{w_G(c)}$ is the learned mask token inserted at the masked positions $i \in M$. Both W_{proj} and $\mathbf{m}$ are sliced under the same weight-sharing policy as Section 3.2.

Teacher. Targets $z_{T,i} = T(X)_i$, $i \in \mathcal{M}$, are produced by a frozen, pretrained V-JEPA 2 ViT-G teacher [1], denoted T, on the full (unmasked) input. An EMA teacher is ill-suited to the elastic setting: since the student is sampled at a different configuration $c \in \mathcal{C}$ each iteration, an EMA target would average parameters across configurations of varying capacity, yielding an unstable, width-dependent signal. A frozen full-capacity teacher instead anchors every subnetwork to the same representation space. A lightweight learned head h_ϕ aligns student dimensions with the teacher; for notational simplicity we fold h_ϕ into $z^{(c)}_{S,i}$ in what follows.

Loss. Following V-JEPA, we use an ℓ_1 regression loss on the masked tokens:

$$\mathcal{L}(c) \;=\; \frac{1}{|M|} \sum_{i \in M} \left\| z^{(c)}_{S,i} - z_{T,i} \right\|_1 . \tag{10}$$

This objective is applied to all configurations in the triplet $\mathcal{T}_{T3} = \{f_\theta^{(w_s)}, f_\theta^{(w_s,w_l,k)}, f_\theta^{(w_l)}\}$, giving

$$\mathcal{L}_{\text{small}} = \mathcal{L}(w_s), \qquad \mathcal{L}_{\text{hybrid}} = \mathcal{L}((w_s, w_l, k)), \qquad \mathcal{L}_{\text{large}} = \mathcal{L}(w_l), \tag{11}$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{small}} + \mathcal{L}_{\text{hybrid}} + \mathcal{L}_{\text{large}}. \tag{12}$$

Since T is frozen, the configurations are regressed toward a single consistent target, stabilizing elastic training and divergence between subnetworks observed under distributional sampling (Section 3.3).

4 Experiments

Data & teacher. We pretrain all elastic backbones using the objective described in Sec. 3.4. The pretraining set is the union of unlabeled videos proposed in [2], including SSv2 [17] Kinetics-710 (K710) [23], and HowTo100M [28]. Each input clip has 16 frames sampled at 4 FPS and resized to 256×256. Videos are tokenized with $16 \times 16 \times 2$ tubelets, so each token spans over two consecutive frames. Masking follows the long-range spatiotemporal strategy of V-JEPA [2]: we sample two spatial block masks with scale up to 70% and extend them across the full temporal duration of the clip. We use a frozen V-JEPA 2 ViT-G as the teacher.

Model configurations. All elastic models instantiate the architecture of Sec. 3.2 with attention-head dim of 64 and an MLP expansion ratio of $4\times$. We train three families, EV-JEPA-B/L/H, whose width sets, depths, and parameter ranges are given in Table 1. LayerNorm parameters are shared across widths. Hybrid configurations use the shared elastic projection P, sliced per (w_s, w_l). Unless otherwise noted, hybrid execution splits at the middle layer ($k=6$ for the 12-layer models and $k=7$ for the 14-layer EV-JEPA-H) and follows our default many-to-one setting, in which only transitions into $w_{\max}$ are supported; Table 1 reports the resulting configuration counts $|\mathcal{C}|$.

Training strategies. We compare four training strategies. The *standalone* baseline trains fixed-width model for each width for the same number of epochs and with the same per-model compute capacity as the elastic ones. The elastic EV-JEPA variants, *MatFormer-style*, *US-Net-style*, and our proposed *T3*, share the same architecture, teacher, data, masking, optimizer, and configuration set. They differ only in how configurations are selected during pretraining, as formalized in Sec. 3.3.

Evaluation. Following the video understanding protocol of [1], we evaluate classification on SSv2 and K400 with an attentive probe on the pretrained encoder, and action anticipation on EK100, followed by ablations. Appendices A.2- A.4 add a comparison against SOTA models and show the adaption of EV-JEPA on vision-language transfer under VL-JEPA, and the application of T3 to CNNs.

4.1 Elastic Training and Range Scaling on SSv2

We begin with SSv2, whose temporal sensitivity demands fine-grained spatial-temporal reasoning, making it a rigorous testbed for elastic representations. Three questions guide our analysis. **1)** Under

Table 1: **Model Setup.** In default settings, every smaller width transitions to the largest width in its range, i.e., $w \to w_{\max},\ \forall\, w \in \mathcal{W} \backslash \{w_{\max}\}$. The number of configurations is $|\mathcal{C}| = |\mathcal{W}| + (|\mathcal{W}| - 1) \cdot |K|$, where $|K|$ is the number of candidate split layers. Reported values correspond to $K=1$.

Model	$\mathcal{W}$	Hybrid	L	Params Range (M)	$\lvert\mathcal{C}\rvert$
EV-JEPA-B	$\mathcal{W}_1 : \{192, 384, 768, 1024\}$	$w \to 1024$	12	5.6–152.6	7
EV-JEPA-L	$\mathcal{W}_2 : \{192, 384, 768, 1024, 1280\}$	$w \to 1280$	12	5.6–238.0	9
EV-JEPA-H	$\mathcal{W}_3 : \{192, 384, 768, 1024, 1280, 1408\}$	$w \to 1408$	14	6.5–335.5	11

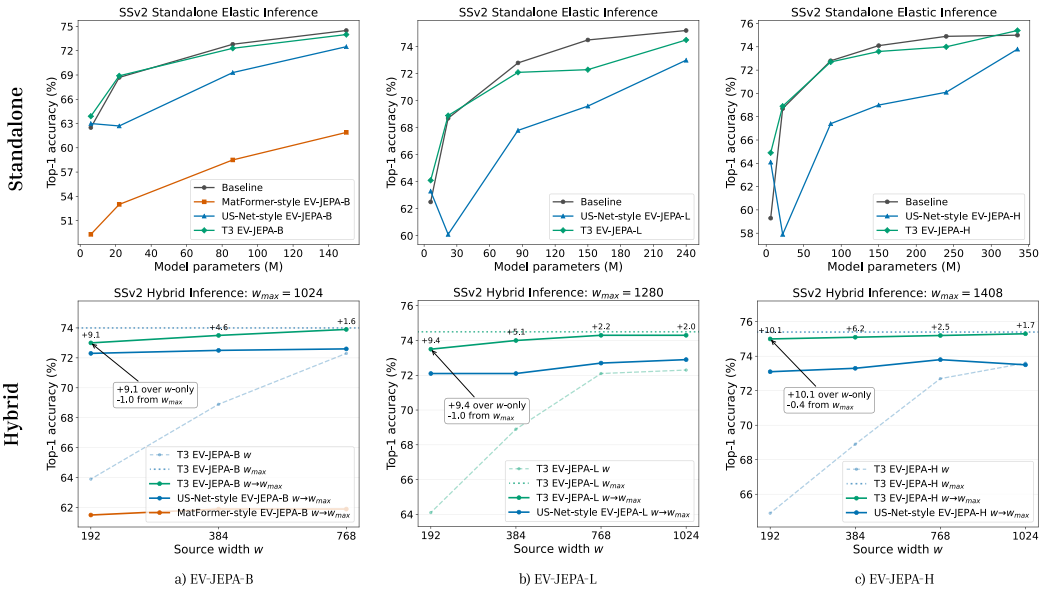


Figure 3: **SSv2 standalone and hybrid inference across increasing elastic ranges.** Columns correspond to a) EV-JEPA-B ($\mathcal{W}_1$), b) EV-JEPA-L ($\mathcal{W}_2$), and c) EV-JEPA-H ($\mathcal{W}_3$). The top row reports standalone elastic inference as a function of model parameters. The bottom row reports many-to-one hybrid execution $w \to w_{\max}$, where the first half of the encoder runs at source width w, and the remaining layers run at w_{max}. In the bottom row, *T3 EV-JEPA* w and *T3 EV-JEPA* $w_{\max}$ are shown as w-only and full-width T3 references, respectively, to measure hybrid recovery.

a controlled setting sharing all the configurations, how does T3 compare to MatFormer- and US-Net-style training? **2)** Does it hold as the elasticity scales from EV-JEPA-B ($\mathcal{W}_1$)to EV-JEPA-H ($\mathcal{W}_3$)? **3)** Does the hybrid inference help smaller models recover performance when transitioning to w_{max}?

Standalone. Figure 3(a) **Standalone** row reports the controlled comparison on EV-JEPA-B (widths $\mathcal{W}_1$) and addresses our first question. Notably, MatFormer-style training trails the other regimes by a wide margin, suggesting that sampling a single configuration and having isolated parameter updates in each step produces unstable performance across configurations. US-Net-style sandwich training mitigates this by supervising the boundary widths at every iteration, ensuring that all shared parameters receive a gradient signal at each step. However, it comes at the cost of configurations between the two endpoints degrading. T3 avoids both failure modes. By coupling a smaller path (f^{w_s}), a larger path (f^{w_l})[2], and its hybrid transition ($f^{(w_s,w_l,k)}$) within the same iteration, every parameter is supervised at every step, and intermediate widths are explicitly drawn into the gradient. As a result, T3 tracks the independently trained baseline references most closely across the range. Having established US-Net as the strongest competing elastic baseline, we use it as the main benchmark in Figures 3 b) & c), which extend the elastic range up to $\mathcal{W}_3$ in EV-JEPA-H to address our second question. The trend is clear, T3 tracks the references throughout, whereas US-Net progressively worsens at intermediate widths, with the gap reaching up to 11 accuracy points in EV-JEPA-H.

Hybrid. Figure 3 **Hybrid** addresses our third question. Hybrid accuracy is upper-bounded by the destination width $w_{\max}$ and we question whether a smaller model can be compatible with a high-capacity continuation to recover the performance gap between w_s (source width) and w_{max}. We plot two T3 references alongside the hybrid curve: the standalone w-only execution as a lower bound and the standalone $w_{\max}$ execution as an upper bound. The vertical distance from the w-only curve quantifies the accuracy gained by continuing computation at $w_{\max}$, while the distance from the hybrid curve to the $w_{\max}$ reference captures the residual loss relative to the standalone w_{max} model. Across all three ranges, T3 hybrid execution recovers nearly w_{max}-level accuracy even with source widths as small as $w = 192$, falling within 1 point of the standalone w_{max} model.

[2]Note, for this specific experiment settings, $w_l = w_{max}$.

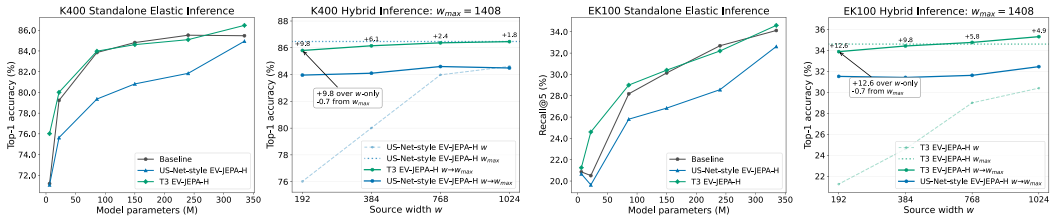


Figure 4: **Experiments on K400 and EK100.** *Standalone* panels (1st, 3rd) report elastic inference as a function of model parameters. *Hybrid* panels (2nd, 4th) report hybrid execution $w \rightarrow w_{\max}$.

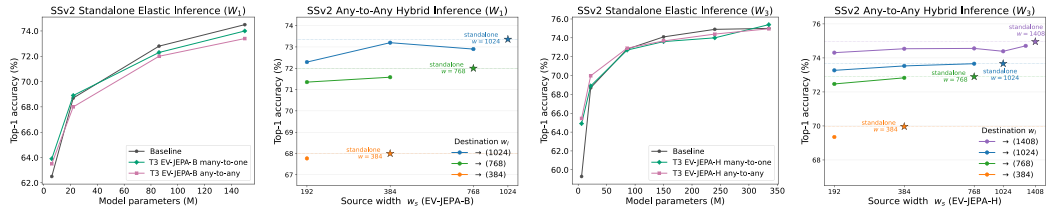


Figure 5: **SSv2 any-to-any.** A any-to-any model can upscale from any source w_s to *any* larger width w_l. On the ***Hybrid Inference*** plots, each curve fixes a destination w_l and sweeps the w_s (dotted lines). The star mark represents the standalone accuracy of that destination width (matching colors).

4.2 Evaluation on Recognition (K400) and Action Anticipation (EK100)

We test whether similar conclusions transfer to K400 (classification) and EK100 (*action anticipation*). For anticipation, we use our frozen pretrained elastic predictor (following Equation 8) for evaluation similar to [1]. We retain US-Net as the strongest competing elastic baseline and report all results on the full-range EV-JEPA-H model ($\mathcal{W}_3$, $w_{\max} = 1408$). K400 is evaluated with Top-1 accuracy and EK100 with mean Top-5 recall (recall@5) [15; 11] reported in Figure 4. The standalone panels of Figure 4 (1st, 3rd) show that on both K400 and EK100, T3 tracks the independently trained references across the full width range, whereas US-Net degrades at intermediate widths, with the gap reaching up to 5 accuracy points on K400 and 5 recall@5 points on EK100. The hybrid panels (2nd, 4th) report many-to-one execution ($w \rightarrow w_{\max}$), with the T3 w-only and $w_{\max}$ standalone curves plotted as lower and upper bounds. On both benchmarks, continuing a small-width encoder at $w_{\max}$ recovers most of the gap to the standalone $w_{\max}$ model. Even at source width $w = 192$, hybrid execution gains +9.8 Top-1 points on K400 and +12.6 recall@5 points on EK100, falling within 1 point(s) of the full-width reference, while consistently outperforming US-Net-style hybrid execution. Cross-width compatibility, therefore, holds for appearance-biased recognition and for egocentric anticipation alike, supporting T3 as a general training regime for distributed inference.

4.3 Any-to-Any Hybrid Execution

The hybrid experiments so far adopt the many-to-one setting ($w \rightarrow w_{\max}$). We now test the more general claim that computation can transition from *any* source width to *any* larger destination width. Because the any-to-any configuration set is more expensive to train, we conduct this study on the smallest and largest ranges, $\mathcal{W}_1 = \{192, 384, 768, 1024\}$ and $\mathcal{W}_3 = \{192, 384, 768, 1024, 1280, 1408\}$, with a single mid-network split ($k = 6$ for the 12-layer EV-JEPA-B and $k = 7$ for the 14-layer EV-JEPA-H). Figure 5 reports SSv2 Top-1 for every valid source–destination pair (w_s, w_l) with $w_s < w_l$. As in the many-to-one setting, all pairs are served by the single shared projection $P_{w_s \rightarrow w_l}$. Two observations emerge. First, the any-to-any T3 model retains standalone quality, tracking both the independently trained baselines and the many-to-one T3 (Fig. 5, 1st and 3rd), giving up <1 point relative to many-to-one as the price of supporting the larger configuration space. Second, hybrid accuracy (Fig. 5, 2nd and 4th) is governed primarily by the destination width w_l. For $\mathcal{W}_1$ ($w_{\max}$=1024), all sources fall within 0.9 points of one another regardless of w_l and within 1.1 points of the standalone of the same w_l (star marked), including the smallest source w_s=192; for $\mathcal{W}_3$

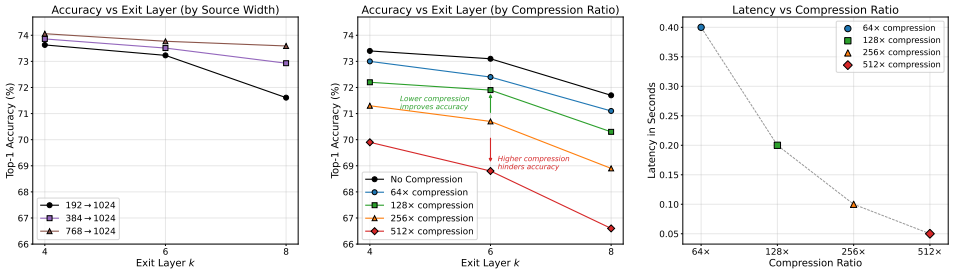


Figure 6: **Variable *k* and elastic token compression. (left)** SSv2 Top-1 for hybrid execution as a function of the exit layer k, for source widths $w_{\in}\{192, 384, 768\}$ and target width $w_l = 1024$. **(middle)** SSv2 Top-1 versus k for the single path $192 \rightarrow 1024$ under different elastic compression ratios. **(right)** Transmission latency at each compression ratio over a theoretical 1 Mbps link, computed as the transmitted payload divided by link bandwidth.

($w_{\max}$=1408) the corresponding spreads shrink to 0.3 and 0.65 points. A small originating device can therefore hand off to any larger continuation device and recover near-destination accuracy.

4.4 Variable Exit Points and Elastic Token Compression

Variable splitting layer k**.** Supporting multiple exit points k adds no parameters and training time. We therefore run a variable-k experiment on EV-JEPA-B with $k \in \{4, 6, 8\}$. Figure 6 (*left*) reports SSv2 Top-1 versus k for the three hybrid paths $w \rightarrow 1024$, $w \in \{192, 384, 768\}$. Two effects are visible. First, accuracy decreases monotonically with k for every source width: an earlier handoff leaves more layers at w_l and recovers more accuracy. Second, the cost of a late exit is governed by the source width. At $k=4$ the three paths lie within 0.4 points of one another, so even $w=192$ nearly matches a $4\times$ wider source; by $k=8$ they fan out and $w=192$ falls 2.0 points below $w=768$. Source capacity and on-device depth are therefore substitutable, exposing k as a deployment knob.

Elastic token compression. Hybrid execution must transmit representations to the continuation device, and on the low-power links typical of wearable sensors this transfer can dominate latency. We therefore attach a learned compressor at the exit point: a single encoder–decoder 3D convolution that reduces the number of tokens, made elastic by slicing the convolution kernel along its spatiotemporal extent, so one set of kernel weights spans every compression ratio (Fig. 7; architecture and compression details in Appendix A.5). The compressor is trained in a post-training stage with all other components frozen using the loss in Equation 15 (Appendix A.5). Figure 6 (*middle*) sweeps k for the $192 \rightarrow 1024$ path at $64\times$–$512\times$ compression. The ordering in k is preserved at every ratio, so exit point and compression ratio act as independent knobs. Accuracy falls monotonically with the ratio, tracking the uncompressed path within 0.7 points at $64\times$ and within 5.1 points at $512\times$. Over a theoretical 1 Mbps link, going from $64\times$ to $512\times$ buys up to roughly $8\times$ latency reduction (Fig. 6, *right*). All accuracies are end-to-end, so the two knobs expose a controllable trade-off among on-device computation, transmission latency, and accuracy.

5 Conclusions

We propose EV-JEPA, a model that is width-elastic and natively partitionable across heterogeneous devices. Its enabler is T3, which at every step optimizes a width pair both as two standalone and as the endpoints of a hybrid path connecting them under a dense predictive objective for the first time. Across SSv2, K400, and EK100 anticipation, the model stays within 1 point of independently standalone baselines while improving over prior elastic training by up to 11 points at intermediate widths. Hybrid mode allows cross-width execution, recovering to within 1.1 points of the higher-capacity model under variable-split settings, and at full width the same weights match or exceed fixed-capacity baseline encoders. The elastic principle further extends beyond the backbone, to a token compressor that cuts transmitted payload by up to 512×. Three limitations remain: 1) training requires a frozen pretrained V-JEPA teacher, so training without an external teacher is unexplored; 2) elasticity is confined to width, leaving depth L fixed and extension non-trivial; 3) T3 operates over a fixed width set whose size drives training cost, so broader device coverage means longer training.

REFERENCES

[1] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[2] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-jepa: latent video prediction for visual representation learning. In *ICLR*, 2024.

[3] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Bangalath, et al. Perception encoder: The best visual embeddings are not at the output of the network. *NeurIPS*, 38:60884–60937, 2026.

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020.

[5] Yun-Hao Cao, Peiqin Sun, and Shuchang Zhou. Three guidelines you should know for universally slimmable self-supervised learning. In *CVPR*, 2023.

[6] Delong Chen, Mustafa Shukor, Theo Moutakanni, Willy Chung, Jade Yu, Tejaswi Kasarla, Yejin Bang, Allen Bolourchi, Yann LeCun, and Pascale Fung. Vl-jepa: Joint embedding predictive architecture for vision-language. *arXiv preprint arXiv:2512.10942*, 2025.

[7] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *ICCV*, 2021.

[8] Minghao Chen, Kan Wu, Bolin Ni, Houwen Peng, Bei Liu, Jianlong Fu, Hongyang Chao, and Haibin Ling. Searching the search space of vision transformer. In *NeurIPS*, 2021.

[9] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *IJCV*, 2022.

[10] Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference. In *NeurIPS*, 2024.

[11] Anxhelo Diko, Danilo Avola, Bardh Prenkaj, Federico Fontana, and Luigi Cinque. Semantically guided representation learning for action anticipation. In *ECCV*, pp. 448–466. Springer, 2024.

[12] Xin Dong, Barbara De Salvo, Meng Li, Chiao Liu, Zhongnan Qu, Hsiang-Tsung Kung, and Ziyun Li. Splitnets: Designing neural architectures for efficient distributed computing on head-mounted systems. In *CVPR*, pp. 12559–12569, 2022.

[13] Schahram Dustdar, Victor Casamayor Pujol, and Praveen Kumar Donta. On distributed computing continuum systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(4): 4092–4105, 2022.

[14] Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew Turner, Arjang Talattof, Arnie Yuan, Bilal Souti, Brighid Meredith, et al. Project aria: A new tool for egocentric multi-modal ai research. *arXiv preprint arXiv:2308.13561*, 2023.

[15] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE TPAMI*, 43(11):4021–4036, 2020.

[16] Chengyue Gong, Dilin Wang, Meng Li, Xinlei Chen, Zhicheng Yan, Yuandong Tian, Qiang Liu, and Vikas Chandra. Nasvit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training. In *ICLR*, 2022.

[17] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[18] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pp. 544–560, 2020. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123610528.pdf.

[19] Janek Haberer, Ali Hojjat, and Olaf Landsiedel. Hydravit: Stacking heads for a scalable vit. In *NeurIPS*, 2024.

[20] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. In *NeurIPS*, 2020.

[21] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[22] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 615–629, 2017.

[23] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[24] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *NeurIPS*, 2022.

[25] Stefanos Laskaridis, Stylianos I. Venieris, Mário Almeida, Ilias Leontiadis, and Nicholas D. Lane. Spinn: Synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 1–14, 2020.

[26] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. *URL: https://arxiv. org/abs/2603.19312*, 2026.

[27] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. In *ICLR*, 2021.

[28] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *CVPR*, pp. 2630–2640, 2019.

[29] Himangi Mittal, Nakul Agarwal, Shao-Yuan Lo, and Kwonjoon Lee. Can't make an omelette without breaking some eggs: Plausible action anticipation using large video-language models. In *CVPR*, pp. 18580–18590, 2024.

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[31] Debaditya Roy, Ramanathan Rajendiran, and Basura Fernando. Interaction region visual transformer for egocentric action anticipation. In *CVPR*, pp. 6740–6750, 2024.

[32] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. In *Int. Conf. on Robot. and Automation*, pp. 7226–7233. IEEE, 2023.

[33] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pp. 6105–6114. PMLR, 2019.

[34] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[35] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *CVPR*, pp. 6418–6427, 2021.

[36] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *CVPR*, pp. 14549–14560. IEEE, 2023.

[37] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, et al. Internvideo2: Scaling foundation models for multimodal video understanding. In *ECCV*, pp. 396–416. Springer, 2024.

[38] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019.

[39] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. In *ICLR*, 2019.

[40] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc V. Le. Bignas: Scaling up neural architecture search with big single-stage models. In *ECCV*, 2020.

[41] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. In *Empirical Methods in Natural Lang. Process.*, pp. 543–553, 2023.

[42] Yitian Zhang, Huseyin Coskun, Xu Ma, Huan Wang, Ke Ma, Xi Chen, Derek Hao Hu, and Yun Fu. Slicing vision transformer for flexible inference. In *NeurIPS*, 2024.

[43] Long Zhao, Nitesh B Gundavarapu, Liangzhe Yuan, Hao Zhou, Shen Yan, Jennifer J Sun, Luke Friedman, Rui Qian, Tobias Weyand, Yue Zhao, et al. Videoprism: A foundational visual encoder for video understanding. *arXiv preprint arXiv:2402.13217*, 2024.

[44] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

A Appendix

A.1 Optimization.

All EV-JEPA models are pretrained with 300 epochs and AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$) in `bf16` precision on 128 GPUs (16 nodes $\times$ 8), with a constant weight decay of 0.04. The learning rate follows a three-phase schedule: a linear warmup from 1×10^{-4} to the peak, a constant phase at the peak, and a half-period cosine anneal toward a final learning rate of 5×10^{-5}. Following V-JEPA, the cosine period is stretched by a factor of 1.25 relative to the training length, so training stops before the anneal completes and the final learning rate remains above. Schedule lengths and peak learning rates differ by student family. See Table 2 for the detailed training configurations for each student family. Inputs are 16-frame clips at 256×256 sampled at 4 fps, drawn from a mixture of Kinetics-710, Something-Something v2 and HowTo100M with sampling weights $0.335/0.100/0.565$.

A.2 Comparison With SOTA

Table 3 compares EV-JEPA-H at full width (w=1408) against state-of-the-art image and video foundation models on SSv2 (motion) and K400 (appearance), all evaluated under the same attentive-probe protocol. Because EV-JEPA-H is distilled from a frozen V-JEPA 2 ViT-G teacher, this is not a like-for-like comparison against models trained from scratch, and we therefore include a *Non-Elastic*

Student	Layers	w_{max}	Warm/Const/Decay	Peak LR	Clip	Batch/GPU
EV-JEPA-B/L	12	1024/1280	40 / 200 / 60	5.25×10^{-4}	0.025	24
EV-JEPA-H	14	1408	40 / 200 / 60	5.25×10^{-4}	0.025	24
CNN	20	384	20 / 80 / 200	3×10^{-3}	0.025	12

Table 2: Optimization schedules by student family. All runs decay to a final learning rate of 5×10^{-5}.

Table 3: Comparison with state-of-the-art models for action recognition on SSv2 and K400. **Bold** marks the best result per column, underline the second best. Avg. is the mean of SSv2 and K400.

		Motion	*Appearance*	
Method	Param.	SSv2	K400	Avg.
Image encoders Evaluated Using Same Protocol				
DINOv2 [30]	1.1B	50.7	83.6	67.2
$PE_{core}G$ [3]	1.9B	55.4	<u>88.5</u>	72.0
SigLIP2 [34]	1.2B	49.9	87.3	68.6
Video encoders Evaluated Using Same Protocol				
VideoMAEv2 [36]	1.0B	56.1	82.8	69.5
InternVideo2-1B [37]	1.0B	67.3	87.9	77.6
VideoPrism [43]	1.0B	68.5	87.6	78.0
$InternVideo2_{s2}$-1B [37]	1.0B	69.7	**89.4**	79.6
V-JEPA 2 ViT-L [1]	300M	73.7	85.1	79.4
V-JEPA 2 ViT-H [1]	600M	74.0	85.3	<u>79.7</u>
V-JEPA 2 ViT-G [1]	1.0B	<u>75.3</u>	86.6	**81.0**
Distilled from V-JEPA 2 ViT-G				
Non-Elastic Baseline (w=1408)	335M	75.0	85.5	80.3
EV-JEPA-H (w=1408)	335M	**75.4**	86.5	**81.0**

Baseline distilled under identical teacher, data, masking, and compute budget. This control isolates the accuracy cost of elasticity from the effect of distillation, which is a quantity of interest.

Read this way, three results stand out. First, elasticity is obtained at no measurable accuracy cost. EV-JEPA-H matches the *Non-Elastic* Baseline across the board (75.4 vs. 75.0 on SSv2, 86.5 vs. 85.5 on K400, 81.0 vs. 80.3 average), despite the same weights additionally serving every smaller width and every cross-device hybrid configuration. We do not read the small margin in our favour as evidence that elasticity *improves* full-width accuracy, though it is consistent with the multi-configuration objective of T3 acting as a regulariser. Second, this operating point is competitive with fixed-capacity specialists: EV-JEPA-H attains the best SSv2 accuracy (75.4), edging its own ViT-G teacher (75.3, 1.0B) at roughly $3\times$ fewer parameters, and far surpassing appearance-biased image encoders such as $PEcore_G$ (55.4) and DINOv2 (50.7) on this motion-centric benchmark. On the SSv2/K400 average it reaches 81.0, matching V-JEPA 2 ViT-G while exceeding ViT-L (79.4, 300M) and ViT-H (79.7, 600M). Third, on the appearance-dominated K400, EV-JEPA-H (86.5) falls within 0.1 points of ViT-G and is competitive with the strongest video encoders, though encoders trained on larger labelled corpora (e.g. $InternVideo2_{s2}$, 89.4) remain ahead, consistent with the motion-versus-appearance trade-off inherent to the V-JEPA objective.

On action anticipation (EK100, Table 4), EV-JEPA-H at full width reaches 34.6 action recall@5, again matching the *Non-Elastic* Baseline distilled under identical conditions (34.1). Elasticity is therefore free on anticipation as well as on recognition. This also clarifies how to read the remaining gap to the frozen V-JEPA 2 ViT-H backbone (36.5): it is a capacity effect rather than a cost of our training regime, since a fixed-width model at the same 335M parameters and the same teacher lands below EV-JEPA-H. Relative to the broader literature, EV-JEPA-H outperforms recent 7–8B video–language models (PlausiVL, 27.6, and Video-LLaMA, 26.0) by roughly seven points at a fraction of their size, and sits between the frozen ViT-L (32.7) and ViT-H (36.5) backbones at only 335M parameters. Crucially, these numbers are the full-width operating point of a *single elastic* model, not a separately trained specialist. The same weights that make EV-JEPA-H competitive with state of the art at w=1408 and comparabale parameter count, also serve every smaller width and cross-device hybrid configuration. EV-JEPA thus competes with similar fixed-capacity state-of-the-art models at its high

Table 4: Action anticipation on EK100. **Bold** marks the best result per column. Frozen-backbone rows attach an anticipation head to a frozen encoder.

		Action anticipation (EK100)		
Method	Param.	Verb	Noun	Action
InAViT [31]	160M	51.9	52.0	25.8
Video-LLaMA [41]	7.0B	52.9	52.0	26.0
PlausiVL [29]	8.0B	55.6	54.2	27.6
Frozen backbone				
V-JEPA 2 ViT-L [1]	300M	57.8	53.8	32.7
V-JEPA 2 ViT-H [1]	600M	59.2	54.6	36.5
V-JEPA 2 ViT-G [1]	1.0B	**61.2**	**55.7**	**38.0**
Frozen backbone Distilled from V-JEPA 2 ViT-G				
Non-Elastic Baseline (w=1408)	335M	58.0	53.2	34.1
EV-JEPA-H (w=1408)	335M	58.2	53.2	34.6

Table 5: Vision–language transfer on SSv2 (Top-1, open-vocabulary classification) under the VL-JEPA framework. All elastic rows use a single frozen EV-JEPA-H encoder.

Configuration	Param.	SSv2
V-JEPA 2 ViT-L [6]	300M	73.2
Standalone		
w=1408	335M	**73.6**
w=1024	179M	72.2
Hybrid		
384→1408	203M	72.9
192→1408	194M	72.6

Table 6: Elasticity beyond Transformers. SSv2 Top-1 for a slimmable factorized 3D-CNN with a lightweight linear-attention layer. A single nested model serves all standalone widths and their cross-width hybrids.

Configuration	Param.	GFLOPs	SSv2
Standalone			
w=128	2.3M	11	56.0
w=192	4.1M	20	58.6
w=256	6.4M	30	60.7
w=384	12.2M	57	**63.5**
Hybrid			
128→384	11.7M	50	63.2
192→384	11.8M	51	63.2
256→384	12.0M	53	63.4

end while additionally offering the reconfigurable compute and collaborative execution that none of these baselines provide.

A.3 Vision-Language Transfer with VL-JEPA

Sections 4.1 and 4.2 probe the visual representation directly through task-specific heads. We now ask whether the same elastic encoder also transfers to vision-language understanding, where the representation must align with language rather than a discriminative head. We adopt VL-JEPA [6], a joint-embedding vision-language model that predicts continuous text embeddings, and replace its frozen V-JEPA 2 ViT-L vision encoder with our frozen EV-JEPA-H encoder, training only the connector and predictor. Following VL-JEPA, we report Top-1 accuracy on SSv2 open-vocabulary classification.

As shown in Table 5, in standalone inference, the full-width configuration (w=1408) matches and slightly exceeds the VL-JEPA ViT-L baseline, while reducing to w=1024 stays within one point at only 179M parameters, about 60% of the ViT-L encoder. In hybrid cross-device execution, where the early layers run at a small source width and the remaining layers continue at $w_{\max}$ so that only compact intermediate features are transmitted, the 384→1408 and 192→1408 configurations reach within one point of the baseline while activating only 194M–203M parameters. A single frozen EV-JEPA encoder therefore extends both its standalone width-elasticity and its cross-device hybrid capability from recognition and anticipation to vision-language tasks, confirming that elasticity and language alignment compose without a per-budget encoder.

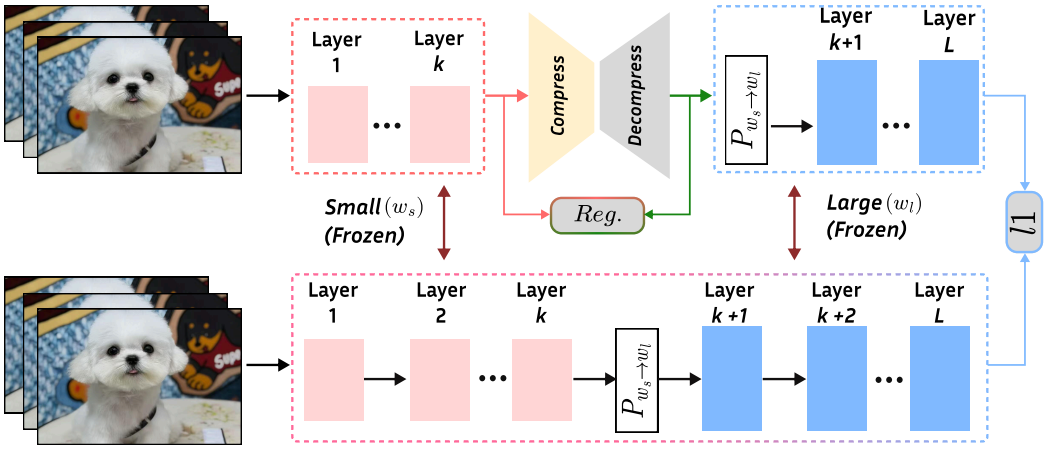


Figure 7: **Elastic token compression for hybrid execution.** A learned compressor is inserted at the exit point k of the source device. The 'compress module' compresses the intermediate representation before transmission, reducing the number of tokens; the 'decompress module' reconstructs the token grid on the target device, after which the shared elastic projection $P_{w_s \to w_l}$ upscales to w_l, and the remaining $L - k$ layers execute. During training, all modules except the compress–decompress pair remain frozen.

A.4 Elasticity Beyond Transformers: Convolutional Backbones

The transition-aware training behind EV-JEPA is not specific to the Transformer. We distill the ViT-G teacher into a slimmable factorized 3D-CNN, augmented with a single lightweight linear-attention layer for global context. The CNN channel widths are sliced under the same weight-sharing policy (Section 3.2), so smaller subnets are strict channel slices of the largest and one set of weights serves every width. We use channels-first LayerNorm so that a single set of statistics-free normalizers stays valid across widths. Trained with the same T3 regime (Section 3.3), a single nested CNN scales monotonically on SSv2 (Table 6): from 56.0 at w=128 (2.3M parameters, 11 GFLOPs), to 63.5 at w=384 (12.2M, 57 GFLOPs). This extends EV-JEPA's elasticity to the low-cost, low-memory regime most relevant to always-on, on-device deployment, confirming that transition-aware triplet training yields a single reconfigurable, partitionable model for convolutional backbones as well.

A.5 Elastic Token Compression

Hybrid execution assumes the transmission of an intermediate representation from the source to the continuation device, and on the low-power links typical of wearable sensors this transfer imposes significant latency. To reduce it, we attach a learned compressor at the exit of the source device, realized as a single encoder-decoder 3D convolution. The encoder runs as one additional layer and the decoder reconstructs the representation on the destination device before the remaining $L - k$ layers execute. The compressor reduces the number of transmitted tokens. Following the nested weight-sharing scheme of Section 3.2, compression is made elastic by slicing the convolution kernel along its spatiotemporal extent,

$$K^{(\kappa)} = K\big[\ :,\ :,\ 1{:}\kappa_t,\ 1{:}\kappa_h,\ 1{:}\kappa_w\ \big], \tag{13}$$

where the input and output channel dimensions are fixed at $C_{\text{in}} = C_{\text{out}} = w_s$, and the sliced extent $(\kappa_t, \kappa_h, \kappa_w)$ sets the spatiotemporal downsampling factor and hence the number of retained tokens N'. We distinguish two ratios throughout. The *token-reduction factor* $\tau = N/N'$ is the quantity the sliced kernel controls directly, and indexes the configurations of Table 7. The *compression ratio* ρ (Eq. 16) is the transmitted payload measured against that of the raw RGB clip, and is the quantity reported in Sec. 4.4 and Fig. 6. A single set of kernel weights thus spans the entire range of compression ratios, so one on-device layer can adjust compression elastically at inference time in the same way the backbone adjusts width.

Table 7: Per-ratio configurations of the elastic encoder–decoder for an input grid of $8\times16\times16 = 2048$ tokens. Kernel, stride, and padding are given as (time, height, width). $\tau = N/N'$ is the token-reduction factor set by the sliced kernel (Eq. 13), and ρ is the resulting payload compression ratio relative to raw RGB (Eq. 16), which is the quantity reported in Fig. 6.

τ	Kernel	Stride	Padding	Latent grid ($T'\times H'\times W'$) / tokens	ρ
$8\times$	$(4,4,4)$	$(2,2,2)$	$(1,1,1)$	$4\times8\times8$ / 256	$64\times$
$16\times$	$(6,4,4)$	$(4,2,2)$	$(1,1,1)$	$2\times8\times8$ / 128	$128\times$
$32\times$	$(4,6,6)$	$(2,4,4)$	$(1,1,1)$	$4\times4\times4$ / 64	$256\times$
$64\times$	$(6,6,6)$	$(4,4,4)$	$(1,1,1)$	$2\times4\times4$ / 32	$512\times$

Compressor configuration. We instantiate the compressor on EV-JEPA-B ($\mathcal{W}_1 = \{192, 384, 768, 1024\}$, $L = 12$) for the hybrid configuration $f_\theta^{(w_s,w_l,k)}$ with source width $w_s = 192$, destination width $w_l = w_{\max} = 1024$ at a given split-layer k, so the compressor sits exactly where the source device hands off to the continuation device. Up to layer k the encoder E_θ runs at w_s, so the quantity to be transmitted is the intermediate representation $h_k^{(w_s)} \in \mathbb{R}^{N\times w_s}$. For a 16-frame 256×256 clip tokenized with $16\times16\times2$ tubelets, this is $N = T\times H\times W = 8\times16\times16 = 2048$ tokens of width $w_s = 192$. The *compress* module reshapes these N tokens back into a $w_s\times T\times H\times W$ volume and applies a single strided 3D convolution with the sliced kernel $K^{(\kappa)}$ of Eq. 13, followed by GroupNorm (8 groups) and GELU. The sliced extent $(\kappa_t, \kappa_h, \kappa_w)$, together with the matching stride $(\sigma_t, \sigma_h, \sigma_w)$ and per-axis padding $p = \lfloor(\kappa - \sigma)/2\rfloor$, fixes the retained grid $N' = T'\times H'\times W'$ and hence the token-reduction factor $\tau = N/N'$, while the channel dimension/width is kept to $C_{\text{out}} = w_s = 192$. The N' tokens of width w_s are what crosses the link, giving the payload P_{comp} of Eq. 16. The *decompress* module mirrors this on the continuation device: a $3\times3\times3$ convolution with GroupNorm and GELU, a strided 3D transposed convolution with GroupNorm and GELU that restores the original $T\times H\times W$ grid at width w_s, and a LayerNorm over the channel dimension. The reconstructed tokens $\hat{h}_k^{(w_s)}$ then enter the same shared elastic projection $P_{w_s\to w_l} = P[1{:}w_l,\ 1{:}w_s]$ of Sec. 3.2 (here $P_{192\to1024}$), after which the remaining $L - k$ layers execute at w_l, exactly as in uncompressed hybrid execution (Fig. 7). Elasticity follows the same nested weight-sharing policy as the backbone: the strided convolution and the transposed convolution each store a single $6\times6\times6$ kernel K, and every token-reduction factor τ selects the leading sub-kernel $K^{(\kappa(\tau))}$ via Eq. 13. One set of kernel weights therefore spans the whole range, so τ becomes an inference-time knob in the same sense as the standalone width w, the hybrid pair (w_s, w_l), and the split layer k. Table 7 lists (κ, σ, p), the retained grid N', and the corresponding payload ratio $\rho = 8\tau$ for each configuration.

Compressor training. The compressor is trained in a second, post-training stage on top of a T3-pretrained EV-JEPA-B (stage 1), with every other module frozen, so the elastic encoder behaves exactly as during pretraining and the reported numbers isolate the cost of compression alone. Writing θ for the frozen pretrained parameters, the reference path is the uncompressed hybrid configuration $f_\theta^{(w_s,w_l,k)}$ and the trained path is the same configuration with the compress–decompress pair inserted at layer k, denoted $f_\theta^{(w_s,w_l,k,\tau)}$ at token-reduction factor τ; only the compressor parameters receive gradients. Two things differ from Sec. 3.4. First, this stage operates on unmasked clips: there is no masked index set $\mathcal{M}$, no mask token $\mathbf{m}$, and no predictor G_θ, so supervision is applied at the encoder output over *all* N tokens rather than over $\mathcal{M}$ alone. Second, the objective is the ℓ_1 regression of Eq. 10 *regularized* by a reconstruction term at the split layer, which anchors the transmitted representation itself rather than only its downstream effect. Concretely, let $h_k^{(w_s)} \in \mathbb{R}^{N\times w_s}$ be the source tokens at split layer k and $\hat{h}_{k\tau}^{(w_s)} \in \mathbb{R}^{N\times w_s}$ their reconstruction after compress–decompress at factor τ, and let z^τ and z denote the layer-normalized final encoder features of the compressed and uncompressed (frozen model from pretraining or stage 1) paths on an unmasked clip x. For a single τ,

$$\mathcal{L}_{\text{comp}}(\tau) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} \left\| z_i^\tau - z_i \right\|_1}_{\textbf{L1 Term (Eq. 10)}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} \left\| h_{k,i}^{(w_s)} - \hat{h}_{k\tau,i}^{(w_s)} \right\|_1}_{\text{Regularization Term}}, \tag{14}$$

and the stage optimizes the mean over the set of token-reduction factors $\mathcal{R} = \{8, 16, 32, 64\}$ (equivalently $\rho \in \{64\times, 128\times, 256\times, 512\times\}$):

$$\mathcal{L}_{\text{comp}} = \frac{1}{|\mathcal{R}|} \sum_{\tau \in \mathcal{R}} \mathcal{L}_{\text{comp}}(\tau). \tag{15}$$

At every iteration the trained path is run once at each $\tau \in \mathcal{R}$ and the gradients are summed before a single optimizer step. This mirrors the T3 principle of Sec. 3.3 one level down: rather than sampling one ratio per step, all ratios are supervised jointly, so the shared kernel K is optimized for every compression level at once and no ratio is left to compete for a sampling slot. Optimization: 100 epochs of 300 iterations with a global batch of 4096 clips drawn from the same pretraining corpus (K710, SSv2, and HowTo100M, sampling weights $0.335/0.100/0.565$) in `bf16`, AdamW with weight decay 0.04, a 10-epoch linear warmup from 10^{-4} to 5.25×10^{-4} and cosine decay to 5×10^{-5}.

Payload and trade-off. We define compression relative to transmitting the input as raw 8-bit RGB, the worst case for an always-on sensor without a hardware video codec. For an intermediate representation of N tokens reduced to N' tokens of dimension w_s, the transmitted payload is $P_{\text{comp}} = N' \cdot (w_s) \cdot b$ bits, and the compression ratio is

$$\rho = \frac{P_{\text{RGB}}}{P_{\text{comp}}}, \qquad P_{\text{RGB}} = F \cdot H \cdot W \cdot 3 \cdot 8, \tag{16}$$

with F, H, W the clip length and frame resolution and $b = 8$ the bits per transmitted element. The elastic token reduction reaches up to $512\times$ payload reduction relative to raw RGB. All reported accuracies are end-to-end, measured after the decoder reconstructs the token grid and the continuation layers run, so they reflect the true cost of compressing the transmitted representation. Together, the exit point and the compression ratio form two orthogonal knobs that expose a controllable trade-off among on-device computation, transmission latency, and accuracy.

A.6 AI Assistance

We used LLMs as a writing assistant only for this manuscript. Its role was limited to refining text that the authors had already written, improving clarity, grammar, and concision, and to checking that notation and terminology were used consistently across the paper. The model did not contribute to the research ideation, the design of the method, the experimental protocol, or the analysis of results, and every suggestion it made was reviewed and verified by the authors before being included. The authors are fully responsible for all content of this paper, including every claim, number, figure, and citation reported herein.