ACTOR-JEPA: MULTI-AGENT MOTION FORECASTING USING LATENT-PREDICTIVE PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Motion forecasting models must capture how agents influence one another, and are trained on large labeled trajectory datasets. Self-supervised pretraining promises to reduce that dependence on labels, yet reported results conflict, ranging from substantial gains to null results. We introduce Actor-JEPA, which applies the Joint-Embedding Predictive Architecture to multi-agent scene encoding. Its encoder factorizes the scene into a cross-agent interaction stage followed by per-agent temporal encoding, and that factorization defines the prediction target: agents are mixed before masking, so the model predicts masked agent-timestep positions in a representation that already carries interaction context, rather than reconstructing coordinates. Across four label regimes and 14 independent pretraining runs on the Waymo Open Motion Dataset, Actor-JEPA raises peak mAP from 0.2089 ± 0.0111 to 0.2245 ± 0.0159 at 20% labels ($+7.4\%$ relative, $p = 0.006$, $n = 14$) and from 0.2799 ± 0.0086 to 0.2938 ± 0.0096 at full labels ($+5.0\%$, $p = 0.026$, $n = 6$), in each case against the identical architecture trained from scratch. Within a dataset, two conditions govern whether any benefit appears: the cross-agent stage must be transferred, since the per-agent core alone performs at from-scratch level, and pretraining must stop before a representation collapse that neither the pretext loss nor the variance regularizer reveals, which we detect by monitoring the effective rank of the encoder's features. Code will be made publicly available.

1 INTRODUCTION

Every robot that shares space with people has to answer the same question before it moves: what will the others do next? An automated vehicle has to decide whether a lane change is executed or aborted, a delivery robot riding along the sidewalk whether it yields or passes. Motion forecasting sits between perception and planning in every such stack, and errors directly propagate into decisions. Methods benchmarked on the Waymo Open Motion Dataset (Ettinger et al., 2021) and Argoverse 2 (Wilson et al., 2021) have converged on a family of transformer-based forecasters like Scene Transformer (Ngiam et al., 2022), MultiPath++ (Varadarajan et al., 2022), Wayformer (Nayakanti et al., 2023), MTR (Shi et al., 2022) and QCNet (Zhou et al., 2023) to autoregressive formulations such as MotionLM (Seff et al., 2023). All these models share a fundamental dependency on labeled trajectories, which means that each new operational domain or sensor configuration requires collecting and curating an entirely new labeled dataset before the forecaster can be deployed with confidence.

Self-supervised pretraining is widely expected to reduce how many annotations a model needs, as it has for perception, where contrastive and masked-image pretraining recover most of the fully supervised accuracy from 1–10% of ImageNet labels (Chen et al., 2020). For motion forecasting, the same expectation has produced contradictory results. Masked trajectory–map reconstruction transfers well on Argoverse 2 (Cheng et al., 2023) and pretraining on pseudo-labelled trajectories helps most when labels are scarce (Xu et al., 2025a), yet pretraining a standard forecaster on more than a thousand hours of in-domain trajectories brings no significant benefit after fine-tuning (Xu et al., 2025b). These results come from comparable transformer architectures and datasets, and nothing in the reports themselves explains the disagreement.

We argue that two things decide such a comparison and are rarely recorded in it: which pretrained parameters actually reach the downstream model, and whether the encoder is still healthy by the time

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.

they do. Both fail silently, producing the flat result a genuine null produces, so a report that does not check them cannot tell the two apart. We therefore study latent-predictive pretraining with both controlled and asserted experiment arms for every run, and introduce **Actor-JEPA**, which applies the Joint-Embedding Predictive Architecture (Assran et al., 2023) to the multi-agent scene encoder of a forecasting model.

The benefit we observe is real but conditional: it appears only when pretraining occurs within the same dataset, requires transferring the cross-agent stage to downstream models, and disappears (or even reverses) if training continues past a representation collapse that the objective fails to detect. Crucially, each of these conditions is empirically verifiable in a pretraining run, eliminating reliance on unverified assumptions and providing actionable guidance for deployment while also establishing clear evaluation protocols for future research.

Our contributions are four-fold:

- We introduce **Actor-JEPA**, a two-stage scene encoder for latent-predictive pretraining. Cross-agent interaction runs first and per-agent temporal encoding second, so the pretext task masks and predicts latents that already carry interaction context. This is a different target from the joint agent-and-map token sets used by masked-autoencoding approaches to this problem (Cheng et al., 2023; Shi et al., 2025). We motivate the choice in Section 3.1.
- We show that this pretraining **improves motion forecasting**, by 7.4% peak mAP at 20% labels ($p = 0.006$) and 5.0% at full labels ($p = 0.026$) over from-scratch controls. Every pretrained measurement comes from an independent pretraining run.
- We localize the benefit to the cross-agent stage. Splitting five pretrained encoders and transferring one part at a time, the per-agent core alone is worth nothing over random initialization (-1.6%, $p = 0.31$), while the cross-agent stage alone is worth $+7.1\%$ ($p = 0.011$). Representational similarity analysis explains why: supervised training independently recovers a core as close to the pretrained one (CKA 0.672) as the pretrained model's own core is (0.654), so the core alone has little to add.
- We identify a silent failure mode in which pretraining degrades an encoder while its objective reports improvement. Effective rank falls from 56.2 to 2.3 in a single epoch while the pretext loss falls through the collapse and reaches its minimum for the run afterwards, and the variance regularizer is uninformative, one of its two terms sitting near a constant and the other contributing no gradient at all, being blind to an anisotropic collapse. Encoders taken after the collapse initialize the forecaster 11% worse than random.

2 Related Work

Self-supervised pretraining for motion forecasting. Forecast-MAE (Cheng et al., 2023) introduced masked autoencoders for trajectory pretraining at the joint trajectory–map level on Argoverse 2. Two later works provide useful boundary conditions for our findings. Valeo4Cast (Xu et al., 2025b) reports that pretraining on 1300+ h of in-domain vehicle trajectories does *not* bring significant benefit over training from scratch after fine-tuning, evidence that in-domain SSL pretraining for motion forecasting can be null in the full-data regime. PPT (Xu et al., 2025a) shows that pseudo-labeled-trajectory pretraining *does* help, with the headline gain concentrated in the low-data regime: SSL with 10% labeled data approaches scratch with 100%. DECAMP (Shi et al., 2025) pretrains for multi-agent motion prediction with disentangled context-aware pretext tasks on Argoverse 2, and like most work in this area reports downstream performance without isolating which parts of the pretrained encoder carry the benefit or whether the pretrained checkpoint is healthy. Those are the two questions our analysis addresses.

Our results sharpen this picture. In-domain, we find a benefit in the full-data regime as well, smaller than at low labels ($+5.0\%$ vs. $+7.4\%$) but present, so the low-label concentration PPT reports is a matter of degree rather than a threshold.

JEPA for autonomous driving. A concurrent line of work applies latent-predictive objectives to driving, and is uniformly aimed at end-to-end planning from pixels rather than at the scene encoder. Drive-JEPA (Wang et al., 2026a) pretrains a ViT on driving video with the V-JEPA objective (Bardes et al., 2024) and distils multimodal trajectory proposals for a lightweight planner. Auto-JEPA (Yang

et al., 2026) keeps the visual encoder frozen and predicts an intent embedding aligned with the latent representation of the future ego trajectory, requiring no perception annotations. WA-JEPA (Wang et al., 2026b) couples future-masked video pretraining on nuPlan with conditional flow matching over latent futures, denoising future scene tokens and ego trajectories jointly. All three pretrain on video and evaluate planning on NAVSIM. We instead pretrain the multi-agent scene encoder directly on trajectory state and evaluate forecasting, so the settings are complementary and the results are not directly comparable.

One finding in that line corroborates ours from the opposite direction. Auto-JEPA reports that masking dynamic-agent regions changes the predicted intent $2.97\times$ as much as masking an equal area at random, indicating that the learned representation is organized around other agents. Our factorial (Section 6.1) reaches a compatible conclusion by intervening on the encoder rather than the input: the cross-agent stage is the one whose transfer is necessary for any benefit, even though it holds only 12 of 94 parameter tensors.

Joint Embedding Predictive Architectures. I-JEPA (Assran et al., 2023) introduced masked prediction in representation space for images, using a context encoder, an EMA target encoder and a narrow predictor, with per-token LayerNorm on the targets and no explicit anti-collapse regulariser. C-JEPA (Mo & Tong, 2024) adds VICReg-style variance–covariance regularisation (Bardes et al., 2022), whose variance term we adopt; data2vec (Baevski et al., 2022) applies the same EMA-teacher masked prediction across speech, vision and language, also with per-position LayerNorm. T-JEPA (Li et al., 2024) applies the framework to GPS trajectory similarity retrieval. We analyze the failure mode that arises when batch normalization replaces the per-token LayerNorm in this family (Appendix B).

Normalization and collapse in self-supervised learning. The role of normalization in SSL is context-dependent. Richemond et al. (2020) show that batch normalization in BYOL's projector *prevents* collapse by introducing implicit contrastive signals through cross-sample statistics. The mechanism we describe in Appendix B runs the other way: in the JEPA target-normalization denominator, batch statistics enable collapse, because that denominator operates on representations that can themselves become degenerate rather than on the stable embeddings a projection head sees. LayerNorm (Ba et al., 2016) is the per-sample alternative, and Colton (2025) analyse its effect on I-JEPA representation quality, finding that it equalizes token norms.

3 METHOD

3.1 THE ACTOR-JEPA ENCODER

Actor-JEPA maps a scene of A agents, each observed over T timesteps, to a per-agent latent, and occupies the context-encoder position of an MTR decoder (Shi et al., 2022) downstream. We are not using MTR's encoder as a forecasting encoder of that style embeds agent and map polylines. It then applies joint self-attention over the combined token set of agent and map, which leaves no point at which per-agent and cross-agent computation can be separated, and no per-agent representation for a latent-predictive objective to predict.

Actor-JEPA is organized instead into two functionally distinct stages, applied in this order. The cross-agent interaction stage consisting of input projection, multi-head attention across agents, residual and layer normalization, output projection operates first, on the raw per-timestep agent features, mixing information between agents at each timestep. The per-agent temporal core then consumes the fused sequence by adding a sinusoidal positional encoding over timesteps, applying a depthwise temporal convolution for local context adapted from T-Jepa (Li et al., 2024), and running a six-layer transformer with $d{=}256$ and eight heads along the time axis of each agent. Figure 1 gives an overview of the implemented architecture for both the pre-training and downstream task.

The ordering is what makes the pretext task well posed. Masking is applied to the fused sequence, so each masked position is a scene latent that already carries the agents present at the timestep, and the EMA target encoder encodes the same fused representation. The target is therefore defined by the factorization itself: an agent-timestep latent is a unit the architecture already produces, whereas over a joint agent-and-map token set the corresponding unit would have to be designated by the objective

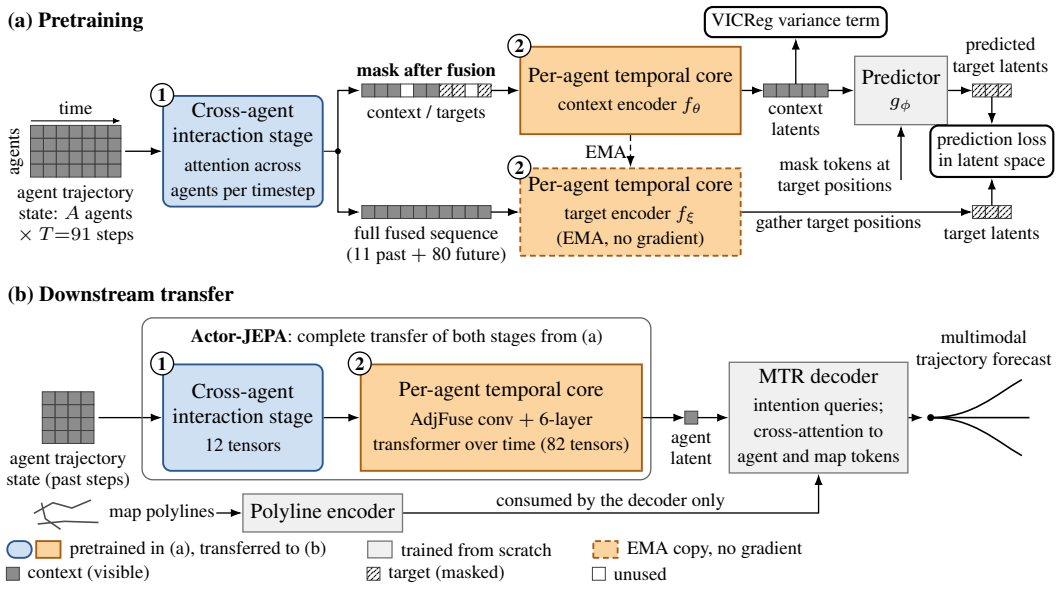

Figure 1: **Architecture and Training Setup. (a) Pretraining.** The agent trajectory state (A agents $\times$ T=91 steps, 11 past and 80 future) first passes through the cross-agent interaction stage (1), which attends across agents at every timestep. Masking is applied to the *fused* sequence: the context encoder f_θ, a per-agent temporal core (2), sees only the visible context tokens, while the EMA target encoder f_ξ processes the full fused sequence without gradient. The predictor g_ϕ fills mask tokens at the target positions from the context latents, and the loss is computed in latent space against the gathered target latents; a VICReg variance term is applied to the context latents. **(b) Downstream transfer.** Both pretrained stages (12 cross-agent and 82 per-agent tensors) are transferred as the agent encoder of an MTR decoder trained from scratch. Map polylines are encoded separately and consumed only by the decoder, so the pretrained encoder never sees the map.

rather than by the encoder. We do not test that alternative here. The factorization also creates an explicit boundary between cross-agent and per-agent computation, which is what the intervention in Section 6.1 cuts along: the two stages hold very different parameter counts, and in our measurements it is the smaller one whose transfer is necessary for any benefit.

3.2 Pretraining objective

Our proposesd masking scheme runs on the fused sequence, so a masked position is a scene state at an instant, not one agent's. Each scene yields two target sets drawn uniformly over all 91 timesteps, each predicted from its own context: a random subset of timesteps with that target's own removed (Appendix B gives the ratios). The context branch encodes the visible portion while an EMA teacher encodes the targets with a predictor mapping context latents to predicted target latents, trained with a smooth-L_1 loss in latent space and a VICReg variance term (Bardes et al., 2022). This is the trajectory analogue of I-JEPA (Assran et al., 2023). Appendix B goes into more detail here.

That window is the entire recording, 11 past and 80 future timesteps, not the past alone, so pretraining observes the segments the downstream task later predicts. No forecasting objective, target trajectory or label is used: the model predicts its own latents from its own inputs. Pretraining and fine-tuning both draw on the Waymo training split and all evaluation is on the held-out validation split, so no evaluation scenario is seen during pretraining.

3.3 Stable pretraining

Normalizing the teacher's targets by batch statistics makes this objective collapse. The batch standard deviation σ falls, the $1/\sigma$ scaling amplifies what remains, the predictor's task eases, and σ falls further. Trajectory tokens enter that loop faster than image patches, because neighboring timesteps

of a smooth 10 Hz signal are far more mutually predictable. Normalizing each token independently with a per-token LayerNorm removes the loop structurally, since the normalizer then depends on the token alone. Every encoder in this paper uses that configuration and Appendix B gives the analysis and the empirical comparison. This failure is distinct from the collapse in Section 6.3, which occurs despite it.

3.4 Complete transfer

Both encoder stages are carried into the downstream model as visualized in Figure 1. This is not a free design choice: Section 6.1 shows that a transfer carrying only the per-agent core performs at from-scratch level, so what matters is not how much of the encoder is reused but whether the cross-agent stage is among it.

3.5 Detecting collapse

Pretraining under this objective can collapse abruptly (Section 6.3) without any signal in the loss. We monitor the effective rank of the encoder's mean-pooled scene embeddings on a fixed validation batch,

$$\mathrm{erank}(X) \;=\; \exp\Big(-\sum_i p_i \log p_i\Big), \qquad p_i = \lambda_i \Big/ \textstyle\sum_j \lambda_j, \tag{1}$$

where λ_i are the eigenvalues of the embedding covariance. This statistic, and the family of rank-based diagnostics it belongs to, is standard in the self-supervised literature, where dimensional collapse is generally treated as something to regularize against (Huang et al., 2024; Almpanakis & Kreshuk, 2026). We use it for the narrower purpose of detecting collapse, because in our setting no other monitored quantity moves when the representation fails. A healthy encoder in our setting sits at $\mathrm{erank} \approx 20$–$63$ of 256 dimensions while a collapsed one sits at $\mathrm{erank} < 3$. The difference is an initialization that helps against one that is worse than random. It is a diagnostic, not a step in our protocol: the fourteen runs behind Table 1 used a fixed 25-epoch budget, so none was stopped or excluded by it, and all fourteen measure inside the healthy band (Appendix A).

4 Experimental setup

Data and downstream task. Pretraining uses the full training split of the Waymo Open Motion Dataset (Ettinger et al., 2021). Downstream, the encoder initializes an MTR decoder fine-tuned for 30 epochs and evaluated with the standard mAP metric. Every run is scored twice. Peak mAP, the maximum over the evaluated epochs, is the pre-registered primary metric. It is biased upward, the more so the more a run oscillates. Late-window mAP, the mean over the evaluated epochs in the last six (25 to 30, two of them on Waymo and six on Argoverse 2, where the resumed legs checkpoint every epoch), is the conservative summary. We report both for every comparison rather than selecting the favorable one. Label availability is controlled by the scenario sample interval, which keeps every SI-th training scenario. Fractions are relative to the standard MTR downstream configuration, SI=5, which we label 100%; SI=25, 50 and 100 retain a fifth, a tenth and a twentieth of it. Pretraining is not subject to this sampling and uses every scenario in the split, so the pretext task sees five times as many scenarios as the 100% supervised arm, without their labels.

The unit of replication. For a pretrained arm the quantity that varies between experiments is the pretraining run, not the fine-tuning seed. In our measurements fourteen independent pretraining runs spread over 0.016 mAP at SI=25, whereas three fine-tuning seeds sharing a single pretrained encoder spread over only 0.0024 mAP. Several seeds drawn from one encoder therefore produce a confidence interval that is far too narrow. (The from-scratch arm's 0.0111 in Table 1 is a different quantity: seed variance with no pretraining stage to hold fixed.) Accordingly, each datapoint in a pretrained arm comes from an independent pretraining run. The results below draw on 14 such runs, the sparser regimes reusing encoders from the same pool. Each from-scratch datapoint is an independent fine-tuning seed, that arm having no pretraining stage to vary.

Scope of the comparison. Every comparison in this paper is internal: the two arms share the architecture, the data pipeline, the decoder, the training schedule and the evaluation code, and differ

Table 1: Downstream forecasting with and without pretraining. Each pretrained datapoint is an independent pretraining run; each from-scratch datapoint an independent fine-tuning seed. p-values are Welch's t-test and the Mann–Whitney U test. The 20% (SI=25) row is the pre-registered primary analysis.

Labels	Metric	Pretrained	From scratch	Δ	Welch p	MWU p
20% (n=14)	peak	$\mathbf{0.2245 \pm 0.0159}$	0.2089 ± 0.0111	$+\mathbf{7.4}\%$	$\mathbf{0.0063}$	0.0077
	late	0.1986 ± 0.0202	0.1883 ± 0.0149	$+5.5\%$	0.136	0.046
100% (n=6)	peak	$\mathbf{0.2938 \pm 0.0096}$	0.2799 ± 0.0086	$+\mathbf{5.0}\%$	$\mathbf{0.026}$	0.009
	late	0.2879 ± 0.0096	0.2760 ± 0.0086	$+4.3\%$	0.047	0.026
10% (n=12)	peak	$\mathbf{0.2010 \pm 0.0132}$	0.1891 ± 0.0058	$+\mathbf{6.3}\%$	$\mathbf{0.013}$	0.012
	late	0.1917 ± 0.0137	0.1828 ± 0.0065	$+4.9\%$	0.063	0.043
5% (n=12)	peak	$\mathbf{0.1783 \pm 0.0108}$	0.1664 ± 0.0061	$+\mathbf{7.2}\%$	$\mathbf{0.0048}$	0.0077
	late	$\mathbf{0.1710 \pm 0.0096}$	0.1591 ± 0.0068	$+\mathbf{7.5}\%$	$\mathbf{0.0035}$	0.0062

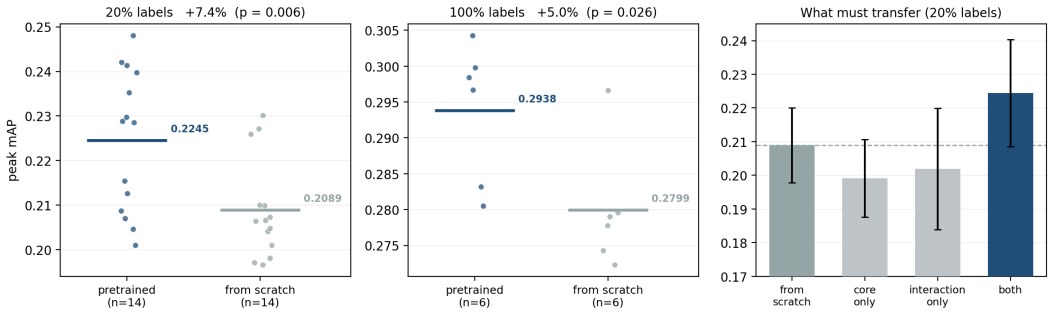

Figure 2: Result of every run. *Left, center*: per-run peak mAP at 20% and 100% labels. Each pretrained point is an independent pretraining run and each from-scratch point an independent fine-tuning seed. Horizontal bars are arm means. The distributions overlap, with four of the fourteen pretraining runs at 20% labels falling below the from-scratch mean, so the claim is distributional rather than a guarantee for an individual run. *Right*: transferring the per-agent core alone leaves performance at from-scratch level (dashed line), while the cross-agent stage alone recovers about half of the composed gain and the two together the rest. Error bars are one standard deviation.

only in whether the encoder was pretrained. That isolates the effect of pretraining, which is the question we ask. It is not a benchmark comparison, and the absolute mAP values reported here should not be read against published numbers for MTR or other forecasters. Our downstream model keeps the MTR decoder but replaces its context encoder with Actor-JEPA (Section 3.1), which consumes agent trajectory state only, with map polylines encoded separately and consumed by the decoder as introduced in MTR (Shi et al., 2022). The training configuration also differs from the published MTR setup. We therefore report what pretraining changes within one fixed system, and make no claim about the state of the art.

Controls and pre-registration. Every comparison uses a from-scratch arm produced by the same script, differing only in initialization, with the initialization asserted in-log before scoring.

5 Results

Table 1 and Figure 2 give the main result. Pretraining improves downstream mAP at every label fraction we measured: $+5.0\%$ at full labels, $+7.4\%$ at 20%, $+6.3\%$ at 10% and $+7.2\%$ at 5%. The three label-scarce regimes sit close to one another, and the benefit is smallest where labels are abundant. The relationship with label fraction is therefore not a steadily growing one: supervision recovers on its own part of what pretraining supplies, which bounds the benefit at full labels, but the benefit does not keep growing as labels become scarcer. On the late-window metric the effect

Table 2: Transferring subsets of one encoder, SI=25. All three transfer arms are subsets of the *same* pretrained encoder, split by parameter tensor, so the arms differ only in which stage arrives; each varies the fine-tuning seed. The cross-agent stage alone recovers about half of the composed gain. The per-agent core alone recovers none. Each transfer row is five independent pretrained encoders, split by parameter tensor and fine-tuned with one seed each, so the replication unit is the pretraining run as elsewhere in the paper. Peak mAP p is a one-sample test of the five encoders against the from-scratch mean.

Transferred	Peak mAP	vs. scratch	p
Neither (from scratch, n=14 runs)	0.2089 ± 0.0111	–	–
Per-agent core only	0.2056 ± 0.0062	-1.6%	0.31
Cross-agent stage only	$\mathbf{0.2237 \pm 0.0074}$	$\mathbf{+7.1\%}$	**0.011**
Complete transfer	0.2147 ± 0.0264	$+2.8\%$	0.65

is 4–8%, and it remains significant on both tests at 5% and at 100% labels. At 20% and 10% the late-window effect is significant under the rank test alone.

5.1 Cross-Dataset Experiment

We repeat the comparison across datasets by pretraining on Waymo and fine-tuning on Argoverse 2 (Wilson et al., 2021) under the same harness: complete transfer, plain fine-tuning, 30 epochs, and six independent pretraining runs against six fine-tuning seeds. Peak mAP is 0.3043 ± 0.0037 pretrained against 0.3026 ± 0.0048 from scratch, a difference of $+0.6\%$ (Welch $p = 0.50$, MWU $p = 0.47$) and the late-window means are 0.3003 ± 0.0035 against 0.2980 ± 0.0040, $+0.8\%$ ($p = 0.31$, MWU $p = 0.52$). Absolute values are higher than the Waymo numbers because the datasets and their metrics differ. We don't intend to compare the results of the two datasets with each other.

This null is powered rather than merely observed. The pooled standard deviation is 0.0043, so at n=6 per arm the smallest difference detectable with 80% power is 0.0077 mAP, 2.6% relative. Power to detect the $+5.0\%$ we measure in-domain at full labels is 1.000. An effect of the size this paper reports would therefore have been found with near-certainty, which makes this an upper bound on cross-dataset transfer and not an absence of evidence.

The in-domain results of Section 5 and the localization of Section 6.1 should be read within a dataset. What the cross-agent stage carries appears to be interaction structure specific to the pretraining distribution, which is consistent with both the size of the in-domain effect and its disappearance here.

6 Analysis

6.1 The cross-agent stage carries the benefit

Table 2 isolates the contribution of each stage by splitting five pretrained encoders along the boundary of Section 3.1 and transferring one part at a time. The per-agent core, 82 of the encoder's 94 parameter tensors, is worth nothing on its own (-1.6%, $p = 0.31$). The cross-agent stage, 12 tensors, is worth $+7.1\%$ by itself ($p = 0.011$), with a spread across encoders of 0.0074. The benefit is localized in the smaller stage.

We cannot say that the two stages together do better than the cross-agent stage alone. Complete transfer across the same five encoders averages $+2.8\%$ ($p = 0.65$) with a spread of 0.0264, three and a half times the cross-agent arm's, and three of the five fall below the from-scratch mean; paired by encoder, the cross-agent stage alone is numerically ahead of the complete encoder ($p = 0.46$). Whatever the composition contributes, this design does not resolve it, and an earlier single-encoder version of this table, which appeared to show the composition adding a further 6.6%, does not survive four more encoders.

This is a decomposition within one encoder, two or one fine-tuning seed per encoder, so within-encoder noise is not separated from the variation between them. And this cohort's complete-transfer

arm sits below Table 1's measurement of the same condition (0.2147 ± 0.0264 against 0.2245 ± 0.0159), so the two do not sit comfortably together.

The natural reading is that the cross-agent stage carries the benefit and remains useful even when paired with a randomly initialized core, whereas the per-agent core contributes nothing on its own, for the reason Section 6.2 gives: supervision recovers an equivalent core unaided. It may contribute alongside the cross-agent stage, so the stages are not separable, but the asymmetry between them is one-sided.

6.2 Why the gain is moderate

Linear centered kernel alignment (CKA) between encoder representations on a fixed validation set, with the interaction stage held fixed to isolate the core, gives a CKA for a pretrained core and a scratch-trained core of 0.672 while a pretrained core and a pretrained-initialized core yields a CKA of 0.654. Supervised training on the forecasting task arrives at a per-agent core as functionally similar to the pretrained core as the pretrained-initialized model's own core is. Pretraining that component is, to this resolution, redundant on its own. The further gain it adds alongside the cross-agent stage (Table 2) is not resolved by a core-level comparison.

Computed on the composed encoder, the fine-tuned representation retains little of its initialization: CKA to the pretrained encoder is 0.083, while the two fine-tuned models, one pretrained and one from scratch, are 0.619 alike. Fine-tuning drives both initializations toward a common functional solution. Consistently, weight drift from the pretrained initialization is 0.44 for the pretrained arm against 0.78 for the from-scratch arm. The pretrained model moves less in parameter space while ending further from the pretrained encoder in function space (CKA 0.083 against 0.132 for the from-scratch model), a small displacement producing a large functional reorganization.

Together these explain the size of the effect. Most of what pretraining learns is recoverable from labels. The residual that is not recoverable accounts for several percent of mAP, and more of it survives when labels are scarce.

6.3 Pretraining collapse

Figure 3 and Table 4, both in Appendix A, show the failure. Over four epochs the representation is stable at an effective rank near 56. In the fifth it falls to 2.3, with the embeddings becoming nearly parallel (mean absolute cosine $0.13 \rightarrow 0.75$) and roughly doubling in norm. It does not recover.

The pretext loss improves rather than merely failing to warn. The end-of-epoch pretext loss falls from 0.17–0.19 over epochs 150–163 to 0.103 at epoch 164, its lowest value in 150 epochs, is unchanged at the rank drop (0.105 at epoch 165), and reaches the run minimum of 0.039 at epoch 200 inside the collapsed regime as seen in Table 3. Checkpoint selection by pretraining loss therefore actively prefers collapsed encoders, which is why pretext loss fails to predict transfer quality. After entering the collapsed regime transfer quality and the loss are anticorrelated.

The anti-collapse regularizer is uninformative. The VICReg variance term sits at 0.050 from epoch 10 onward, exactly half its ceiling, and that value is diagnostic rather than reassuring. It averages two hinges, one on the predictor's output and one on the context encoder's. The predictor's hinge is high and nearly unchanging: its per-dimension standard deviation measures 0.15 to 0.23 on healthy checkpoints, well inside the penalized region, so that hinge contributes close to a constant. The context-encoder hinge, the one that could signal the collapse, is identically zero because it penalizes per-dimension standard deviation, which a high-variance rank-two representation satisfies exactly. Across the epochs spanning the collapse the term logs 0.050 on seventy of eighty-six recorded iterations, with sporadic excursions to 0.051–0.080 that carry no systematic relation to the rank drop. The unchanging number a practitioner sees is one dead term beside one blind one, not a satisfied constraint.

The consequence downstream is severe. An encoder taken after collapse initialises the forecaster 10.2% worse than random initialisation at full labels (0.2515 against 0.2799, p=0.0003) and 17.3% worse at 20% labels (0.1728 against 0.2089, p=0.0001), measured against the same controls as Table 1. A pretrained initialisation is not a weakly-informative prior that fine-tuning can overcome.

Table 3: Six checkpoints from one pretraining run, fine-tuned at SI=25 with a common seed. The lowest pretext loss of the run belongs to the checkpoint that transfers *worst*. Across these six, pretext loss correlates *positively* with downstream mAP (Spearman $\rho = +0.77$, $p = 0.072$), so selecting the lowest-loss checkpoint is worse than selecting at random; effective rank correlates in the expected direction ($\rho = +0.89$, $p = 0.019$). The from-scratch reference at this protocol is 0.2089.

Pretraining epoch	5	10	25	50	100	200
Pretext loss (smooth-L_1)	0.077	0.046	0.185	0.248	0.220	**0.039**
Effective rank	25.1	21.1	31.3	36.6	**62.6**	2.0
Downstream peak mAP	0.2057	0.2083	0.2233	0.2094	**0.2384**	0.1820

Effective rank separates the two conditions which we measure across eight checkpoints with measured downstream performance. Every checkpoint with $\mathrm{erank} > 10$ outperformed every checkpoint with $\mathrm{erank} < 10$, and every collapsed checkpoint fell below the from-scratch baseline. Within the healthy band, however, it is not a graded quality score. Checkpoints span 0.206 to 0.238 and rank orders them only approximately. We therefore use it only as a detector of collapse, which is what early stopping requires.

7 Limitations

The gain depends on pretraining beyond the labeled subset. We pretrain on the full training split and fine-tune on a fraction of it, so the pretrained arm sees scenarios the from-scratch arm never does, and those scenarios carry labels our controls leave unused. Confining the pretext task to the same 20% subset the decoder is fine-tuned on, at equal gradient steps, leaves $+1.6\%$ over from-scratch (0.2122 ± 0.0092 against 0.2089, $p = 0.62$, three pretraining runs) where pretraining on the full split gives $+7.4\%$. What we demonstrate is therefore that pretraining on a large corpus helps when labels are scarce, not that the pretext task extracts extra value from the labeled subset itself.

Scope. The label-regime results are on the Waymo Open Motion Dataset with a single downstream decoder, and the absolute mAP values are below published numbers for state-of-the-art forecasters, for the reasons given in Section 4. Our one cross-dataset test (Section 5.1) returns a null powered to detect the in-domain effect size, so the benefit is specific to the pretraining dataset rather than untested. We also do not know whether the effect size holds for other forecasting architectures, and the cross-agent structure identified in Section 6.1 may be specific to encoders that separate per-agent and cross-agent processing as ours does.

8 Conclusion

Actor-JEPA improves multi-agent motion forecasting on the Waymo Open Motion Dataset by 5 to 7% relative peak mAP over matched controls across four label regimes and 14 independent pretraining runs. The gain requires the cross-agent stage and is bounded because supervision largely recovers the per-agent features on its own. The effect is conditional to the pretraining stopping before a collapse that neither the objective nor its variance regularizer detects, though one forward pass measuring effective rank does. This condition fails silently as a collapsed encoder performs below from-scratch level, which we suspect contributes to the conflicting results in this area. We therefore recommend asserting the initialization under test, treating the pretraining run rather than the fine-tuning seed as the unit of replication, and monitoring a rank statistic during pretraining rather than the pretext loss. A matched cross-dataset test on Argoverse 2 returns a null powered to detect that effect size. We publish our code and model weights to invite further research in this area.

Reproducibility statement

The pretraining recipe and its normalization are specified in Section 3.3 and Appendix B. The encoder architecture and the transfer protocol are described in Sections 3.1 and 3.4. We detail the downstream protocol, control discipline, and pre-registered analysis in Section 4. The dataset is

publicly available. Two details are load-bearing for reproducing our numbers and are stated explicitly rather than left implicit: every pretrained datapoint comes from an independent pretraining run rather than a fine-tuning seed, and every run asserts its own initialization in-log before being scored. The effective-rank statistic used for checkpoint selection is defined in Section 3.5, and the per-run values behind Figure 2 and Table 1 are included in the supplementary material along with the analysis and figure-generation scripts. We additionally publish our code and model weights.

ETHICS STATEMENT

This work uses a public, de-identified driving dataset and involves no human subjects. Motion forecasting is a safety-relevant component of autonomous driving, and we note two properties of our results that bear on deployment. First, the benefit we report is distributional: individual pretraining runs vary, and four of 14 fell below the from-scratch mean, so a pretrained initialization should not be assumed to improve any particular model without measurement. Second, and more consequentially, we document a failure mode in which pretraining silently degrades an encoder while its training objective reports improvement. Monitoring the pretext loss alone could ship an initialization that is worse than random. We report the detection statistic precisely so that this failure is checkable rather than latent.

AI USE STATEMENT

Generative AI tools were used in this work for orchestrating and monitoring cluster runs of experiments, statistical analysis, and reviewing the writing style and grammar of the paper. They were not used to generate synthetic data, and no experimental measurement reported here was produced by a language model. All numbers come from executed training and evaluation runs whose logs are retained.

We have reviewed all AI-assisted work. Every result in this paper was regenerated from raw evaluation logs. Each experimental arm asserts its own initialization programmatically before scoring and the primary analysis (metric, regime and sample size) was fixed in advance of the corresponding runs. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Konstantinos Almpanakis and Anna Kreshuk. Icone: Batch independent collapse prevention for self-supervised representation learning. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 445–463, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37624-4.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629, 2023.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning (ICML)*, pp. 1298–1312, 2022.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 22243–22255, 2020.

Jie Cheng, Xiaodong Mei, and Ming Liu. Forecast-MAE: Self-supervised pre-training for motion forecasting with masked autoencoders. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8645–8655, 2023.

Adam Colton. Elucidating the role of feature normalization in IJEPA. *arXiv preprint arXiv:2508.02829*, 2025.

Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9710–9719, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*, pp. 2790–2799, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, 2018.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Hanxun Huang, Ricardo J. G. B. Campello, Sarah Monazam Erfani, Xingjun Ma, Michael E. Houle, and James Bailey. LDReg: Local dimensionality regularized self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Lihuan Li, Hao Xue, Yang Song, and Flora Salim. T-JEPA: A joint-embedding predictive architecture for trajectory similarity computation. In *Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pp. 569–572, 2024.

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, Zeyu Chen, and Jun Huan. DELTA: DEep learning transfer using feature map with attention for convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning (ICML)*, pp. 2825–2834, 2018.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

Shentong Mo and Shengbang Tong. Connecting joint-embedding predictive architecture with contrastive self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2348–2377, 2024.

Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2980–2987, 2023.

Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Ben Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified architecture for predicting multiple agent trajectories. In *International Conference on Learning Representations (ICLR)*, 2022.

Pierre H. Richemond, Jean-Bastien Grill, Florent Alteché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. BYOL works even without batch statistics. *arXiv preprint arXiv:2010.10241*, 2020.

Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S. Refaat, Rami Al-Rfou, and Benjamin Sapp. MotionLM: Multi-agent motion forecasting as language modeling. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8579–8590, 2023.

Jianxin Shi, Zengqi Peng, Xiaolong Chen, Tianyu Wo, and Jun Ma. DECAMP: Towards scene-consistent multi-agent motion prediction with disentangled context-aware pre-training. *arXiv preprint arXiv:2509.10426*, 2025.

Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6531–6543, 2022.

Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S. Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, and Benjamin Sapp. MultiPath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7814–7821, 2022.

Linhan Wang, Zichong Yang, Chen Bai, Guoxiang Zhang, Xiaotong Liu, Xiaoyin Zheng, Xiao-Xiao Long, Chang-Tien Lu, and Cheng Lu. Drive-JEPA: Video JEPA meets multimodal trajectory distillation for end-to-end driving. *arXiv preprint arXiv:2601.22032*, 2026a.

Xinlin Wang, Yujiao Xiang, Yuheng Zhou, Jingqi Wang, Minqing Huang, Jiajie Huang, Dongxu Wei, Tingguang Zhou, Xiyang Wang, Gong Chen, Zhi Xu, Feiyang Tan, Hangning Zhou, and Mu Yang. WA-JEPA: Rethinking the video JEPA paradigm for world-action modeling in autonomous driving. *arXiv preprint arXiv:2608.20974*, 2026b.

Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Yihong Xu, Yuan Yin, Eloi Zablocki, Tuan-Hung Vu, Alexandre Boulch, and Matthieu Cord. PPT: Pretraining with pseudo-labeled trajectories for motion forecasting. In *Workshop on Making Sense of Data in Robotics: Composition, Curation, and Interpretability at Scale at CoRL 2025*, 2025a.

Yihong Xu, Éloi Zablocki, Alexandre Boulch, Gilles Puy, Mickael Chen, Florent Bartoccioni, Nermin Samet, Oriane Siméoni, Spyros Gidaris, Tuan-Hung Vu, Andrei Bursuc, Eduardo Valle, Renaud Marlet, and Matthieu Cord. Valeo4cast: A modular approach to end-to-end forecasting. In Alessio Del Bue, Cristian Canton, Jordi Pont-Tuset, and Tatiana Tommasi (eds.), *Computer Vision – ECCV 2024 Workshops*, pp. 1–14, Cham, 2025b. Springer Nature Switzerland. ISBN 978-3-031-91767-7.

Jiwei Yang, Zhengxian Chen, Chaosheng Huang, and Jun Li. Auto-JEPA: A latent world model of continuous intent for end-to-end autonomous driving. *arXiv preprint arXiv:2607.29031*, 2026.

Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17863–17873, 2023.

A THE SILENT COLLAPSE: FIGURE AND TABLE

Figure 3 and Table 4 document the single-epoch rank collapse analyzed in Section 6.3. It occurs under the per-token target normalization of Section 3.3 and is distinct from the batch-statistics feedback loop analyzed in Appendix B.

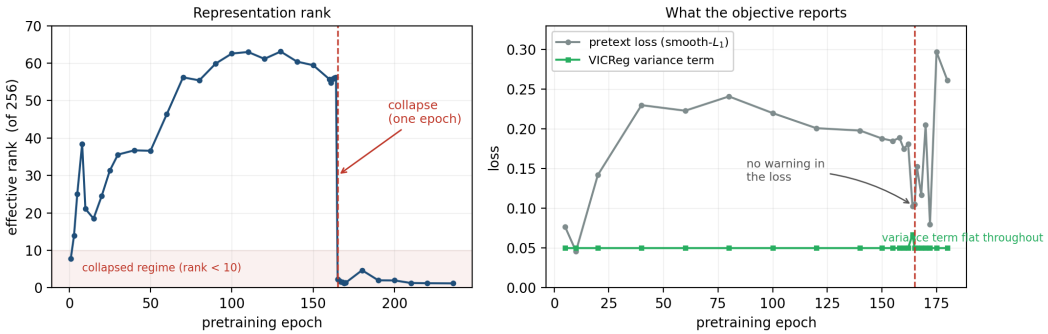

Figure 3: Pretraining collapses silently. *Left*: effective rank of the encoder's scene embeddings (d=256) over pretraining. The representation is stable near rank 56 for four consecutive epochs and falls to 2.3 in the fifth. *Right*: The pretext loss and the VICReg variance term over the same run. The pretext loss falls through the collapse rather than rising, because predicting a rank-one target is easy. The lowest loss of the run belongs to a collapsed checkpoint (see Table 3). The VICReg variance term, whose purpose is to prevent collapse, is flat throughout: its predictor hinge sits near a constant and its context hinge, which contributes no gradient, cannot see an anisotropic collapse. Encoders taken after this point initialise the downstream forecaster worse than random.

Table 4: Effective rank of the encoder's scene embeddings (d=256) across consecutive pretraining epochs, on a fixed validation batch. The transition at epoch 165 occupies a single epoch. Columns 161–169 are pretraining run T6-RETAIN, the run of Figure 3 and Table 3; the epoch-300 column is a *different* run under the same recipe, shown only for its end state. The pretext loss is the smooth-L_1 objective at the last iteration of the epoch, and the variance term is the VICReg variance penalty at the same point.

Epoch	161	162	163	164	**165**	166	169	300
Effective rank	54.8	55.8	56.1	56.2	**2.3**	1.9	1.3	2.7
Mean $\lvert\cos\rvert$	0.132	0.131	0.131	0.132	**0.745**	0.824	0.877	0.601
Feature norm	24.1	23.9	24.3	24.9	**53.7**	54.8	53.2	46.4
Pretext loss (smooth-L_1)	–	–	–	0.103	0.105	0.153	–	0.470
VICReg variance term	–	–	–	0.050	0.050	0.050	–	0.050

Effective rank of every Table 1 encoder. The criterion of Section 3.5 was characterized on the long runs above. For completeness we measure it on all fourteen pretraining runs behind Table 1, at the epoch each one stopped, on the same fixed validation batch and with the same settings, so the values are comparable to those above. We show these results in Table 5. Every run used the same fixed 25-epoch budget: none was stopped early by the criterion and none was excluded by it. All fourteen sit inside the healthy band, at $\mathrm{erank} = 21.7$–29.4 (mean 26.3 ± 2.3 of 256 dimensions), an order of magnitude above the collapsed regime. The collapse of Table 4 appeared at epoch 165, far beyond this budget, so on these runs the criterion never had occasion to fire.

Within the healthy band the statistic does not grade quality, which matches what Section 6.3 reports over the long runs. Across the fourteen, effective rank and downstream peak mAP are uncorrelated (Pearson $r = 0.07$, $p = 0.82$; Spearman $\rho = 0.11$, $p = 0.71$), and the four runs that initialized the forecaster below the from-scratch mean are all inside the band, at $\mathrm{erank} = 22.2$, 26.4, 26.5 and 27.7. The statistic detects collapse, however, it does not predict which healthy encoder will transfer best, and it would not have screened out the four weak runs.

B STABLE PRETRAINING: ANALYSIS AND COMPARISON

B.1 MASKING PARAMETERS.

The masker runs on the fused sequence, so it selects timesteps rather than individual agents. Per scene it draws two target sets of 10% of the 91 timesteps each, sampled uniformly over the whole

Table 5: Effective rank at the stopping epoch for all fourteen pretraining runs behind Table 1, with the downstream peak mAP each produced at 20% labels. Every run stopped at its fixed 25-epoch budget. V1-ORIG also appears in Table 3, where it was measured independently at the same value, 21.70.

Pretraining run	erank	Peak mAP
REP-F	29.35	0.2352
REP-K	28.52	0.2414
REP-H	28.49	0.2421
REP-E	28.48	0.2126
REP-J	27.70	0.2070
REP-A	26.56	0.2154
REP-M	26.48	0.2087
REP-L	26.39	0.2010
REP-I	26.21	0.2285
REP-G	25.79	0.2297
REP-C	25.42	0.2481
REP-B	24.59	0.2288
REP-D	22.22	0.2046
V1-ORIG	21.70	0.2397
From scratch (mean of 14)	–	0.2089

window with a cap on the overlap between them, and gives each target its own context: an initial context of 80% of the timesteps drawn at random, minus that target's timesteps, with a floor of 10% of the window. Context and target ratios are resampled per scene within ± 0.05 and ± 0.03 of 0.75 and 0.10, following a cosine warm-up over the first tenth of training. Targets are not biased toward later timesteps.

B.2 JOINT EMBEDDING PREDICTIVE ARCHITECTURES

This appendix concerns the batch-normalization collapse that motivated the per-token target normalization of Section 3.3. The objective, its variance term, the feedback loop and the fix are stated in turn.

A JEPA consists of three components: a context encoder f_θ, an EMA target encoder f_ξ (with $\xi \leftarrow \tau\xi + (1-\tau)\theta$), and a predictor g_ϕ. Given an input sequence, a subset of positions are designated as the context and the remainder as targets. The model is trained to minimize:

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \big\| g_\phi\big(f_\theta(\mathbf{x}_{\text{ctx}}), m_i\big) - \text{normalize}\big(f_\xi(\mathbf{x}_i)\big) \big\|_2^2, \tag{2}$$

where m_i is a positional mask token indicating which target region to predict, and normalize$(\cdot)$ is the critical step under study. The implemented objective replaces the squared norm by a smooth-L_1 residual and standardizes the prediction as well as the target (Appendix B.5.1). The squared form is the tractable case analyzed below.

I-JEPA (Assran et al., 2023) applies layer normalization to the target encoder's output representations. data2vec (Baevski et al., 2022) uses the same per-position normalization. In our implementation we experimented with both layer and batch normalization.

B.3 VICREG-STYLE VARIANCE REGULARIZATION

Our architecture also incorporates a variance regularization term from VICReg (Bardes et al., 2022), applied as the average of two hinges, one on the context encoder's output z and one on the predictor's output $\hat{z}$:

$$\mathcal{L}_{\text{std}} = \frac{\lambda}{2} \left[\frac{1}{D} \sum_{d=1}^{D} \max\big(0,\ 1 - \text{std}(z_d)\big) + \frac{1}{D} \sum_{d=1}^{D} \max\big(0,\ 1 - \text{std}(\hat{z}_d)\big) \right], \qquad \lambda = 0.1, \tag{3}$$

where z_d and $\hat{z}_d$ are the d-th features and each standard deviation is taken across the tokens of the batch. Each hinge is bounded by 1, so the term's ceiling is 0.1 and the constant 0.050 of Section 6.3

is one hinge fully saturated. With λ=0.1 the term provides mild anti-collapse pressure but acts slowly, and the gradient shock from batch-norm amplification overwhelms it within a single epoch.

B.4 Applying Batch Normalization

Empirically, we observe a dilemma:

- With batch normalization: training learns effectively early on (eval loss = 0.018 at epoch 5) but collapses catastrophically at peak LR.
- Without normalization: training never collapses (grad norms $\approx$0.07 through 56 epochs) but representations plateau at eval loss $\approx$0.22 and never improve.

The normalization that enables learning is also the mechanism of collapse.

B.5 The Batch Normalization Collapse Feedback Loop

B.5.1 Formal Analysis

Let $\mathbf{T} \in \mathbb{R}^{N \times D}$ be the target encoder outputs for a batch of N tokens, each with D features. Batch normalization along $\dim$=0 computes, for each feature d:

$$\hat{T}_{n,d} = \frac{T_{n,d} - \mu_d}{\sigma_d + \varepsilon}, \quad \sigma_d^2 = \frac{1}{N}\sum_{n=1}^{N}(T_{n,d} - \mu_d)^2. \tag{4}$$

Consider the scenario where batch collapse begins: all N tokens converge toward a common vector $\mathbf{v}$, so $T_{n,d} = v_d + \delta_{n,d}$ with $|\delta_{n,d}| \ll 1$. Then $\sigma_d \approx \mathrm{std}(\delta_{\cdot,d}) \to 0$ as the collapse progresses.

Relation to the implemented objective. Our implementation uses a smooth-L_1 objective with threshold 1, applied after both the prediction and the target have been standardized, with the standardization statistics detached. The analysis below uses the squared form, and normalizes only the target, because it is the tractable case and the feedback loop it exposes is a property of the normalization denominator rather than of the residual's shape. Two consequences of the difference are worth stating. A smooth-L_1 residual has gradient bounded by 1, so the divergence below is an upper bound on what the objective can produce rather than a description of it. Because the prediction is standardized on the same footing as the target, the $1/(\sigma_d + \varepsilon)$ factor reaches the encoder through the prediction branch as well.

The squared training loss becomes:

$$\mathcal{L}_{\text{sq}} = \frac{1}{ND}\sum_{n,d}\left(\hat{P}_{n,d} - \hat{T}_{n,d}\right)^2 \tag{5}$$

$$= \frac{1}{ND}\sum_{n,d}\left(\hat{P}_{n,d} - \frac{T_{n,d} - \mu_d}{\sigma_d + \varepsilon}\right)^2. \tag{6}$$

As $\sigma_d \to 0$, the denominator $\sigma_d + \varepsilon \to \varepsilon = 10^{-6}$, amplifying the loss by $1/\varepsilon = 10^6$. The gradient with respect to the context encoder's parameters contains a factor of $1/(\sigma_d + \varepsilon)$; as collapse accelerates, this factor grows without bound, creating a catastrophic gradient spike.

Positive feedback. The amplified gradient causes a large parameter update that further homogenizes the context encoder's outputs. More homogeneous outputs lead to more similar target encoder outputs (through the EMA coupling). This reduces σ_d further in the next step. The loop: *collapse* $\to$ *smaller* σ_d $\to$ *larger loss amplification* $\to$ *larger gradient* $\to$ *faster collapse* is a positive feedback loop with no stable equilibrium.

Empirical verification. We observe this sequence in our experiments:

- **Peak LR** 3×10^{-4}: gradient norm at epoch 5 (mid-warmup): 0.066 (pristine); epoch 10 (peak LR): 9.24; epoch 11: 1270. Eval loss at epoch 10: 0.218.

- **With EMA momentum warmup**: same LR, more severe collapse. Epoch 10 grad norm: 1.4×10^6. Lower initial EMA momentum ($0.996 \rightarrow 0.9999$) made the target encoder track the context encoder more closely, so when the context encoder destabilized, the target encoder collapsed simultaneously, further reducing σ_d.
- **Peak LR** 1×10^{-4}: survived peak LR (epoch 10 grad norm: 0.045) but still collapsed at epoch 20. Lower LR reduces the gradient magnitude, slowing collapse accumulation, but the positive feedback loop is still present and eventually dominates.

B.5.2 Why Trajectory Tokens Collapse Faster Than Image Patches

In image JEPA (Assran et al., 2023), spatial patches from natural scenes are visually diverse. Different patches contain different textures, colors, and structures. This natural diversity maintains non-trivial σ_d even as learning progresses.

Trajectory tokens encode agent motion over time. In the Waymo dataset, the majority of agents exhibit slow-changing, nearly periodic motion (constant velocity, gentle turns). This means many trajectory tokens in a batch are genuinely similar, both because the same agent's successive tokens are similar and because many agents in the same scene exhibit similar motion patterns imposed by the traffic flow structure and the environment. As a result, σ_d is naturally lower in trajectory data, providing a smaller buffer before the amplification regime. The feedback loop triggers at lower LR and earlier epochs than it would for image data.

B.6 Per-Token Layer Normalization

B.6.1 Why Layer Normalization is Immune

Changing the normalization from batch statistics to per-token statistics, matching the layer normalization used in the original I-JEPA implementation (Assran et al., 2023), fixes the above described issue.

With per-token normalization, σ_n measures diversity within a single token's D features:

$$\sigma_n^2 = \frac{1}{D} \sum_{d=1}^{D} (T_{n,d} - \mu_n)^2. \tag{7}$$

For $\sigma_n \rightarrow 0$, it is necessary that all D features of token n become identical. This is a much stronger condition than batch collapse (all N tokens becoming similar). Specifically:

- **Batch collapse** (all tokens converge): $T_{n,d} \approx v_d$ for all n. Per-token std: $\sigma_n \approx \mathrm{std}_d(v_d)$, which is the std of the common vector's feature values. As long as the representation is not completely flat (which would imply no information), σ_n remains bounded away from zero.
- **Per-token collapse** (single token n becomes uniform): $T_{n,d} = c$ for all d. This would require the target encoder to map one specific input position to a constant, regardless of context. This is a pathological per-token collapse that is far harder to trigger through ordinary gradient dynamics.

The $10^6\times$ amplification feedback loop cannot form because the condition $\sigma_n \rightarrow 0$ decouples from the batch-level collapse dynamics.

B.6.2 Connection to the JEPA Family

The per-token normalization is consistent with the entire family of EMA-teacher predictive architectures:

- I-JEPA (Assran et al., 2023): layer normalization on targets, confirmed by Colton (2025).
- data2vec (Baevski et al., 2022): layer normalization per sequence position (over feature dimension).

Table 6: Pretext-task variant results, each pretrained and fine-tuned through the same fixed pipeline at SI=25. No variant separates from the base recipe by more than the run-to-run standard deviation of a single evaluation (0.016, Section 4), so this table bounds the effect of these changes rather than ranking them.

Pretext task	Peak mAP
Base recipe (Actor-JEPA)	0.2397
I-JEPA-style multi-block temporal masking	0.2407
VICReg variance measured at a fixed timestep	0.2394
Agent-level context masking	0.2354

C Additional related work: adapting frozen and pretrained backbones

Parameter-efficient transfer learning. Houlsby et al. (2019) introduced residual bottleneck adapter modules for NLP, inserting small two-layer MLP modules within each transformer layer while keeping the backbone frozen. This achieves near-full-fine-tuning performance with a fraction of trainable parameters. LoRA (Hu et al., 2022) proposes low-rank decomposition of weight update matrices as an alternative, achieving comparable efficiency with fewer adapter parameters. Both methods target a common challenge: how to adapt a large frozen backbone to a new task or domain at low parameter cost. Our frozen-encoder experiments use the Houlsby design in the trajectory setting and find a structural ceiling: output-layer adapters can re-project the representation the frozen encoder already computes, but cannot supply information it never computed.

Source-anchored fine-tuning. A family of methods regularizes fine-tuning toward the pretrained source. L2-SP (Li et al., 2018) regularizes fine-tuned parameters toward the source weights. DELTA (Li et al., 2019) regularizes features toward source features with attention weighting. Learning without Forgetting (Li & Hoiem, 2018) regularizes outputs toward a frozen teacher. An orthogonal mechanism is the differential learning rate popularized by ULMFiT (Howard & Ruder, 2018), where lower layers receive a smaller step than task-specific upper layers. We do not evaluate these methods here. We note them because Section 6.2 shows that fine-tuning moves the representation far from its initialization, which is the regime they target.

D Pretext-task variants

In addition to the base variant, we investigated three variants of the pretext task (multi-block temporal masking, a fixed-timestep variance term, agent-level context masking). None separates from the base recipe by more than the run-to-run standard deviation of a single evaluation.

Each variant changed one aspect of the pretext task and was pretrained and fine-tuned through the otherwise fixed pipeline. What each did:

- **I-JEPA-style multi-block temporal masking.** Replaces the base adaptive-context sampler, which draws a 75% context ratio with a $3\times$ bias toward future targets, with two contiguous blocks of roughly $1.8\,\mathrm{s}$ masked bidirectionally. The intent was to make the prediction span a coherent interval rather than scattered timesteps.
- **VICReg variance measured at a fixed timestep.** The base variance term pools tokens from all 91 timesteps, so the positional code alone can satisfy the hinge and a scene-agnostic representation goes unpenalized. This variant centers each token on the mean of the tokens at its own timestep, so the term measures spread across scenes at a fixed time.
- **Agent-level context masking.** The EMA target encoder still sees the scene fused from every agent, while the context path sees the focal agent plus a random subset of its neighbors, each non-focal agent dropped with a per-sample probability. Temporal masking is unchanged and applied on top, so the model must infer a neighbor's contribution from the agents it can see.

Table 6 reports the results of the different pretext task variants. All four entries (that includes our base version) fall within one standard deviation of one another, so the honest reading is that this evaluation cannot distinguish them, not that the base recipe is optimal.

E ADDITIONAL LIMITATIONS

The benefit is distributional. Four of fourteen pretraining runs at 20% labels produced encoders that initialized the forecaster below the from-scratch mean. Pretraining improves the expected outcome, however, it does not guarantee an improvement for a particular run, and a practitioner obtaining a single pretrained encoder has no assurance from our results alone.

Metric sensitivity. Peak mAP over the fine-tuning trajectory is upward-biased on runs that oscillate, which ours do. The late-window means are the conservative estimate and are weaker: at 20% labels the late-window difference reaches significance under the rank test but not under Welch's t-test. We report both throughout rather than selecting the favorable one.

One pre-registered regime. The metric, regime and sample size were fixed in advance for $\mathrm{SI}{=}25$ only. The other three regimes are secondary analyzes whose sample sizes were chosen after seeing that result, and should be weighted accordingly. All four point the same way and three of the four agree closely in magnitude, but only one of them was specified before the data were seen.

Asymmetric replication units. The pretrained arm varies the pretraining run and the from-scratch arm varies the fine-tuning seed, because the latter has no pretraining stage. This is the correct comparison for the question we ask, but the two arms do not have the same variance structure, and the resulting test is not a paired design.

A detector, not a remedy. We characterize the collapse and give a statistic that detects it, but we do not prevent it, and the recipe that produces it is otherwise the one we recommend. Whether it admits a regularization fix of the kind proposed for isotropic collapse (Huang et al., 2024; Almpanakis & Kreshuk, 2026) is open. We also locate the onset epoch in one run. A second reaches the same collapsed state, but we do not characterize how the onset varies across recipes or seeds.