LEARNING FROM THE PAST: EXPERIENCE REPLAY STRENGTHENS JOINT-EMBEDDING PREDICTIVE LEARNING IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) excel at token-level generation but may learn undesirable abstract semantics and lack comprehensive perception. LLM-JEPA mitigates this by aligning different views of the same underlying knowledge via a joint-embedding predictive architecture (JEPA). However, strong alignment does not necessarily lead to accurate, stable predictions. To address this, we propose ER-JEPA, which adds an episodic replay path to LLM-JEPA. ER-JEPA stores training pairs in a memory. At each step, it stores and retrieves relevant data to provide additional supervision. This enables learning from both the current batch and stored training pairs, providing additional supervision for token prediction and representation alignment. Experiments across multiple datasets (NL-RX, GSM8K, Spider, and NQ-Open) demonstrate that ER-JEPA consistently outperforms LLM-JEPA.

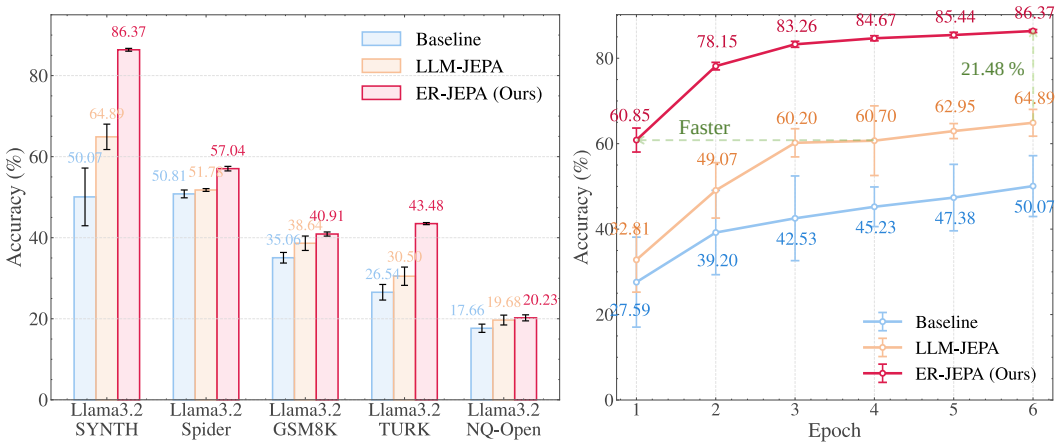


Figure 1: **ER-JEPA outperforms LLM-JEPA across tasks.**

1 INTRODUCTION

Large language models (LLMs) have achieved remarkable advances in fluent generation (Pillutla et al., 2021; Yu et al., 2023; Li et al., 2025; Gong et al., 2024; Li et al., 2024; Liu et al., 2024), few-shot learning (Brown et al., 2020; Wei et al., 2021; Wang et al., 2024), and broad downstream transfer (Rafailov et al., 2023; Zeng et al., 2022; Huan et al., 2025). These abilities require models to extract shared structure from large-scale data and convert it into dependable predictions on new problems. Autoregressive pre-training addresses both needs through next-token prediction (NTP) (see Figure 2(a)), which provides a strong learning signal and can lead to semantic representations under the right conditions (Jin & Rinard, 2024). However, a token-level objective may also favor local coherence over the representations needed for comprehensive perception and reasoning.

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.

Joint-Embedding Predictive Architectures (JEPAs) shift prediction from the token level to the semantic level. Recent work on JEPAs has shown promising advances in representation learning and provable benefits for perception tasks (Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025; Chen et al., 2026). Inspired by JEPAs, Huang et al. introduced LLM-JEPA (Huang et al., 2026), which adds an embedding space predictive loss to the standard language modeling loss (see Figure 2(b)). This loss aligns different views of the same underlying knowledge while preserving generative capability. It predicts both the next token and the target representation, improving the perception capability of LLMs.

Despite these advances, better representation alignment does not by itself guarantee reliable predictions. We observe that LLM-JEPA continues to make errors even after its alignment loss has converged (Figure 3). Some previously correct predictions also become incorrect later in training. Therefore, progress in representation learning is not automatically converted into predictions that are learned and retained. This leaves an open question: how can LLM-JEPA correct persistent errors and retain correct predictions during training?

We explore experience replay as a way to address this question by revisiting previously seen training examples and providing additional supervision as the model changes. Like standard fine-tuning, LLM-JEPA supervises an example only when it appears in the current mini-batch. Between two visits, many updates change the model, and the example is not checked again until the data schedule reaches it in the next epoch. Replay can reduce this gap by revisiting past examples during training. The replay mechanism is widely used in continual learning to preserve learned knowledge (Shi et al., 2024; Huang et al., 2024; Deng et al., 2025), and Complementary Learning Systems (CLS) theory gives it a similar role in the brain (Kumaran et al., 2016). In CLS, a fast system stores individual experiences and replays them to a slow system that extracts shared structure. From this perspective, LLM-JEPA is similar to the former but has no counterpart to the fast system. This motivates our research question: **can experience replay improve the predictive performance of LLM-JEPA?**

To answer it, we propose ER-JEPA, which adds an episodic replay (ER) pathway that stores past training examples and replays them during later updates (see Figure 2(c)). It enables the model to learn from useful past experiences beyond the current mini-batch, providing additional supervision signals and strengthening the perception capabilities of LLMs. The replay path is removed after training, so ER-JEPA has the same architecture and inference cost as LLM-JEPA.

In summary, the contributions of this paper are as follows:

- We propose ER-JEPA, which extends LLM-JEPA with an episodic replay path that revisits past training examples to provide additional supervision for token prediction and representation alignment. We show that learning from past samples continues to provide alignment updates and reinforces the knowledge important for perception and reasoning.
- We implement content-based, uniform, and hard selection within a shared replay path. We demonstrate that the replay branch itself, rather than any specific retrieval rule, is the central source of improvement.
- We evaluate ER-JEPA across multiple datasets. The results show that ER-JEPA consistently outperforms LLM-JEPA, demonstrating the benefit of episodic replay for abstraction and generalization.

2 METHODOLOGY

ER-JEPA extends LLM-JEPA with a replay path over stored training examples. We first introduce the joint training objective in Section 2.1. We then explain how replay provides additional supervision in Section 2.2.

2.1 ER-JEPA OVERVIEW AND OBJECTIVE

Our ER-JEPA builds on LLM-JEPA, retaining its token prediction and representation alignment objectives while adding an episodic replay path to learn from past training examples. The original

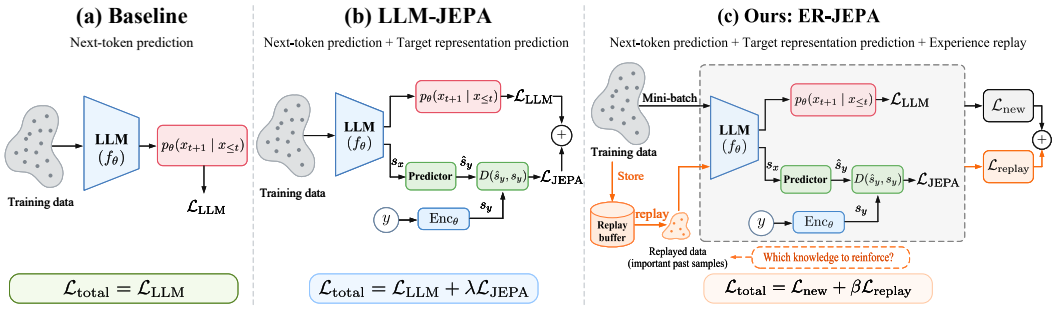


Figure 2: **Comparison of baseline training, LLM-JEPA, and ER-JEPA.** (a) The baseline learns through next token prediction. (b) LLM-JEPA adds prediction in the embedding space to learn abstract relations between inputs and targets. (c) ER-JEPA adds a replay path to LLM-JEPA. It stores past training examples and replays their tokens, providing additional supervision for both token prediction and representation alignment. Past experience thus continues to shape learning as the model changes, helping it correct errors and reinforce learned knowledge for more reliable predictions.

LLM-JEPA objective is

$$\mathcal{L}_{\text{LLM-JEPA}} = \underbrace{\sum_{\ell=2}^{L} \mathcal{L}_{\text{LLM}}(\text{Text}_{1:\ell-1}, \text{Text}_{\ell})}_{\text{generative capabilities (LLM)}} + \lambda \underbrace{d(\text{Pred}_\phi(\text{Enc}_\theta(\text{Text})),\ \text{Enc}_\theta(\text{Code}))}_{\text{abstraction capabilities (JEPA)}}. \tag{1}$$

Here Text and Code denote the source and target views, and L is the source sequence length. Enc_θ and Pred_ϕ denote the encoder and predictor. The function d measures cosine distance, and λ controls the JEPA loss weight.

For a fresh mini-batch $\mathcal{B}_t$ of size B, let x_i^u and x_i^a denote the source and target tokens of example i, and $x_i^{\text{full}} = x_i^u \| x_i^a$. Define

$$z_i^u = \text{Enc}_\theta(x_i^u), \qquad p_i = \text{Pred}_\phi(z_i^u), \qquad z_i^a = \text{Enc}_\theta(x_i^a). \tag{2}$$

Let $\mathcal{T}_i^{\text{new}}$ contain the target-token positions in x_i^{full}. The loss on new data is

$$\mathcal{L}_{\text{LLM-JEPA-new}}(\mathcal{B}_t) = \frac{\gamma}{Z_t^{\text{new}}} \sum_{i=1}^{B} \sum_{\ell \in \mathcal{T}_i^{\text{new}}} \mathcal{L}_{\text{LLM}}\big(x_{i,<\ell}^{\text{full}}, x_{i,\ell}^{\text{full}}\big) + \frac{\lambda}{B} \sum_{i=1}^{B} d(p_i, z_i^a), \tag{3}$$

where $Z_t^{\text{new}} = \sum_{i=1}^{B} |\mathcal{T}_i^{\text{new}}|$ and $\gamma = 1$ is the language-modeling loss weight.

The full ER-JEPA objective adds a replay term,

$$\mathcal{L}_{\text{ER-JEPA}}(\mathcal{B}_t) = \mathcal{L}_{\text{LLM-JEPA-new}}(\mathcal{B}_t) + \beta\, \mathcal{L}_{\text{LLM-JEPA-replay}}(\mathcal{B}_t), \tag{4}$$

where $\beta \ge 0$ controls the strength of replay. $\mathcal{L}_{\text{LLM-JEPA-replay}}$ is defined in Section 2.2.

2.2 Replay-Based Training

Content-based, uniform, and hard replay are three implementations of the same replay path. They share the replay objective below. The selection rules for all three strategies are given in Appendix D.3. Unless otherwise specified, ER-JEPA uses content-based replay.

At each step, a replay strategy selects up to R entries from the episodic memory, where R is the replay budget. Let R_t be the number of selected entries and $(\widetilde{\tau}_i^u, \widetilde{\tau}_i^a)$ be the stored source and target tokens for entry i. Memory storage and retrieval are described in Appendix D.1.

For each replay entry i, we concatenate the stored source and target tokens as $\widetilde{\tau}_i^{\text{full}} = \widetilde{\tau}_i^u \| \widetilde{\tau}_i^a$. The current model recomputes the source and target representations,

$$z_i^{u,\text{rep}} = \text{Enc}_\theta(\widetilde{\tau}_i^u), \qquad z_i^{a,\text{rep}} = \text{Enc}_\theta(\widetilde{\tau}_i^a), \qquad p_i^{\text{rep}} = \text{Pred}_\phi(z_i^{u,\text{rep}}). \tag{5}$$

Let $\mathcal{T}_i^{\text{rep}}$ contain the target-token positions in $\widetilde{\tau}_i^{\text{full}}$. For $R_t > 0$, the replay loss is

$$\mathcal{L}_{\text{LLM-JEPA-replay}}(\mathcal{B}_t) = \frac{\gamma}{Z_t^{\text{rep}}} \sum_{i=1}^{R_t} \sum_{\ell \in \mathcal{T}_i^{\text{rep}}} \mathcal{L}_{\text{LLM}}\left(\widetilde{\tau}_{i,<\ell}^{\text{full}}, \widetilde{\tau}_{i,\ell}^{\text{full}}\right) + \frac{\lambda}{R_t} \sum_{i=1}^{R_t} d(p_i^{\text{rep}}, z_i^{a,\text{rep}}), \quad (6)$$

where $Z_t^{\text{rep}} = \sum_{i=1}^{R_t} |\mathcal{T}_i^{\text{rep}}|$. The loss is zero when $R_t = 0$. The normalization matches the new-data loss in Eq. (3).

The current and replay losses jointly update the shared model through Eq. (4). Replay-time JEPA errors refresh the selected memory scores, and replay counts are incremented as specified in Appendix D.2. At inference time, the episodic pathway is removed.

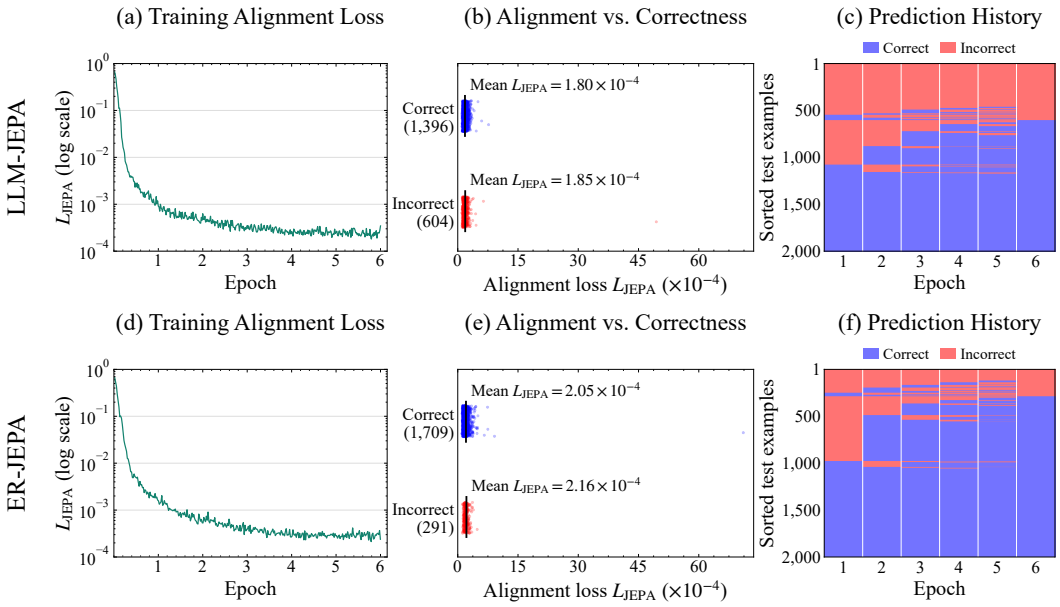


Figure 3: **LLM-JEPA can achieve strong alignment yet still make errors and lose previously correct predictions.** For LLM-JEPA, the JEPA loss converges in (a). Yet in (b) and (c), errors remain even when cosine similarities between inputs and targets are close to one. Some correct predictions also become incorrect as training continues. For ER-JEPA, the JEPA loss also converges in (d). In (e) and (f), more predictions are correct despite slightly lower cosine similarities. At each checkpoint, ER-JEPA consistently has more correct answers than LLM-JEPA. These results suggest that replay supervision helps resolve errors that alignment alone leaves unresolved. This figure uses Llama-3.2-1B on 2000 SYNTH test examples with seed 82. Heatmap rows are sorted independently within each method by final-checkpoint correctness and prediction history.

3 Experiments: Episodic Replay Improves LLM-JEPA

This section evaluates whether episodic replay improves JEPA-based LLM training. After describing the experimental setup in Section 3.1, we report results across tasks and compare accuracy at matched training compute in Section 3.2. We then examine the role of historical replay in Section 3.3, analyze prediction quality across regex structures and lengths and report resource costs in Section 3.4. Further ablations are provided in Appendix E.

3.1 Experimental Setup

We evaluate ER-JEPA under the same task families used in the LLM-JEPA protocol. Experiments are conducted on Llama-3.2-1B-Instruct (Grattafiori et al., 2024). We use five datasets with naturally paired input–target structures: NL-RX-SYNTH and NL-RX-TURK (Locascio et al., 2016) for

generating regular expressions from descriptions in natural language, GSM8K (Cobbe et al., 2021) for mathematical reasoning, Spider (Yu et al., 2018) for text-to-SQL generation, and NQ-Open (Lee et al., 2019) for open-domain question answering. Following the evaluation protocol of LLM-JEPA, we report exact match accuracy of the generated regular expression for NL-RX-SYNTH and NL-RX-TURK, exact match accuracy of the final answer for GSM8K and NQ-Open, and execution accuracy for Spider. Detailed experimental settings are provided in Appendix A.

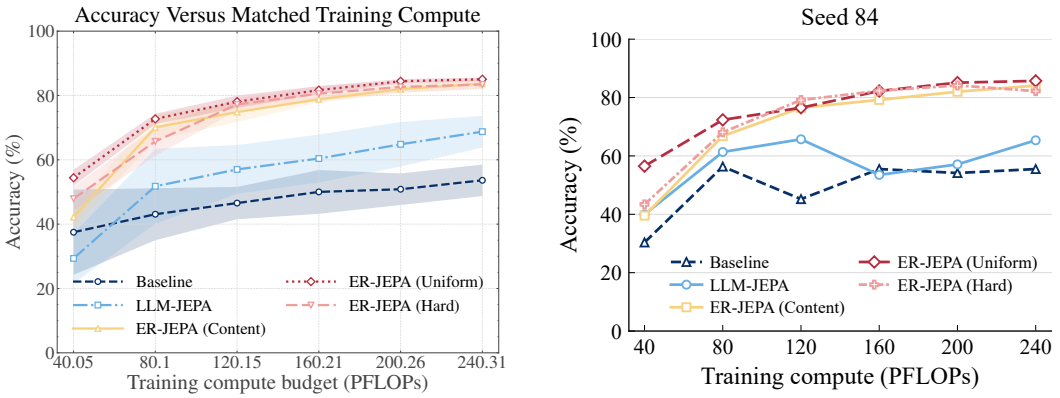
3.2 ER-JEPA Outperforms LLM-JEPA

Episodic replay improves LLM-JEPA across tasks. We first test whether replay provides benefits beyond representation alignment alone. We fine-tune Llama-3.2-1B-Instruct with different training objectives on all five datasets described above. As shown in Figure 1 (left), ER-JEPA achieves higher performance than LLM-JEPA on all five datasets.

The consistent gains show that the benefit of ER-JEPA is not limited to one type of task. The method improves both structured generation tasks, such as regular expression and SQL generation, and more general generation tasks, such as mathematical reasoning and open-domain question answering. These results suggest that episodic replay improves the use of training data in LLM-JEPA. By storing separated traces of past examples and retrieving relevant traces for the current context, ER-JEPA allows the model to reuse useful past experiences during optimization. This mechanism provides additional training signals beyond the current mini-batch and leads to more effective fine-tuning.

ER-JEPA achieves higher accuracy at the same training compute. To test whether the gains persist at the same training compute, we evaluate all methods at the same budgets from 40.05 to 240.31 PFLOPs on SYNTH with Llama-3.2-1B. We also evaluate three replay policies, including content, uniform, and hard replay, to test whether the benefit depends on the replay selection rule. The three policies share the same replay path and differ only in how stored examples are selected.

Figure 4(a) shows that all three replay policies outperform LLM-JEPA at every evaluated compute budget. At 240.31 PFLOPs, content replay reaches $83.65 \pm 1.48\%$, while LLM-JEPA reaches $68.75 \pm 4.92\%$, a gain of 14.90 percentage points. The normalized AUC also increases from $56.61 \pm 3.41\%$ to $73.73 \pm 1.15\%$. The gains across all three rules show that the benefit is not limited to content-based selection. At matched training compute, ER-JEPA achieves higher mean accuracy than LLM-JEPA on SYNTH with all three replay policies.



(a) Five seeds. (b) Seed $= 84$.

Figure 4: **ER-JEPA improves accuracy at matched training compute and reduces overfitting.** (a) All three replay policies outperform LLM-JEPA at every evaluated budget. These gains support the benefit of introducing the replay path. (b) Baseline loses accuracy after an early increase. LLM-JEPA delays this decline but still loses accuracy later. Content and uniform replay maintain their gains, while hard replay shows only a small final decline. These trends suggest that replay mitigates overfitting. They complement the mean curves in Figure 1 (right). Results use Llama-3.2-1B on SYNTH at six compute budgets from 40.05 to 240.31 PFLOPs.

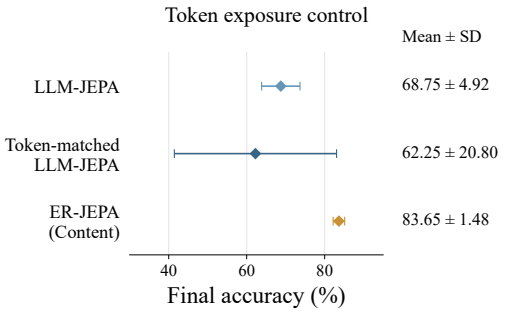


Figure 5: **ER-JEPA achieves higher mean accuracy with replay under matched token exposure.** Results use Llama-3.2-1B on SYNTH over five seeds.

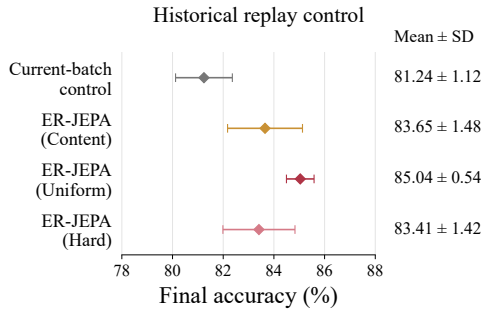


Figure 6: **Historical samples provide additional value.** Results use Llama-3.2-1B on SYNTH over five seeds.

Replay reduces accuracy drops during training. The mean curves in Figure 4(a) rise at every budget for all methods, but individual runs behave differently. Baseline and LLM-JEPA often lose accuracy that they reached earlier in training. These drops happen at different budgets in different seeds, so averaging hides them. Figure 4(b) illustrates this pattern for seed 84. Baseline loses accuracy after an initial increase, and LLM-JEPA shows a decline at a later checkpoint. ER-JEPA shows strong resistance to overfitting under all three replay policies. These results suggest that the replay path mitigates overfitting and helps retain the accuracy gained earlier in training. Appendix Figure 13 shows the other four seeds.

3.3 Understanding the Role of Historical Replay

To analyze these gains, we first compare ER-JEPA with token-matched LLM-JEPA to assess whether additional token exposure alone can reproduce the improvement. We then compare historical replay with a control that applies the same replay loss to current-batch examples, assessing the additional benefit of revisiting past samples. We also examine which errors replay corrects and whether it keeps correct predictions. Appendix B analyzes a single replay update.

Content replay achieves higher mean accuracy than token-matched LLM-JEPA. We first test whether extra token exposure can explain the gain. The token-matched control uses the token budget of content replay and retains the LLM-JEPA objective. Figure 5 shows that content replay reaches $83.65 \pm 1.48\%$, compared with $62.25 \pm 20.80\%$ for token-matched LLM-JEPA. These results indicate that extra token exposure alone does not reproduce the gain.

Historical replay achieves higher mean accuracy than the current-batch control. We next examine whether historical samples provide value beyond an additional loss on the current batch. The current-batch control applies the replay loss to current-batch examples instead of stored examples. Figure 6 shows that all three historical replay policies have higher mean accuracy than this control. Uniform replay reaches $85.04 \pm 0.54\%$, which is 3.80 percentage points (pp) above the control, and it is higher in all five seeds. Content replay reaches 83.65% ($+2.41$ pp), and hard replay reaches 83.41% ($+2.17$ pp). These results are consistent with a benefit from reusing historical samples.

ER-JEPA achieves higher correction rates on examples mispredicted by the baseline. As discussed in the Introduction, a model can align well with its target representation and still predict the wrong answer, sometimes even losing ground it had already covered. This raises a natural question: **when Baseline errs, can LLM-JEPA and ER-JEPA fix its mistakes?** To find out, we test both models on the SYNTH inputs where Baseline fails at the final matched compute point, checking whether each model recovers the exact target expression. We sort these failures by length relative to the target, including over-generation (extra regex tokens), under-generation (missing tokens), and same-length mismatch (correct length, wrong expression). Table 1 shows that ER-JEPA achieves a higher mean correction rate in all three categories. The largest gain occurs for over-generation, where the rate rises from 53.07% with LLM-JEPA to 84.51% with ER-JEPA. Complete correction counts

Table 1: **ER-JEPA achieves higher mean correction rates across all three Baseline error categories.** Results use Llama-3.2-1B on SYNTH at 240.309 PFLOPs. Values are means ± one sample standard deviation over five seeds. Correction rates are computed separately for each seed before averaging.

Baseline error category	Baseline errors	Method	Corrected	Correction rate (%)
Over-generation	718.4 ± 112.5	LLM-JEPA	378.8 ± 64.6	53.07 ± 7.92
		ER-JEPA	610.8 ± 126.1	84.51 ± 4.70
Under-generation	2.6 ± 1.5	LLM-JEPA	0.6 ± 0.5	24.00 ± 25.10
		ER-JEPA	1.2 ± 0.8	54.67 ± 44.07
Same-length mismatch	168.4 ± 10.9	LLM-JEPA	34.8 ± 5.8	20.58 ± 2.35
		ER-JEPA	42.4 ± 7.0	25.07 ± 2.62

and rates for each seed are reported in Appendix Table 3. Appendix Figure 10 further compares the two methods on inputs that LLM-JEPA predicts incorrectly.

To further examine whether replay helps correct errors and retain correct predictions during training, we track predictions on the same test inputs across successive checkpoints. Figure 3 shows more correct predictions for ER-JEPA in (f) than for LLM-JEPA in (c) at every evaluated checkpoint. Figure 7(b) shows a higher mean correction rate for ER-JEPA than for LLM-JEPA. In Figure 7(c), the mean rate of correct predictions becoming incorrect decreases from 8.85% with LLM-JEPA to 6.65% with ER-JEPA. These results suggest that replay supports both error correction and the retention of correct predictions. Example inputs, targets, and predictions from the three methods are shown in Appendix Figure 11.

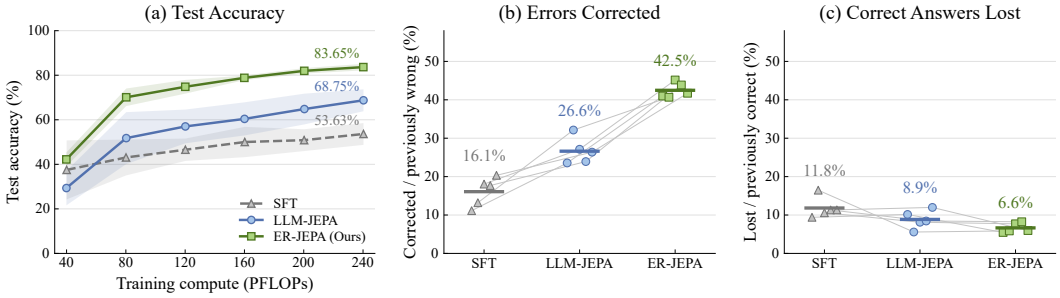


Figure 7: **ER-JEPA achieves higher accuracy and corrects more errors during training.** (a) Test accuracy at comparable training compute for SFT, LLM-JEPA, and ER-JEPA. (b) The fraction of wrong predictions that become correct at the next checkpoint. (c) The fraction of correct predictions that become wrong at the next checkpoint. Compared with LLM-JEPA, ER-JEPA improves error correction across all five seeds, while the reduction in lost correct predictions is smaller and varies across seeds. Results use Llama-3.2-1B on the same 2,000 SYNTH test examples.

3.4 Fine-grained Analysis of Prediction Quality

We next examine where the accuracy gains occur within the SYNTH test set, focusing on the structure and length of the target regular expressions.

Accuracy gains span multiple forms of symbolic composition. To examine how these gains relate to symbolic structure, we analyze the SYNTH test set using six structural subsets defined by the target regular expressions, including complement, intersection, alternation, quantifiers, boundaries and anchors, and character classes. Detailed definitions and classification rules are provided in Appendix Table 2. Figure 8 shows that ER-JEPA achieves higher mean exact match accuracy than LLM-JEPA in all six subsets. The largest gains occur for expressions containing quantifiers, intersections, and character classes, with improvements of 16.90, 16.07, and 15.42 percentage points, respectively. These structures encode repetition, conjunction, and restrictions on allowed characters, all of which

must be preserved when translating a natural language description into a regular expression. The gains therefore span several forms of constraint composition. This pattern is consistent with replay reinforcing the mapping from linguistic constraints to their symbolic implementation.

Replay reduces sensitivity to expression length. We next examine whether this advantage persists for longer expressions. Target regex length serves as a proxy for complexity. We divide the SYNTH test set into four groups: ≤ 10, 11–15, 16–20, and > 20 regex tokens. Figure 9 shows that Regular is highly sensitive to expression length. Its mean accuracy drops from 83.64% in the shortest group to 33.53% in the longest group, a decrease of 50.11 percentage points. LLM-JEPA reduces this gap, but its accuracy still falls from 85.15% to 56.76%. ER-JEPA achieves the highest accuracy in all four groups. Its accuracy decreases from 89.80% to 78.53%, a drop of 11.27 percentage points, compared with 28.39 for LLM-JEPA. These results suggest that episodic replay helps the model generate longer symbolic expressions more reliably.

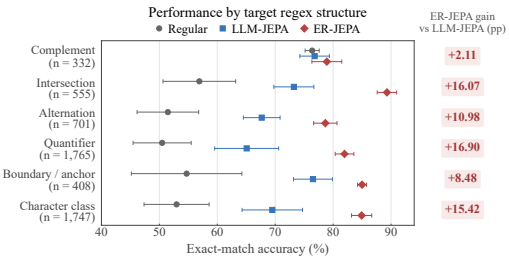


Figure 8: **ER-JEPA achieves higher prediction accuracy across all six evaluated regex structures with episodic replay.** Compared with LLM-JEPA, ER-JEPA improves accuracy by 16.90, 16.07, and 15.42 percentage points for expressions containing quantifiers, intersections, and character classes, respectively.

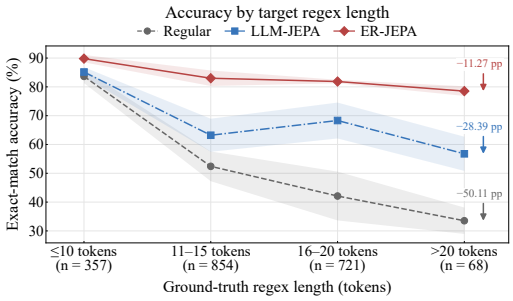


Figure 9: **ER-JEPA maintains higher accuracy on longer regular expressions with episodic replay.** ER-JEPA leads in all four length groups and shows a smaller decline from the shortest to the longest group than Regular and LLM-JEPA. Lines show mean exact match accuracy over five seeds, and shaded bands indicate one standard deviation.

Training gains come without additional inference cost. At inference, ER-JEPA uses the same generation procedure as LLM-JEPA. We remove the episodic memory and replay path after training. Following LLM-JEPA, we use only the fine-tuned language model for inference. Replay therefore adds no overhead to inference time and requires no additional computation or memory during inference.

Ablation studies on memory addressing, memory capacity and JEPA objective hyperparameters are provided in Appendix E.

4 Related Work

Joint-Embedding Predictive Architectures (JEPAs) learn representations by predicting target embeddings from context. LeCun (2022) proposed JEPA as a framework for learning world models that predict abstract states without reconstructing every input detail. A related approach, data2vec (Baevski et al., 2022), predicts contextualized representations from masked inputs using the same learning method across speech, vision, and language.

In vision, I-JEPA (Assran et al., 2023) predicts the representations of target image regions from a context region, learning semantic features without relying on hand-crafted augmentations. V-JEPA (Bardes et al., 2024) extends latent prediction to video and learns representations that support downstream tasks with a frozen encoder. Image World Models (Garrido et al., 2024) broaden the prediction task beyond masked regions to include the effects of photometric transformations. To better understand latent prediction, Littwin et al. (2024) analyze its learning dynamics in deep linear models. Their analysis identifies an implicit bias toward features with high regression coefficients.

Recent work also applies JEPA to language and vision-language learning. VL-JEPA (Chen et al., 2026) predicts continuous text embeddings for vision-language tasks and uses a decoder when text output is needed. For language models, LLM-JEPA (Huang et al., 2026) combines next-token prediction with a JEPA loss over paired textual views. This joint objective encourages semantic structure in the learned representations while preserving generative capabilities. Our work builds on LLM-JEPA by adding episodic replay, which reuses past training examples to provide additional supervision for token prediction and representation alignment.

5 Conclusion

We propose ER-JEPA, which adds an episodic replay path to LLM-JEPA. The replay path revisits stored training pairs with the current model and provides additional learning signals. At matched training compute, ER-JEPA outperforms LLM-JEPA with all three selection rules. Replay also corrects more errors during training and avoids most large accuracy drops. These results show that revisiting past examples is a simple and effective way to improve LLM-JEPA.

We view the current implementation of ER-JEPA as a first step toward introducing episodic replay into the JEPA framework, and future work could further refine and extend this direction. In particular, since the overall training objective is formulated as a linear combination of the new data JEPA loss and the replay loss, the relative weighting between different loss terms, such as the replay weight β, currently needs to be selected through grid search. This tuning process introduces additional computational cost and may motivate future work on adaptive weighting strategies. Moreover, ER-JEPA inherits the need of LLM-JEPA for datasets with paired views, such as text and code. We hope future work will explore more efficient and general data augmentation strategies for constructing informative paired views, thereby broadening the application regime of ER-JEPA.

AI use statement

In this work, we used generative AI tools to check English grammar and spelling. We also used generative AI tools to assist with debugging code errors and resolving implementation issues to ensure that the code executed correctly. All AI-assisted edits and code modifications were reviewed and verified by the authors.

Ethics statement

This work does not involve human subjects or personally identifiable information. The datasets used in our experiments are publicly available. We do not foresee any direct ethical concerns or negative societal impacts arising from this work.

Reproducibility statement

To ensure reproducibility, we provide detailed descriptions of the experimental settings, implementation details, hyperparameters, datasets, and evaluation protocols in the main text and appendix. These details are provided to facilitate the reproduction of our experimental results.

References

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15619–15629, 2023.

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.

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*, pp. 1298–1312. PMLR, 2022.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Delong Chen, Mustafa Shukor, Théo Moutakanni, Willy Chung, Lei Yu, Tejaswi Kasarla, Allen Bolourchi, Yann LeCun, and Pascale Fung. VL-JEPA: Joint embedding predictive architecture for vision-language. In *The Fourteenth International Conference on Learning Representations*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, John Schulman, Jacob Hilton, Melanie Knight, Adrian Weller, Dario Amodei, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Junze Deng, Qinhang Wu, Peizhong Ju, Sen Lin, Yingbin Liang, and Ness Shroff. Unlocking the power of rehearsal in continual learning: A theoretical perspective, 2025. URL https://arxiv.org/abs/2506.00205.

Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. Learning and leveraging world models in visual representation learning. *arXiv preprint arXiv:2403.00504*, 2024.

Linyuan Gong, Sida Wang, Mostafa Elhoushi, and Alvin Cheung. Evaluation of llms on syntax-aware code fill-in-the-middle tasks. *arXiv preprint arXiv:2403.04814*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Maggie Ziyu Huan, Yuetai Li, Tuney Zheng, XiaoYu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. *ArXiv*, abs/2507.00432, 2025.

Hai Huang, Yann LeCun, and Randall Balestriero. LLM-JEPA: Large language models meet joint embedding predictive architectures. In *The Fourteenth International Conference on Learning Representations*, 2026.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal, 2024. URL https://arxiv.org/abs/2403.01244.

Charles Jin and Martin Rinard. Emergent representations of program semantics in language models trained on programs. In *Forty-first International Conference on Machine Learning*, 2024.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7): 512–534, 2016.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1612.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *ArXiv*, abs/2406.11939, 2024.

Xianhang Li, Haoqin Tu, Mude Hui, Zeyu Wang, Bingchen Zhao, Junfei Xiao, Sucheng Ren, Jieru Mei, Qing Liu, Huangjie Zheng, Yuyin Zhou, and Cihang Xie. What if we recaption billions of web images with LLaMA-3? In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35957–35976. PMLR, 13–19 Jul 2025.

Etai Littwin, Omid Saremi, Madhu Advani, Vimal Thilak, Preetum Nakkiran, Chen Huang, and Joshua Susskind. How jepa avoids noisy features: The implicit bias of deep linear self distillation networks. *Advances in Neural Information Processing Systems*, 37:91300–91336, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1918–1923, 2016.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey, 2024. URL https://arxiv.org/abs/2404.16789.

Song Wang, Zihan Chen, Chengshuai Shi, Cong Shen, and Jundong Li. Mixture of demonstrations for in-context learning. *Advances in Neural Information Processing Systems*, 37:88091–88116, 2024.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 36:78808–78823, 2023.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

A Detailed Experimental Setup

The experimental design follows the LLM-JEPA protocol. This design allows us to isolate the effect of the proposed episodic replay pathway, rather than introducing confounding changes in data, model scale, or training configuration. All experiments are conducted under the same LoRA fine-tuning setting with rank 512. Under this setting, we compare three fine-tuning objectives: the standard supervised fine-tuning objective, the LLM-JEPA objective, and the proposed ER-JEPA objective.

For each combination of model and dataset, we use a learning rate of $1e{-}5$ and train for 6 epochs. We adopt this LoRA configuration because the LLM-JEPA study found that it outperforms fine-tuning all model parameters under the same setting. The standard supervised objective, LLM-JEPA, and ER-JEPA all share this same configuration, learning rate, and training schedule. This helps isolate the effects of the training objective and the episodic-replay extension on performance.

All experiments are repeated with five fixed random seeds, $\{82, 23, 37, 84, 4\}$. We report the mean performance across seeds. Statistical significance is assessed using paired one-tailed t-tests over seed-matched runs. All training runs are conducted on a server equipped with four NVIDIA A800 GPUs.

B How Historical Replay Can Provide Additional Correction Directions

LLM-JEPA combines token generation with representation alignment, but both losses directly supervise only the current batch. We give a conditional explanation of how replay changes one gradient-descent step. Replay can supply a descent direction for the selected historical loss that current-batch gradients do not provide.

The local limitation of the LLM-JEPA objective. Let ω collect the trainable LoRA coordinates, counting shared parameters once. For a nonempty current batch $\mathcal{B}$, let $\mathcal{T}_\mathcal{B}$ contain its supervised-token positions and $Z_\mathcal{B} = |\mathcal{T}_\mathcal{B}| > 0$. Write $\pi_q = \mathrm{softmax}(f_q(\omega))$ for the token probabilities, y_q for the target token, and $e_i = p_i/\|p_i\|_2 - z_i^a/\|z_i^a\|_2$ for the alignment residual. Equation (3) gives the LLM-JEPA objective

$$L_\mathcal{B}(\omega) = \underbrace{\frac{\gamma}{Z_\mathcal{B}} \sum_{q \in \mathcal{T}_\mathcal{B}} -\log \pi_{q,y_q}}_{\text{token generation}} + \underbrace{\frac{\lambda}{2|\mathcal{B}|} \sum_{i \in \mathcal{B}} \|e_i\|_2^2}_{\text{representation alignment}}, \tag{7}$$

where $\gamma, \lambda > 0$ are fixed. Fix a parameter state ω_0. Hold the examples, sequences, masks, and forward-pass randomness fixed. Repeated occurrences use identical views and forward realizations. For all current and replay examples, assume finite logits, twice continuously differentiable maps, and representation norms above the normalization floor near ω_0.

At ω_0, let $J_q = \partial f_q/\partial\omega$ and $A_i = \partial e_i/\partial\omega$, differentiating both representation branches. The two losses supply the gradient

$$g_\mathcal{B} = \nabla L_\mathcal{B}(\omega_0) = \frac{\gamma}{Z_\mathcal{B}} \sum_{q \in \mathcal{T}_\mathcal{B}} J_q^\top(\pi_q - \mathbf{e}_{y_q}) + \frac{\lambda}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} A_i^\top e_i, \tag{8}$$

where $\mathbf{e}_{y_q}$ is the target-token basis vector. Neither term provides a component along directions in

$$\mathcal{K} = \{v : \ J_q v \in \mathrm{span}\{\mathbf{1}\}\ \forall q \in \mathcal{T}_\mathcal{B}, \quad A_i v = 0\ \forall i \in \mathcal{B}\}. \tag{9}$$

These directions preserve current token probabilities and alignment residuals to first order. They may still change losses on historical examples. Let P be the orthogonal projector onto $\mathcal{K}$. The subspace $\mathcal{K}$ may contain only the zero vector.

ER-JEPA adds $\beta L_\mathcal{R}$, with fixed $\beta > 0$, where $L_\mathcal{R}$ applies the same joint objective to a nonempty replay multiset with supervised tokens, as in Eq. (6). The analysis also allows replay to overlap $\mathcal{B}$. Occurrences are counted separately and representations are recomputed. Hold the selected multiset $\mathcal{R}$ fixed throughout the comparison. Write $g_\mathcal{R} = \nabla L_\mathcal{R}(\omega_0)$.

Proposition B.1 (Conditional local effect of historical replay)**.** *At the fixed state above, $Pg_\mathcal{B} = 0$. The same holds for a gradient obtained by repeating current examples or reweighting them with fixed weights. Define $d = -Pg_\mathcal{R}$. If $Pg_\mathcal{R} \neq 0$, then*

$$g_\mathcal{B}^\top d = 0, \qquad g_\mathcal{R}^\top d = -\|Pg_\mathcal{R}\|_2^2 < 0. \tag{10}$$

For the gradient-descent updates $\omega_\mathrm{C} = \omega_0 - \eta g_\mathcal{B}$ and $\omega_\mathrm{E} = \omega_0 - \eta(g_\mathcal{B} + \beta g_\mathcal{R})$, we have

$$P(\omega_\mathrm{E} - \omega_0) = \eta\beta d, \qquad P(\omega_\mathrm{C} - \omega_0) = 0. \tag{11}$$

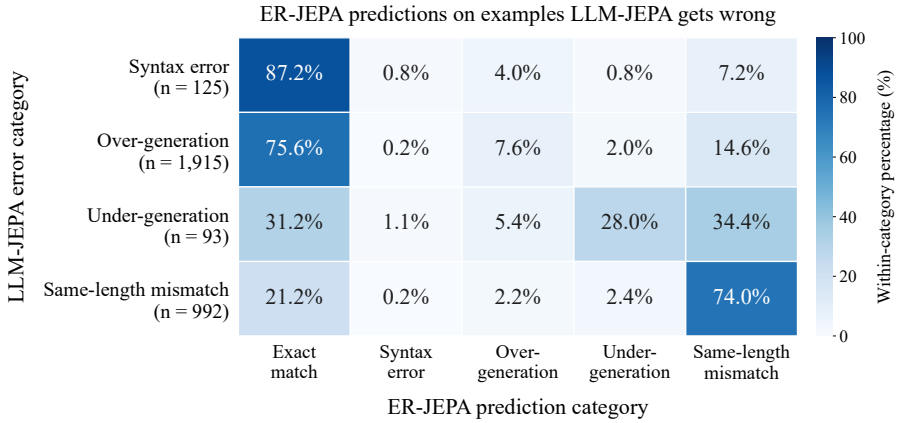


Figure 10: ER-JEPA predictions on examples that LLM-JEPA predicts incorrectly. Rows indicate LLM-JEPA error categories, and columns indicate ER-JEPA prediction categories for the same input and seed. The row total n counts LLM-JEPA errors pooled across five seeds. Each cell shows its count divided by n, expressed as a percentage to one decimal place. Results use Llama-3.2-1B on 2,000 SYNTH test examples per seed at a matched training compute budget of 240.309 PFLOPs.

Their replay losses satisfy

$$L_{\mathcal{R}}(\omega_{\mathrm{E}}) - L_{\mathcal{R}}(\omega_{\mathrm{C}}) = -\eta\beta\left(\|Pg_{\mathcal{R}}\|_2^2 + \|(I-P)g_{\mathcal{R}}\|_2^2\right) + O(\eta^2) \tag{12}$$

as $\eta \to 0$. If $g_{\mathcal{R}} \neq 0$, then $L_{\mathcal{R}}(\omega_{\mathrm{E}}) < L_{\mathcal{R}}(\omega_{\mathrm{C}})$ for all sufficiently small $\eta > 0$. This loss comparison does not require $Pg_{\mathcal{R}} \neq 0$.

Proof. Let $v \in \mathcal{K}$. For each supervised token, write $J_q v = c_q \mathbf{1}$. Then

$$(J_q v)^\top(\pi_q - \mathbf{e}_{y_q}) = c_q(1-1) = 0, \qquad (A_i v)^\top e_i = 0.$$

Thus every token and alignment contribution is orthogonal to $\mathcal{K}$. Equation (8) gives $Pg_{\mathcal{B}} = 0$. Repeating these contributions or changing their fixed weights preserves this property.

Since $P = P^\top = P^2$, we have $g_{\mathcal{B}}^\top d = 0$ and $g_{\mathcal{R}}^\top d = -g_{\mathcal{R}}^\top P g_{\mathcal{R}} = -\|Pg_{\mathcal{R}}\|_2^2$. This proves Eq. (10) when $Pg_{\mathcal{R}} \neq 0$. Applying P to the two updates gives Eq. (11).

Taylor expansion at ω_0 gives

$$L_{\mathcal{R}}(\omega_{\mathrm{C}}) = L_{\mathcal{R}}(\omega_0) - \eta g_{\mathcal{R}}^\top g_{\mathcal{B}} + O(\eta^2),$$
$$L_{\mathcal{R}}(\omega_{\mathrm{E}}) = L_{\mathcal{R}}(\omega_0) - \eta g_{\mathcal{R}}^\top g_{\mathcal{B}} - \eta\beta\|g_{\mathcal{R}}\|_2^2 + O(\eta^2).$$

Subtracting cancels the shared term $-\eta g_{\mathcal{R}}^\top g_{\mathcal{B}}$. The orthogonal decomposition $g_{\mathcal{R}} = Pg_{\mathcal{R}} + (I-P)g_{\mathcal{R}}$ then gives Eq. (12). The smoothness assumptions bound the absolute remainder by $C\eta^2$ for some fixed $C > 0$ and all sufficiently small η. If $g_{\mathcal{R}} \neq 0$, choosing η small enough that $C\eta < \beta\|g_{\mathcal{R}}\|_2^2$ makes the difference strictly negative.

The term $\|Pg_{\mathcal{R}}\|_2^2$ is the contribution of the additional directions to the first-order loss difference. It is positive only when $Pg_{\mathcal{R}} \neq 0$. The projector is an analytical device; ER-JEPA uses the full replay gradient. The loss comparison also applies to ordinary replay or any added smooth loss with a nonzero gradient. This analysis characterizes a conditional local effect on the replay loss; its implications for predictive performance are evaluated empirically.

C Additional Empirical Analyses

C.1 Regex Structural Subsets

Classification follows the regex grammar. Escaped literal symbols and symbols inside character classes are not counted as separate operators. Each example is assigned to every applicable subset, so the subsets may overlap.

Table 2: **Definitions of the six regex structural subsets.** Each subset contains examples whose target regular expressions include the specified feature.

Subset	Classification rule	Meaning
Complement	Contains the operator `~`.	Excludes strings matched by a subexpression.
Intersection	Contains the operator `&`.	Requires multiple patterns to hold simultaneously.
Alternation	Contains the operator `|`.	Allows a choice between alternative patterns.
Quantifier	Contains `*`, `+`, `?`, or a brace quantifier such as `{m,n}`.	Specifies repetition counts or optionality.
Boundary / anchor	Contains `\b`, `\B`, `^`, or `$`.	Constrains matching positions relative to word boundaries or string endpoints.
Character class	Contains a bracketed character class, such as `[a-z]`.	Specifies a set of allowed characters.

Table 3: **Correction counts and rates for each seed on the same Baseline errors.** Both methods use the same inputs and seed within each row. Corrected subsets may overlap between methods. The counts and rates in Table 1 are summarized across these five seeds.

Seed	Baseline errors	LLM-JEPA corrected	Correction rate (%)	ER-JEPA corrected	Correction rate (%)
(a) Over-generation					
4	901	448	49.72	807	89.57
23	697	435	62.41	583	83.64
37	705	375	53.19	608	86.24
82	591	344	58.21	455	76.99
84	698	292	41.83	601	86.10
(b) Under-generation					
4	1	0	0.00	1	100.00
23	5	1	20.00	2	40.00
37	3	0	0.00	1	33.33
82	2	1	50.00	2	100.00
84	2	1	50.00	0	0.00
(c) Same-length mismatch					
4	161	29	18.01	36	22.36
23	156	31	19.87	38	24.36
37	173	41	23.70	46	26.59
82	184	41	22.28	53	28.80
84	168	32	19.05	39	23.21

C.2 Error Correction Across Seeds

Table 3 reports the individual results summarized in Table 1. For each seed, we identify Baseline errors among the same 2,000 SYNTH test inputs and evaluate both methods on those inputs. Syntax errors are excluded from the three length-based categories. A prediction counts as corrected only when it exactly matches the target. Each correction rate uses the Baseline error count in its row as the denominator. ER-JEPA uses content replay.

Table 4: Some regular expressions generated by Llama-3.2-1B-Instruct after fine-tuning with $\mathcal{L}_{\text{LLM}}$, $\mathcal{L}_{\text{LLM-JEPA}}$, and $\mathcal{L}_{\text{ER-JEPA}}$ losses. Color code: wrong, extra, missing.

Model / target	Regular expression
Input: lines not having the string "dog" followed by a number, 3 or more times	
Ground truth	˜((dog.*[0-9].*){3,})
$\mathcal{L}_{\text{LLM}}$	˜((dog.*[0-9].*){3,})
$\mathcal{L}_{\text{LLM-JEPA}}$	˜((dog.*[0-9].*){3,})
$\mathcal{L}_{\text{ER-JEPA}}$	˜((dog.*[0-9].*){3,})
Input: lines containing ending with a vowel, zero or more times	
Ground truth	.*(.*)(([AEIOUaeiou])*).*
$\mathcal{L}_{\text{LLM}}$	.*(.*)(([AEIOUaeiou])*).* .*.*
$\mathcal{L}_{\text{LLM-JEPA}}$	.*(.*)(([AEIOUaeiou])*).* .*
$\mathcal{L}_{\text{ER-JEPA}}$	.*(.*)(([AEIOUaeiou])*).*
Input: lines with a number or a character before a vowel	
Ground truth	(([0-9])|(.)).*([AEIOUaeiou]).*
$\mathcal{L}_{\text{LLM}}$	(([0-9])|(.)).*([AEIOUaeiou]).* .*
$\mathcal{L}_{\text{LLM-JEPA}}$	(([0-9])|(.)).*([AEIOUaeiou]).*
$\mathcal{L}_{\text{ER-JEPA}}$	(([0-9])|(.)).*([AEIOUaeiou]).*
Input: lines ending with containing the string "dog", 7 or more times	
Ground truth	((.*)(.*dog.*)){7,}
$\mathcal{L}_{\text{LLM}}$	((.*)(.* (dog.*)){7,} .*)*
$\mathcal{L}_{\text{LLM-JEPA}}$	((.*) ((.*dog.*)){7,})
$\mathcal{L}_{\text{ER-JEPA}}$	(.*) ((.*dog.*)7,))

C.3 Examples Selected by a Fixed Rule

Figure 11 illustrates Baseline errors corrected by ER-JEPA, grouped by the Baseline error taxonomy. The selection contains exactly two LLM-JEPA-correct cases and seven LLM-JEPA errors; all three methods are compared on the same input and seed for each example. The full evaluation, using 2,000 test inputs for each of five seeds, gives mean accuracies of 53.63%, 68.75%, and 83.65% for Baseline, LLM-JEPA, and ER-JEPA, respectively. Figure 10 reports ER-JEPA predictions on inputs that LLM-JEPA predicts incorrectly.

C.4 Individual Seed Accuracy at Matched PFLOPs

Figure 13 complements the seed 84 results in Figure 4(b). Across all five seeds, Baseline has a local accuracy decline in four seeds, and LLM-JEPA has a decline in all five. Such declines occur less often with replay. These results suggest that replay mitigates overfitting across individual runs.

D ER-JEPA Implementation Details

D.1 Episodic Memory and Retrieval

ER-JEPA maintains a fixed-capacity episodic memory $\mathcal{M}_t$. Each active slot stores a source–target token pair, a sparse address, and memory-management metadata. We denote these entries by

$$m_n = \big(s_n, \tau_n^u, \tau_n^a, \sigma_n, r_n\big), \qquad n \in \mathcal{A}_t, \tag{13}$$

Model performance on regex generation -SYNTH

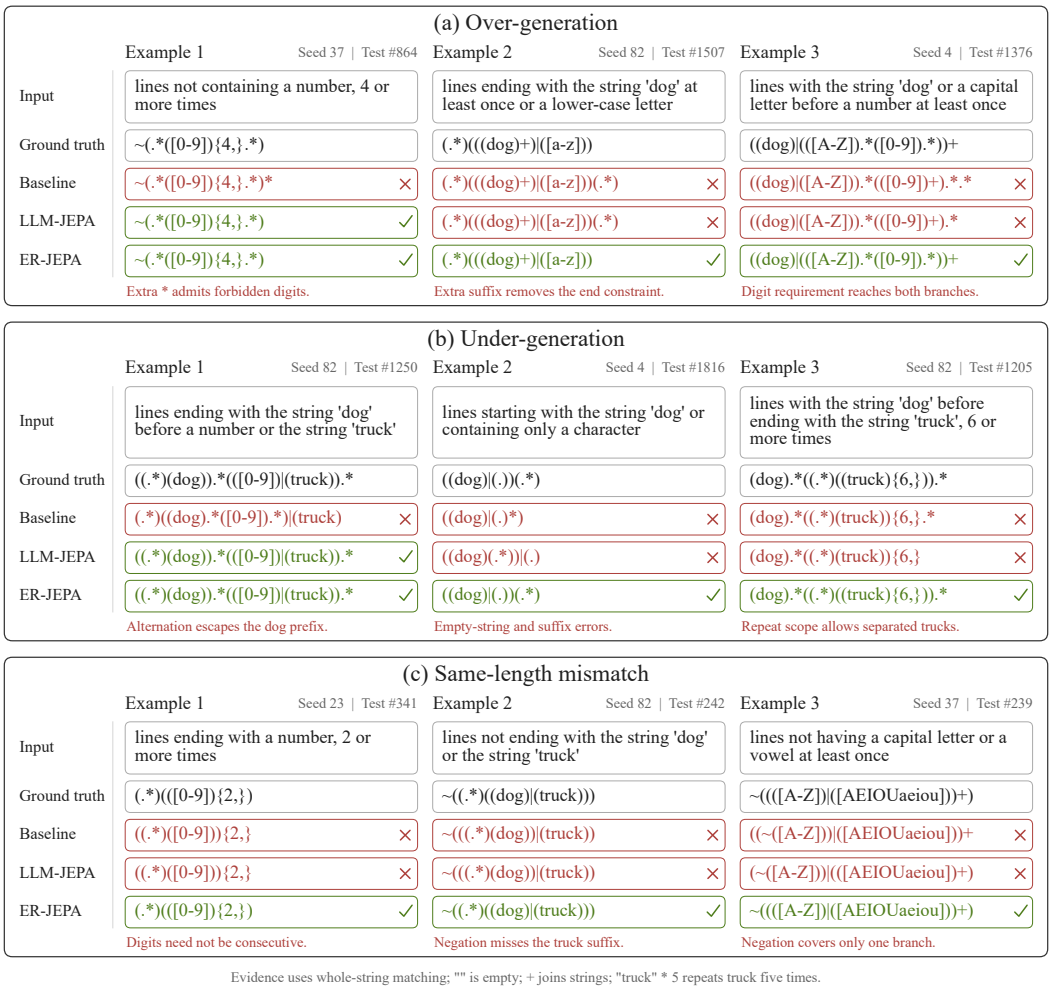


Figure 11: Example predictions from Baseline, LLM-JEPA, and ER-JEPA on the SYNTH dataset.

NQ-OPEN prediction examples

Question	Reference	Baseline	LLM-JEPA	ER-JEPA
who did the jets play in super bowl 3	Baltimore Colts	Pittsburgh Steelers	Pittsburgh Steelers	Baltimore Colts
medical term for the cause of a disease	etiology	etiology	disease process	etiology
how many sisters does joey have in friends	seven	two	two	two
who played bat masterson in the tv show	Gene Barry	Gene Kelly	Gene Barry	Gene Barry
when did the war of the roses start	22 May 1455; 1455	1485	1485	1485

Figure 12: Example predictions from Baseline, LLM-JEPA, and ER-JEPA on the NQ-OPEN dataset. Correct and incorrect answers are shown in dark green and dark red, respectively. For each prediction, only the first semicolon-separated answer is displayed.

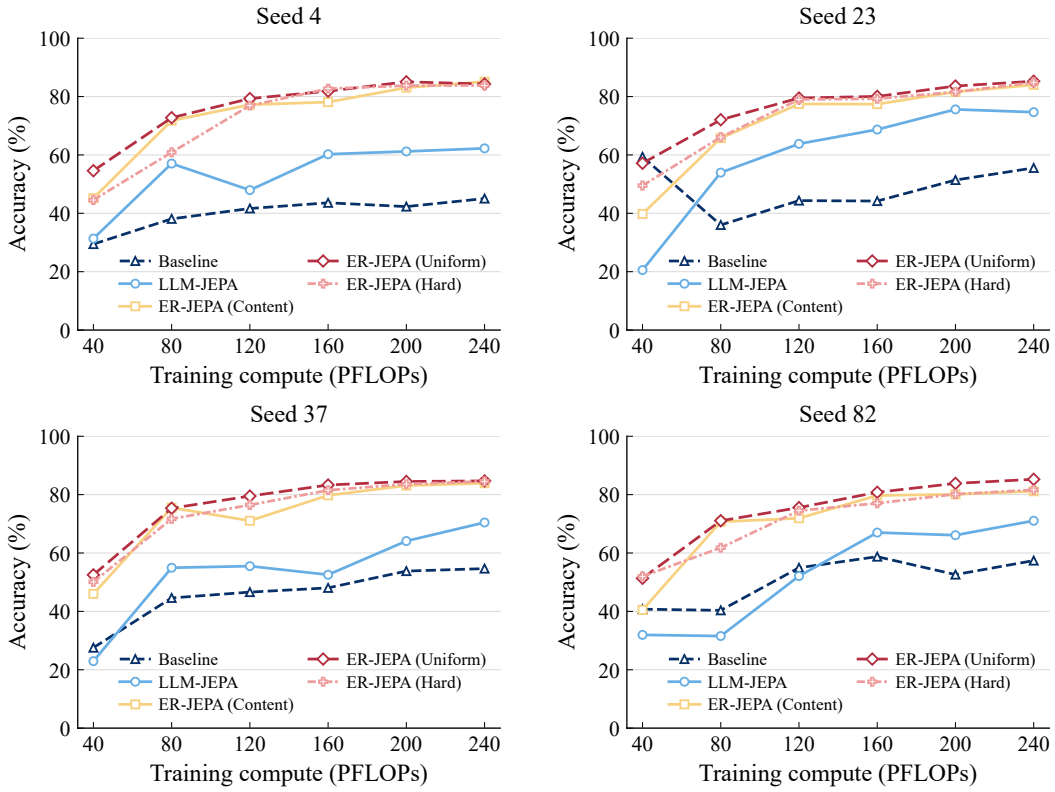


Figure 13: Raw accuracy at six shared target PFLOPs for seeds 4, 23, 37, and 82. Each panel shows one seed. Seed 84 is shown in Figure 4(b). The panels do not show means or error bars.

where $\mathcal{A}_t$ is the set of active slots, s_n is the address stored at insertion, σ_n is the running JEPA error, and r_n is the replay count. Examples in the current mini-batch are excluded from replay. **Writing, replacement**, and score **updates** follow Appendix D.2.

Content-based retrieval uses the current source representation $z_j^u = \mathrm{Enc}_\theta(x_j^u)$ as a cue. Let Φ be the frozen random projection with sparse Top-K selection defined in Appendix D.3.1. We normalize the cue and stored addresses as

$$\hat{s}_j^{\mathrm{cue}} = \frac{\Phi(z_j^u)}{\|\Phi(z_j^u)\|_2}, \qquad \hat{s}_n = \frac{s_n}{\|s_n\|_2}. \tag{14}$$

For each cue, we retrieve up to κ slots with the highest cosine similarities,

$$\mathcal{K}_{t,j} = \mathrm{Top}_{\min(\kappa,N_t)}\left(\left\{\left(n, \langle \hat{s}_j^{\mathrm{cue}}, \hat{s}_n\rangle\right) \mid n \in \mathcal{A}_t\right\}\right), \qquad N_t = |\mathcal{A}_t|. \tag{15}$$

Here Top returns slot indices in descending similarity order. We concatenate the lists in batch order, keep the first occurrence of each slot, and retain at most R entries, where R is the replay budget. If fewer than R distinct slots are retrieved, we replay only those slots. Let $\mathcal{S}_t = (n_{t,1}, \ldots, n_{t,R_t})$ denote the selected slot list, where $R_t = |\mathcal{S}_t| \le R$. The token pair for replay entry i is

$$(\widetilde{\tau}_i^u, \widetilde{\tau}_i^a) = (\tau^u_{n_{t,i}}, \tau^a_{n_{t,i}}), \qquad i = 1, \ldots, R_t. \tag{16}$$

The selection rules for all three replay strategies are given in Appendix D.3.

D.2 Shared Episodic Memory

Each new slot is initialized with

$$\sigma_n = d(p_n, z_n^a), \qquad r_n = 0, \tag{17}$$

where p_n is the predicted target representation and z_n^a is the target readout. When the store is full, the slot with the lowest replay-adjusted significance is evicted,

$$n^\star = \arg\min_{n \in \mathcal{A}_t} \frac{\sigma_n}{1 + r_n}. \tag{18}$$

After a replay forward pass, the significance of each selected slot is updated as

$$\sigma_n \leftarrow (1-\eta)\sigma_n + \eta\, d(p_n^{\mathrm{rep}}, z_n^{a,\mathrm{rep}}), \qquad r_n \leftarrow r_n + 1, \qquad \eta \in (0,1), \tag{19}$$

where η is the update rate and $p_n^{\mathrm{rep}}, z_n^{a,\mathrm{rep}}$ are recomputed during replay. Unselected slots retain their scores. These rules are shared by all three selection policies.

D.3 Replay Selection Policies

The content-based, uniform, and hard replay strategies differ only in how replay slots are selected. All use the memory rules in Appendix D.2 and the replay loss in Eq. (6). Let $\mathcal{A}_t$ be the active slots, $N_t = |\mathcal{A}_t|$, and R the replay budget. A policy $\pi \in \{\mathrm{content}, \mathrm{uniform}, \mathrm{hard}\}$ returns an ordered slot list $\mathcal{S}_t^\pi$, and $\mathcal{S}_t = \mathcal{S}_t^\pi$. Repeated slots contribute once per occurrence. All policies return an empty selection when $N_t = 0$.

D.3.1 Content Replay

Content replay retrieves stored examples using current source representations z^u. Representations are mapped to sparse addresses with a frozen random projection and Top-K selection,

$$\begin{aligned} W_{\mathrm{PS}} \in \mathbb{R}^{S\times H}, \qquad & [W_{\mathrm{PS}}]_{\ell k} \sim \mathcal{N}(0, 1/H), \\ [\Phi(z)]_\ell = \mathrm{stop\text{-}grad}(\mathbb{I}[\ell \in \mathcal{I}_K(W_{\mathrm{PS}} z)](W_{\mathrm{PS}} z)_\ell)&. \end{aligned} \tag{20}$$

H and $S > H$ are the representation and address dimensions, and $\mathcal{I}_K(h)$ contains the indices of the K largest-magnitude entries of h, retaining their sign and magnitude. The projection is fixed throughout training.

For current example j, the cue is $\hat{s}_j^{\mathrm{cue}} = \Phi(z_j^u)/\|\Phi(z_j^u)\|_2$. Let s_n be the address stored at insertion and $\hat{s}_n = s_n/\|s_n\|_2$. The $\min(\kappa, N_t)$ slots with the highest cosine similarity are retrieved,

$$\mathcal{K}_{t,j} = \mathrm{Top}_{\min(\kappa,N_t)}\left(\left\{\left(n, \langle \hat{s}_j^{\mathrm{cue}}, \hat{s}_n\rangle\right) \mid n \in \mathcal{A}_t\right\}\right), \tag{21}$$

returned in descending score order. Lists from all examples in the batch are concatenated in batch order, deduplicated, and truncated to R entries,

$$\mathcal{C}_t = \mathrm{Prefix}_R\big(\mathrm{Unique}\big(\mathcal{K}_{t,1} \| \cdots \| \mathcal{K}_{t,|\mathcal{B}_t|}\big)\big), \qquad \mathcal{S}_t^{\mathrm{content}} = \mathcal{C}_t, \tag{22}$$

where Unique keeps the first occurrence of each slot and Prefix_R keeps at most R entries. The default policy uses the resulting list without repetition. For controls with a fixed replay count, we instead use $\mathcal{S}_t^{\mathrm{content}} = \mathrm{Fill}_R(\mathcal{C}_t)$, where Fill_R repeats a shorter nonempty list until it reaches R entries; an empty list remains empty. Cosine similarity is used only for retrieval and contributes no training loss.

D.3.2 Uniform Replay

Uniform replay samples stored examples with equal probability, without using cue similarity or significance. When $R \le N_t$, R slots are sampled without replacement: for any subset $\mathcal{U} \subseteq \mathcal{A}_t$ of size R,

$$\Pr\big(\mathrm{set}(\mathcal{S}_t^{\mathrm{uniform}}) = \mathcal{U} \mid \mathcal{M}_t\big) = \binom{N_t}{R}^{-1}, \qquad \Pr(n \in \mathcal{S}_t^{\mathrm{uniform}} \mid \mathcal{M}_t) = \frac{R}{N_t}. \tag{23}$$

When $0 < N_t < R$, R slots are drawn independently with replacement, each with probability $1/N_t$.

Uniform replay estimates the mean JEPA error over active memory. Let $d_n(t)$ be the error of stored example n under the current model. For a nonempty memory and fixed parameters,

$$\mathbb{E}\left[\frac{1}{R}\sum_{n \in \mathcal{S}_t^{\mathrm{uniform}}} d_n(t) \,\middle|\, \mathcal{M}_t, \theta\right] = \frac{1}{N_t}\sum_{n \in \mathcal{A}_t} d_n(t). \tag{24}$$

D.3.3 Hard Replay

Hard replay selects the $\min(R, N_t)$ slots with the largest significance σ_n (Appendix D.2),

$$\mathcal{H}_t = \mathrm{Top}_{\min(R,N_t)}\left(\{(n, \sigma_n) \mid n \in \mathcal{A}_t\}\right), \qquad \mathcal{S}_t^{\mathrm{hard}} = \mathrm{Fill}_R(\mathcal{H}_t), \tag{25}$$

returned in descending score order, with Fill_R as in Appendix D.3.1. A large JEPA error indicates poor representation alignment, not necessarily an incorrect generated answer.

E Ablation Studies

E.1 Sparse and Dense Memory Addressing

We compare content replay using sparse addresses and dense source representations. Sparse addresses are formed by a frozen random projection and Top-K selection. Both variants use cosine similarity, with all other settings fixed. Table 5 reports mean accuracies of 86.37% for sparse retrieval and 86.21% for dense retrieval. The two strategies achieve similar mean accuracy, with a difference of 0.16 percentage points.

Table 5: Sparse and dense memory addressing for content replay. All other training settings are fixed. Results report mean accuracy, standard deviation, minimum, and maximum over five seeds.

Method	**Accuracy (%) ↑**	**Min**	**Max**
ER-JEPA (Sparse Retrieval)	$\mathbf{86.37 \pm 0.35}$	85.75	86.60
Dense Retrieval	86.21 ± 0.37	85.85	**86.70**

E.2 Hyperparameter Ablations

Memory capacity. Table 6 reports accuracy after four epochs, and Figure 14 shows the learning curves. Increasing memory capacity from 10 to 10^4 raises accuracy from 84.22% to 84.98%.

Objective hyperparameters. We also vary the JEPA loss weight λ and predictor depth k, following the LLM-JEPA design. Figure 15 reports the results.

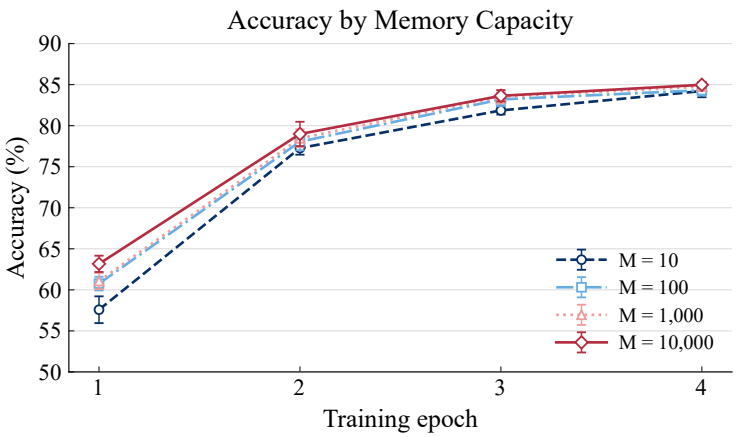


Figure 14: Accuracy across four training epochs for four episodic memory capacities. Points show means over five seeds. Error bars show one sample standard deviation. These runs are not compute matched.

Table 6: Ablation on episodic memory capacity $\mathcal{M}$ for Meta-Llama-3.2-1B-Instruct on NL-RX-SYNTH. Values at epoch 4 are mean $\pm$ standard deviation over five seeds. The observed training costs are not compute matched.

Memory capacity $\mathcal{M}$	**Accuracy (%) $\uparrow$**	**PFLOPs**	**Time (min)**	**Tokens (M)**
10	84.22 ± 0.75	181.536 ± 0.211	10.48 ± 0.07	8.581 ± 0.166
10^2	84.32 ± 0.65	219.605 ± 0.533	11.62 ± 0.22	9.427 ± 0.175
10^3	84.76 ± 0.52	219.766 ± 0.439	16.74 ± 1.33	10.969 ± 0.097
10^4	$\mathbf{84.98 \pm 0.35}$	219.667 ± 0.393	22.23 ± 0.17	11.252 ± 0.010

	$k = 0$	$k = 1$	$k = 2$	$k = 3$
$\lambda = 0.5$	85.05%	85.55%	86.05%	85.70%
$\lambda = 1.0$	85.80%	85.95%	**86.60%**	86.30%
$\lambda = 2.0$	86.30%	86.40%	85.85%	86.10%

Figure 15: Ablation on the JEPA hyperparameters λ and k. Each entry reports accuracy (%), with the best result in bold.