# LEVERAGING SELF-SUPERVISED LEARNING FOR FRAUD DETECTION IN TABULAR DATA

**José Lucas De Melo Costa**[1]
**Fabrice Popineau**[1] **Arpad Rimmel**[1] **Fabrice Daniel**[2] **Bich-Liên Doan**[1]

[1] Université Paris-Saclay, CNRS, CentraleSupélec,
Laboratoire Interdisciplinaire des Sciences du Numérique,
91190 Gif-sur-Yvette, France
[2] LUSIS, France

{name}.{surname}@centralesupelec.fr[1] fabrice.daniel@lusis.fr[2]

## ABSTRACT

Financial fraud poses significant economic and societal risks, yet detecting fraud in tabular data remains challenging due to extreme class imbalance and the difficulty of learning meaningful representations. Traditional gradient-boosting models dominate this domain but struggle to generalize effectively. In this work, we propose $\tau$-JEPA, a lightweight self-supervised pretraining framework designed to enhance fraud detection in imbalanced datasets. We introduce a theoretical analysis of JEPA loss dynamics and show that early-stage representation collapse can be mitigated through adaptive momentum scheduling. Our scheduling improves pretraining, giving $\tau$-JEPA a 12–11–7 win–tie–loss ratio advantage over original models while retaining a shallow MLP encoder. We provide empirical evidence that the learned representations enhance downstream fraud classifiers, particularly in low-data regimes. This work demonstrates that self-supervised learning can be an effective tool for tabular fraud detection, offering a scalable alternative to traditional methods.

## 1 INTRODUCTION

Financial fraud in digital transactions is an escalating challenge, with cybercriminals continuously developing more sophisticated evasion techniques. According to the Nilson Report (HSN, 2024), financial fraud in banking alone is projected to exceed $34.75 billion by 2025, reflecting both the rapid adoption of digital payments and the increasing complexity of fraudulent schemes. Beyond financial losses, fraud erodes consumer trust in financial systems, necessitating advanced and adaptive fraud detection methodologies (Baesens et al., 2015).

Detecting fraud in tabular datasets—which comprise structured transaction records—remains particularly challenging due to extreme class imbalance, concept drift, and feature sparsity. Traditional fraud detection models rely on gradient boosting techniques, which, while effective in structured data, struggle to generalize across varying fraud patterns and highly imbalanced distributions. Despite the success of deep learning architectures in other domains (e.g., vision and NLP), their application to tabular data remains limited due to issues such as lack of inductive biases and suboptimal feature encoding. Hence, gradient boosting-based techniques remain the predominant choice for anomaly detection in tabular data (Zhao & Hryniewicki, 2018), primarily due to their ability to focus on hard-to-predict instances.

To address these challenges, several solutions have been proposed, including the development of tabular-specific deep learning architectures Gorishniy et al. (2023). One promising approach involves representation learning techniques based on Self-Supervised Learning (SSL). This method aims to characterize samples by optimizing a proxy objective to construct robust latent spaces (Shwartz Ziv & LeCun, 2024), upon which predictions are subsequently made using the transformed representation of the data.

Two major aproaches of tabular representation learning are constrastive and regularized methods. Contrastive SSL strategies, which perform well in image domains (He et al., 2020; Chen et al., 2020), bring similar samples closer and push dissimilar ones apart, though their success heavily depends on the creation of negative samples and typically requires large datasets. Non-contrastive methods (Bardes et al., 2022; Chaoning Zhang et al., 2022; Grill et al., 2020), tackle this limitation by regularizing the latent space differently, avoiding explicit negative pairs. Recently, Joint Embedding Predictive Architectures (JEPAs) (Assran et al., 2023) have emerged as a variant of non-contrastive methods and presents promising results when applied to tabular data Thimonier et al. (2024).

Despite the empirical success of non-contrastive methods, fundamental questions remain about their training dynamics, particularly regarding the phenomenon of *collapse* — a state where the learned representations collapse to a single point or low-dimensional subspace. While previous studies (Tian et al., 2021; Esser et al., 2023) have explored why JEPAs do not collapse asymptotically, recent findings (Thimonier et al., 2024) reveal an initial collapse phase in the early epochs of training. The mechanics and implications of this transient behavior, especially in domains like images, text, or tabular data, remain under-investigated.

In this work, we introduce $\tau$-JEPA, a novel self-supervised pretraining framework specifically designed to enhance fraud detection in tabular datasets. Building upon the promising results of T-JEPA, we recognize that its loss landscape remains unexplored and challenging to track. To gain a deeper understanding of the collapse phenomenon, we design a lightweight version of T-JEPA, replacing its transformer-based architecture with a multi-layer perceptron (MLP). This simplification allows us to analyze the collapse process and its influence on the loss trajectory. Additionally, we assess whether this lightweight JEPA retains practical benefits, such as faster computation compared to T-JEPA, while maintaining its effectiveness in fraud detection. Incidentally, we also investigate a momentum scheduling to optimize training stability and performance for this lightweight JEPA variant.

The contributions of this work are:

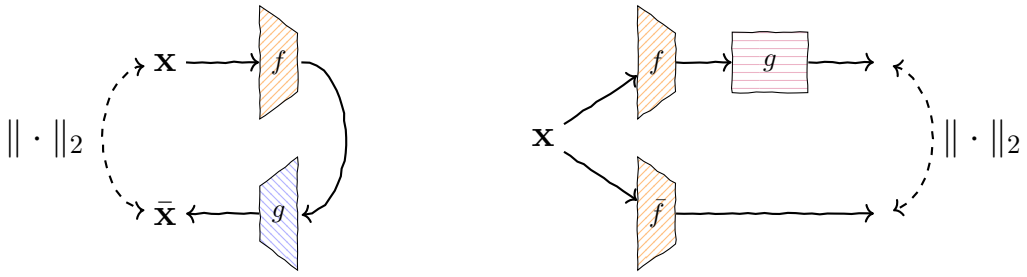
1. **Theoretical Model of JEPA for Tabular Data:** We provide a novel analysis of the loss landscape for linear JEPA training, presenting conditions under which collapse is avoided.
2. **Lightweight SSL Pretraining for Fraud Detection:** We introduce a simplified JEPA-based approach that mitigates the high computational cost typically associated with SSL methods.
3. **Empirical Validation on Fraud Benchmarks:** We show consistent improvements over baseline methods on real-world fraud datasets, thereby demonstrating the practical effectiveness of our approach.

## 2 Related Work

**Joint Embedding Predictive Architectures (JEPAs)** Joint Embedding Predictive Architectures (JEPAs) were initially introduced for images (Assran et al., 2023) and operate by predicting hidden representations from partially masked input views (Tsai et al., 2021; Shwartz Ziv & LeCun, 2024). By training a context and a target encoder on complementary views, JEPAs promote representations that are both informative and robust to noise. However, as highlighted by Thimonier et al. (2024), the training dynamics of JEPAs follow four distinct stages—initialization, collapse, collapse exit, and convergence.

Figure 1 illustrates a key difference between generative and JEPA-based self-supervised architectures. In a generative framework (Figure 1a), the loss is computed in the original space, preserving structural integrity throughout training. In contrast, JEPAs (Figure 1b) compute the loss in the latent space, making them susceptible to representational collapse. This issue arises because, instead of reconstructing the input explicitly, the model learns to predict embeddings, which can lead to degenerate solutions where all representations collapse into trivial forms.

**Training dynamics of non-constrastive SSL** Theoretical advances in understanding collapse avoidance in non-contrastive self-supervised learning (SSL) primarily follow two perspectives: temporal dynamics and gradient analysis. The work by Tian et al. (2021) employs a temporal framework




(a) Generative Architecture (b) Joint-Embedding Predictive Architecture

Figure 1: Common architecture for self-supervised learning. In the generative approach (a), the system learns to reconstruct a sample $\mathbf{x}$ using an encoder-decoder structure. The loss is computed in the original space and no collapse occurs in this setup. In the Joint-Embedding Predictive Architecture represented in (b), the model learns to predict the embedding of a sample $(g \circ f)(\mathbf{x})$ from a corrupted version of it. The loss is computed in the latent space, thus this setup may suffer from representational collapse.

to study the evolution of parameter matrices during training. Chaoning Zhang et al. (2022) proposed a complementary gradient-based analysis. Using a vector decomposition framework, they unified the gradient contributions that prevent representational collapse in non-contrastive methods. These insights guide the theoretical foundation for our JEPA-based model, particularly in understanding how to maintain non-trivial representations without contrastive samples.

## 3 Unveiling training dynamics of JEPA models

This section outlines our theoretical and practical approach to building a lightweight JEPA for fraud detection in tabular data. We first present the linear model formulation and the assumptions under which the JEPA learning dynamics can be analyzed. We then introduce our simplified model and an algorithmic strategy to adaptively schedule momentum parameters.

### 3.1 Linear Model of JEPAs

The JEPA architecture comprises a *context encoder* $f_\theta$ and a *target encoder* $f_{\bar{\theta}}$, which jointly generate and align latent representations. Given a sample $\mathbf{x} \in \mathbb{R}^N$, specific subsets of features are masked to produce distinct views of $\mathbf{x}$. The context encoder processes one subset, while the target encoder processes the other. A predictor module $g_\phi$ then learns how to match the target encoder's latent space from the output of the context encoder. Figure 2 illustrates this workflow.

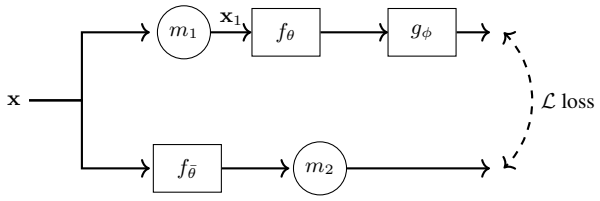



Figure 2: Overview of the JEPA architecture. The context encoder $f_\theta$ processes masked inputs, and the predictor $g_\phi$ minimizes the reconstruction loss with respect to the target encoder $f_{\bar{\theta}}$.

**Notation.** Let $\mathbf{m}_1 \in \{0,1\}^N$ and $\mathbf{m}_2 \in \{0,1\}^d$ be masking vectors, where $\mathbf{m}^j = 0$ indicates feature $j$ is masked out. The model uses different masks $m_1$ and $m_2$ for context and target encoders, respectively. The overall loss is:

$$\mathcal{L}(\mathbf{x}) = \frac{1}{|M_1|} \cdot \frac{1}{|M_2|} \sum_{\mathbf{m}_1 \in M_1} \sum_{\mathbf{m}_2 \in M_2} \left\| g_\phi(h^{\mathbf{m}_1}_{\text{context}}) - h^{\mathbf{m}_2}_{\text{target}} \right\|_2^2, \tag{1}$$

where $M_1$ and $M_2$ are the sets of masking vectors for context and target subsets, respectively. The predictor's input is defined as $h_{context}^{\mathbf{m_1}} = f_\theta(x_1)$, with $x_1 = x \circ m_1$ and $\circ$ the Hadamard product. Moreover, $h_{\text{target}}^{\mathbf{m_2}} = f_{\bar{\theta}}(x) \circ m_2$.

In the following, we consider the case where $f_\theta$, $f_{\bar{\theta}}$ and $g_\phi$ are represented by matrices $W$, $W_a$, $W_p$ respectively.

**Lemma 1** (Linear JEPA Learning Dynamics)**.** *Let $W \in \mathbb{R}^{d\times N}$ and $W_p \in \mathbb{R}^{d\times d}$ be the weight matrices of the context encoder and the predictor, respectively. Define $X_1 \in \mathbb{R}^{B\times N}$ as the masked feature subsets, $W_a$ as the weight matrix of the target encoder, and $m_2$ as the target masking. Then, the JEPA weight updates can be written in the form:*

$$\dot{W} \;=\; -\,W_p^\top\, W_p\, W\, X_1^\top X_1 \;+\; W_p^\top \left(X\, W_a^\top \circ m_2\right)^\top X_1, \tag{2}$$

$$\dot{W_p} \;=\; -\alpha_p \Big[\, W_p\, W\, X_1^\top\, X_1\, W^\top \;-\; \left(X\, W_a^\top \circ m_2\right)^\top X_1\, W \Big], \tag{3}$$

*where $\alpha_p$ is the relative predictor learning rate, and $\circ$ denotes the Hadamard product.*

It is possible to simplify the binary mask as a diagonal operation. Instead of an element-wise product one can represent it as a matrix multiplication, leading to the following assumption.

**Assumption 1** (Masking features)**.** *The binary mask $m_2 \in \{0,1\}^d$, sampled independently at each iteration ($m_2[i] \sim$ Bernoulli$(p)$), activates a random subset of hidden dimensions. Represented as $M_2 =$ diag$(m_2)$, it reformulates $\left(XW_a^\top \circ m_2\right)$ in equation 2 as $XW_a^\top M_2$, where now we assume that the selected features are masked for each row (the entire batch).*

Next, we define another assumption that models the exponential moving average as a proportional scaling factor, aligning with prior theoretical work (Tian et al., 2021)

**Assumption 2** (Proportional Weights in Target Encoder)**.** *We assume the target encoder weights $W_a$ maintain a fixed proportional relationship $\tau$ with the context encoder weights $W$, i.e., $W_a = \tau W$, throughout training.*

### 3.2 Stable Point Analysis

We analyze the stationary points of the JEPA training dynamics to understand collapse and stability. Below is a key theoretical result. We refer the reader to the appendix C for more information.

**Theorem 1** (Stationary Points of the JEPA Training Dynamics)**.** *Consider small perturbations around the origin. Under the assumption that the relevant inverses exist, the stationary points of the JEPA training dynamics are given by*

$$W^* \;=\; A_L\, W_a\, A_R. \tag{4}$$

*Moreover, the momentum parameter $\tau$ must satisfy*

$$\frac{1}{\tau} \;\in\; \mathrm{Spec}\big(A_R^\top \otimes A_L\big), \tag{5}$$

*where*

$$A_R = X^\top X_1 \left(X_1^\top X_1\right)^{-1}, \quad A_L = \left(W_p^\top W_p\right)^{-1} W_p^\top\, M_2^\top,$$

*and $\mathrm{Spec}(\cdot)$ denotes the set of eigenvalues of a matrix or linear operator. The point $W^*$ nullifies the gradient of the loss with respect to $W$, and the condition on $\tau$ determines when nontrivial solutions emerge beyond the trivial $W = 0$.*

**Geometric Interpretation.** The trivial solution $W = 0$ lies at the origin in the space of weight matrices. Its stability depends on the parameter $\tau$. If the gradient at $W = 0$ is restorative, small perturbations revert to $W = 0$, maintaining stability. As $\tau$ increases, a bifurcation occurs when $\frac{1}{\tau}$ matches an eigenvalue of $(A_R \otimes A_L)$. Beyond this threshold, $W = 0$ becomes unstable, and new equilibrium solutions $W \neq 0$ appear in the corresponding eigen-directions. As so, two main stability regimes emerge:

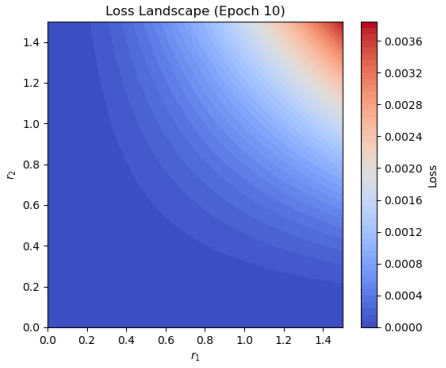
- **Subcritical** ($\tau < \tau_{\text{crit}}$): The trivial solution $W = 0$ is stable; no other fixed points exist.
- **Supercritical** ($\tau \geq \tau_{\text{crit}}$): $W = 0$ becomes unstable, causing non-trivial solutions $W \neq 0$ to emerge.

### 3.3 LOSS LANDSCAPE

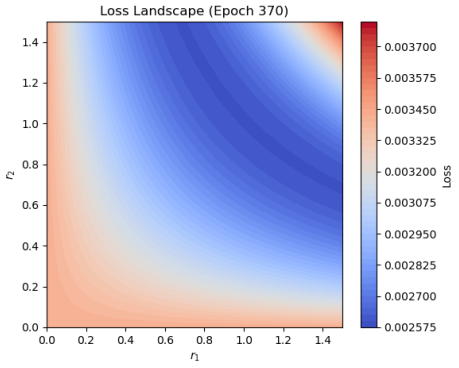
The loss landscape provides insight into the training dynamics of the JEPA model on the `ccfraud` dataset. Our analysis follows the theoretical framework established in Ziyin et al. (2023), where the geometry of self-supervised learning (SSL) loss landscapes is linked to stability, collapse, and the emergence of meaningful representations.

At the beginning of training (Epoch 0, Figure 3a), the model exhibits collapse, characterized by a dominant stable region near the origin. As training progresses (Epoch 370, Figure 3b), new minima emerge in the loss landscape, signifying the escape from collapse. This evolution is predicted by Theorem 1, where the stationary points $W^* = A_L W_a A_R$ depend on the interaction between the data structure and learning dynamics. Specifically, when the spectral condition $\frac{1}{\tau} \in \text{Spec}(A_R^\top \otimes A_L)$ is met, the trivial solution $W = 0$ loses stability, and the model learns structured representations aligned with informative directions in the data.

Figure 3 illustrates this transition, where the encoder matrix $W$ is parameterized by scaling factors $r_1$ and $r_2$. The color gradient represents loss variations, with red indicating regions of higher loss. Initially, the loss is minimized when $W$ remains near zero, consistent with theoretical predictions for early-stage SSL models. However, as training continues, the stability regime shifts, leading to more diverse and structured solutions.




(a) Loss landscape at epoch 0, illustrating collapse, as the loss is minimized when the weight values are zero.




(b) Loss landscape at epoch 370, showing newly emerged local minima.

Figure 3: Loss landscapes on the `ccfraud` dataset running the theoretical model. The encoder matrix $W$ is scaled by factors $r_1$ and $r_2$ for its upper and lower halves, respectively. The color gradient represents loss variations, with red indicating areas of increasing loss.

## 4 SIMPLIFIED JEPA MODEL AND MOMENTUM SCHEDULING

We propose $\tau$-JEPA, an adaptation of the JEPA model within a streamlined framework for fraud detection, reducing computational overhead while preserving its theoretical advantages:

1. **Context Encoder:** A shallow MLP receiving masked features.
2. **Target Encoder:** Mirrors the context encoder, with weights set to a fraction $\tau$ of the context encoder's weights.
3. **Predictor:** A single linear layer mapping context embeddings to target embeddings.

To ensure stable training, we introduce a dynamic momentum scheduling algorithm that iteratively adjusts $\tau$ based on eigenvalue bounds derived from Theorem 1. The procedure initializes $\tau$ with an initial value $\tau_0$ and iteratively updates it to ensure that it remains within stable eigenvalue bounds. Given the computed eigenvalues of matrices $A_L$ and $A_R$, the permissible range for $\tau$ is defined as minBound $= 1/\lambda_{\max}$ and maxBound $= 1/\lambda_{\min}$. If $\tau$ falls outside these limits, it is adjusted by a predefined increment $\delta$ until it returns within the valid range.

The target encoder weights are updated using $W_a \leftarrow \tau W$, followed by standard JEPA parameter updates by gradient descent for the context encoder and predictor. If $\tau$ remains out of bounds for a prolonged period, training halts to prevent divergence. Pseudocode is provided in Algorithm 1.

**Algorithm 1** Momentum Scheduling for $\tau$-JEPA Anomaly Detection

**Require:** Context encoder $W$, predictor $W_p$, target encoder $W_a$, data batches $(X, X_1, m_2)$, initial momentum $\tau_0 > 0$, maximum iterations $T$, bounding window size $\Delta$, threshold parameter $\epsilon$ (maximum consecutive out-of-bounds updates allowed)

$\tau \leftarrow \tau_0$
countInsideBound $\leftarrow 0$
**for** $t = 1$ to $T$ **do**
  **if** countInsideBound $< \epsilon$ **then**
    $\tau \leftarrow \tau + \delta$
  **end if**
  Compute $W_p^\top W_p$ and $X_1^\top X_1$
  **if** $W_p^\top W_p$ and $X_1^\top X_1$ are invertible **then**
    $P_L \leftarrow (W_p^\top W_p)^{-1}$, $P_R \leftarrow (X_1^\top X_1)^{-1}$
  **else**
    $P_L \leftarrow (W_p^\top W_p)^{+}$, $P_R \leftarrow (X_1^\top X_1)^{+}$
  **end if**
  $A_L \leftarrow P_L W_p^\top M_2^\top$
  $A_R \leftarrow X^\top X_1 P_R$
  Compute eigenvalues: $\lambda(A_L)$, $\lambda(A_R^\top)$
  $\lambda_{\min} \leftarrow \min(\lambda(A_L)) \times \min(\lambda(A_R^\top))$
  $\lambda_{\max} \leftarrow \max(\lambda(A_L)) \times \max(\lambda(A_R^\top))$

$$\text{minBound} \leftarrow \frac{1}{\lambda_{\max}}, \quad \text{maxBound} \leftarrow \frac{1}{\lambda_{\min}}$$

  **if** $\tau \geq$ minBound **and** $\tau \leq$ maxBound **then**
    countInsideBound $\leftarrow$ countInsideBound $+ 1$
  **else**
    countInsideBound $\leftarrow$ countInsideBound $- 1$
  **end if**
  *Update target encoder weights:* $W_a \leftarrow \tau W$
  *Perform standard JEPA parameter updates on* $W$ *and* $W_p$
**end for**

## 5 EXPERIMENTS

### 5.1 DATASETS

To evaluate the proposed method, we employ multiple fraud datasets spanning real-world and simulated scenarios. Table 1 provides a high-level overview of each dataset, including the number of training and test samples, along with feature dimensionality.

Table 1: Overview of Datasets Used for Fraud Detection

| Dataset | Fraud category | # samples | #Categorical | # Numerical |
|---|---|---|---|---|
| `fraudecom` | Card Not Present Transactions Fraud | 42312 | 2 | 3 |
| `ieeecis` | Card Not Present Transactions Fraud | 589540 | 61 | 6 |
| `ccfraud` | Card Not Present Transactions Fraud | 284807 | 0 | 28 |

**IEEE-CIS Fraud Detection (`ieeecis`)** This dataset, provided by Vesta Corporation and prepared by the IEEE Computational Intelligence Society, focuses on card-not-present transaction

fraud. The original release contains 393 features (anonymized variables such as product, card, address, email domain, and device, plus numeric columns prefixed by V, C, D, and M).

**Credit Card Fraud Detection (`ccfraud`)** This dataset comprises card-not-present transactions by European cardholders collected in September 2013. Features are largely anonymized via PCA transformations, accompanied by non-transformed time and amount. Of the 284,807 transactions over two days, 492 are labeled as fraud, yielding a highly imbalanced dataset. The original time and amount variables remain, while the PCA components obscure sensitive information.

**Fraud E-commerce (`fraudecom`)** This e-commerce dataset includes variables such as sign-up time, purchase time, purchase value, device ID, user ID, browser, and IP address.

### 5.2 MODELS

We employed multiple machine learning models to detect fraudulent activities, leveraging both gradient boosting and ensemble learning techniques. **LightGBM** (Ke et al.), developed by Microsoft, offers efficient tree-based learning with fast training speeds. **CatBoost** (Prokhorenkova et al., 2019), from Yandex, is optimized for categorical features, reducing preprocessing efforts. **XGBoost** (Chen & Guestrin, 2016) provides an optimized gradient boosting framework with parallel tree boosting for high accuracy. Additionally, we used ensemble methods such as **RandomForestClassifier** and **ExtraTreesClassifier** (Pedregosa et al., 2018), which build multiple decision trees and average predictions to enhance robustness. Finally, we included the **KNeighborsClassifier** (Pedregosa et al., 2018), a non-parametric model relying on majority voting among nearest neighbors for classification.

### 5.3 EXPERIMENTAL SETTING

**Data Preprocessing** Each dataset was split into training, validation, and test sets using an 80/10/10 ratio. When required by the model, numerical features were normalized, while categorical features were encoded using either label encoding or one-hot encoding. Missing values were imputed using the mean strategy, and columns with a high proportion of missing values were discarded.

**$\tau$-JEPA Pretraining** The first phase of training involved tuning the hyperparameters of the shallow MLP, including the hidden layer dimension, number of layers, and dropout rate. The model was trained using the AdamW optimizer (Loshchilov & Hutter, 2019) and a learning rate scheduler based on cosine annealing (Loshchilov & Hutter, 2017). The $\tau$ parameter regulates the linear dependence between the parameters of the context encoder and the target encoder. This parameter was adjusted following the algorithm detailed in Algorithm 1. A linear probe was used to evaluate the learned representation and determine the optimal hyperparameters. We refer the reader to the appendix A for more details.

**Fraud Detection Task** For the fraud detection task, baseline models were trained on the new representation space. Each experiment was conducted five times using different random seeds, and we report the average performance across these runs. The fraud detection models were evaluated using the AUC-ROC metric. The goal was to determine whether the learned representation improved the robustness and generalization of the models, particularly in highly imbalanced datasets where fraudulent transactions are significantly rarer than legitimate ones.

### 5.4 RESULTS

As presented in Table 2, the proposed pretraining scheme improved the performance of most machine learning models.

As presented in Table 3,the highest performance improvements were observed in the `fraudecom` dataset, where most models benefited from the learned representations. The improvements ranged from small but consistent gains (e.g., LightGBM improved from $0.764$ to $0.775$) to more significant enhancements (e.g., XGBoost improved from $0.759$ to $0.776$).

Table 2: Overall Win/Tie/Loss for $\tau$-JEPA vs. Original Models.

| **Dataset** | **Wins** | **Ties** | **Losses** |
|---|---|---|---|
| fraudecom | 8 | 0 | 2 |
| ccfraud | 0 | 10 | 0 |
| ieeecis | 4 | 1 | 5 |
| **Overall** | 12 | 11 | 7 |

Interestingly, for datasets where the models already performed exceptionally well in detecting fraud, such as `ccfraud`, no degradation in performance was observed. This suggests that $\tau$-JEPA does not interfere with the inherent capacity of the models to learn from clean and well-separated data distributions.

On the `ieeecis` dataset, results varied depending on the model. While LightGBM and some tree-based models slightly improved, XGBoost and CatBoost saw minor performance declines. This could be attributed to the complex nature of the dataset, and models with more capacity should be used instead of the proposed MLP.

Table 3: **Performance metrics.** Comparison of Original Models vs. $\tau$-JEPA Enhanced Models. We report in **bold** the metric that wins between the original space and the learned representation. We underline the overall best for a given dataset.

| **Model** | `fraudecom` | `ccfraud` | `ieeecis` |
|---|---|---|---|
| LightGBM | 0.764 | 1.0 | 0.886 |
| + $\tau$-JEPA | **0.775** | 1.0 | **0.888** |
| XGBoost | 0.759 | 1.0 | **0.881** |
| + $\tau$-JEPA | **0.776** | 1.0 | 0.871 |
| CatBoost | 0.768 | 1.0 | **0.886** |
| + $\tau$-JEPA | **0.774** | 1.0 | 0.866 |
| RandomForestGini | 0.764 | 1.0 | 0.862 |
| + $\tau$-JEPA | **0.767** | 1.0 | **0.863** |
| ExtraTreesEntr | 0.762 | 1.0 | **0.879** |
| + $\tau$-JEPA | **0.767** | 1.0 | 0.877 |
| LightGBMLarge | 0.769 | 1.0 | **0.888** |
| + $\tau$-JEPA | **0.772** | 1.0 | 0.874 |
| RandomForestEntr | 0.762 | 1.0 | **0.866** |
| + $\tau$-JEPA | **0.766** | 1.0 | 0.858 |
| ExtraTreesGini | 0.763 | 1.0 | 0.871 |
| + $\tau$-JEPA | **0.768** | 1.0 | 0.871 |
| KNeighborsDist | **0.765** | 1.0 | 0.774 |
| + $\tau$-JEPA | 0.761 | 1.0 | **0.786** |
| KNeighborsUnif | **0.765** | 1.0 | 0.773 |
| + $\tau$-JEPA | 0.761 | 1.0 | **0.783** |

## 6 DISCUSSION

**Impact of Scheduling in Non-Linear Scenarios** As illustrated in Figure 4, $\tau$ must be dynamically regulated to maintain performance within an optimal range. When $\tau$ drifts outside the non-collapse region (stage a), performance degrades, necessitating an increase in $\tau$. Once stability is achieved (stage b), the scheduling remains consistent. However, if instability arises (stage c), a further increase is required to recover the downstream performance.

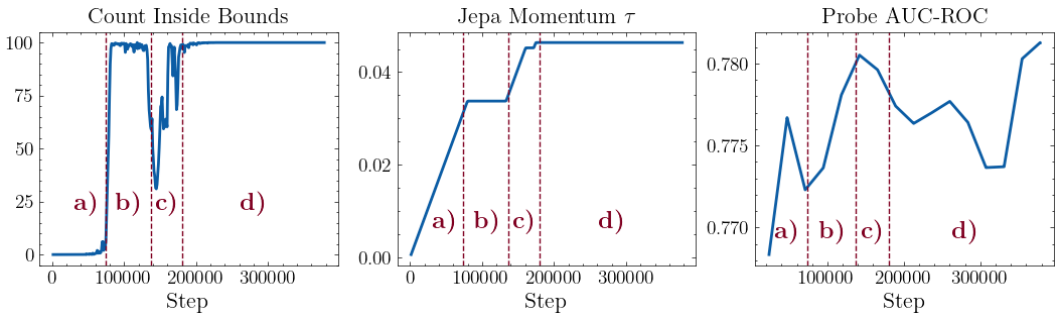



Figure 4: Training steps of $\tau$-JEPA in the `ccfraud`. The first plot shows the count of occurrences where $\tau$ remains within the desired range. When this condition is met, $\tau$ is kept constant, as depicted in the middle plot. The third plot illustrates the impact on probing accuracy. Three key regions are highlighted: (a) $\tau$ is initially outside the non-collapse region, prompting an increase; (b) once $\tau$ satisfies the condition, it remains stable; and (c) an instability arises, causing $\tau$ to increase further. This instability results in a temporary drop in probe detection performance. Eventually, the model regains stability, $\tau$ returns to the desired range, and the AUC begins to improve again.

**Fraud detection improvement** The experimental results demonstrate that the $\tau$-JEPA approach provides advantages in fraud detection. The impact of $\tau$-JEPA varies depending on the base classifier. While models such as LightGBM and RandomForest show steady gains, some variations in performance are observed for XGBoost and CatBoost, particularly on the `ieeecis` dataset. One reason for this may be the insufficient model capacity (MLP encoder) to learn an effective representation space.

**Limitations and Future Directions** Despite the demonstrated improvements, $\tau$-JEPA has certain limitations that warrant further investigation. The most important one is that the dynamic scheduling depends highly on the soundness of the matrices $W_p W_p^\top$ and $X_1^\top X_1$, but in practice those matrices may be poorly-conditioned, which leads to instabilities in calculating the eigenvalues.

In addition to that, while the method enhances performance for most classifiers, cases where performance decreases suggest that additional hyperparameter tuning may be required to optimize generalization across diverse datasets. Second, the low capacity of the chosen MLP architecure may constraint the learned representation.

## 7 Conclusion

In this paper, we presented a novel framework for fraud detection in tabular data that combines Joint Embedding Predictive Architectures (JEPAs) with a dynamic momentum scheduling mechanism, denoted as $\tau$-JEPA. Building on theoretical insights regarding loss landscape analysis and representational collapse, we demonstrated that properly regulating the momentum parameter $\tau$ helps the model escape trivial solutions and learn meaningful embeddings. Our empirical results on real-world fraud datasets highlight the effectiveness of these learned representations: when downstream classifiers operate on these embeddings, their fraud detection performance generally improves over standard baselines, especially for tree-based models.

## 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, April 2023. URL http://arxiv.org/abs/2301.08243. arXiv:2301.08243 [cs, eess].

Bart Baesens, Véronique Van Vlasselaer, and Wouter Verbeke. Fraud: Detection, Prevention, and Analytics! In *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques*, pp. 1–36. John Wiley & Sons, Ltd, 2015. ISBN 978-1-119-14684-1. doi: 10.1002/9781119146841.ch1. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119146841.ch1. Section: 1 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119146841.ch1.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning, January 2022. URL http://arxiv.org/abs/2105.04906. arXiv:2105.04906 [cs].

Chaoning Zhang, Kang Zhang, Chenshuang Zhang, T. Pham, C. D. Yoo, and I. Kweon. How Does SimSiam Avoid Collapse Without Negative Samples? A Unified Understanding with Self-supervised Contrastive Learning. *International Conference on Learning Representations*, 2022. doi: 10.48550/arxiv.2203.16262. ARXIV_ID: 2203.16262 S2ID: 2b9455fceb0ff58f28a46aebfb8df6f7003e9e40.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, August 2016. doi: 10.1145/2939672.2939785. URL http://arxiv.org/abs/1603.02754. arXiv:1603.02754 [cs].

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations, June 2020. URL http://arxiv.org/abs/2002.05709. arXiv:2002.05709 [cs, stat].

Pascal Esser, Satyaki Mukherjee, and Debarghya Ghoshdastidar. Representation Learning Dynamics of Self-Supervised Models, September 2023. URL http://arxiv.org/abs/2309.02011. arXiv:2309.02011.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale, July 2018. URL http://arxiv.org/abs/1807.01774. arXiv:1807.01774 [cs].

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting Deep Learning Models for Tabular Data, October 2023. URL http://arxiv.org/abs/2106.11959. arXiv:2106.11959 [cs].

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning, September 2020. URL http://arxiv.org/abs/2006.07733. arXiv:2006.07733 [cs, stat].

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning, March 2020. URL http://arxiv.org/abs/1911.05722. arXiv:1911.05722 [cs].

Consultants HSN. The Nilson report. Technical report, HSN Consultants, Inc, 2024.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts, May 2017. URL http://arxiv.org/abs/1608.03983. arXiv:1608.03983 [cs].

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs].

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python, June 2018. URL http://arxiv.org/abs/1201.0490. arXiv:1201.0490 [cs].

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features, January 2019. URL http://arxiv.org/abs/1706.09516. arXiv:1706.09516 [cs].

Ravid Shwartz Ziv and Yann LeCun. To Compress or Not to Compress—Self-Supervised Learning and Information Theory: A Review. *Entropy*, 26(3):252, March 2024. ISSN 1099-4300. doi: 10.3390/e26030252. URL https://www.mdpi.com/1099-4300/26/3/252. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

Hugo Thimonier, José Lucas De Melo Costa, Fabrice Popineau, Arpad Rimmel, and Bich-Liên Doan. T-JEPA: Augmentation-Free Self-Supervised Learning for Tabular Data. 2024. ARXIV_ID: 2410.05016 S2ID: cff3fa0ee682923d0321cf350467d3907dbba358.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised Learning Dynamics without Contrastive Pairs, October 2021. URL http://arxiv.org/abs/2102.06810. arXiv:2102.06810.

Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Self-supervised Learning from a Multi-view Perspective, March 2021. URL http://arxiv.org/abs/2006.05576. arXiv:2006.05576 [cs, stat].

Yue Zhao and Maciej K. Hryniewicki. XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2018. doi: 10.1109/IJCNN.2018.8489605. URL https://ieeexplore.ieee.org/document/8489605. ISSN: 2161-4407.

Liu Ziyin, Ekdeep Singh Lubana, Masahito Ueda, and Hidenori Tanaka. What shapes the loss landscape of self-supervised learning?, March 2023. URL http://arxiv.org/abs/2210.00638. arXiv:2210.00638 version: 2.

# A TRAINING DETAILS

This section presents the implementation details of the project.

## A.1 PROGRAMMING ENVIRONMENT

All the code was develop in Python, using the Pytorch libraries. In addition to that, we employed the ssl4tab library to access some useful resources for tabular self-supervised learning. The baseline models training was done using the Autogluon library, an AutoML python library. The most important libraries are presented in Table 4.

Table 4: Main libraries used in the project.

| Library | Description |
|---|---|
| **Python** `v3.12.5` | The programming language used for the project |
| **ssl4tab** `v0.1.0` | Self-supervised learning utils for tabular data |
| **autogluon** `v1.2.0` | Automates machine learning tasks for tabular data |
| **einops** `v0.8.0` | A flexible and powerful tool for tensor operations |
| **matplotlib** `v3.8.4` | A library for creating static, animated, and interactive plots |
| **numpy** `v2.1.0` | Fundamental package for scientific computing with arrays |
| **pandas** `v2.2.2` | Data manipulation and analysis tool |
| **pytorch_lightning** `v2.2.1` | A PyTorch wrapper for high-performance deep learning research |
| **scikit_learn** `v1.4.1.post1` | Machine learning library for Python |
| **scipy** `v1.14.1` | Library for scientific and technical computing |
| **torch** `v2.3.0.post301` | PyTorch deep learning library |
| **torchinfo** `v1.8.0` | Module to show model summaries in PyTorch |
| **tqdm** `v4.66.2` | Progress bar utility for Python |
| **xgboost** `v2.1.1` | Optimized gradient boosting library |

## A.2 HYPERPARAMETER SEARCH

We employed Bayesian optimization (Falkner et al., 2018) to tune the hyperparameters. The hyperparameters are described in Table 5.

Table 5: Hyperparameter Search Configuration for Bayesian Optimization

| Parameter | Values | Description |
|---|---|---|
| `dropout` | $\mathcal{U}(0.1, 0.5)$ | Dropout rate applied to prevent overfitting. |
| `lr_ctx` | $\mathcal{U}_{log}(0.003, 1e-05)$ | Context learning rate |
| `lr_pred` | $\mathcal{U}_{log}(0.003, 1e-05)$ | Predictor learning rate |
| `n_hidden` | $[1, 2, 3, 4, 5, 6, 7, 8]$ | Number of hidden layers in the model. |
| `hidden_dim` | $[16, 32, 64, 128, 256, 512]$ | Hidden dimension |
| `mask_min_ctx_share` | $(0.1, 0.3)$ | Minimum proportion of masked context features. |
| `mask_max_ctx_share` | $(0.3, 0.9)$ | Maximum proportion of masked context features. |
| `mask_min_trgt_share` | $(0.1, 0.3)$ | Minimum proportion of masked target features. |
| `mask_max_trgt_share` | $(0.3, 0.9)$ | Maximum proportion of masked target features. |

## A.3 MLP MODEL ARCHITECTURE

The Multi-Layer Perceptron (MLP) architecture employed in this work is designed to effectively transform input features. The transformation begins with a linear projection of the input $\mathbf{z}$, which can either be the raw input $\mathbf{x}$ or an embedding projection, into a hidden representation of dimensionality $h$:

$$\mathbf{h}^{(0)} = \mathbf{W}_1\mathbf{z} + \mathbf{b}_1.$$

The model then applies the hidden layers sequentially:

$$\mathbf{h}^{(l+1)} = \text{BatchNorm}\big(\text{Dropout}\big(\text{ReLU}\big(\mathbf{W}_{l+1}\mathbf{h}^{(l)} + \mathbf{b}_{l+1}\big)\big)\big),$$

where $l = 0, \ldots, L-1$. Here is the selected hyper-parameters for each dataset:

## B TRAINING DYNAMICS OF LINEAR MODEL

The loss is defined as

$$\mathcal{L}(W, W_p) = \frac{1}{2}\|X_1 W^\top W_p^\top - X W_a^\top \circ m_2\|_2^2 \tag{6}$$

with $X, X_1 \in \mathbb{R}^{N\times d}$ the input matrices and $X_1 = X \circ m_1$ the masked input that serves as input for the context encoder. And $W, W_p, W_a \in \mathbb{R}^{d\times d}$ the corresponding matrices from the context, target and predictor respectively. As so,

$$\begin{aligned}
\nabla_W \mathcal{L} = & \quad \frac{1}{2}\nabla_W \text{tr}[(X_1 W^\top W_p^\top - F_2)^\top (X_1 W^\top W_p^\top - F_2)] && (7)\\
= & \quad \frac{1}{2}\nabla_W \text{tr}[(W_p W X_1^\top - F_2^\top)(X_1 W^\top W_p^\top - F_2)] && (8)\\
= & \quad \frac{1}{2}\nabla_W [\text{tr}(F_{1,1}) - \text{tr}(F_{2,1}) - \text{tr}(F_{1,2}) + \text{tr}(F_{2,2})] && (9)
\end{aligned}$$

with $F_{1,1} = W_p W X_1^\top X_1 W^\top W_p^\top$, $F_{1,2} = W_p W X_1^\top F_2$, $F_{2,1} = F_{1,2}^\top$, $F_{2,2} = F_2^\top F_2$ and $F_2 = X W_a^\top \circ m_2$.

$$\begin{aligned}
\nabla_W \text{tr}(F_{1,1}) &= \nabla_W \text{tr}(W_p W X_1^\top X_1 W^\top W_p^\top) && (10)\\
&= \nabla_W \text{tr}(W X_1^\top X_1 W^\top W_p^\top W_p) && (11)\\
&= 2 W_p^\top W_p W X_1^\top X_1 && (12)
\end{aligned}$$

$$\begin{aligned}
\nabla_W \text{tr}(F_{1,2}) = \nabla_W \text{tr}(F_{2,1}) &= \nabla_W \text{tr}(W_p W X_1^\top F_2) && (13)\\
&= \nabla_W \text{tr}(W X_1^\top F_2 W_p) && (14)\\
&= W_p^\top F_2^\top X_1 && (15)
\end{aligned}$$

As so,

$$\begin{aligned}
\nabla_W \mathcal{L} &= \frac{1}{2}\nabla_W [\text{tr}(F_{1,1}) - \text{tr}(F_{2,1}) - \text{tr}(F_{1,2}) + \text{tr}(F_{2,2})] && (16)\\
&= W_p^\top W_p W X_1^\top X_1 - W_p^\top F_2^\top X_1 && (17)
\end{aligned}$$

and

$$\dot{W} = -W_p^\top W_p W X_1^\top X_1 + W_p^\top (X W_a^\top \circ m_2)^\top X_1 \tag{18}$$

Now,

$$\begin{aligned}
\nabla_{W_p} \text{tr}(F_{1,1}) &= \nabla_{W_p} \text{tr}(W_p W X_1^\top X_1 W^\top W_p^\top) && (19)\\
&= \nabla_{W_p} \text{tr}(W X_1^\top X_1 W^\top W_p^\top W_p) && (20)\\
&= 2 W_p W X_1^\top X_1 W^\top && (21)
\end{aligned}$$

$$\begin{aligned}
\nabla_{W_p} \text{tr}(F_{1,2}) = \nabla_{W_p} \text{tr}(F_{2,1}) &= \nabla_{W_p} \text{tr}(W_p W X_1^\top F_2) && (22)\\
&= \nabla_{W_p} \text{tr}(W X_1^\top F_2 W_p) && (23)\\
&= F_2^\top X_1 W && (24)
\end{aligned}$$

As so,

$$\nabla_{W_p}\mathcal{L} = \frac{1}{2}\nabla_{W_p}[\text{tr}(F_{1,1}) - \text{tr}(F_{2,1}) - \text{tr}(F_{1,2}) + \text{tr}(F_{2,2})] \tag{25}$$

$$= W_p W X_1^\top X_1 W^\top - F_2^\top X_1 W \tag{26}$$

and

$$\dot{W}_p = -\alpha_p \left[W_p W X_1^\top X_1 W^\top - (X W_a^\top \circ m_2)^\top X_1 W\right] \tag{27}$$

## C STABLE POINTS ANALYSIS

From Equation 16, we have that

$$\nabla_W \mathcal{L} = W_p^\top W_p W X_1^\top X_1 - W_p^\top M_2^\top W_a X^\top X_1 \tag{28}$$

and so the stable points satisfy $\nabla_W \mathcal{L} = 0$:

$$W^* \;=\; A_L\, W_a\, A_R. \tag{29}$$

where

$$A_R \;=\; X^\top X_1 \left(X_1^\top X_1\right)^{-1} \quad \text{and} \quad A_L \;=\; \left(W_p^\top W_p\right)^{-1} W_p^\top\, M_2^\top.$$

Under the assumption $W_a = \tau W$, we have that:

$$W = \tau A_L\, W\, A_R \tag{30}$$

$$\text{vec}(W) = \tau(A_R^\top \otimes A_L)\text{vec}(W) \tag{31}$$

$$[I - \tau(A_R^\top \otimes A_L)]\text{vec}(W) = 0 \tag{32}$$