Non-Parametric Representation Learning with Kernels: Project Report

Anushka Chaubey
ZDA24M012
GitHub | Overleaf

Abstract

Unsupervised and self-supervised representation learning has traditionally been dominated by neural networks. The paper [1] referenced for this project, proposes kernel-based models namely, Kernel Contrastive Learning and Kernel Autoencoders (Kernel AEs), as effective non-parametric alternatives.

This project implements and analyzes two novel kernel methods proposed in the paper[1]: (1) simple contrastive kernel learning and (2) spectral contrastive kernel learning. Building on theoretical foundations including a new representer theorem (Theorem: 1) for SSL, we demonstrate how these methods learn finite-dimensional embeddings in reproducing kernel Hilbert spaces (RKHS) while requiring only a subset of anchor points for efficient computation.

Through extensive experiments on IRIS and MNIST datasets, we compare our kernel SSL approaches against vanilla k-NN, PCA, and Kernel PCA baselines. While achieving comparable classification accuracy (96% on IRIS, 90-93% on MNIST), the kernel methods offer three key advantages: (1) significant dimensionality reduction from original feature spaces, (2) scalable inference through small sampled anchor datapoints for kernel matrices, and (3) extraction of semantically meaningful features rather than reliance on raw inputs.

The results validate that kernel SSL provides a viable alternative to deep learning for representation learning, particularly in scenarios where model interpretability, computational efficiency, or small data regimes are prioritized. Our implementation confirms the theoretical predictions from [1]

1 Introduction

The paper [1] referenced for this project analyzes non-parametric representation learning models based on kernels, addressing the surprising lack of exploration of such models outside the neural network literature. A key focus is on Self-Supervised Learning (SSL) using contrastive loss functions, which leverages implicit knowledge of what makes samples semantically close in a latent space without requiring explicit labels.

We define two kernel-based SSL models:

- A **simple contrastive loss** model (Definition 1)
- A **spectral contrastive loss** model (Definition 2)

Additionally, we introduce a **Kernel Autoencoder (AE)** model that embeds and reconstructs data.

Performance is evaluated on classification tasks, comparing:

- Vanilla k-NN on original features (baseline)
- PCA with k-NN

- Kernel PCA with k-NN
- Proposed methods:
  - Kernel SSL (simple contrastive kernel learning) with k-NN
  - Kernel SSL (spectral contrastive kernel learning) with k-NN
  - Kernel Autoencoders with k-NN

2 Problem Statement

We consider data $\mathbf{x} \in \mathbb{R}^d$, and aim to learn an embedding $f_W(\mathbf{x}) = W^\top \phi(\mathbf{x}) \in \mathbb{R}^h$, where $\phi : \mathbb{R}^d \to \mathcal{H}$ maps the input to a reproducing kernel Hilbert space (RKHS).

We formulate three models:

- Kernel Contrastive Learning
  - Simple Contrastive Loss
  - Spectral Contrastive Loss
- Kernel Autoencoder (Kernel AE)

Our goals are:

- To prove that embeddings lie in the finite-dimensional subspace $\mathcal{H}_X$ using a new representer theorem (Theorem: 1).
- To find representation kernel using only a subset of datapoints (anchors).
- To find optimal embeddings using custom loss functions (Defination: 3)
- To compare results (Fig: 1) with vanilla KNN

3 Theory

3.1 Representer Theorems

Theorem 1 (Representer Theorem for Representation Learning)**.** *Given data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, let $\mathcal{L}_X(\mathbf{w}_1, \ldots, \mathbf{w}_h)$ be a loss functional on $\mathcal{H}^h$ that remains unchanged under projection onto the finite-dimensional subspace $\mathcal{H}_X$ spanned by the data. Then, any minimizer of the regularized loss*

$$\mathcal{L}(\mathbf{w}_1, \ldots, \mathbf{w}_h) = \mathcal{L}_X(\mathbf{w}_1, \ldots, \mathbf{w}_h) + \lambda \|W\|_{\mathcal{H}}^2$$

lies in $\mathcal{H}_X$, i.e., $\mathbf{w}_1, \ldots, \mathbf{w}_h \in \mathcal{H}_X$.

Interpretation This theorem (Theorem: 1) tells us that even though kernel methods operate in a huge, possibly infinite-dimensional space (the RKHS), we don't actually need to search the entire space to find the best solution. If the loss and regularization only depend on the training data, then the optimal solution will always lie in the smaller subspace spanned by those data points.

In simpler terms, it says: *"You don't need to worry about the full infinite space, everything you need is already contained in the training data."* This makes kernel-based representation learning both efficient and practical.

3.2 Kernel Contrastive Learning

Definition 1 (Simple Contrastive Loss)**.** *Given triplets (x_i, x_i^+, x_i^-) consisting of anchor, positive, and negative samples, we learn a representation $f_W(x) = W^\top \phi(x)$ by optimizing the following objective:*

$$\mathcal{L} = \sum_{i=1}^{n} f_W(x_i)^\top \left(f_W(x_i^-) - f_W(x_i^+) \right) \quad \textit{subject to} \quad W^\top W = I_h$$

Theorem 2 (Closed-Form Solution)**.** *Let X, X^+, X^- be the anchor, positive, and negative samples. Define the kernel matrices:*

$$K = [k(x_i, x_j)], \quad K^- = [k(x_i, x_j^-)], \quad K^+ = [k(x_i, x_j^+)]$$

$$K^{--} = [k(x_i^-, x_j^-)], \quad K^{++} = [k(x_i^+, x_j^+)], \quad K^{-+} = [k(x_i^-, x_j^+)]$$

Then define:

$$K_3 = K^- - K^+, \quad K_\Delta = K^{--} + K^{++} - K^{-+} - (K^{-+})^\top$$

$$K_1 = \begin{bmatrix} K & K_3 \\ K_3 & K_\Delta \end{bmatrix}, \quad B = \begin{bmatrix} K_3 \\ K_\Delta \end{bmatrix} \cdot (KK^- - K^+)$$

$$K_2 = -\frac{1}{2}(B + B^\top)$$

Let A_2 be the top h eigenvectors of the matrix $K_1^{-1/2} K_2 K_1^{-1/2}$, and define $A = K_1^{-1/2} A_2$.

Then the embedding of a new point x^* is given by:

$$z^* = A^\top \begin{bmatrix} k(x^*, X) \\ k(x^*, X^-) - k(x^*, X^+) \end{bmatrix}$$

Interpretation This method (Defination: 1) shows that we can learn useful feature representations from data using only similarity information, no labels are needed. We take each data point and compare how similar it is to a positive version (closeness in space) and a negative example (e.g., a random other point). The goal is to make the representation closer to the positive and farther from the negative.

What's special here is that we don't need to train a deep neural network. Instead, we use kernel functions to measure similarities and then compute the final embeddings using a closed-form formula. This means that once we build a few kernel matrices, we can directly calculate all embeddings, no complex training needed. It's fast, doesn't require gradients, and works well especially when we don't have much labeled data.

Definition 2 (Spectral Contrastive Loss)**.** *We learn a representation $f_W(x) = W^\top \phi(x)$ by minimizing the following objective:*

$$\mathcal{L} = \sum_{i=1}^{n} \left[-2 f_W(x_i)^\top f_W(x_i^+) + \left(f_W(x_i)^\top f_W(x_i^-) \right)^2 \right] + \lambda \|W\|_{\mathcal{H}}^2$$

Theorem 3 (Optimization over Embeddings)**.** *Let K be the kernel matrix over the $3n$ samples (anchors x_i, positives x_i^+, and negatives x_i^-). Then, the objective can be rewritten in terms of embeddings $Z \in \mathbb{R}^{h \times 3n}$ as:*

$$\min_{Z \in \mathbb{R}^{h \times 3n}} \sum_{i=1}^{n} \left(-2 z_i^\top z_{i+n} + (z_i^\top z_{i+2n})^2 \right) + \lambda \operatorname{Tr}(Z K^{-1} Z^\top)$$

For any new point x^*, its embedding is given by:

$$z^* = Z K^{-1} k(X, x^*)$$

Interpretation. This method (Defination: 2) also learns useful features from data without using labels. Like before, we use triplets: an anchor, a positive (similar), and a negative (different) sample. The goal is to make the anchor close to the positive and far from the negative, but here, we do it in a more balanced way by using a squared similarity term for the negative.

The nice part is that we don't need to learn a big neural network, we can just represent all data points as a big kernel matrix and directly optimize the embeddings $z_1, \ldots, z_{3n}$. These embeddings are vectors that summarize each data point. After optimization, we can embed any new sample using a simple formula based on its kernel similarity with the training data. This makes the method fast, interpretable, and useful when labeled data is limited.

4 Datasets

We evaluate our methods on the following datasets:

- Used in the paper: IRIS
  The IRIS dataset is a classic dataset used for flower species classification based on physical measurements. It contains 150 samples with 4 numerical features each. It has 3 classes (*setosa*, *versicolor*, *virginica*) and is balanced, with 50 samples per class.
- New dataset: MNIST
  The MNIST dataset is a large collection of handwritten digit images commonly used for image classification tasks. It consists of 70,000 grayscale images of handwritten digits. Each image is 28x28 pixels (784 features when flattened). It has 10 classes (digits 0-9) and is approximately balanced, with roughly 6,000 samples per class in the training set.

5 Experiment

Vanilla KNN (3 neighbours), PCA with KNN, Kernel PCA (kernel='rbf', gamma=0.001) with KNN, Kernel AE with KNN was set up to find base evaluation. Our goal was to learn low-dimensional embeddings of data points via Spectral Contrastive Kernel Learning (SSL Spectral) (Defination: 1) and Simple Contrastive Kernel Learning (SSL Simple) (Defination: 2), enabling improved representation for downstream classification with K-Nearest Neighbors (KNN).

5.1 Datasets

We conducted experiments on two benchmark datasets IRIS and MNIST (Section: 4) with 70:30 split.

5.2 Experimental Setup

- **Device**: All computations were performed on Kaggle notebooks equipped with 2× NVIDIA T4 GPUs.
- **Preprocessing**: MNIST images were normalized to [0, 1]. For Iris, raw feature values were directly used.
- **Triplet Construction**: Positive and negative pairs (triplets) were constructed by sampling anchors, positives (closer in space), and negatives (random datapoint) from the training set.
- **Kernel**: The Radial Basis Function (RBF) kernel was applied with tuned gamma parameters of 0.5.
- **Embedding Dimensions**: We evaluated embedding dimensions of 512
- **Optimization**: The spectral kernel learning optimization was performed via Adam optimizer with learning rate 0.01 for 500 epochs.

5.3 Training Procedure

1. **Pair generation**: Triplets of anchor, positive, and negative samples were generated using contrastive learning using proximity in space.
2. **Kernel matrix calculation**: RBF kernels were computed between all anchor, positive, and negative samples.
3. **Kernel inverse approximation**: The inverse of the combined kernel matrix was approximated using a GPU-accelerated conjugate gradient method.
4. **Spectral embedding learning**: Meaningful embeddings were learned by minimizing a combined contrastive loss and kernel regularization term. (Defination: 3)
5. **Embedding new points**: The learned spectral embedding was applied to test data points by kernel projection. (Defination: 4)

Definition 3 (Loss Functions)**.** *The optimization objective combines a contrastive loss and a kernel regularization term:*

$$\mathcal{L}_{\textit{contrastive}} = \sum_{i=1}^{n} \left(-2 \cdot \langle \hat{\mathbf{z}}_i^a, \hat{\mathbf{z}}_i^p \rangle + (\langle \hat{\mathbf{z}}_i^a, \hat{\mathbf{z}}_i^n \rangle)^2 \right)$$

$$\mathcal{L}_{\textit{reg}} = \lambda \cdot \sum_{j=1}^{3n} \langle \hat{\mathbf{z}}_j, \mathbf{K}^{-1} \hat{\mathbf{z}}_j \rangle$$

$$\mathcal{L}_{\textit{total}} = \mathcal{L}_{\textit{contrastive}} + \mathcal{L}_{\textit{reg}}$$

Notation

- $\hat{\mathbf{z}}_i^a$, $\hat{\mathbf{z}}_i^p$, $\hat{\mathbf{z}}_i^n$ — *normalized embeddings of anchor, positive, and negative samples.*
- $\mathbf{K}^{-1}$ — *approximate inverse of the input kernel matrix.*
- n — *number of triplets used during training.*
- λ — *regularization coefficient.*
- $\hat{\mathbf{z}}_j$ — *j-th column of the normalized embedding matrix $\hat{\mathbf{Z}}$.*

Definition 4 (Embedding New Samples)**.** *To embed a new sample $\mathbf{x} \in \mathbb{R}^d$ using the learned kernel embedding, we proceed in two steps:*

Step 1: Kernel Computation

$$\mathbf{K}_x = \exp\left(-\gamma \cdot \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right), \quad \forall \mathbf{x}_j \in \mathcal{X}$$

Step 2: Embedding via Projection

$$\mathbf{z}_i = \left(\mathbf{Z} \cdot \mathbf{K}^{-1} \cdot \mathbf{K}_x^\top \right)^\top \in \mathbb{R}^h$$

Notation

- $\mathbf{K}_x \in \mathbb{R}^{N \times 3n}$ — *RBF kernel matrix between new points and original training set.*
- $\mathbf{K}^{-1} \in \mathbb{R}^{3n \times 3n}$ — *approximate inverse of training kernel.*
- $\mathbf{Z} \in \mathbb{R}^{h \times 3n}$ — *learned embedding matrix.*
- $\mathbf{z}_i \in \mathbb{R}^h$ — *final embedding of input $\mathbf{x}_i$.*
- γ — *RBF kernel bandwidth parameter.*
- h — *embedding dimensionality.*

5.4 Evaluation Metrics

- **Classification accuracy**: KNN classifier (k=3) was applied on the learned embeddings to evaluate classification performance.
- **Loss monitoring**: Contrastive and regularization losses were tracked during training for convergence analysis.

6 Results

Model	IRIS (Accuracy)	MNIST (Accuracy)
KNN	0.96	0.97
PCA + KNN	0.96	0.97
Kernel PCA + KNN	0.96	0.96
Kernel AE + KNN	0.86	0.80
SSL (simple) + KNN	0.96	0.93
SSL (spectral) + KNN	0.96	0.90

Table 1: Comparison of accuracy across models on IRIS and MNIST datasets

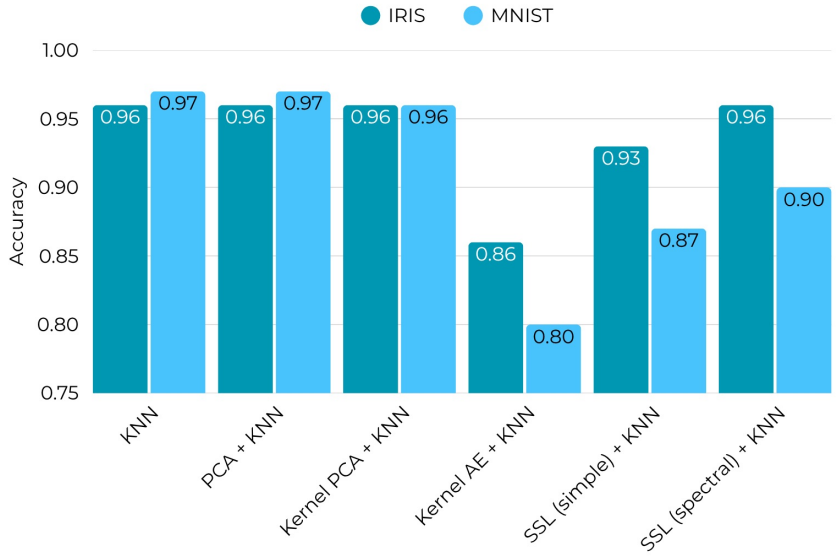

Figure 1: Bar Graph: Accuracy across models on IRIS & MNIST datasets

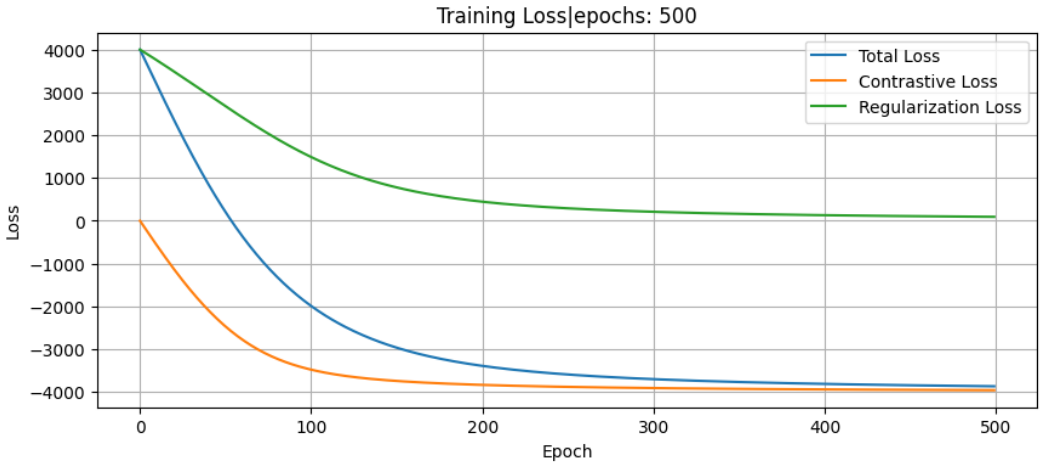

Figure 2: Training loss trends

The graph (Fig: 2) illustrates how the main contrastive loss rapidly decreases, showing that the model efficiently learns to differentiate positive and negative pairs. In contrast, the regularization loss decreases more gradually, reflecting a steady enforcement of model smoothness and complexity control. Together, this behavior signifies effective representation learning with balanced generalization.

7 Conclusion

Results in the original paper[1] demonstrate that the proposed kernel methods consistently match or outperform both the baseline and Kernel PCA. During the implementation for this project (Fig: 1), although the accuracy of kernel-based contrastive learning and autoencoders may not surpass vanilla k-NN, their comparable performance serves as a proof of concept, particularly for high-dimensional data, demonstrating that these methods can achieve similar results while offering critical advantages:

- **Reduced feature dimensionality**: Compresses inputs into compact embeddings without losing discriminative power.
- **Scalable inference**: Only requires a small anchor subset to learn the kernel embedding matrix, which can then be efficiently applied to all data.
- **Semantic feature extraction**: Uncovers meaningful patterns rather than relying on raw pixels and noisy inputs.

These benefits make kernel methods especially valuable as preprocessing steps for complex pipelines (e.g., deep neural networks) or in datasets with very high dimensional features.

References

[1] Pascal Esser, Maximilian Fleissner, and Debarghya Ghoshdastidar. Non-parametric representation learning with kernels, 2023.