Rank-N-Contrast: Learning Continuous Representations for Regression

Kaiwen Zha[1,*] **Peng Cao**[1,*] **Jeany Son**[2] **Yuzhe Yang**[1] **Dina Katabi**[1]

[1]MIT CSAIL [2]GIST

NeurIPS 2023 Spotlight

<u>Group 3</u>

Rishav Mukherji, Karan Bania, Tejas Agrawal, Arnav Goyal, Jinam Keniya

Introduction

- Regression tasks are one of the most fundamental real world problems.
- To carry out such tasks and make continuous value predictions, the widely utilised methods are distance-based loss functions such as L1 and L2 distance.

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

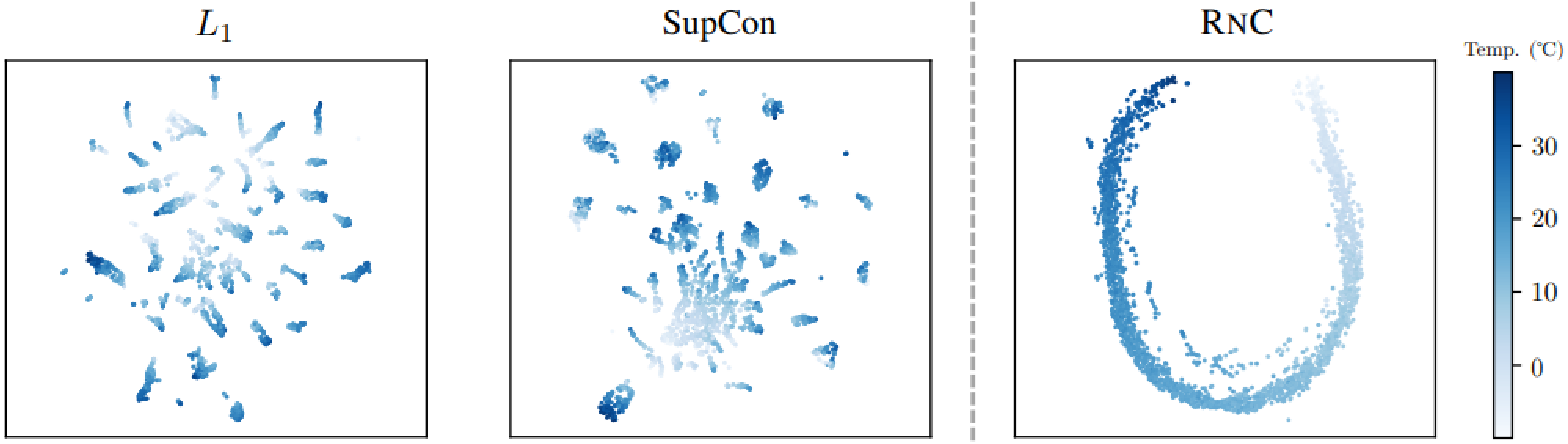
L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

- Earlier approaches concentrate on the final predictions in an end-to-end manner, without explicitly highlighting the representations acquired by the model.

Introduction

- Furthermore, there has been a lack of research regarding algorithms that capture the intrinsic continuity in data for regression
- To fill this gap the authors introduce Rank-N-Contrast (R*N*C), a novel framework for generic regression learning.

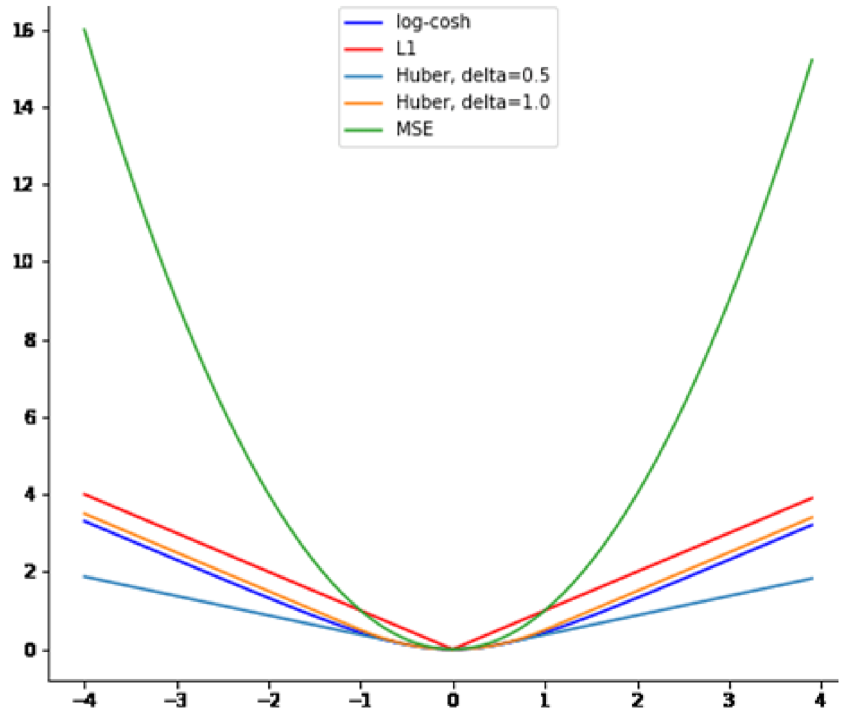


Contributions

- They propose R*N*C, a simple and effective method designed to learn continuous representations for regression.
- Extensive experiments are conducted on five diverse regression datasets across various domains such as vision, human-computer interaction, and healthcare. The results demonstrate the superior performance of R*N*C compared to state-of-the-art schemes.
- Further analysis reveals properties of R*N*C regarding its data efficiency, robustness to spurious targets and data corruptions, and improved generalization to unseen targets.

Related work

- Current deep regression models lack "regression-aware" representations due to the end-to-end training focus on final predictions, not representation continuity
- Several works casts regression as an ordinal classification problem using multiple binary classifiers based on ordered thresholds



$$L1LossFunction = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$$

$$Huber = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \hat{y}_i)^2 \qquad |y_i - \hat{y}_i| \leq \delta$$

$$MSE = \frac{1}{N}\sum_{i=1}^{i=N}(y\_true_i - y\_pred_i)^2$$

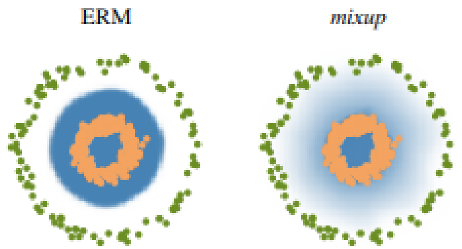
$$Huber = \frac{1}{n}\sum_{i=1}^{n}\delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right) \qquad |y_i - \hat{y}_i| > \delta$$

Related work

- C-Mixup adapts the original *mixup* by adjusting the sampling probability of the mixed pairs according to the target similarities

```python
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

[C-Mixup, NeurIPS 2022]

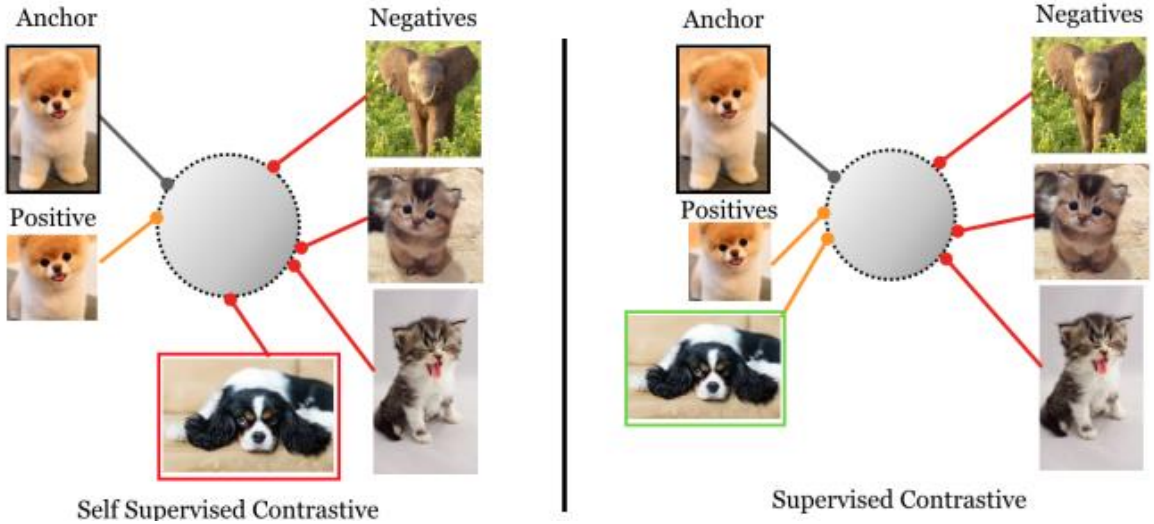
Algorithm 1 Training with C-Mixup

Require: Learning rates η; Shape parameter α
Require: Training data $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^N$
1: Randomly initialize model parameters θ
2: Calculate pairwise distance matrix P via Eqn. (6)
3: **while** not converge **do**
4: Sample a batch of examples $\mathcal{B} \sim \mathcal{D}$
5: **for** each example $(x_i, y_i) \in \mathcal{B}$ **do**
6: Sample (x_j, y_j) from $P(\cdot \mid (x_i, y_i))$ and λ from $\mathrm{Beta}(\alpha, \alpha)$
7: Interpolate (x_i, y_i), (x_j, y_j) to get $(\tilde{x}, \tilde{y})$ according to Eqn. (2)
8: Use interpolated examples to update the model via Eqn. (3)

[Original mixup, ICLR 2018]

Related work

- Contrastive learning excels in representation learning for classification
- The supervised version of contrastive learning, SupCon, has been shown to outperform the conventional cross-entropy loss
- Recent works adapt SupCon to tackle ordered labels in specific downstream applications



[SupCon, NeurIPS 2020]

Approach

1. Train a neural network consisting:
   a. a feature encoder - $f(\cdot) : X \to \mathbb{R}^{d_e}$
   b. a predictor - $g(\cdot) : \mathbb{R}^{d_e} \to \mathbb{R}^{d_t}$

1. Augmentations are applied

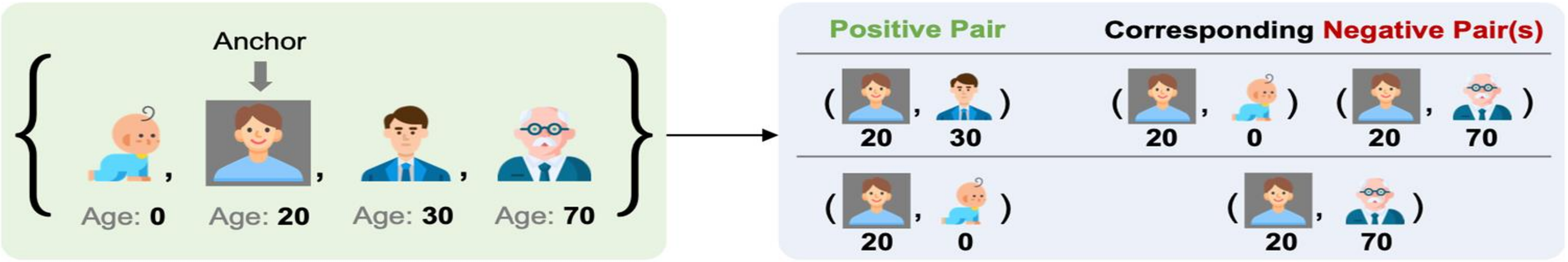
$$\tilde{\boldsymbol{x}}_{2n} = t(\boldsymbol{x}_n) \text{ and } \tilde{\boldsymbol{x}}_{2n-1} = t'(\boldsymbol{x}_n) \qquad \tilde{\boldsymbol{y}}_{2n} = \tilde{\boldsymbol{y}}_{2n-1} = \boldsymbol{y}_n$$

(t and t’ are 2 separate augmentations)

This creates the augmented batch $\{(\tilde{\boldsymbol{x}}_\ell, \tilde{\boldsymbol{y}}_\ell)\}_{\ell \in [2N]}$

Approach

3. Create – and + pairs!



(a) A Batch of Samples

(b) Pair Construction for RNC

$$\mathcal{S}_{i,j} := \{\boldsymbol{v}_k \mid k \neq i, d(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_k) \geq d(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_j)\}$$

They choose an anchor vi and introduce a set Si,j which denotes the set of samples which are higher rank than vj in terms of label distance wrt vi.

d(,) is the distance measure (eg. L1)

Approach

4. Loss function, basically, given an anchor, contrast each example with it's negatives to **impose an ordering.**

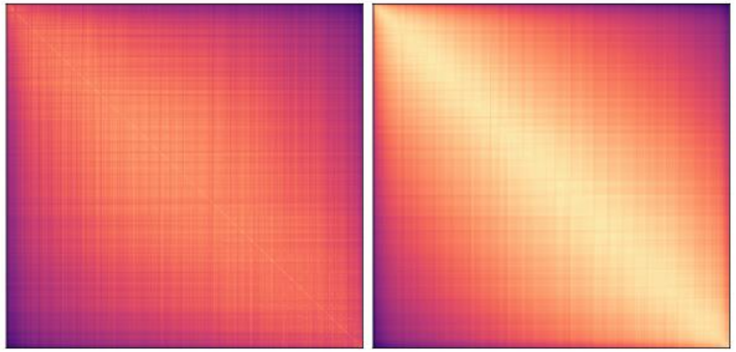
$$l_{\mathrm{RNC}}^{(i)} = \frac{1}{2N-1} \sum_{j=1,\ j\neq i}^{2N} -\log \frac{\exp(\mathrm{sim}(\boldsymbol{v}_i, \boldsymbol{v}_j)/\tau)}{\sum_{\boldsymbol{v}_k \in \mathcal{S}_{i,j}} \exp(\mathrm{sim}(\boldsymbol{v}_i, \boldsymbol{v}_k)/\tau)}$$

5. L(rnc) is then enumerating over all 2N samples as anchors to enforce the entire feature embeddings ordered according to their orders in the label space:

$$\mathcal{L}_{\mathrm{RNC}} = \frac{1}{2N} \sum_{i=1}^{2N} l_{\mathrm{RNC}}^{(i)} = \frac{1}{2N} \sum_{i=1}^{2N} \frac{1}{2N-1} \sum_{j=1,\ j\neq i}^{2N} -\log \frac{\exp(\mathrm{sim}(\boldsymbol{v}_i, \boldsymbol{v}_j)/\tau)}{\sum_{\boldsymbol{v}_k \in \mathcal{S}_{i,j}} \exp(\mathrm{sim}(\boldsymbol{v}_i, \boldsymbol{v}_k)/\tau)}.$$

Approach

6. Feature Ordinality (data points sorted by ground truth) & Correlation.



L_1 RnC

	Spearman's $\rho\uparrow$	Kendall's $\tau\uparrow$
L_1	0.822	0.664
RnC	**0.971**	**0.870**

Two qualitative metrics - Spearman's rho and Kendall's tau, both measures the strength of association between two ranked variables.

Theoretical Analysis –

- **Definition 1** (δ-ordered feature embeddings). For any $0 < \delta < 1$, the feature embeddings $\{\boldsymbol{v}_l\}_{l \in [2N]}$ are δ-ordered if $\forall i \in [2N], j, k \in [2N] \backslash \{i\}$,

$$\begin{cases} s_{i,j} > s_{i,k} + \frac{1}{\delta} & \text{if } d_{i,j} < d_{i,k} \\ |s_{i,j} - s_{i,k}| < \delta & \text{if } d_{i,j} = d_{i,k} \\ s_{i,j} < s_{i,k} - \frac{1}{\delta} & \text{if } d_{i,j} > d_{i,k} \end{cases} .$$

$$L^\star := \frac{1}{2N(2N-1)} \sum_{i=1}^{2N} \sum_{m=1}^{M_i} n_{i,m} \log n_{i,m}$$

Theoretical Analysis –

The paper proves these 3 theorems.

Theorem 1 (Lower bound of $\mathcal{L}_{\mathrm{RNC}}$). *$L^\star$ is a lower bound of $\mathcal{L}_{\mathrm{RNC}}$, i.e., $\mathcal{L}_{\mathrm{RNC}} > L^\star$.*

Theorem 2 (Lower bound tightness). *For any $\epsilon > 0$, there exists a set of feature embeddings such that $\mathcal{L}_{\mathrm{RNC}} < L^\star + \epsilon$.*

Theorem 3 (Main theorem). *For any $0 < \delta < 1$, there exist $\epsilon > 0$, such that if $\mathcal{L}_{\mathrm{RNC}} < L^\star + \epsilon$, then the feature embeddings are δ-ordered.*

Theoretical Analysis –

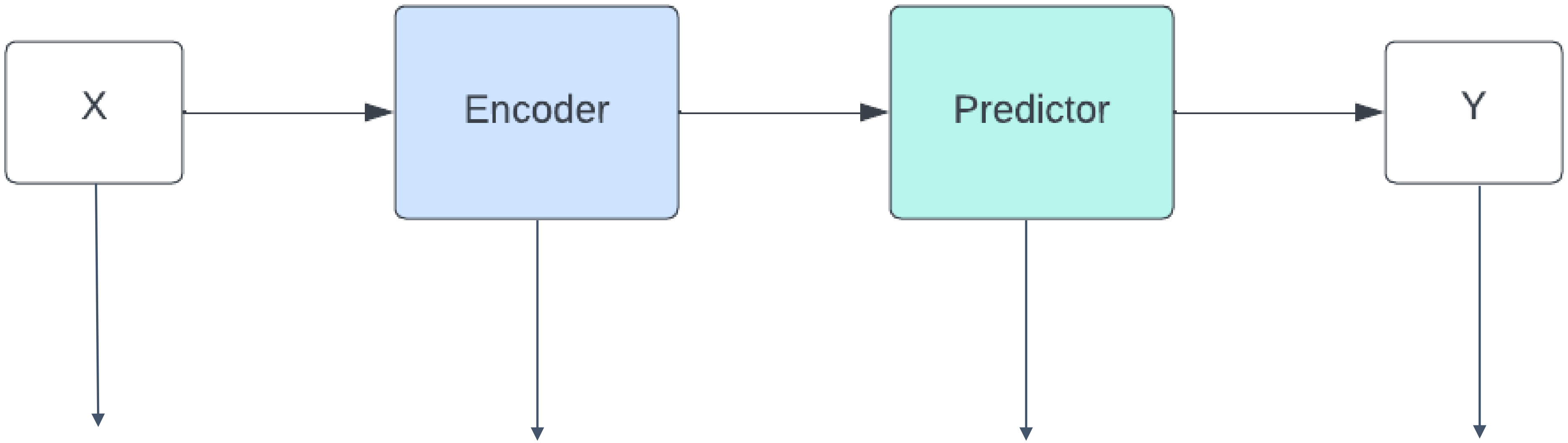
- Connections of a δ-ordered embedding space to **final performance** and **generalizability**.

- **Monotonic function** - a monotonic function (or monotone function) is a function between ordered sets that preserves or reverses the given order.

- **Rademacher Complexity** - In computational learning theory (machine learning and theory of computation), Rademacher complexity, measures richness of a class of sets with respect to a probability distribution.

$$2R(\mathcal{A}_i) + 4c_i\sqrt{\frac{2\ln(4/\epsilon)}{m}}$$

Methodology



Datasets Used
AgeDB
TUAB
SkyFinder
MPIIFaceGaze

Encoder is trained on the proposed $\mathbf{L}_{RNC}$ to obtain feature embedding

Predictor is then trained on $\mathbf{L}_1$ loss

MAE is used as the metric for evaluation except for MPIIFaceGaze which uses **Angular Error**.

Experiments Results

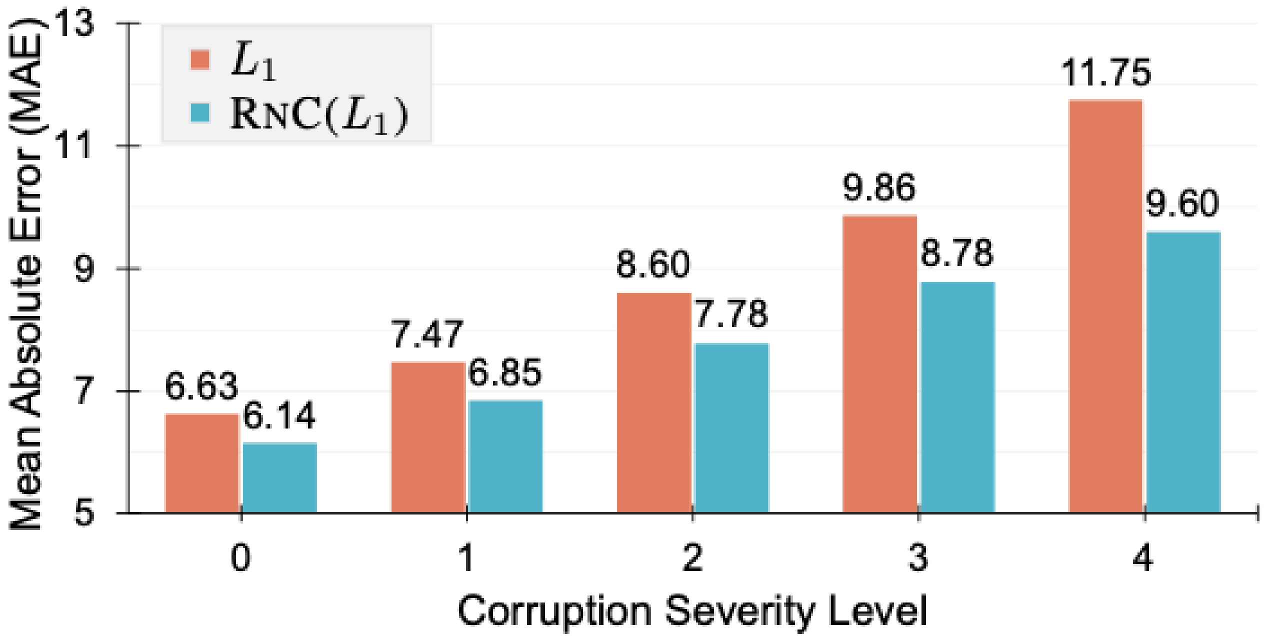
- Comparison to **State of the Art.**

Method	AgeDB	TUAB	MPIIFaceGaze	SkyFinder
Representation learning methods (Linear Probing):				
SimCLR [4]	9.59	11.01	9.43	4.70
DINO [3]	10.26	11.62	11.92	5.63
SupCon [25]	8.13	8.47	9.27	3.97
Representation learning methods (Fine-tuning):				
SimCLR [4]	6.57	7.57	5.50	2.93
DINO [3]	6.61	7.58	5.80	2.98
SupCon [25]	6.55	7.41	5.54	2.95
Regression learning methods:				
L_1	6.63	7.46	5.97	2.95
LDS+FDS [44]	6.45	–	–	–
L2CS-Net [1]	–	–	5.45	–
LDE [7]	–	–	–	2.92
RankSim [17]	6.51	7.33	5.70	2.94
Ordinal Entropy [50]	6.47	7.28	–	2.94
RnC(L_1)	**6.14**	**6.97**	**5.27**	**2.86**
Gains	**+0.31**	**+0.31**	**+0.18**	**+0.06**

Experiments Results

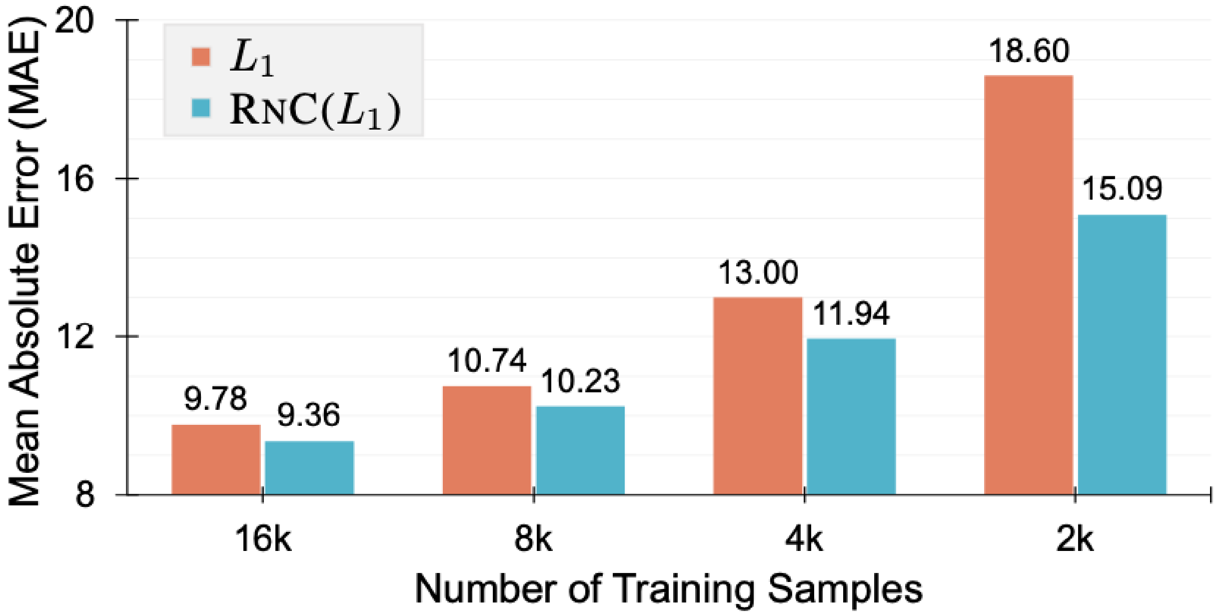
	AgeDB		TUAB		MPIIFaceGaze		SkyFinder	
Metrics	MAE↓	R^2↑	MAE↓	R^2↑	Angular↓	R^2↑	MAE↓	R^2↑
L_1	6.63	0.828	7.46	0.655	5.97	0.744	2.95	0.860
RNC(L_1)	**6.14** (+0.49)	**0.850** (+0.022)	**6.97** (+0.49)	**0.697** (+0.042)	**5.27** (+0.70)	**0.815** (+0.071)	2.86 (+0.09)	**0.869** (+0.009)
MSE	6.57	0.828	8.06	0.585	6.02	0.747	3.08	0.851
RNC(MSE)	**6.19** (+0.38)	**0.849** (+0.021)	**7.05** (+1.01)	**0.692** (+0.107)	**5.35** (+0.67)	**0.802** (+0.055)	**2.86** (+0.22)	**0.869** (+0.018)
HUBER	6.54	0.828	7.59	0.637	6.34	0.709	2.92	0.860
RNC(HUBER)	**6.15** (+0.39)	**0.850** (+0.022)	**6.99** (+0.60)	**0.696** (+0.059)	**5.15** (+1.19)	**0.830** (+0.121)	**2.86** (+0.06)	**0.869** (+0.009)
DEX [36]	7.29	0.787	8.01	0.537	5.72	0.776	3.58	0.778
RNC(DEX)	**6.43** (+0.86)	**0.836** (+0.049)	**7.23** (+0.78)	**0.646** (+0.109)	**5.14** (+0.58)	**0.805** (+0.029)	**2.88** (+0.70)	**0.865** (+0.087)
DLDL-v2 [14]	6.60	0.827	7.91	0.560	5.47	0.799	2.99	0.856
RNC(DLDL-v2)	**6.32** (+0.28)	**0.844** (+0.017)	**6.91** (+1.00)	**0.697** (+0.137)	**5.16** (+0.31)	**0.802** (+0.003)	**2.85** (+0.14)	**0.869** (+0.013)
OR [33]	6.40	0.830	7.36	0.646	5.86	0.770	2.92	0.861
RNC(OR)	**6.34** (+0.06)	**0.843** (+0.013)	**7.01** (+0.35)	**0.688** (+0.042)	**5.13** (+0.73)	**0.825** (+0.055)	**2.86** (+0.06)	**0.867** (+0.006)
CORN [40]	6.72	0.811	8.11	0.597	5.88	0.762	3.24	0.819
RNC(CORN)	**6.44** (+0.28)	**0.838** (+0.027)	**7.22** (+0.89)	**0.663** (+0.066)	**5.18** (+0.70)	**0.820** (+0.058)	**2.89** (+0.35)	**0.862** (+0.043)

Robustness to Data Corruptions



- Generated corruptions on AgeDB test set using ImageNet-C benchmark at various severity levels.
- RNC consistently more robust and shows less performance degradation.

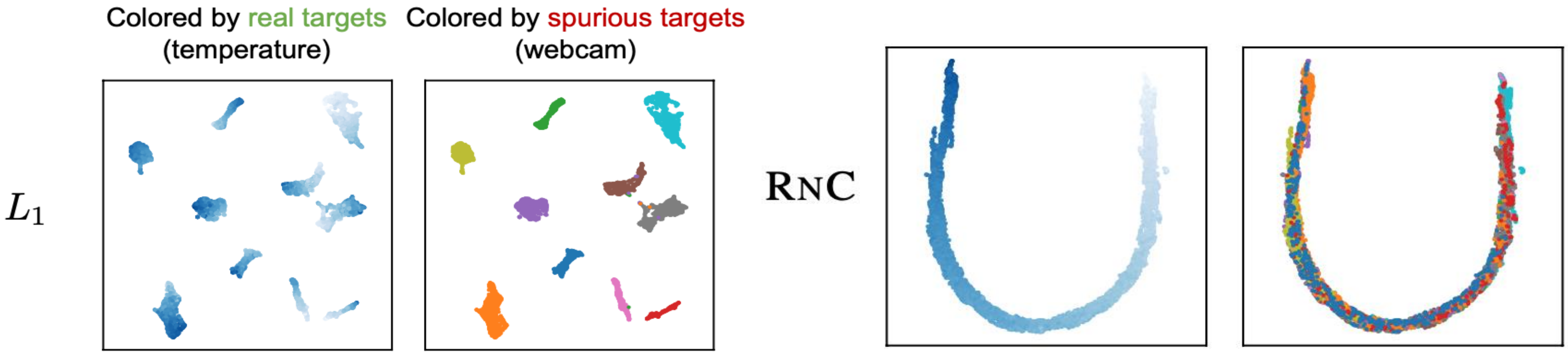
Resilience To Reduced Training Data



- Subsampled **IMDB-WIKI** to generate training sets of various sizes.
- RNC displays less performance degradation with decreasing training samples.

Ablation study –

- Robustness to spurious targets.



Ablation study -

- Is RnC actually good or is it the 2-stage training?

Method	End-to-End	Two-Stage
L_1	6.63	6.68
MSE	6.57	6.57
Huber	6.54	6.63
DEX [36]	7.29	7.42
DLDL-v2 [14]	6.60	7.28
OR [33]	6.40	6.72
CORN [40]	6.72	6.94
RnC(L_1)	–	**6.14**