Beyond Binning: Soft Task Reformulation for Deep Regression

Lawrence Stewart
ENS & INRIA

Francis Bach
ENS & INRIA

Quentin Berthet
Google DeepMind

Abstract

Whilst neural networks are powerful predictors, it has been observed and theoretically analyzed that training such models by minimizing the square loss can lead to suboptimal results on regression problems, where the targets are real-valued. In this work, we propose a novel method aimed at improving test-time performance of neural networks on regression tasks. Our method is based on casting this task in a different fashion, using a target encoder, and a prediction decoder, inspired by approaches in classification and clustering. We demonstrate our method on a wide range of real-world datasets.

1 INTRODUCTION

Neural network architectures have become ubiquitous in machine learning, becoming the de facto go-to models for a wide array of tasks. This is particularly true for classification tasks, where the goal is to predict a discrete label based on observed features, e.g., in image classification Krizhevsky et al. (2012); He et al. (2016), language modeling Sutskever et al. (2014); Bahdanau et al. (2015); Vaswani et al. (2017), and audio generation Borsos et al. (2023); Dieleman et al. (2016). The amount of scientific work applying neural networks to classification tasks significantly outweighs that for regression problems —where the objective is to predict a real-valued target $y \in \mathbb{R}^m$— see e.g., (Stewart et al., 2023) and references therein. Yet, neural networks have obtained state-of-the-art results on numerous regression problems, such as pose estimation, point estimation, and robotics (Sun et al., 2013; Toshev and Szegedy, 2014; Belagiannis et al., 2015; Liu et al., 2016).

The reformulation of regression problems as a classification, via a discretization (commonly known as "binning"), has been a recent point of interest when training neural networks (Stewart et al., 2023; Pintea et al., 2023; Guha et al., 2024). Such an approach transforms real-valued labels into one-hot vectors, enabling optimization of neural network weights through cross-entropy loss minimization rather than the typical square loss, used in regression. Real-valued predictions can be obtained from the classification model's predicted distribution by calculating the expected value over the bin midpoints. Interestingly, such discretization techniques often yield superior performance, despite cross-entropy loss lacking any inherent notion of distance between classes McCarter (2024). This behavior has been reported across a range of disciplines, e.g., computer vision (Zhang et al., 2016; Van Den Oord et al., 2016), robotics (Rogez et al., 2017; Akkaya et al., 2019), reinforcement learning (Schrittwieser et al., 2020; Farebrother et al., 2024), biology (Gao et al., 2024; Picek et al., 2024), among others (Lee et al., 2024; Abe et al., 2023; Ansari et al., 2024).

Understanding the cause of this pattern remains an open research problem. Analyzing the gradient dynamics of over-parametrized neural networks (Chizat and Bach, 2018; Chistikov et al., 2023; Boursier et al., 2022), (Stewart et al., 2023) shows that the implicit bias of models trained on the square loss can lead to convergence to spurious minima; reformulating the problem to classification was observed to alleviate the underfitting due to a change in the implicit bias. Neural networks have been observed to under-perform on regression problems due to their bias to overly-smooth solutions and the lack of robustness of dense multilayer perceptron (MLP) layers to uninformative features, supporting the prior claim (Grinsztajn et al., 2022). There are still limitations to reformulating regression problems as classification, including excessive quantization in the outputs of the model and inefficient binning of the target space, which can harm the test-time performance and hurt training performance.

In this work, we introduce a novel generalization of the *"regression as classification"* reformulation (Stewart et al., 2023). We propose using a target encoder-

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

decoder model, which can (1) generate a distributional representation of the regression targets (an encoding), and (2) decode the distributional representation back into the original target space. Our proposed target encoder-decoder (abbreviated to "target codec"), can be initialized in a fashion that can be viewed as a probabilistic latent model, and optionally trained in an unsupervised manner to learn a target space encoding, providing a smoother alternative to the traditional one-hot encodings generated from binning.

In addition to our novel "model-based" task reformulation, we explore the benefits of interpolating between different objective functions i.e., regression, classification and target reconstruction. In conclusion, we propose an **end-to-end** framework that involves jointly training the neural network in conjunction with the target codec, in order to solve both the original regression problem and the reformulated (soft) classification problem.

Our methodology offers several advantages: firstly, we empirically observe that our target codec yields notable improvements in test-time regression performance, compared to baseline methods. One of the explanations for this observed improvement is that embedding the low-dimensional target space (especially when it is scalar) into an intermediate continuous space (distributions over k classes) improves the training dynamics when using high-dimensional features $x \in \mathcal{X}$. This echoes insights from recent literature exploring the implicit bias induced by gradient based training of neural networks (Chizat and Bach, 2018; Chistikov et al., 2023; Boursier et al., 2022; Stewart et al., 2023). Moreover, such gains are achievable with only a simple target codec architecture —namely a logistic model for the encoder and a linear mapping for the decoder— making the approach both computationally efficient and straightforward to integrate into existing frameworks.

Main contributions. In this work, we introduce a general framework for training neural networks on supervised regression tasks. To summarize, we make the following contributions:

- We introduce a novel generalization to reformulating regression as classification, via a light-weight target encoder-decoder model. Such a model can be initialized as a probabilistic latent model (with frozen weights), or trained in an unsupervised manner, clustering the target space.
- Differing from prior works (Stewart et al., 2023; Shah et al., 2022; Pintea et al., 2023; Zhang et al., 2023), we propose interpolating between the reformulated classification and original regression objective. More precisely, we propose an end-to-end framework which involves jointly training both the neural network and target codec. We break our method into components, in order to ablate the gains yielded from different terms.
- We showcase the improvements over existing approaches that our method obtains for key regression metrics, across a wide range of real world datasets, data modalities and model architectures.

Related Work. (Stewart et al., 2023) explore reformulating regression as classification when training neural networks, providing theoretical insights into how implicit bias affects neural network features, and empirical evidence demonstrating the benefits of task formulation. Both (Guha et al., 2024; Pintea et al., 2023) build on this work; the former applies the method to conformal prediction, while the latter empirically shows that task reformulation can aid in imbalanced regression datasets, a challenge also explored in other recent studies (Ren et al., 2022; Yang et al., 2021; Gong et al., 2022). Our work relates closely to that of (Stewart et al., 2023), building upon the idea of task reformulation via binning (which we compare as one of the baselines methods). Our proposed target codec can be viewed as a generalization of task reformulation. However, our method has multiple distinctions: our reformulation is smooth, and moreover, can be optionally learned. Furthermore, our end-to-end approach includes the original square loss term, which we see via ablation brings performance gains. A first exploration of reformulations of regression problems for linear models is found in (Torgo and Gama, 1996). More recently a proposed reformulation of regression as classification and learning considers an ensemble of random forest classifiers (which can be combined to obtain a real prediction) (Ahmad et al., 2018). Whilst both of these works share the reformulating regression as classification, they are not concerned with neural networks. Similarly, learning a set of regressors from a neural network's feature, in order to predict a binary encoding of a real value is considered in (Shah et al., 2022). This approach differs from ours in multiple fashions, i) we reformulate to a classification problem, ii) our encoding-decoding is smooth and trainable, iii) we do not require ensembles, iv) we bridge losses from both problem formulations.

Notations. We denote by $\mathcal{X}$ a general space of features, and by $\mathbb{R}^m$ the canonical real vector space of dimension m for some positive integer $m \geq 1$, and e_i the i-th element of its canonical basis (i.e., the one-hot vector for label i). For any positive integer k, we denote by $[k]$ the finite set $\{1, \ldots, k\}$, and by $\Delta_k \subset \mathbb{R}^k$ the unit simplex in dimension k, of vectors with nonnegative coefficients that sum to 1. It is the convex hull of $e_1, \ldots, e_k$, and the space of discrete probabilities

over k elements. We denote by H the entropy function from Δ_k to $\mathbb{R}$, defined for any $p \in \Delta_k$ by $H(p) = -\sum_{i\in[k]} p_i \log(p_i)$, and by KL the associated Kullback-Leibler divergence, defined for any $p, q \in \Delta_k$ by $\mathsf{KL}(p,q) = \sum_{i\in[k]} p_i \log(p_i/q_i)$.The softmax function from $\mathbb{R}^k$ to Δ_k is defined for $x \in \mathbb{R}^k$, element-wise for all $i \in [k]$ by $\text{softmax}(x)_i = \exp(x_i)/\sum_{j\in[k]} \exp(x_j)$.

2 OVERVIEW: REGRESSION WITH NEURAL NETWORKS

In regression, the aim is to infer a potentially multivariate continuous target $y \in \mathbb{R}^m$ based on observed features $x \in \mathcal{X}$. This supervised learning problem can be tackled by using a parametrized predictor function that can be trained on a dataset of coupled examples $(x_i, y_i) \in \mathcal{X} \times \mathbb{R}^m$, $i \in [n]$. In this work, we concern ourselves with training a neural network f_η parameterized by learnable weights $\eta \in \mathbb{R}^p$. Throughout this section, we will provide a brief overview of existing methodologies, which will later form the baselines we benchmark our approach against.

2.1 Least-Squares Regression: [Baselines 1-2]

Perhaps the most commonplace approach for regression is to train the model f_η to directly predict y, by minimizing a loss function of the form $\ell(y,z) = L(y-z)$, where $z = f_\eta(x)$:

$$\min_\eta \ \mathbf{E}_{(x,y)}\big[L(y - f_\eta(x))\big]. \tag{1}$$

Typically $L(y,z)$ is the square loss $L(y,z) = \|y-z\|_2^2$. Prior works have explored robust choices of L, such as the Huber loss Huber (1964) (which we also include as a baseline), or even nonconvex functions (see, e.g., Barron, 2019, and references therein). As shown in earlier work, implicit bias in regression sometimes leads to underfitting Grinsztajn et al. (2022); Stewart et al. (2023).

2.2 Classification: [Baseline 3]

For the case of scalar targets $y \in \mathbb{R}$, an alternative existing approach is to discretize the target space, and reformulate the regression task as a classification problem. This can be done by partitioning the target space into uniform sized bins, with midpoints $c_1, \dots, c_k \in \mathbb{R}$. For each scalar target $y \in \mathbb{R}$, one can assign a new one-hot label $y' \in \Delta_k$, based on which of the k bins y falls into. For further information on binning, see (Torgo and Gama, 1996; Stewart et al., 2023; Bhat et al., 2021). A classification model, producing logits $g_\theta : \mathcal{X} \to \mathbb{R}^k$, is then trained to minimize the cross-entropy loss between one-hot targets y' and $\pi_\theta(x) = \text{softmax}(g_\theta(x))$. If a real-valued prediction is necessary, this can be obtained by calculating $\hat{y}(x) = c^T \pi_\theta(x)$, where $c = (c_1, \dots, c_k) \in \mathbb{R}^k$. It has been observed that this simple methodology can yield superior results when training neural networks, (compared to least squares regression), despite the cross-entropy loss having no notion of class ordering (McCarter, 2024). Recent works attribute this phenomenon to factors such as implicit bias of over-parameterized models (Chizat and Bach, 2018; Chistikov et al., 2023; Boursier et al., 2022; Stewart et al., 2023) and target distribution shift (Pintea et al., 2023).

3 PROPOSED FRAMEWORK

3.1 Motivation: Viewing Class Reformulation as Target Encoding-Decoding

Our approach offers a new perspective: that the reformulation introduced in Section 2.2 can be interpreted as an encoding-decoding process over the target space.

In the more general case of $y \in \mathbb{R}^m$, one can consider a map $\psi_c : \mathbb{R}^m \to \Delta_k$, parameterized by k centroids $c_1, \dots, c_k \in \mathbb{R}^m$, which maps real-valued labels to one-hot encodings. Such a discretization can be achieved by mapping each y to the one-hot label representing the nearest center[1] i.e., $\psi_c(y) = e_i$, where $i = \text{argmin}_{i\in[k]} \|y - c_i\|_2^2$. For this reason, one can consider the mapping ψ_c as a **target encoder**, encoding the targets into one-hot vectors.

Again, a classification model, producing logits $g_\theta : \mathcal{X} \to \mathbb{R}^k$, is then trained by minimizing a classification loss between the encoded target $\psi(y)$ and $\pi_\theta(x) = \text{softmax}(g_\theta(x))$, such as the Kullback-Leibler divergence (KL divergence)

$$\min_\theta \ \mathbf{E}_{(x,y)}\big[\mathsf{KL}(\psi_c(y) \| \pi_\theta(x))\big]. \tag{2}$$

We note that up to a constant, this is equivalent to the more common cross-entropy loss, as the entropy term in the KL-divergence is zero for one-hot labels:

$$\begin{aligned}&\min_\theta \ \mathbf{E}_{(x,y)}\big[-\psi_c(y)^\top \log \pi_\theta(x)\big] \\ &= \min_\theta \ \mathbf{E}_{(x,y)}\big[-\psi_c(y)^\top \log \text{softmax}(g_\theta(x))\big].\end{aligned}$$

Given a a probability distribution $\pi_\theta(x)$, obtained from the trained classification model's logits, one can perform a **decoding** to obtain a prediction in $\mathbb{R}^m$ (the original target space), by calculating $c^T \pi_\theta(x)$. In other words, binning technique of (Torgo and Gama, 1996; Stewart et al., 2023) (and generalizations to

[1] This approach can be also be viewed as vector quantization in the target space (Van Den Oord et al., 2017).

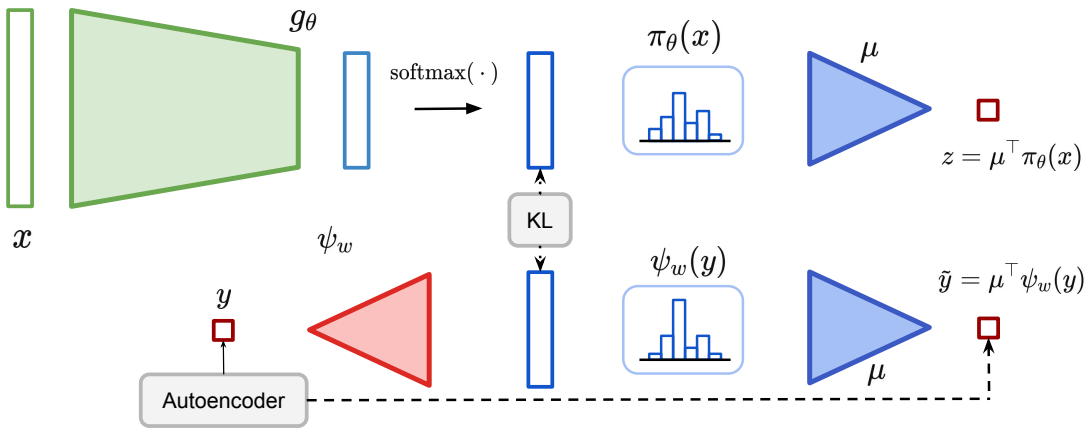

Figure 1: Framework description. Our framework is based on a **target encoder** ψ_w (in red) that yields for each y an encoded distribution $\psi_w(y)$ over k classes. A **classification model** $\pi_\theta = \text{softmax}(g_\theta)$ is trained with a KL objective on this distribution. A **decoder model** μ (in blue) decodes this distribution in the target space $\mathbb{R}^m$. The target encoder and decoder can be trained using an auto-encoding loss, jointly with the neural network using our end-to-end training objective (see Section 3.4, Equation 8).

higher dimensional examples), as a **target encoding-decoding**, where the neural network objective is shifted to a classification problem in the encoded target space. This method is illustrated in Figures 1 and 2. We make the following remarks:

1. The above target encoder and decoder mappings both shared the same parameters c, but one could equally consider independent parameterizations for both mappings, i.e., ψ_w (encoder parameterized by $w \in \mathbb{R}^{k\times m}$) and $\mu \in \mathbb{R}^{k\times m}$ (decoder).

2. In the above, the encoder function only maps onto the extremal points of Δ_k, i.e., produces only one-hot encodings. One could equally consider a soft mapping ψ_w whose range is all of Δ_k, encoding into probability vectors.

3. One can train on both classification and regression objectives jointly. Moreover, the target encoder and decoder weights can also be learned.

In this section, we introduce our novel *end-to-end* framework for training neural networks on regression problems, which is founded on contributions relating to the remarks above. Our approach can be viewed as: i) a generalization to task reformulation via a target codec, and ii) jointly optimizing multiple objective functions, bridging problem formulations. For clarity of exposition, we present our work as distinct modular contributions, building on one another, which when combined in Section 3.4, result in our proposed *end-to-end* framework. This also allows for systematic ablation studies to verify the contribution of each component to overall performance gains.

3.2 Proposal I: Soft Target Encoder-Decoder

The first generalization that we propose in this work is to modify the classification method, by using more general target encoders that utilize the whole simplex (not only one-hot vectors). In particular, to generalize binning for k centroid $c = (c_1, \ldots, c_k) \in \mathbb{R}^{k\times m}$, we propose a *soft target encoder*, (akin to performing a soft binning of the target space in the case of $m = 1$). Similarly to the previous approach, a classification model $\pi_\theta = \text{softmax}(g_\theta)$ is then trained on these soft labels using the KL divergence as described in Equation (2).

One way to implement this *soft partition* is by taking a target encoder that approximates the one-hot binning by replacing max by softmax:

$$\psi_c(y) = \text{softmax}\big(-\tfrac{\|c_1-y\|_2^2}{2\sigma^2}, \ldots, -\tfrac{\|c_k-y\|_2^2}{2\sigma^2}\big) \tag{3}$$

$$= \text{softmax}\left(\tfrac{c_1^\top y - \frac{1}{2}\|c_1\|_2^2}{\sigma^2}, \ldots, \tfrac{c_k^\top y - \frac{1}{2}\|c_k\|_2^2}{\sigma^2}\right), \tag{4}$$

for $\sigma > 0$. The two representations are mathematically equivalent (all k values differ only by $\|y\|_2^2/2\sigma^2$, and the softmax function is invariant by constant shifts). Looking at Equation (4), one can introduce a trivial change of variable to w, allowing for the encoder to conveniently take the following form of a linear-layer followed by a softmax:

$$\psi_w(y) = \text{softmax}(w_1^\top y + w_2)\,. \tag{5}$$

The formulation in Equation (4) is connected to a classical probabilistic interpretation of softmax regression by a generative model (see, e.g., Bach, 2024, Section 14.2), since we then have $\psi_w(y)_i = \mathbf{P}(Z = i|Y = y)$, in a probabilistic model with a latent variable $Z \in [k]$, and isotropic Gaussian class-conditional densities with

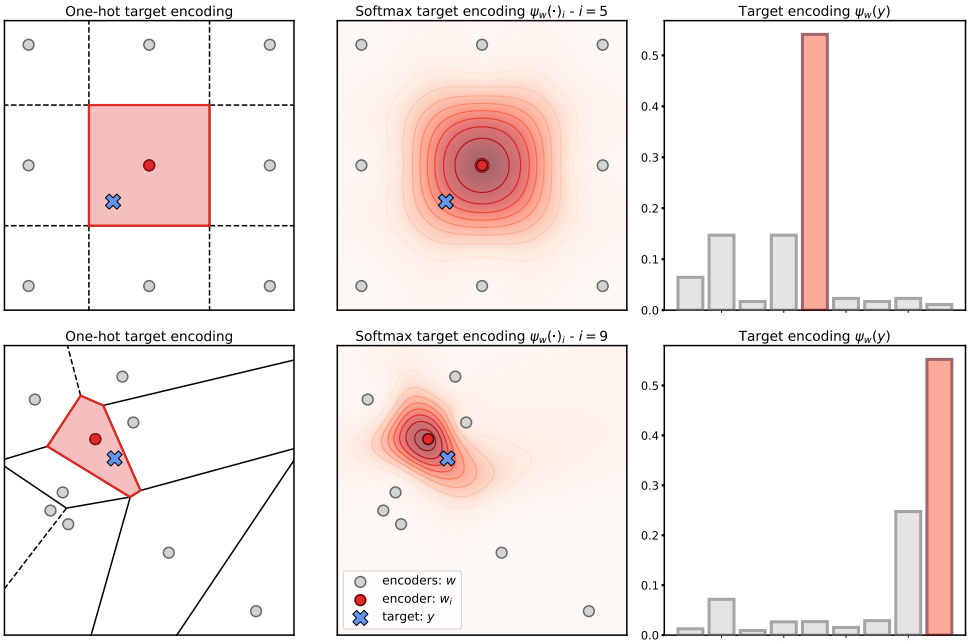

Figure 2: Embedding and binning the target space $\mathbb{R}^m$ (here $m = 2$) into Δ_k (here $k = 9$), for both a fixed grid of encoders (**Top**) and a learnt encoder (**Bottom**). For both cases we display the encoders, including an highlighted one, for a fixed $i \in [k]$ and a target $y \in \mathbb{R}^m$ (blue cross). We illustrate first *hard binning* (**Left**) where y (and any y in the same highlighted region) is assigned to one class (via a one-hot), and *soft binning* both with the contour plot of $\psi_w(\cdot)_i$ for one $i \in [k]$ (**Center**), and $\psi_w(y)$ as a distribution in Δ_k (**Right**).

mean c_i and variance $\sigma^2 I$ for the distribution of y given $Z = i$. This approach, in its full generality, extends upon soft labelling methods used by, e.g., (Imani and White, 2018; Farebrother et al., 2024).

The prediction model $\pi_\theta = \text{softmax}(g_\theta)$ is then trained by minimizing the KL divergence between $\pi_\theta(x)$ and $\psi_w(y)$, both in Δ_k as in Equation (2).

3.3 Proposal II: Pre-trained Target Encoder-Decoder

The second proposal we make is a further generalization on this method, by pre-training a target encoder-decoder (ψ_w, μ), instead of hand-picking it, e.g., by minimizing an auto-encoding objective:

$$\min_{w,\mu} \mathbf{E}_y \left[L(y - \mu^\top \psi_w(y)) \right], \tag{6}$$

and then to use this frozen target encoder to generate soft-label targets $\psi_w(y)$, to train the classification model $\pi_\theta = \text{softmax}(g_\theta)$ as in Equation (2) (Stage 2).

Note that the first stage can be done without access to the features $x \in \mathcal{X}$, and could even be performed with synthetic data (e.g., uniform sampling on the target space if it is compact). To generalize hand-picked soft encoders, it can be chosen as a simple model, with architecture

$$\psi_w(y) = \text{softmax}(w_{\text{lin}}^\top y + w_{\text{bias}}) .$$

Naively minimizing the auto-encoder objective in Stage 1 can afflict an implicit bias to the encoder, and yield close-to-uniform $\psi_w(y)$. To avoid this effect we can penalize the entropy, that is, minimize instead

$$\min_{w,\mu} \mathbf{E}_y \left[L(y - \mu^\top \psi_w(y)) - \alpha H(\psi_w(y)) \right], \tag{7}$$

with a positive parameter $\alpha > 0$. During methodology development, we also explored using the isotropic Gaussian log-likelihood (i.e., the K-Means objective) in place of the auto-encoder objective, but found it empirically less performant than our proposed approach.

Initialization of encoder-decoder. For $m = 1$, we propose initializing the decoder weights μ as a uniform spacing over the target space, where δ_μ denotes the magnitude of the spacing. We remark that this closely resembles discretized binning Stewart et al. (2023). For the encoder weights, we propose setting $\sigma = \lambda_\sigma \cdot \delta_\mu$, e.g,. $\lambda_\sigma = 1$, and initializing with $c = \mu$ using the connection between Equations (4) and (5). For this initialization, the autoencoder loss $L(y - \mu^\top \psi_w(y))$ goes to 0 for growing values of k, but we show experimentally that it is not necessary. For $m > 1$, we suggest using a clustering algorithm such as K-means++ (Arthur and Vassilvitskii, 2006) to initialize μ. In this case δ_μ would refer to average intra-cluster distance, and one can initialize the encoder weights in the same fashion as for $m = 1$.

3.4 Proposal III: End-to-End Approach (Main Proposal)

We now proceed to present our proposed *end-to-end* framework. In particular, we explore training the neural network jointly with the target codec, by minimizing the following loss, with scalar hyperparameters $\lambda_{\mathsf{auto}}, \lambda_{\mathsf{KL}}, \lambda_{\mathsf{pred}} \geq 0$:

$$\begin{aligned}\min_{w,\mu,\theta} \quad & \lambda_{\mathsf{auto}} \mathbf{E}_y \left[L(y - \mu^\top \psi_w(y)) - \alpha H(\psi_w(y)) \right] \\ & + \lambda_{\mathsf{KL}} \mathbf{E}_{(x,y)} \left[\mathsf{KL}\big(\psi_w(y) \| \pi_\theta(x)\big) \right] \\ & + \lambda_{\mathsf{pred}} \mathbf{E}_{(x,y)} \left[L(y - \mu^\top \pi_\theta(x)) \right]. \end{aligned} \tag{8}$$

Differing from previous approaches (Torgo and Gama, 1996; Stewart et al., 2023), in this approach the neural network is trained to produce features which are suitable for classification when encoded, and for regression when subsequently decoded. Crucially, by utilizing a learnable decoder μ, the representatives (the predicted values) are optimized directly via gradient descent rather than relying on fixed bin midpoints. This allows the model to automatically find the optimal centroids for the regression task.

By smoothing over the transition between a discrete and a continuous task, the method that we propose leads to possible interpretability of the learnt codes as representations of the target data. As noted above, the decoded predictions are necessarily in the convex hull of the μ_i's, that can be interpreted as a quantization of the data. When there is an *a priori* natural underlying clustering to the feature and target space, it is natural to investigate whether the learnt classes correspond to the natural ones. We observed in several experiments (see Section 4) that while the entropy of learnt encoded distributions $\psi_w(y) \in \Delta_k$ for targets y from the data is quite low, these distributions are not typically very close to one-hots, as is more common in classification. The reason for this behavior could be connected to implicit biases and training dynamics as observed in a classification setting Stewart et al. (2023).

4 EXPERIMENTS

Methods Evaluated. In this section, we compare the performance our *end-to-end approach* against multiple baselines, namely: (1) *Least squares regression with the square loss* see Section 2.1. (2) *Least squares regression with the Huber loss.* (3) *Hard-binning regression as classification*, as described by (Stewart et al., 2023) and briefly reviewed in Section 2.2. (4) *Least squares with a softmax layer.* This final baseline ensures that any observed gains are not simply arising from the effective architectural change (and added parameters count) incurred by the decoder, applied after the softmax (effectively an extra linear layer). For more details on this baseline, see Appendix A. In addition to evaluating our end-to-end approach, we provide ablations by evaluating the individual proposals detailed in sections 3.2 (*soft binning classification*) and 3.3 (*classification with pre-trained encoder*). To measure performance, we report test time mean-absolute error (MAE), root mean square error (rMSE) and r-squared (R2), in line with prior literature (Grinsztajn et al., 2022). We first present results evaluating all methods across a range of real world regression datasets, providing empirical evidence supporting our methodology. After, we compare our proposed end-to-end method with the least squares regression baseline on a larger scale data set and architecture, to demonstrate that the observed gains indeed translate at scale.

Datasets. We demonstrate our methodology across a diverse set of real-world regression datasets, spanning engineering, social sciences, medicine, physics, geography and other interdisciplinary fields, all of which are publicly available. In particular we use the following OpenML (Vanschoren et al., 2014) datasets (CC BY 4.0 licence): Ailerons (AE), Elevators (EL), Computer Activity (CA), Diamonds (DM); the following UCI datasets: Wine Quality (WN) (Cortez et al., 2009), Bike Sharing (BS) Fanaee-T and Gama (2014), Superconductivity (SC) (Hamidieh, 2018), as well as the Retina MNIST dataset (RM) from the Medical MNIST benchmark Yang et al. (2023) (CC BY 4.0 licence). For the larger-scale experiment, we use the Open Street View 5M dataset (Astruc et al., 2024) (CC-BY-SA license). The train, validation, test split sizes and feature dimensions for each of the datasets are listed in Table 1. For tabular data points, we applied min-max scaling, for images we standardize across channels, and all labels are scaled to $[0, 1]$. Our tabular regression datasets were selected following the rigorous filtering methodology of (Grinsztajn et al., 2022). This ensures the inclusion of only high-quality, heterogeneous data exhibiting non-trivial baselines, deliberately excluding simpler deterministic tasks from broader collections (such as those by Torgo and Gama (1996)).

Models. For tabular datasets we followed the convention of prior literature (Gorishniy et al., 2021), by using a multilayer perceptron (MLP), with hidden dimension 128, ReLU non-linearity, and a dropout Srivastava et al. (2014) of 0.3. For image datasets we used a convolutional neural network LeCun et al. (1998), using three layers of convolutions with average pooling between layers, followed by two fully-connected layers. For the convolutions, we use $(3, 3)$ kernel size, with a stride of one, and for the average pooling we use a $(2, 2)$ size

Table 1: Dataset properties.

	Tabular							Computer Vision	
	WN	AE	BS	SC	EL	CA	DM	RM	OSV5M
#num. features	7	33	6	79	16	21	6	(3, 28, 28)	-
#num. train points	5,197	11,000	13,903	17,010	13,279	6,553	43,152	1,080	4.9M
#num. val points	650	1,375	1,738	2,126	1,660	819	5,394	120	210k
#num. test points	650	1,375	1,738	2,127	1,660	819	5,394	400	210k

with a stride of two. The two fully-connected layers have hidden dimension 256, and use a dropout of 0.5, with ReLU as the non-linearity. For the large scale experiment, we used a ViT-L Transformer (Dosovitskiy et al., 2020) with a 12-Layer MLP head, similar to (Dufour et al., 2024). For the exact implementations of all models, data processing and training, we refer the reader to our code repository, (implemented in PyTorch).

Training. All models were trained via backpropagation, using the Adam optimizer Kingma and Ba (2014) with an ℓ_2 weight decay of 10^{-4} for the MLP and encoder-decoder, ℓ_2 decay of 10^{-2} for the CNN, and ℓ_2 decay of 0.05 for the ViT-L, as in (Dufour et al., 2024). All models use a unit gradient clipping norm. Hyper-parameters for experiments i.e., max learning rate, λ_{KL}, λ_{auto}, λ_{pred}, were selected for each model via a log-space sweep. We evaluate methodologies for varied $k \in \{5, 15, 25\}$. To ensure statistical significance, we run repeat trials for all experiments with varied random seeds —10 random seeds for all experiments, except for the large-scale runs, for which we used 5 seeds— permitting us to report the mean and 2σ confidence intervals of all test metrics.

4.1 Results

For $k = 25$, Figure 3 depicts the test MAE for each of the datasets and evaluated methodologies[2]. Complementarily, Figure 4 (top) depicts the test MAE for each methodology, averaged across all datasets. We remark that the same general behaviour was observed across all values of k; see Appendix B for further visualization of the results for all metrics, and Tables 3, 4, 5 for all numerical results with 2σ confidence intervals.

We remark that across all datasets, reformulating regression as a classification problem via both hard-binning and soft-binning yielded improvements, with soft-binning performing globally better. This reinforces the observations of prior literature (Stewart et al., 2023; Farebrother et al., 2024), and secondly demonstrates the benefits of mapping targets into the interior of the simplex (our proposed initialized encoder-decoder), rather than to an extremal one-hot vector (discretized binning).

Further, we observe that training a classification model on targets generated from a trained target codec, yielded better performance across datasets than both hard-binning and an initialized target codec (with frozen weights). Fitting our proposed softmax encoder-decoder on the train targets is both fast and computationally light-weight, and is moreover promising for scenarios where the auto-encoding loss (Equation (6)) at initialization can be decreased substantially (for example, in a case where k is not large enough for the target distribution of $y \in \mathbb{R}^m$).

For Baseline 4 (least-squares objective for a model with softmax layer), we can see that adding the decoder's extra trainable parameters to the regression model and training with the square loss results in varied results. For some datasets (e.g., Super Conductivity), it leads to performance gains (likely due to a greater model capacity), whilst for others (and globally on average), it leads to degraded performance, even compared to the initial least-squares baseline. Our gains are therefore not due to architectural choices and the presence of a softmax layer.

It can be seen from Figures 3 and 4 (and results in Appendix B) that our proposed "end-to-end" objective (Equation (8)) leads to the best performance across all datasets. We stipulate this is because this approach (1) optimizes the encoder-decoder to attain a low auto-encoding error, (so decoding of classification model that has learned to predict with high accuracy the target encoding would result in a low prediction error), and (2) the approach bridges classification and regression, with the prior potentially yielding the benefits of task reformulation (Stewart et al., 2023), and the latter ensuring both the classification model and decoder are jointly trained by gradients coming from the regression objective, (additionally providing positional information of the soft-bins).

Hyper-parameters. A key hyper-parameter of both hard binning and our proposed end-to-end method-

[2]For clarity, the Huber loss results are omitted from the Figure 3 as there was no statistical difference between those of the square loss.

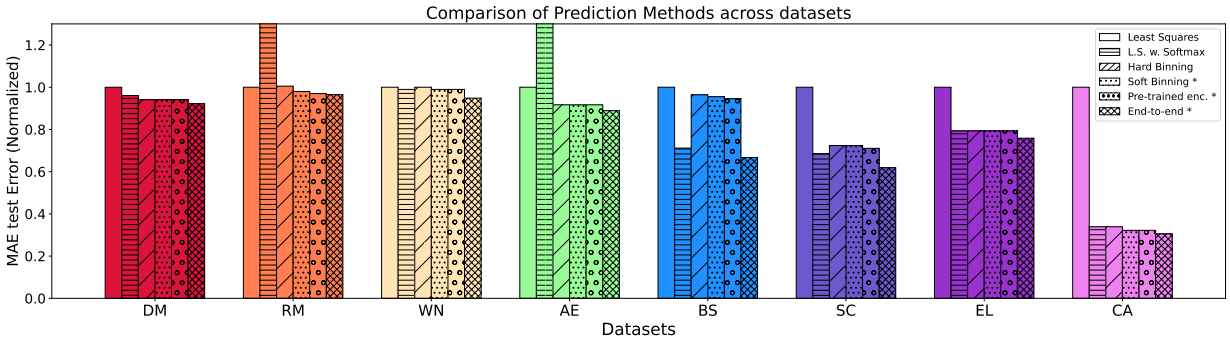

Figure 3: Average test MAE across datasets, normalized to error of the first baseline (least squares with regression with the square loss).

ology is the choice of k, the number of classification classes (or size of the encoded distribution). For small k, the encoder-decoder will have less capacity to auto-encode the targets, which may hurt performance on the regression task, but larger values of k yield increased optimization costs. Figure 4 (center) depicts the relationship between k and the final test MAE for soft-binning encoder classification. As k increases, we can see improvements in performance, followed by a plateau with no further gains. As with determining the optimal number of clusters in methods like K-means, the choice of k remains dependent on the specific dataset and task, and will typically require empirical tuning or cross-validation.

For end-to-end training, we explored the balance between regression and classification loss terms, via the choice of λ_{KL} and λ_{pred}. Whilst for some select datasets and values of k, training with only the KL objective produced similar performance to our end-to-end approach, it was observed that no single values of λ_{KL}, λ_{pred} that were globally optimal across all datasets and models. To explore the robustness in model performance to these loss terms, we set $\lambda_{\mathsf{pred}} = 1$ and performed a sweep to find λ^*_{KL}, the best value for each dataset (with the results listed in Table 6 within Appendix B). Figure 4 (bottom) depicts the impact of this parameter for the *diamonds* dataset, and highlights how the combination of the regression and classification loss can lead to better results than just one of the losses alone. We remark that overall, the dependency of the final results on these hyper-parameters was low, indicating some robustness to these choices.

Larger-scale Experiment. We further demonstrate that the gains observed when using our end-to-end methodology translate to larger architectures and datasets. In particular, we use the Open StreetView 5M (Astruc et al., 2024), a large-scale, open-access dataset comprising over 5.1 million georeferenced street view images, covering 225 countries and territories. The learning problem involves predicting a 2D GPS coordinate given an image. The dataset has strict train / test spatial separation, and hence tests for the relevance of learned geographical features beyond memorization. We use the experimental setup of (Dufour et al., 2024), which uses a ViT-L transformer with a 12-Layer MLP head. A detailed description of the experimental setup is detailed in Appendix C. For our end-to-end method, we used $k = 625$, initializing μ as a uniform spaced 25×25 grid. Our results, unnormalized, are reported in Table 2, with our method notably outperforming the baseline on all metrics, further motivating the applicability of our approach when scaled to higher-capacity models and large-scale, diverse datasets.

Limitations. Our work does not provide a theoretical insight into why re-including the regression loss into the training objective in our proposed end-to-end method yielded superior performance. We speculate that this could relate to: i) the regression loss gives information of soft-bin position / order, which the classification loss does not incorporate ii) additional fitting of the target decoder parameter μ. Exploring this question would make for interesting future work. Whilst we empirically investigated the impact of the hyper-parameter k (Figure 4, center), its optimal value remains inherently dataset- and model-dependent, a limitation shared with baseline approaches such as (Stewart et al., 2023); our study does not provide a theoretical analysis of the asymptotic behavior of k.

Metric	**Least-squares**	**End-to-end**	Δ
MAE ($\downarrow$)	0.073 ± 0.002	0.051 ± 0.002	-30.04%
rMSE ($\downarrow$)	0.114 ± 0.003	0.099 ± 0.000	-13.04%
$\mathbf{R}^2$ ($\uparrow$)	0.749 ± 0.006	0.806 ± 0.001	+7.52%

Table 2: Open Street View 5M: test set statistics

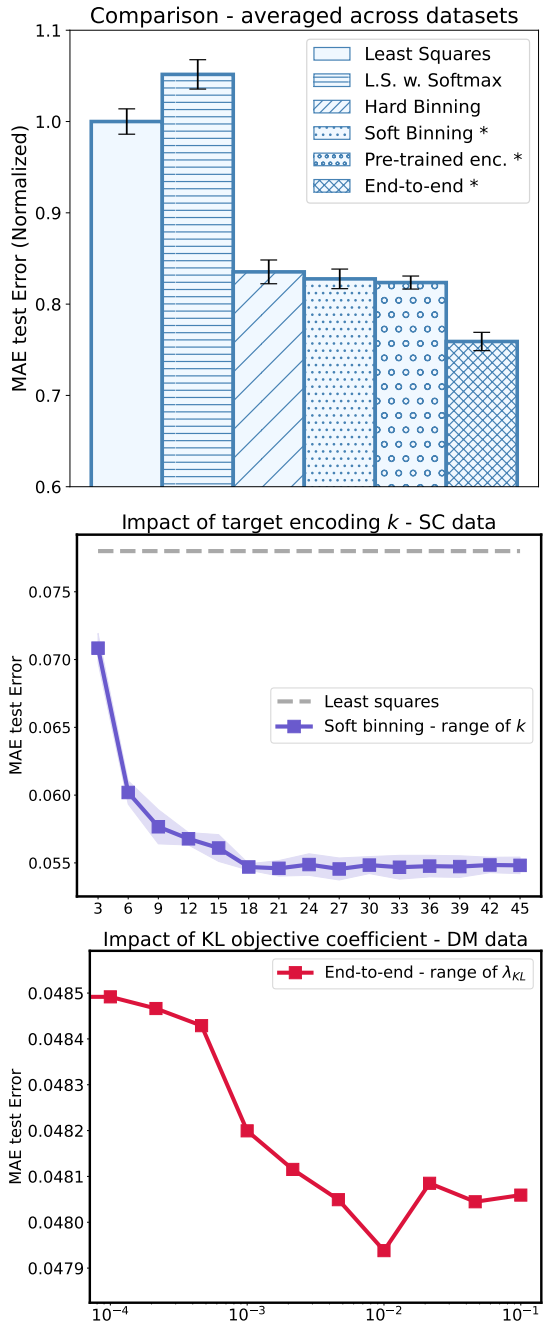

Figure 4: **Top**: test MAE for all methodologies, averaged across datasets with 2σ confidence intervals. We observe an overall hierarchy between the different methods considered. **Center:** Impact of k for values between 3 and 45, for a model trained on labels generated by a target codec using our proposed initialization (Section 3.2). **Bottom**: Impact of the value of λ_{KL} on the test MAE, for our end-to-end framework, with $\lambda_{pred} = 1$. A combination of both losses attains best performance, however the methodology is comparatively robust to choices of the loss hyper-parameters.

Conclusion. We introduced a novel generalization to reformulating regression as classification, via a light-weight target encoder-decoder model. By bridging training objectives and learning a target encoder-decoder mapping, our end-to-end method consistently outperformed prior regression and classification baselines Stewart et al. (2023), across a wide range of real-world datasets and neural network architectures.

References

T. Abe, E. K. Buchanan, G. Pleiss, and J. P. Cunningham. Pathologies of predictive diversity in deep ensembles. *arXiv preprint arXiv:2302.00704*, 2023.

A. Ahmad, S. S. Khan, and A. Kumar. Learning regression problems by using classifiers. *Journal of Intelligent & Fuzzy Systems*, 35(1):945–955, 2018.

I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

G. Astruc, N. Dufour, I. Siglidis, C. Aronssohn, N. Bouia, S. Fu, R. Loiseau, V. N. Nguyen, C. Raude, E. Vincent, et al. Openstreetview-5m: The many roads to global visual geolocation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21967–21977, 2024.

F. Bach. *Learning Theory from First Principles*. MIT Press, 2024.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

J. T. Barron. A general and adaptive robust loss function. In *Cnference on computer vision and pattern recognition*, pages 4331–4339, 2019.

V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab. Robust optimization for deep regression. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2838, 2015.

S. F. Bhat, I. Alhashim, and P. Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4009–4018, 2021.

Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, D. Roblek, O. Teboul, D. Grangier, M. Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *ACM Transactions on Audio, Speech, and Language Processing*, 31:2523–2533, 2023.

E. Boursier, L. Pillaud-Vivien, and N. Flammarion. Gradient flow dynamics of shallow relu networks for square loss and orthogonal inputs. *Advances in Neural Information Processing Systems*, 35:20105–20118, 2022.

D. Chistikov, M. Englert, and R. Lazic. Learning a neuron by a shallow relu network: Dynamics and implicit bias for correlated inputs. *Advances in Neural Information Processing Systems*, 36:23748–23760, 2023.

L. Chizat and F. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in Neural Information Processing Systems*, 31, 2018.

P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12, 2016.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

N. Dufour, D. Picard, V. Kalogeiton, and L. Landrieu. Around the world in 80 timesteps: A generative approach to global visual geolocation. *arXiv preprint arXiv:2412.06781*, 2024.

H. Fanaee-T and J. Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2:113–127, 2014.

J. Farebrother, J. Orbay, Q. Vuong, A. A. Taïga, Y. Chebotar, T. Xiao, A. Irpan, S. Levine, P. S. Castro, A. Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

Z. Gao, C. Tan, J. Wang, Y. Huang, L. Wu, and S. Z. Li. Foldtoken: Learning protein language via vector quantization and beyond. *arXiv preprint arXiv:2403.09673*, 2024.

Y. Gong, G. Mori, and F. Tung. Ranksim: Ranking similarity regularization for deep imbalanced regression. *arXiv preprint arXiv:2205.15236*, 2022.

Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.

E. Guha, S. Natarajan, T. Möllenhoff, M. E. Khan, and E. Ndiaye. Conformal prediction via regression-as-classification. *arXiv preprint arXiv:2404.08168*, 2024.

L. Haas, S. Alberti, and M. Skreta. Learning generalized zero-shot learners for open-domain image geolocalization. *arXiv preprint arXiv:2302.00275*, 2023.

K. Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

P. J. Huber. Robust estimation of a location parameter. *Ann. Math. Statist.*, 35(4):73–101, 1964.

E. Imani and M. White. Improving regression performance with distributional losses. In *International Conference on Machine Learning*, pages 2157–2166, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR 2015*, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

K. Lee, Y. S. Sim, H.-S. Cho, M. Eo, S. Yoon, S. Yoon, and W. Lim. Binning as a pretext task: Improving self-supervised learning in tabular domains. *arXiv preprint arXiv:2405.07414*, 2024.

X. Liu, W. Liang, Y. Wang, S. Li, and M. Pei. 3d head pose estimation with convolutional neural network trained on synthetic images. In *International Conference on Image Processing (ICIP)*, pages 1289–1293. IEEE, 2016.

C. McCarter. Unmasking trees for tabular data. *arXiv preprint arXiv:2407.05593*, 2024.

L. Picek, C. Botella, M. Servajean, C. Leblanc, R. Palard, T. Larcher, B. Deneu, D. Marcos, P. Bonnet, and A. Joly. Geoplant: Spatial plant species prediction dataset. *arXiv preprint arXiv:2408.13928*, 2024.

S. L. Pintea, Y. Lin, J. Dijkstra, and J. C. van Gemert. A step towards understanding why classification helps regression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19972–19981, 2023.

J. Ren, M. Zhang, C. Yu, and Z. Liu. Balanced mse for imbalanced visual regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7926–7935, 2022.

G. Rogez, P. Weinzaepfel, and C. Schmid. Lcr-net: Localization-classification-regression for human pose. In *Conference on Computer Vision and Pattern Recognition*, pages 3433–3441, 2017.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

D. Shah, Z. Y. Xue, and T. M. Aamodt. Label encoding for regression networks. *arXiv preprint arXiv:2212.01927*, 2022.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

L. Stewart, F. Bach, Q. Berthet, and J.-P. Vert. Regression as classification: Influence of task formulation on neural network features. In *International Conference on Artificial Intelligence and Statistics*, pages 11563–11582, 2023.

Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3476–3483, 2013.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in Neural information Processing Systems*, 27, 2014.

L. Torgo and J. Gama. Regression by classification. In *Advances in Artificial Intelligence: 13th Brazilian Symposium on Artificial Intelligence, SBIA'96 Curitiba, Brazil, October 23–25, 1996 Proceedings 13*, pages 51–60. Springer, 1996.

A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

A. Vaswani et al. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

Y. Yang, K. Zha, Y. Chen, H. Wang, and D. Katabi. Delving into deep imbalanced regression. In *International conference on machine learning*, pages 11842–11851. PMLR, 2021.

R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666, 2016.

S. Zhang, L. Yang, M. B. Mi, X. Zheng, and A. Yao. Improving deep regression with ordinal entropy. *arXiv preprint arXiv:2301.08915*, 2023.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not yet - upon publication]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not yet - upon publication]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A LEAST SQUARES WITH SOFTMAX LAYER [BASELINE 4]

Our proposed methodology uses a classification model $\pi_\theta = \text{softmax}(g_\theta)$ with outputs in Δ_k, and prediction the target decoder (a linear layer μ), with $z = \mu^\top \pi_\theta(x) \in \mathbb{R}^m$. To ensure that any observed gains are simply not resulting from this specific architecture change, (or simply the added parameters), we also compare in all our our results to the following baseline:

$$\min_{\theta,\mu} \mathbf{E}_{(x,y)}\big[L(y - \mu^\top \pi_\theta(x))\big] = \min_{\theta,\mu} \mathbf{E}_{(x,y)}\big[L(y - \mu^\top \text{softmax}(g_\theta(x))\big]. \tag{9}$$

This baseline effectively corresponds to adding an extra layer with k neurons after a softmax non-linearity. The parameters θ and μ are optimized jointly by minimizing the square loss.

B EXPERIMENTAL RESULTS: ALL METHODOLOGIES

The test time statistics for all methodologies are displayed in Tables 3, 4, 5, with our end-to-end approach using $\lambda_{pred} = 1$ and $\lambda_{KL} = 0.068$. We observed empirically that when initializing the encoder-decoder weights using our proposed methodology, the results are robust across datasets to the choice of the entropic regularization coefficient α in Equation (7), and taking a very small value, e.g., 10^{-6} suffices, (on the other hand, too large values of α will negatively affect the auto-encoding loss listed in Equation (6)). All experiments were ran on a single NVIDIA V100 GPU.

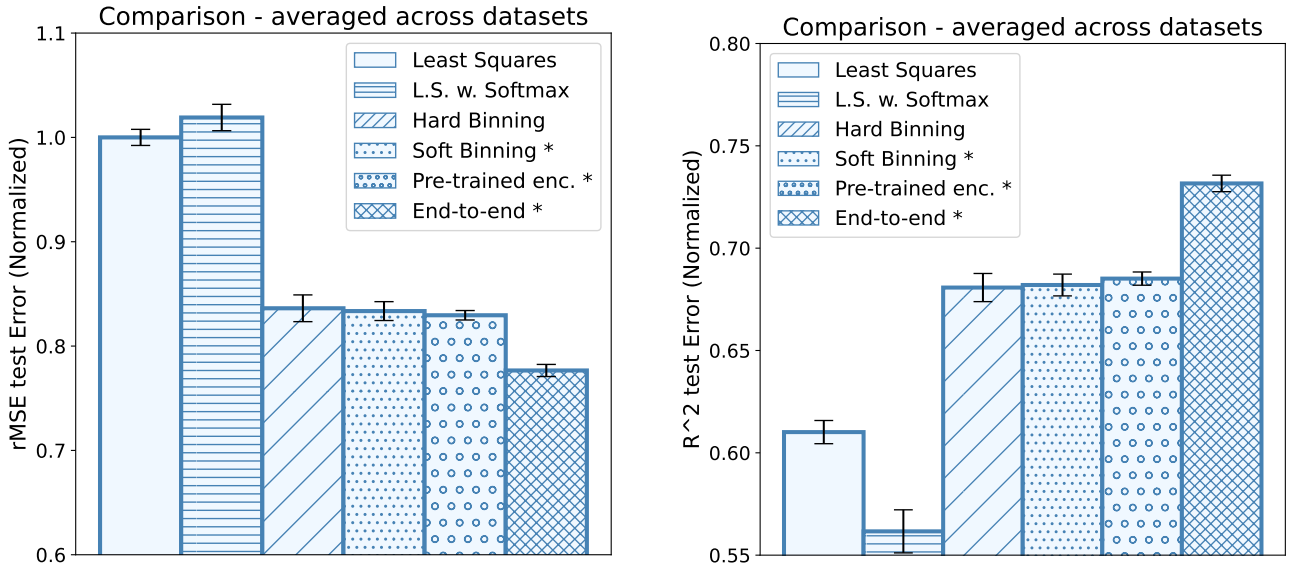

Figure 5: Performance of methodologies averaged across datasets, with 2σ confidence intervals. Left-most figure corresponds to rMSE, right-most figure corresponds to R^2.

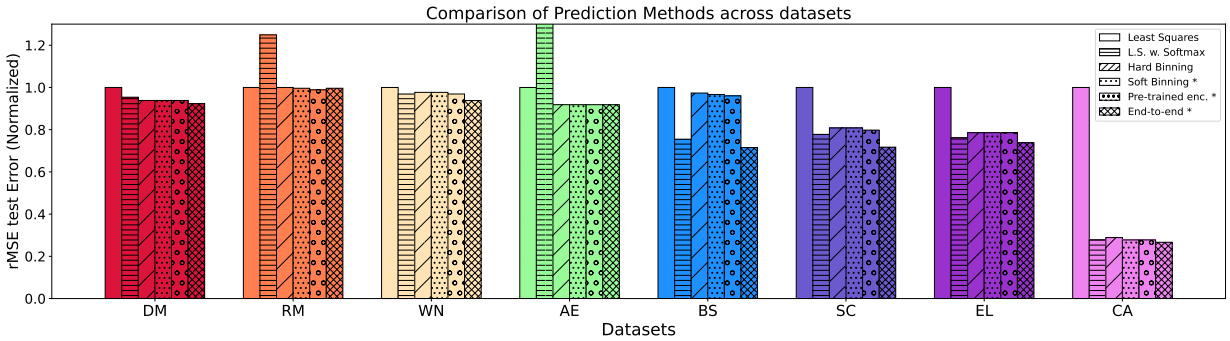

Figure 6: Average test rMSE across datasets, normalized to error of the first baseline (least squares with regression with the square loss).

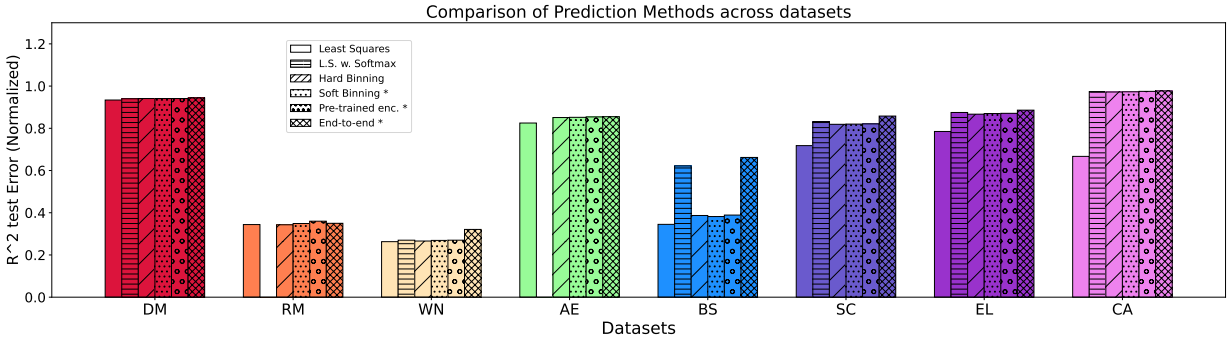

Figure 7: Average test R^2 across datasets, normalized to error of the first baseline (least squares with regression with the square loss).

Table 3: **MAE**: statistics over 10 random seeds, (lower is better).

	WN	AE	BS	SC	EL	CA	DM	RM
Least Squares	$0.097 \pm 2.00e^{-4}$	$0.036 \pm 7.25e^{-5}$	$0.111 \pm 6.84e^{-4}$	$0.076 \pm 2.42e^{-4}$	$0.029 \pm 1.34e^{-4}$	$0.059 \pm 8.28e^{-4}$	$0.051 \pm 9.67e^{-5}$	$0.199 \pm 3.93e^{-3}$
Huber Loss	$0.097 \pm 2.00e^{-4}$	$0.036 \pm 7.27e^{-5}$	$0.111 \pm 6.85e^{-4}$	$0.076 \pm 2.42e^{-4}$	$0.029 \pm 1.34e^{-4}$	$0.059 \pm 8.18e^{-4}$	$0.051 \pm 4.02e^{-5}$	$0.199 \pm 4.29e^{-3}$
Least Squares w. Softmax $k = 5$	$0.096 \pm 4.36e^{-4}$	$0.089 \pm 2.54e^{-5}$	$0.088 \pm 2.58e^{-3}$	$0.057 \pm 2.33e^{-3}$	$0.028 \pm 1.30e^{-2}$	$0.021 \pm 1.45e^{-3}$	$0.049 \pm 1.94e^{-4}$	$0.231 \pm 4.21e^{-2}$
Least Squares w. Softmax $k = 15$	$0.095 \pm 2.75e^{-4}$	$0.089 \pm 3.34e^{-5}$	$0.081 \pm 5.50e^{-3}$	$0.052 \pm 8.36e^{-4}$	$0.023 \pm 2.00e^{-4}$	$0.020 \pm 1.94e^{-4}$	$0.049 \pm 1.82e^{-4}$	$0.278 \pm 2.76e^{-2}$
Least Squares w. Softmax $k = 25$	$0.096 \pm 4.78e^{-4}$	$0.089 \pm 2.61e^{-5}$	$0.079 \pm 3.36e^{-3}$	$0.052 \pm 9.05e^{-4}$	$0.023 \pm 2.11e^{-4}$	$0.020 \pm 2.30e^{-4}$	$0.049 \pm 4.15e^{-5}$	$0.291 \pm 8.98e^{-4}$
Hard Binning Classification $k = 5$	$0.097 \pm 9.56e^{-4}$	$0.043 \pm 1.42e^{-3}$	$0.108 \pm 2.81e^{-4}$	$0.058 \pm 2.90e^{-4}$	$0.043 \pm 1.35e^{-3}$	$0.039 \pm 2.22e^{-4}$	$0.050 \pm 2.05e^{-4}$	$0.199 \pm 2.35e^{-3}$
Hard Binning Classification $k = 15$	$0.096 \pm 2.48e^{-4}$	$0.033 \pm 1.14e^{-4}$	$0.107 \pm 5.98e^{-4}$	$0.055 \pm 3.57e^{-4}$	$0.024 \pm 2.07e^{-4}$	$0.020 \pm 3.13e^{-4}$	$0.048 \pm 5.08e^{-5}$	$0.201 \pm 3.39e^{-3}$
Hard Binning Classification $k = 25$	$0.097 \pm 1.88e^{-4}$	$0.033 \pm 2.02e^{-4}$	$0.107 \pm 3.55e^{-4}$	$0.055 \pm 4.89e^{-4}$	$0.023 \pm 2.31e^{-4}$	$0.020 \pm 2.54e^{-4}$	$0.048 \pm 7.14e^{-5}$	$0.200 \pm 4.22e^{-3}$
Soft Binning Classification $\lambda_\sigma = 0.5, k = 5$	$0.096 \pm 2.05e^{-4}$	$0.033 \pm 1.04e^{-4}$	$0.109 \pm 5.06e^{-4}$	$0.062 \pm 6.27e^{-4}$	$0.027 \pm 1.67e^{-4}$	$0.019 \pm 1.62e^{-4}$	$0.049 \pm 3.46e^{-5}$	$0.197 \pm 2.47e^{-3}$
Soft Binning Classification $\lambda_\sigma = 0.5, k = 15$	$0.096 \pm 3.50e^{-4}$	$0.033 \pm 1.10e^{-4}$	$0.107 \pm 4.56e^{-4}$	$0.055 \pm 3.74e^{-4}$	$0.024 \pm 1.04e^{-4}$	$0.019 \pm 3.90e^{-4}$	$0.048 \pm 4.46e^{-5}$	$0.197 \pm 3.80e^{-3}$
Soft Binning Classification $\lambda_\sigma = 0.5, k = 25$	$0.096 \pm 3.03e^{-4}$	$0.033 \pm 1.03e^{-4}$	$0.106 \pm 6.89e^{-4}$	$0.055 \pm 4.56e^{-4}$	$0.023 \pm 8.23e^{-5}$	$0.019 \pm 2.42e^{-4}$	$0.048 \pm 3.91e^{-5}$	$0.195 \pm 3.39e^{-3}$
Trained Encoder Classification $k = 5$	$0.096 \pm 1.86e^{-4}$	$0.033 \pm 6.84e^{-5}$	$0.109 \pm 3.05e^{-4}$	$0.058 \pm 5.09e^{-4}$	$0.025 \pm 2.09e^{-4}$	$0.019 \pm 3.31e^{-4}$	$0.049 \pm 4.82e^{-5}$	$0.193 \pm 2.21e^{-3}$
Trained Encoder Classification $k = 15$	$0.096 \pm 1.34e^{-4}$	$0.033 \pm 1.70e^{-5}$	$0.105 \pm 2.33e^{-4}$	$0.055 \pm 2.79e^{-4}$	$0.023 \pm 1.04e^{-4}$	$0.019 \pm 9.57e^{-5}$	$0.048 \pm 3.61e^{-5}$	$0.192 \pm 1.54e^{-3}$
Trained Encoder Classification $k = 25$	$0.096 \pm 1.74e^{-4}$	$0.033 \pm 2.67e^{-5}$	$0.105 \pm 3.33e^{-4}$	$0.054 \pm 3.32e^{-4}$	$0.023 \pm 1.04e^{-4}$	$0.019 \pm 1.50e^{-4}$	$0.048 \pm 5.90e^{-5}$	$0.193 \pm 2.26e^{-3}$
End to End $k = 5$	$0.095 \pm 8.06e^{-4}$	$0.032 \pm 2.95e^{-5}$	$0.088 \pm 9.26e^{-4}$	$0.052 \pm 2.77e^{-4}$	$0.023 \pm 2.24e^{-4}$	$0.018 \pm 1.14e^{-4}$	$0.048 \pm 3.58e^{-5}$	$0.193 \pm 3.36e^{-3}$
End to End $k = 15$	$0.092 \pm 8.16e^{-4}$	$0.032 \pm 8.90e^{-5}$	$0.075 \pm 2.45e^{-4}$	$0.048 \pm 4.60e^{-4}$	$0.022 \pm 1.02e^{-4}$	$0.018 \pm 2.24e^{-4}$	$0.047 \pm 9.57e^{-5}$	$0.192 \pm 1.94e^{-3}$
End to End $k = 25$	$0.092 \pm 5.71e^{-4}$	$0.032 \pm 1.05e^{-4}$	$0.074 \pm 3.13e^{-4}$	$0.047 \pm 3.37e^{-4}$	$0.022 \pm 8.31e^{-5}$	$0.018 \pm 3.06e^{-4}$	$0.047 \pm 3.14e^{-5}$	$0.192 \pm 3.07e^{-3}$

Table 4: **rMSE**: statistics over 10 random seeds, (lower is better).

	WN	AE	BS	SC	EL	CA	DM	RM
Least Squares	$0.129 \pm 1.75e^{-4}$	$0.049 \pm 1.89e^{-4}$	$0.151 \pm 7.44e^{-5}$	$0.099 \pm 2.93e^{-4}$	$0.042 \pm 1.07e^{-4}$	$0.090 \pm 9.91e^{-4}$	$0.065 \pm 1.52e^{-4}$	$0.265 \pm 1.67e^{-3}$
Huber Loss	$0.129 \pm 1.75e^{-4}$	$0.049 \pm 1.87e^{-4}$	$0.151 \pm 7.45e^{-5}$	$0.099 \pm 2.93e^{-4}$	$0.042 \pm 1.07e^{-4}$	$0.090 \pm 1.04e^{-3}$	$0.065 \pm 6.59e^{-5}$	$0.265 \pm 2.03e^{-3}$
Least Squares w. Softmax $k = 5$	$0.126 \pm 2.91e^{-4}$	$0.118 \pm 2.64e^{-6}$	$0.127 \pm 2.87e^{-3}$	$0.081 \pm 2.31e^{-3}$	$0.039 \pm 1.84e^{-2}$	$0.027 \pm 1.59e^{-3}$	$0.062 \pm 1.76e^{-4}$	$0.291 \pm 2.97e^{-2}$
Least Squares w. Softmax $k = 15$	$0.125 \pm 1.80e^{-4}$	$0.118 \pm 2.61e^{-6}$	$0.117 \pm 6.11e^{-3}$	$0.077 \pm 8.06e^{-4}$	$0.032 \pm 1.59e^{-4}$	$0.025 \pm 1.38e^{-4}$	$0.062 \pm 1.34e^{-4}$	$0.318 \pm 1.71e^{-2}$
Least Squares w. Softmax $k = 15$	$0.125 \pm 2.17e^{-4}$	$0.118 \pm 1.99e^{-6}$	$0.114 \pm 3.84e^{-3}$	$0.077 \pm 8.29e^{-4}$	$0.032 \pm 3.05e^{-4}$	$0.025 \pm 2.26e^{-4}$	$0.062 \pm 4.49e^{-5}$	$0.331 \pm 1.21e^{-3}$
Hard Binning Classification $k = 5$	$0.127 \pm 4.25e^{-4}$	$0.057 \pm 5.03e^{-4}$	$0.149 \pm 4.27e^{-5}$	$0.083 \pm 4.29e^{-4}$	$0.054 \pm 1.39e^{-3}$	$0.049 \pm 7.08e^{-4}$	$0.063 \pm 1.70e^{-4}$	$0.265 \pm 1.62e^{-3}$
Hard Binning Classification $k = 15$	$0.126 \pm 2.11e^{-4}$	$0.046 \pm 1.87e^{-4}$	$0.147 \pm 8.47e^{-5}$	$0.080 \pm 3.58e^{-4}$	$0.034 \pm 4.21e^{-4}$	$0.026 \pm 5.95e^{-4}$	$0.061 \pm 6.88e^{-5}$	$0.265 \pm 2.00e^{-3}$
Hard Binning Classification $k = 25$	$0.126 \pm 2.41e^{-4}$	$0.045 \pm 3.08e^{-4}$	$0.147 \pm 6.03e^{-5}$	$0.080 \pm 4.98e^{-4}$	$0.033 \pm 4.29e^{-4}$	$0.026 \pm 1.49e^{-3}$	$0.061 \pm 6.48e^{-5}$	$0.265 \pm 2.63e^{-3}$
Soft Binning Classification $\lambda_\sigma = 0.5$, $k = 5$	$0.126 \pm 1.68e^{-4}$	$0.045 \pm 1.49e^{-4}$	$0.149 \pm 3.10e^{-5}$	$0.086 \pm 5.88e^{-4}$	$0.037 \pm 3.72e^{-4}$	$0.025 \pm 1.09e^{-4}$	$0.062 \pm 1.76e^{-5}$	$0.266 \pm 2.06e^{-3}$
Soft Binning Classification $\lambda_\sigma = 0.5$, $k = 15$	$0.126 \pm 2.73e^{-4}$	$0.045 \pm 1.64e^{-4}$	$0.147 \pm 6.06e^{-5}$	$0.080 \pm 3.19e^{-4}$	$0.033 \pm 1.71e^{-4}$	$0.025 \pm 8.93e^{-4}$	$0.061 \pm 2.60e^{-5}$	$0.265 \pm 2.27e^{-3}$
Soft Binning Classification $\lambda_\sigma = 0.5$, $k = 25$	$0.126 \pm 2.82e^{-4}$	$0.045 \pm 1.64e^{-4}$	$0.146 \pm 8.04e^{-5}$	$0.080 \pm 4.69e^{-4}$	$0.033 \pm 2.14e^{-4}$	$0.025 \pm 9.55e^{-4}$	$0.061 \pm 4.86e^{-5}$	$0.264 \pm 2.36e^{-3}$
Trained Encoder Classification $k = 5$	$0.125 \pm 1.41e^{-4}$	$0.045 \pm 1.69e^{-4}$	$0.149 \pm 6.03e^{-5}$	$0.082 \pm 3.69e^{-4}$	$0.034 \pm 3.25e^{-4}$	$0.026 \pm 3.95e^{-4}$	$0.062 \pm 2.80e^{-5}$	$0.263 \pm 1.76e^{-3}$
Trained Encoder Classification $k = 15$	$0.125 \pm 2.03e^{-4}$	$0.045 \pm 7.49e^{-5}$	$0.145 \pm 8.30e^{-5}$	$0.080 \pm 3.14e^{-4}$	$0.033 \pm 1.49e^{-4}$	$0.025 \pm 9.09e^{-5}$	$0.061 \pm 3.83e^{-5}$	$0.262 \pm 6.43e^{-4}$
Trained Encoder Classification $k = 25$	$0.125 \pm 1.64e^{-4}$	$0.045 \pm 1.14e^{-4}$	$0.145 \pm 6.72e^{-5}$	$0.079 \pm 3.27e^{-4}$	$0.033 \pm 1.75e^{-4}$	$0.025 \pm 9.22e^{-5}$	$0.061 \pm 3.23e^{-5}$	$0.262 \pm 1.35e^{-3}$
End to End $k = 5$	$0.125 \pm 1.20e^{-3}$	$0.045 \pm 8.06e^{-5}$	$0.127 \pm 9.28e^{-4}$	$0.076 \pm 5.36e^{-4}$	$0.032 \pm 4.43e^{-4}$	$0.024 \pm 1.61e^{-4}$	$0.061 \pm 3.51e^{-5}$	$0.263 \pm 1.59e^{-3}$
End to End $k = 15$	$0.122 \pm 6.22e^{-4}$	$0.045 \pm 5.14e^{-5}$	$0.109 \pm 4.71e^{-4}$	$0.072 \pm 3.18e^{-4}$	$0.031 \pm 2.67e^{-4}$	$0.024 \pm 3.39e^{-4}$	$0.060 \pm 9.02e^{-5}$	$0.262 \pm 1.46e^{-3}$
End to End $k = 25$	$0.121 \pm 4.41e^{-4}$	$0.045 \pm 1.16e^{-4}$	$0.108 \pm 6.26e^{-4}$	$0.071 \pm 3.74e^{-4}$	$0.031 \pm 1.38e^{-4}$	$0.024 \pm 2.76e^{-4}$	$0.060 \pm 3.79e^{-5}$	$0.264 \pm 6.86e^{-4}$

Table 5: **R2**: statistics over 10 random seeds, (higher is better).

	WN	AE	BS	SC	EL	CA	DM	RM
Least Squares	$0.263 \pm 2.04e^{-3}$	$0.825 \pm 1.34e^{-3}$	$0.345 \pm 6.48e^{-4}$	$0.718 \pm 1.66e^{-3}$	$0.785 \pm 1.10e^{-3}$	$0.667 \pm 7.32e^{-3}$	$0.934 \pm 3.11e^{-4}$	$0.344 \pm 8.25e^{-3}$
Huber Loss	$0.263 \pm 2.04e^{-3}$	$0.825 \pm 1.33e^{-3}$	$0.345 \pm 6.48e^{-4}$	$0.718 \pm 1.66e^{-3}$	$0.785 \pm 1.10e^{-3}$	$0.667 \pm 7.68e^{-3}$	$0.935 \pm 1.33e^{-4}$	$0.346 \pm 1.00e^{-2}$
Least Squares w. Softmax $k = 5$	$0.261 \pm 3.41e^{-3}$	$0.000 \pm 4.49e^{-5}$	$0.536 \pm 2.07e^{-2}$	$0.814 \pm 1.07e^{-2}$	$0.780 \pm 2.77e^{-1}$	$0.969 \pm 3.59e^{-3}$	$0.940 \pm 3.41e^{-4}$	$0.201 \pm 1.67e^{-1}$
Least Squares w. Softmax $k = 15$	$0.270 \pm 2.09e^{-3}$	$0.000 \pm 4.44e^{-5}$	$0.607 \pm 4.14e^{-2}$	$0.831 \pm 3.52e^{-3}$	$0.877 \pm 1.23e^{-3}$	$0.973 \pm 2.88e^{-4}$	$0.940 \pm 2.61e^{-4}$	$0.053 \pm 9.87e^{-2}$
Least Squares w. Softmax $k = 25$	$0.270 \pm 2.53e^{-3}$	$0.000 \pm 3.39e^{-5}$	$0.622 \pm 2.55e^{-2}$	$0.832 \pm 3.62e^{-3}$	$0.875 \pm 2.37e^{-3}$	$0.974 \pm 4.74e^{-4}$	$0.940 \pm 8.71e^{-5}$	$-0.020 \pm 7.49e^{-3}$
Hard Binning Classification $k = 5$	$0.249 \pm 5.02e^{-3}$	$0.767 \pm 4.15e^{-3}$	$0.360 \pm 3.67e^{-4}$	$0.804 \pm 2.02e^{-3}$	$0.647 \pm 1.84e^{-2}$	$0.900 \pm 2.89e^{-3}$	$0.937 \pm 3.37e^{-4}$	$0.343 \pm 8.03e^{-3}$
Hard Binning Classification $k = 15$	$0.267 \pm 2.46e^{-3}$	$0.847 \pm 1.24e^{-3}$	$0.379 \pm 7.18e^{-4}$	$0.819 \pm 1.63e^{-3}$	$0.863 \pm 3.44e^{-3}$	$0.972 \pm 1.28e^{-3}$	$0.942 \pm 1.32e^{-4}$	$0.343 \pm 9.90e^{-3}$
Hard Binning Classification $k = 25$	$0.266 \pm 2.81e^{-3}$	$0.851 \pm 2.03e^{-3}$	$0.387 \pm 5.07e^{-4}$	$0.819 \pm 2.27e^{-3}$	$0.867 \pm 3.45e^{-3}$	$0.972 \pm 3.32e^{-3}$	$0.941 \pm 1.24e^{-4}$	$0.343 \pm 1.30e^{-2}$
Soft Binning Classification $\lambda_\sigma = 0.5$, $k = 5$	$0.262 \pm 1.96e^{-3}$	$0.852 \pm 9.75e^{-4}$	$0.359 \pm 2.67e^{-4}$	$0.790 \pm 2.88e^{-3}$	$0.834 \pm 3.35e^{-3}$	$0.974 \pm 2.23e^{-4}$	$0.939 \pm 3.43e^{-5}$	$0.340 \pm 1.02e^{-2}$
Soft Binning Classification $\lambda_\sigma = 0.5$, $k = 15$	$0.266 \pm 3.18e^{-3}$	$0.852 \pm 1.07e^{-3}$	$0.377 \pm 5.14e^{-4}$	$0.816 \pm 1.46e^{-3}$	$0.869 \pm 1.36e^{-3}$	$0.974 \pm 1.91e^{-3}$	$0.941 \pm 5.00e^{-5}$	$0.345 \pm 1.13e^{-2}$
Soft Binning Classification $\lambda_\sigma = 0.5$, $k = 25$	$0.269 \pm 3.28e^{-3}$	$0.852 \pm 1.08e^{-3}$	$0.382 \pm 6.80e^{-4}$	$0.820 \pm 2.13e^{-3}$	$0.870 \pm 1.70e^{-3}$	$0.973 \pm 2.05e^{-3}$	$0.941 \pm 9.33e^{-5}$	$0.349 \pm 1.16e^{-2}$
Trained Encoder Classification $k = 5$	$0.271 \pm 1.64e^{-3}$	$0.853 \pm 1.11e^{-3}$	$0.362 \pm 5.18e^{-4}$	$0.808 \pm 1.73e^{-3}$	$0.857 \pm 2.70e^{-3}$	$0.973 \pm 8.27e^{-4}$	$0.939 \pm 5.47e^{-5}$	$0.354 \pm 8.67e^{-3}$
Trained Encoder Classification $k = 15$	$0.270 \pm 2.37e^{-3}$	$0.853 \pm 4.88e^{-4}$	$0.389 \pm 6.98e^{-4}$	$0.820 \pm 1.42e^{-3}$	$0.870 \pm 1.18e^{-3}$	$0.975 \pm 1.83e^{-4}$	$0.941 \pm 7.34e^{-5}$	$0.359 \pm 3.14e^{-3}$
Trained Encoder Classification $k = 25$	$0.270 \pm 1.90e^{-3}$	$0.854 \pm 7.42e^{-4}$	$0.389 \pm 5.65e^{-4}$	$0.821 \pm 1.48e^{-3}$	$0.871 \pm 1.39e^{-3}$	$0.975 \pm 1.85e^{-4}$	$0.941 \pm 6.19e^{-5}$	$0.360 \pm 6.61e^{-3}$
End to End $k = 5$	$0.278 \pm 1.38e^{-2}$	$0.856 \pm 5.20e^{-4}$	$0.537 \pm 6.81e^{-3}$	$0.837 \pm 2.30e^{-3}$	$0.876 \pm 3.43e^{-3}$	$0.977 \pm 3.14e^{-4}$	$0.942 \pm 6.69e^{-5}$	$0.353 \pm 7.82e^{-3}$
End to End $k = 15$	$0.314 \pm 7.02e^{-3}$	$0.855 \pm 3.32e^{-4}$	$0.655 \pm 2.97e^{-3}$	$0.853 \pm 1.30e^{-3}$	$0.884 \pm 2.01e^{-3}$	$0.977 \pm 6.62e^{-4}$	$0.944 \pm 1.69e^{-4}$	$0.358 \pm 7.12e^{-3}$
End to End $k = 25$	$0.321 \pm 4.96e^{-3}$	$0.855 \pm 7.49e^{-4}$	$0.662 \pm 3.91e^{-3}$	$0.858 \pm 1.50e^{-3}$	$0.886 \pm 1.02e^{-3}$	$0.977 \pm 5.34e^{-4}$	$0.944 \pm 7.09e^{-5}$	$0.350 \pm 3.37e^{-3}$

Table 6: Optimal sweep λ_{KL} for datasets, with fixed $\lambda_{pred} = 1$.

	WN	AE	BS	SC	EL	CA	DM	RM
$k = 5$	0.068	0.068	0.068	0.068	0.068	0.068	0.068	0.147
$k = 15$	0.068	0.068	0.068	0.147	0.068	0.068	0.068	3.163
$k = 25$	0.316	0.147	0.068	0.068	0.147	0.068	0.068	0.147

C OPEN STREET VIEW 5M EXPERIMENTAL SETUP

Our experimental setup for the large scale Open Streetview 5M problem follows the approach of (Dufour et al., 2024), (where the ViT-L is finetuned on StreetCLIP (Haas et al., 2023) and the soft-tokens are cached to train the MLP weights). Our experimental setup differs from theirs in the following ways: i) We remove the auxillary noise layers from the model, (as our method does not use any diffusion) ii) We used a training batch size of 32,768, and trained until seeing 100M examples. iii) We ramp our loss up over 500 steps to $5e^{-5}$, before running a cosine-decay for the remaining steps. All experiments were ran using 4 x NVIDIA A100 GPUs.

D ADDITIONAL VISUALIZATIONS: TARGET ENCODING FUNCTIONS

We provide more detailed illustration of the target encoding functions $\psi_w(\cdot)_i$ over $\mathbb{R}^2$ for $i \in \{1, \dots, 9\}$ from Figure 2, in Figures 8 and 9 below

These provide a vizualization of the encoder acting as a soft tokenizer.

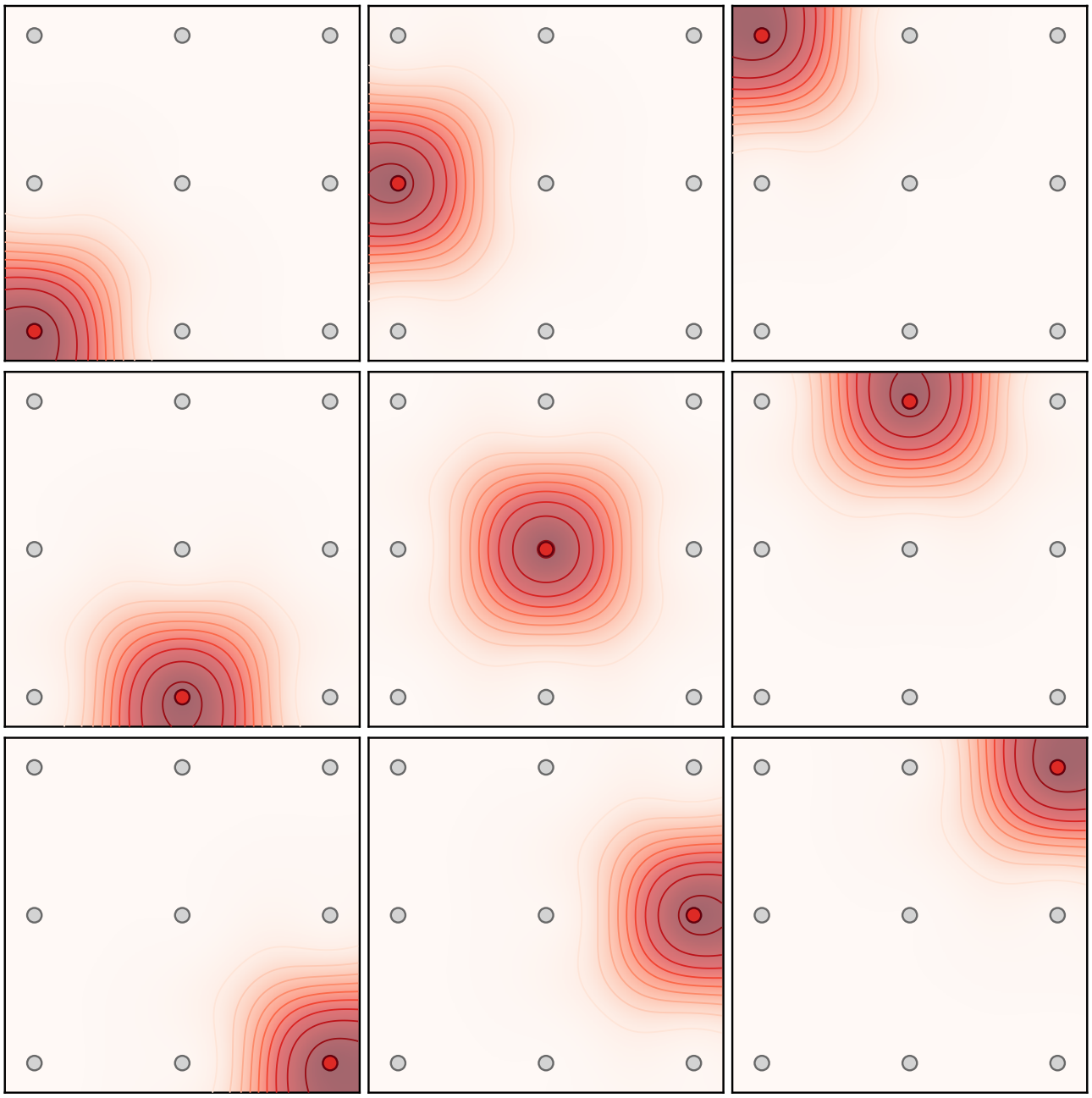

Figure 8: Embedding the target space $\mathbb{R}^m$ (here $m = 2$), representation of all the coefficients of the target encoding

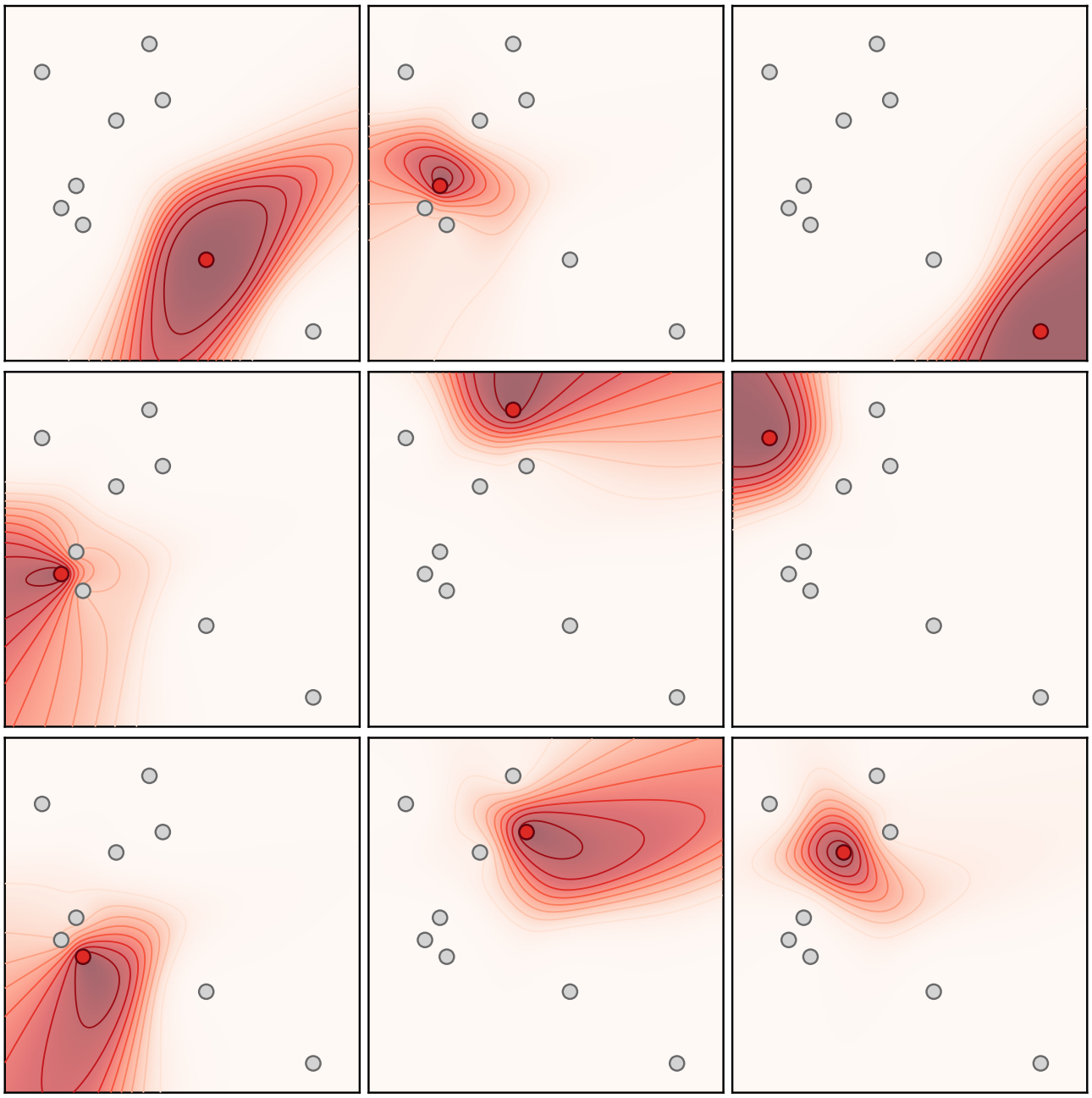

Figure 9: Embedding the target space $\mathbb{R}^m$ (here $m = 2$), representation of all the coefficients of the target encoding