

# Investigating the Histogram Loss in Regression

Ehsan Imani

Kai Luedemann\*<sup>†</sup>

Sam Scholnick-Hughes\*

Esraa Elelimy\*

Martha White

IMANI@UALBERTA.CA

KLUEDEMA@UALBERTA.CA

SCHOLNIC@UALBERTA.CA

ELELIMY@UALBERTA.CA

WHITEM@UALBERTA.CA

*Alberta Machine Intelligence Institute (Amii) and  
Reinforcement Learning and Artificial Intelligence Laboratory  
Department of Computing Science, University of Alberta  
Edmonton, Alberta, Canada T6G 2E8*

**Editor:** Kilian Weinberger

## Abstract

It is becoming increasingly common in regression to train neural networks that model the entire distribution even if only the mean is required for prediction. This additional modeling often comes with performance gains, and the reasons behind the improvement are not fully known. This paper investigates a recent approach to regression, the *histogram loss*, which involves learning the conditional distribution of the target variable by minimizing the cross-entropy between a target distribution and a flexible histogram prediction. The resulting loss corresponds to a classification loss: a cross-entropy between the outputs and a smoothed label vector. We design theoretical and empirical analyses to determine why and when this performance gain appears and how different components of the loss contribute to it. Our results suggest that the benefits of learning distributions in this setup come from improvements in optimization rather than modeling extra information. We then demonstrate the viability of the histogram loss in common deep learning applications without the need for costly hyperparameter tuning.<sup>1</sup>

**Keywords:** Neural Networks, Regression, Conditional Density Estimation

## 1. Introduction

In machine learning, the loss function is a crucial design choice that can influence the optimization landscape and the quality of solutions found by gradient descent. For regression tasks, where the goal is to predict a continuous target  $Y$  from inputs  $\mathbf{x}$ , a common choice is the squared-error loss ( $\ell_2$ ). This loss is theoretically motivated, as minimizing  $\ell_2$  corresponds to maximum likelihood estimation when the conditional distribution of  $Y \mid \mathbf{x}$  is Gaussian with fixed variance.

However,  $\ell_2$  is not the only choice for regression. There are also statistically motivated alternatives. For example, loss functions such as  $\ell_1$  loss, Huber loss, and Charbonnier loss have shown improvements in the presence of outliers (Huber, 2011; Charbonnier et al., 1994;

\*. These authors contributed equally.

†. The author is currently affiliated with the University of Tübingen and Zuse School ELIZA.

1. An implementation is available at [https://github.com/marthawhite/Histogram\\_loss](https://github.com/marthawhite/Histogram_loss).

Barron, 2019). In deep learning, the loss also plays a role in training by influencing the geometry of the optimization landscape. For nonconvex problems, a loss that is more easily optimized may allow gradient descent to converge to a better solution.

A more radical alternative is to reparameterize the model to capture a fuller picture of the output distribution. Examples are predicting the means, variances, and coefficients of a multi-modal mixture of Gaussian distributions (Bishop, 1994), weights of a histogram (Rothe et al., 2015), quantiles of a distribution (Dabney et al., 2018b), and a generative model for sampling from the distribution (Dabney et al., 2018a). In the deep learning literature, such reparameterizations have been beneficial in applications such as age estimation, depth estimation, and value prediction for decision-making (Gao et al., 2017; Reading et al., 2021; Bellemare et al., 2017; Hessel et al., 2018; Bellemare et al., 2023). Understanding why these approaches can be superior to the squared loss, even when only the mean of the predicted distribution is needed, is an ongoing endeavor, and a common explanation is that the extra difficulty helps the network learn a better representation (Bellemare et al., 2017; Lyle et al., 2019, 2021; Lan et al., 2022).

A shorter version of this paper by Imani and White (2018) proposed a family of loss functions called the *histogram loss* (HL). In short, a model trained with HL outputs a *prediction distribution*, a histogram parameterized by a softmax function. Each scalar target in the training data is converted to a *target distribution*, and the model aims to match the histogram to the target distribution by reducing the cross-entropy between the two. The process is shown in Figure 1. This loss was motivated by benefits found in distributional reinforcement learning (RL) but investigated in supervised learning to better understand the phenomenon. Imani and White (2018) motivated that HL might provide performance improvements because of better-behaved gradient norms and provided a small-scale empirical study towards understanding why HL can outperform the squared error loss. Recent work by Farebrother et al. (2024) has shown the utility of HL beyond supervised learning tasks. They showed that using HL for training the value function in RL resulted in a consistent performance improvement across many tasks and improved the scalability of deep RL methods with large networks.

In this paper, we expand the theoretical understanding of this loss, delve deeper empirically into the reasons behind the benefits of HL, explore the viability of HL in larger-scale deep learning tasks, and provide rules of thumb for its hyperparameters. We will also include the results by Imani and White (2018) for completeness. The following three paragraphs summarize the novel contributions.

For theoretical motivation, Imani and White (2018) proved an upper bound on the local Lipschitz constant for HL, showing it is smaller than for  $\ell_2$ ; it is known that a smaller Lipschitz constant improves generalization performance with stochastic gradient descent. This

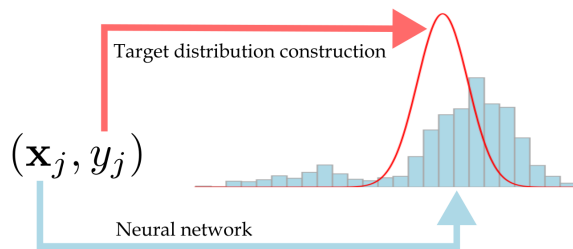


Figure 1: Training with HL. The red curve is the target distribution, and the blue histogram is the prediction distribution. The neural network is trained to minimize the cross-entropy between the two.

previous work also highlighted a connection to maximum entropy reinforcement learning, which can promote efficient search. In this work, we focus on characterizing the *discretization bias* introduced by moving from the  $\ell_2$  loss to HL. For a larger number of output bins, the bias is negligible, and it increases with a small number of bins. We bound the bias based on bin width and further show that, even with a small number of bins, bias can be made small by appropriately selecting a variance parameter. We also motivate HL as a surrogate objective by showing that minimizing HL reduces an upper bound on the squared difference between the predictions and the targets.

We conduct a larger empirical study compared to Imani and White (2018) to test hypotheses regarding the performance of HL. On four regression data sets, we test

1. if the choice of target distribution has a major impact on the bias of HL that can explain the difference in error rates,
2. if properties of the histogram prediction and target distribution give rise to a bias-variance trade-off,
3. if the challenge of predicting a full distribution forces the model to learn a better representation,
4. if switching from the  $\ell_2$  loss to HL makes the landscape smoother
5. if HL is more robust to corrupted targets in the data set, and
6. if HL finds a model whose output is less sensitive to input perturbations.

Finally, we show the viability of HL on large-scale time series and value prediction tasks with different architectures and provide heuristics for choosing the hyperparameters of the loss, namely the number of bins and the amount of padding in the histogram and the degree of smoothing in the target distribution. We show that a fixed set of recommended hyperparameter values works well in a diverse set of experiments. Overall, this paper provides more insights into HL and clarifies some of the choices within this loss to make it easy to use more broadly for regression.

## 2. Distributional Losses for Regression

In this section we introduce the histogram loss (HL). At a high level, defining and using HL is simple: 1) the user selects the number of outputs  $k$  (bins) to discretize the output space, 2) the regression labels  $y$  in the data set are converted to classification labels (probability vectors of length  $k$  that sum to one), and 3) the standard cross-entropy loss is used between the  $k$  outputs and these new targets. The learned network is used for prediction by using the weighted average of outputs with their corresponding bin centers. We first introduce how this loss is constructed, explaining how to define these new classification labels. The construction is defined by the choice of target distribution. We advocate for a Gaussian target distribution as a suitable default in this work but include other examples (e.g., uniform) both to better understand the loss construction and because later we empirically compare these alternatives. This section only explains the construction of this loss; later sections motivate why this loss is useful.

## 2.1 Defining the Histogram Loss

Consider predicting a continuous target  $Y$  with event space  $\mathcal{Y}$ , given inputs  $\mathbf{x}$ . Recall that directly predicting  $Y$  is possible by minimizing the squared error on samples  $(\mathbf{x}_j, y_j)$  where input  $\mathbf{x}_j \in \mathbb{R}^d$  is associated with target  $y_j \in \mathbb{R}$ . Instead of directly predicting  $Y$ , we can learn a distribution on  $Y|\mathbf{x}$  whose expected value will be used as the prediction. Assume we have samples  $(\mathbf{x}_j, q_j)$ , where each input  $\mathbf{x}_j$  is associated with a target distribution with the probability density function (pdf)  $q_j : \mathcal{Y} \rightarrow [0, \infty)$ . We would like to learn a parameterized prediction distribution  $h_{\mathbf{x}} : \mathcal{Y} \rightarrow [0, \infty)$ , conditioned on  $\mathbf{x}$ , by minimizing a measure of difference between  $h_{\mathbf{x}}$  and  $q$ .

Three choices have to be made to learn a distribution on  $Y|\mathbf{x}$  in this setting. First, we need a parameterization for  $h_{\mathbf{x}}$ , which is a continuous pdf. Second, we require a function that measures how the prediction distribution is different from the target distribution. Third, we need a method to convert a regression data set with samples  $(\mathbf{x}_j, y_j)$  with real-valued outputs to one with samples  $(\mathbf{x}_j, q_j)$  whose outputs are target distributions. Below we detail the first two choices, and in the next section we describe several methods for constructing target distributions.

We propose to restrict the prediction distribution  $h_{\mathbf{x}}$  to be a histogram density. Assume an interval  $[a, b]$  on the real line has been uniformly partitioned into  $k$  bins of width  $w_i$ , and let function  $h : \mathcal{X} \rightarrow [0, 1]^k$  provide the  $k$ -dimensional vector  $h(\mathbf{x})$  of coefficients indicating the probability the target is in each bin, given  $\mathbf{x}$ . The distribution  $h_{\mathbf{x}}$  corresponds to a (normalized) histogram and has density values  $h_i(\mathbf{x})/w_i$  per bin. A sample histogram is depicted in Figure 2 (top). The histogram prediction distribution  $h_{\mathbf{x}}$  can be parameterized with a softmax output layer in a neural network. The softmax layer has  $k$  units, and the value of unit  $i$  represents  $h_i(\mathbf{x})$ . Figure 2 (bottom) shows a sample neural network whose output represents a histogram.

For the measure of difference, we use the KL-divergence. Suppose the target distribution has CDF  $F$ , and the support of this distribution is within the range  $[a, b]$ . The KL-divergence between  $q$  and  $h_{\mathbf{x}}$  is

$$\begin{aligned} D_{KL}(q||h_{\mathbf{x}}) &= - \int_a^b q(y) \log \frac{h_{\mathbf{x}}(y)}{q(y)} dy \\ &= - \int_a^b q(y) \log h_{\mathbf{x}}(y) dy - \left( - \int_a^b q(y) \log q(y) dy \right). \end{aligned}$$

Because the second term only depends on  $q$ , the aim is to minimize the first term: the cross-entropy between  $q$  and  $h_{\mathbf{x}}$ . This loss simplifies due to the histogram form on  $h_{\mathbf{x}}$ :

$$- \int_a^b q(y) \log h_{\mathbf{x}}(y) dy = - \sum_{i=1}^k \int_{l_i}^{l_i+w_i} q(y) \log \frac{h_i(\mathbf{x})}{w_i} dy$$

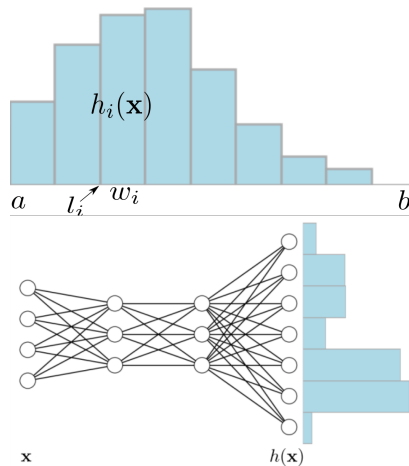


Figure 2: (Top) A sample histogram, and (bottom) a neural network with a softmax output layer that represents a histogram.

$$= - \sum_{i=1}^k \log \frac{h_i(\mathbf{x})}{w_i} \underbrace{(F(l_i + w_i) - F(l_i))}_{c_i}.$$

In the minimization, the width itself can be ignored because  $\log \frac{h_i(\mathbf{x})}{w_i} = \log h_i(\mathbf{x}) - \log w_i$ , giving the histogram loss

$$\text{HL}(q, h_{\mathbf{x}}) = - \sum_{i=1}^k c_i \log h_i(\mathbf{x}). \quad (1)$$

This loss has several useful properties. One important property is that it is convex in  $h_i(\mathbf{x})$ ; even if the loss is not convex in all network parameters, it is at least convex on the last layer. The other two benefits are due to restricting the form of the predicted distribution  $h_{\mathbf{x}}$  to be a histogram density. First, the divergence to the full distribution  $q$  can be efficiently computed. Second, the choice of  $q$  is flexible, as long as its CDF can be evaluated for each bin. The weighting  $c_i = F(l_i + w_i) - F(l_i)$  can be computed offline once for each sample, making it inexpensive to query repeatedly for each sample during training.

## 2.2 Target Distributions

The histogram loss requires a target distribution corresponding to each input. The method for constructing a target distribution from the target  $y_j$  in the data set can be chosen upfront. Different choices simply result in different weightings in HL. Below, we consider some special cases that are of interest.

**Truncated Gaussian on  $Y|\mathbf{x}$  (HL-Gauss).** Consider a truncated Gaussian distribution, on support  $[a, b]$ , as the target distribution. The mean  $\mu$  for this Gaussian is the data point  $y_j$  itself, with fixed variance  $\sigma^2$ . The pdf  $q$  is

$$q(y) = \frac{1}{Z\sigma\sqrt{2\pi}} e^{-\frac{(y-y_j)^2}{2\sigma^2}}$$

where  $Z = \frac{1}{2}(\text{erf}(\frac{b-y_j}{\sqrt{2\sigma}}) - \text{erf}(\frac{a-y_j}{\sqrt{2\sigma}}))$ , and HL has

$$c_i = \frac{1}{2Z} \left( \text{erf} \left( \frac{l_i + w_i - y_j}{\sqrt{2\sigma}} \right) - \text{erf} \left( \frac{l_i - y_j}{\sqrt{2\sigma}} \right) \right).$$

This distribution enables smoothing over  $Y$  through the variance parameter  $\sigma^2$ . We call this loss HL-Gauss, which is defined by the number of bins  $k$  and variance  $\sigma^2$ . Based on positive empirical performance, it is the main HL variant that we advocate for and analyze. Combined with Equation (1), the final form for this loss is:

$$\text{HL-Gauss}(y_j, h_{\mathbf{x}}) = - \sum_{i=1}^k \frac{1}{2Z} \left( \text{erf} \left( \frac{l_i + w_i - y_j}{\sqrt{2\sigma}} \right) - \text{erf} \left( \frac{l_i - y_j}{\sqrt{2\sigma}} \right) \right) \log h_i(\mathbf{x}).$$

**Dirac delta on  $Y|\mathbf{x}$  (HL-OneBin).** Consider Gaussians centered around data points  $y_j$ , with arbitrarily small variances  $\frac{1}{2}a^2$ :

$$\delta_{a,j}(y) = \frac{1}{a\sqrt{\pi}} \exp \left( -\frac{(y-y_j)^2}{a^2} \right).$$

Let the target distribution have  $q(y) = \delta_{a,j}(y)$  for each sample. Define function  $c_{i,j} : [0, \infty) \rightarrow [0, 1]$  as  $c_{i,j}(a) = \int_{l_i}^{l_i+w_i} \delta_{a,j}(y) dy$ . For each  $y_j$ , as  $a \rightarrow 0$ ,  $c_{i,j}(a) \rightarrow 1$  if  $y_j \in [l_i, l_i + w_i]$  and  $c_{i,j}(a) \rightarrow 0$  otherwise. So, for  $\text{bin}_j$  s.t.  $y_j \in [l_{\text{bin}_j}, l_{\text{bin}_j} + w_{\text{bin}_j}]$ ,

$$\lim_{a \rightarrow 0} HL(\delta_{a,j}, h_{\mathbf{x}_j}) = -\log h_{\text{bin}_j}(\mathbf{x}_j).$$

The sum over samples for HL to the Dirac delta on  $Y|\mathbf{x}$ , then, corresponds to the negative log-likelihood for  $h_{\mathbf{x}}$

$$\operatorname{argmin}_{h_1, \dots, h_k} - \sum_{j=1}^n \log h_{\text{bin}_j}(\mathbf{x}_j) = \operatorname{argmin}_{h_1, \dots, h_k} - \sum_{j=1}^n \log h_{\mathbf{x}_j}(y_j).$$

Such a delta distribution on  $Y|\mathbf{x}$  results in one coefficient  $c_i$  being 1, reflecting the distributional assumption that  $Y$  is certainly in a bin. In the experiments, we compare to this loss, which we call HL-OneBin. This makes the final form of the loss:

$$\text{HL-OneBin}(y_j, h_{\mathbf{x}}) = -\log h_{\text{bin}_j}(\mathbf{x}).$$

**Mixture of Dirac delta and uniform on  $Y|\mathbf{x}$  (HL-Uniform).** Using a similar analysis to above,  $q(y)$  can be considered as a mixture between  $\delta_{a,j}(y)$  and a uniform distribution. For a weighting of  $\epsilon$  on the uniform distribution, the resulting loss HL-Uniform has  $c_i = \epsilon$  for  $i \neq i_j$  and  $c_{i_j} = 1 - (k - 1)\epsilon$ . The loss can be expressed as:

$$\text{HL-Uniform}(y_j, h_{\mathbf{x}}) = -(1 - (k - 1)\epsilon) \log h_{\text{bin}_j}(\mathbf{x}) - \sum_{i \neq \text{bin}_j} \epsilon \log h_{\text{bin}_i}(\mathbf{x}).$$

### 3. Theoretical Analysis

This section assembles our theoretical results on the histogram loss (HL). The first part considers the behavior of gradient descent optimization on HL and the  $\ell_2$  loss. Gradient descent is a local search method, and problems like the existence of highly varying regions, unfavorable local minima and saddle points, and poorly conditioned coordinates can hinder the search. A complete characterization of the loss surface is a hard problem. Instead, we provide an upper bound on the norm of the gradient of HL, which suggests that the gradient varies less through training and provides more reliable optimization steps. We also point out how this norm relates to the generalization performance of the model. The second part shows a similarity between minimizing HL and entropy-regularized policy improvement based on work by Norouzi et al. (2016).

The third part discusses the bias of the minimizer of HL. The mean of the target generating distribution can be different from a coarse histogram prediction distribution that approximates it. We show that when HL-Gauss is minimized, this difference is bounded by half of the bin width as long as the target distribution has negligible probability beyond the support of the histogram. The last part shows that minimizing HL on the data reduces an upper bound on the difference between the targets and the mean of prediction distributions.

### 3.1 Stable Gradients for the Histogram Loss

Hardt et al. (2016) have shown that the generalization performance for stochastic gradient descent is bounded by the number of steps that stochastic gradient descent takes during training, even for non-convex losses. The bound is also dependent on the properties of the loss. In particular, it is beneficial to have a loss function with small Lipschitz constant  $L$ , which bounds the norm of the gradient. Below, we discuss how HL with a Gaussian distribution (HL-Gauss) in fact promotes an improved bound on this norm, over both the  $\ell_2$  loss and HL with all weight in one bin (HL-OneBin).

In the proposition bounding the HL-Gauss gradient, we assume

$$h_i(\mathbf{x}) = \frac{\exp(\boldsymbol{\phi}_\theta(\mathbf{x})^\top \mathbf{w}_i)}{\sum_{j=1}^k \exp(\boldsymbol{\phi}_\theta(\mathbf{x})^\top \mathbf{w}_j)} \quad (2)$$

for some function  $\boldsymbol{\phi}_\theta : \mathcal{X} \rightarrow \mathcal{R}^k$  parameterized by a vector of parameters  $\boldsymbol{\theta}$ . For example,  $\boldsymbol{\phi}_\theta(\mathbf{x})$  could be the last hidden layer in a neural network, with parameters  $\boldsymbol{\theta}$  for the entire network up to that layer. The proposition provides a bound on the gradient norm in terms of the *current network parameters*. Our goal is to understand how the gradients might vary *locally for the parameters*, as opposed to globally bounding the norm and characterizing the Lipschitz constant only in terms of the properties of the function class and loss.

**Proposition 1 (Local Lipschitz constant for HL-Gauss)** *Assume  $\mathbf{x}, y$  are fixed, giving fixed coefficients  $c_i$  in HL-Gauss. Let  $h_i(\mathbf{x})$  be as in (2), defined by the parameters  $\mathbf{w} = \{\mathbf{w}_1, \dots, \mathbf{w}_k\}$  and  $\boldsymbol{\theta}$ , providing the predicted distribution  $h_{\mathbf{x}}$ . Assume for all  $i$  that  $\mathbf{w}_i^\top \boldsymbol{\phi}_\theta(\mathbf{x})$  is locally  $l$ -Lipschitz continuous with respect to  $\boldsymbol{\theta}$ :*

$$\|\nabla_{\boldsymbol{\theta}}(\mathbf{w}_i^\top \boldsymbol{\phi}_\theta(\mathbf{x}))\| \leq l. \quad (3)$$

*Then the norm of the gradient for HL-Gauss, with respect to all the parameters in the network  $\{\boldsymbol{\theta}, \mathbf{w}\}$ , is bounded by*

$$\|\nabla_{\boldsymbol{\theta}, \mathbf{w}} HL(q, h_{\mathbf{x}})\| \leq (l + \|\boldsymbol{\phi}_\theta(\mathbf{x})\|) \sum_{i=1}^k |c_i - h_i(\mathbf{x})|. \quad (4)$$

The results by Hardt et al. (2016) suggest it is beneficial for the local Lipschitz constant—or the norm of the gradient—to be small on each step. HL-Gauss provides exactly this property. Besides the network architecture—which we are here assuming is chosen outside of our control—the HL-Gauss gradient norm is proportional to  $|c_i - h_i(\mathbf{x})|$ . This number is guaranteed to be less than 1 but is generally likely to be even smaller, especially if  $h_i(\mathbf{x})$  reasonably accurately predicts  $c_i$ . Further, the gradients should push the weights to stay within a range specified by  $c_i$ , rather than preferring to push some to be very small—close to 0—and others to be close to 1. For example, if  $h_i(\mathbf{x})$  starts relatively uniform, then the objective does not encourage predictions  $h_i(\mathbf{x})$  to get smaller than  $c_i$ . If  $c_i$  are non-negligible, this keeps  $h_i(\mathbf{x})$  away from zero and the loss in a smaller range.

This contrasts both the norm of the gradient for the  $\ell_2$  loss and HL-OneBin. For the  $\ell_2$  loss,  $(f(\mathbf{x}) - y) \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \mathbf{w}^\top \boldsymbol{\phi}_\theta(\mathbf{x}) \\ \boldsymbol{\phi}_\theta(\mathbf{x}) \end{bmatrix}$  is the gradient, giving gradient norm bound  $(l + \|\boldsymbol{\phi}_\theta(\mathbf{x})\|) |f(\mathbf{x}) - y|$ . The constant  $|f(\mathbf{x}) - y|$ , as opposed to  $\sum_{i=1}^k |c_i - h_i(\mathbf{x})|$ , can be much larger, even if  $y$  is

normalized between  $[0, 1]$ , and can vary considerably more. HL-OneBin, on the other hand, shares the same constant as HL-Gauss but suffers from another problem. The Lipschitz constant  $l$  in Equation (3) will likely be larger because  $c_i$  is frequently zero and so pushes  $h_i(\mathbf{x})$  towards zero. This results in larger objective values and pushes  $\mathbf{w}_i^\top \phi_\theta(\mathbf{x})$  to get larger to enable  $h_i(\mathbf{x})$  to get close to 1.

### 3.2 Connection to Reinforcement Learning

HL can also be motivated through a connection to maximum entropy reinforcement learning. In reinforcement learning, an agent iteratively selects actions and transitions between states to maximize (long-term) reward. The agent’s goal is to find an optimal policy in as few interactions as possible. To do so, the agent begins by exploring more to facilitate efficiently finding a better policy. Supervised learning can be expressed as a reinforcement learning problem (Norouzi et al., 2016), where action selection conditioned on a state corresponds to making a prediction conditioned on a feature vector. An alternative view to minimizing prediction error is to search for a policy to make accurate predictions.

One strategy to efficiently find an optimal policy is through a maximum entropy objective. The policy balances between selecting the action it believes to be optimal—making its current best prediction—and acting more randomly with high entropy. For a continuous action set  $\mathcal{Y}$ , the goal is to minimize the following objective

$$\int_{\mathcal{X}} p_s(\mathbf{x}) \left[ -\tau H(h_{\mathbf{x}}) - \int_{\mathcal{Y}} h_{\mathbf{x}}(y) r(y, y_i) dy \right] d\mathbf{x}, \quad (5)$$

where  $\tau > 0$ ;  $p_s$  is a distribution over states  $\mathbf{x}$ ;  $h_{\mathbf{x}}$  is the policy or distribution over actions for a given  $\mathbf{x}$ ;  $H(\cdot)$  is the differential entropy—the extension of entropy to continuous random variables; and  $r(y, y_i)$  is the reward function, such as the negative of the objective  $r(y, y_i) = -\frac{1}{2}(y - y_i)^2$ . Minimizing (5) corresponds to minimizing the KL-divergence across  $\mathbf{x}$  between  $h_{\mathbf{x}}$  and the exponentiated payoff distribution  $q(y) = \frac{1}{Z} \exp(r(y, y_i)/\tau)$  where  $Z = \int \exp(r(y, y_i)/\tau)$  because

$$\begin{aligned} D_{KL}(h_{\mathbf{x}}||q) &= -H(h_{\mathbf{x}}) - \int h_{\mathbf{x}}(y) \log q(y) dy \\ &= -H(h_{\mathbf{x}}) - \tau^{-1} \int h_{\mathbf{x}}(y) r(y, y_i) dy + \log Z. \end{aligned}$$

The connection between HL and maximum-entropy reinforcement learning is that both are minimizing a divergence to this exponentiated distribution  $q$ . HL, however, is minimizing  $D_{KL}(q||h_{\mathbf{x}})$  instead of  $D_{KL}(h_{\mathbf{x}}||q)$ . For example, Gaussian target distribution with variance  $\sigma^2$  corresponds to minimizing  $D_{KL}(q||h_{\mathbf{x}})$  with  $r(y, y_i) = -\frac{1}{2}(y - y_i)^2$  and  $\tau = \sigma^2$ . These two KL-divergences are not the same, but a similar argument to Norouzi et al. (2016) could be extended for continuous  $y$ , showing that  $D_{KL}(h_{\mathbf{x}}||q)$  is upper-bounded by  $D_{KL}(q||h_{\mathbf{x}})$  plus variance terms. The intuition, then, is that minimizing HL is promoting an efficient search for an optimal (prediction) policy.

### 3.3 Bias of the Histogram Loss

Different forms of bias can be induced by components of the histogram loss, namely using histogram densities, using the KL-divergence, and the choice of target distribution. This

section explores the effects of these components on the bias. First, we characterize the minimizer of HL and the effect of target distribution on it. Then, we show to what extent the mean of the minimizer differs from the Bayes optimal solution.

We make two assumptions in this section to simplify the analysis and highlight the key points. First, we assume that the model is flexible enough so that the predicted distribution for each input can be optimized independently. Therefore, we consider the case where all the data points in the data set have the same input and drop the subscripts that show dependence on  $\mathbf{x}$ . Second, we assume that the bins have equal width  $w$  and the support of the histogram is unbounded, and therefore the target distribution is not truncated. For a fixed placement and indexing of the bins, we denote the set of distributions with this form as  $\mathcal{P}(\mathbb{N})$  and the set of distributions without the histogram form restriction as  $\mathcal{P}(\mathbb{R})$ . Later in the experiments, we pad the support of the histogram to ensure that there is negligible truncation.

In the data, each input can be associated with one or multiple targets. More generally, we consider a *target generating distribution* with pdf  $p : \mathcal{Y} \rightarrow [0, \infty)$  for an input. This is the underlying distribution that generates the targets given an input, and it is different from the target distribution that is used in HL. In the data set, targets  $y_j$  are samples from  $p$  and each one is turned into a *target distribution* with pdf  $q_j$  (e.g. a truncated Gaussian centered at  $y_j$  if we use HL-Gauss, or a Dirac delta at  $y_j$  if we use HL-OneBin) which is then used to train the model with HL. We use  $q_y$  to denote the pdf of the target distribution obtained from the target  $y$ . Finally, the model's predicted distribution for the input is denoted by  $h$ .

In this setup, the predictor that is Bayes optimal in squared error predicts  $\mathbb{E}_p[y]$ . We define the bias of a predicted distribution  $h^*$  as  $\text{Bias}(h^*, p) = \mathbb{E}_{h^*}[z] - \mathbb{E}_p[y]$ . The following proposition characterizes the bias incurred from predicting a histogram. The proof is in Appendix A.

**Proposition 2 (Bias Characterization)** *The following statements hold:*

1. *If the form of  $h$  is not restricted, the minimizer of the KL-divergence is the result of smoothing  $p$  with  $q$ . Formally,  $\text{argmin}_{h \in \mathcal{P}(\mathbb{R})} \mathbb{E}_{y \sim p}[D_{KL}(q_y || h)] = s$  where  $s(z) = \int p(y)q_y(z) dy$ .*
2. *The bias of  $s$  is the expected difference between the scalar target and the mean of the target distribution constructed from it. If the target distribution is Gaussian,  $s$  is unbiased. Formally,  $\text{Bias}(s, p) = \mathbb{E}_{y \sim p}[\mathbb{E}_{z \sim q_y}[z] - y]$ .*
3. *If the form of  $h$  is restricted to a histogram with bins of equal widths, the minimizer of HL is  $h^*$ , where the probability in each bin is equal to the probability of  $s$  in the range of that bin. Formally,  $\text{argmin}_{h \in \mathcal{P}(\mathbb{N})} \mathbb{E}_{y \sim p}[\text{HL}(q_y || h)] = h^*$  where  $h_i^* = \int_{l_i}^{l_i+w} s(z) dz$ .*
4. *If the target distribution is Gaussian, the bias of  $h^*$  is at most half of the bin width, and this bound is tight, i.e.  $|\text{Bias}(h^*, p)| \leq w/2$ .*

Recall that this section does not explore the biases incurred due to truncation or the model's lack of capacity to fit different data points. The form of bias in this section is due to restricting the prediction to a histogram, and we call it the *discretization bias*. This analysis characterizes the bias of the mean of the distribution that minimizes HL-Gauss and provides an upper bound on it that can be arbitrarily reduced by using smaller bins,

assuming that the target distributions have negligible probability exceeding the support of the histogram. Note that the variance parameter affects both the quantified bias and the extent to which the assumption is satisfied. Section C provides numerical simulations that show the discretization bias in HL-Gauss reduces as the variance parameter increases and the total bias becomes negligible with proper padding. We will also empirically investigate the bias in Section 4 on regression data sets.

### 3.4 Bound on Prediction Error

An often-sought property of surrogate loss functions is reducing an upper bound on the original objective. In this section we provide a result that shows, if the prediction distribution and the target distributions have a similarly bounded support, minimizing the KL-divergence between them reduces an upper bound on the difference between the means. This result motivates minimizing HL and using the mean of the prediction distribution as the final prediction.

**Proposition 3 (Bound on Prediction Error)** *Assuming that for a data point, the target distribution  $q_y$  and the model’s prediction distribution  $h_{\mathbf{x}}$  have supports bounded by the range  $[a, b]$ , the following bound holds:*

$$(\mathbb{E}_{q_y}[z] - \mathbb{E}_{h_{\mathbf{x}}}[z])^2 \leq 4 \max(|a|, |b|)^2 \min\left(\frac{1}{2}D_{KL}(q_y||h_{\mathbf{x}}), 1 - e^{-D_{KL}(q_y||h_{\mathbf{x}})}\right). \quad (6)$$

The proof is in Appendix A, and the behavior of the two bounds is explained in Appendix B. A model trained with HL reduces  $D_{KL}(q||h_{\mathbf{x}})$  for each training data point. If the mean of the target distribution is close to the original label, the bound above shows that the mean of the predicted distribution is a good predictor for the original labels in the training set if the model achieves a low training loss.

## 4. An Empirical Study on the Histogram Loss

In this section, we investigate the utility of HL-Gauss for regression compared to using the  $\ell_2$  loss. We particularly investigate why the modification to this distributional loss improves performance by first comparing to baselines to test if it is due to each of the various ways in which the two losses differ and then designing experiments to explore other properties of HL.

### 4.1 A Synthetic Experiment

We start with an experiment on sine functions to understand the difference between HL-Gauss and the  $\ell_2$  loss in a controlled setting. The goal is to see how the convergence rates of the two losses compare as the frequency or the offset of the target changes. The function we want to fit has an input  $x \in \mathbb{R}$  and a deterministic target  $\sin(\alpha x) + \beta$ , where  $\alpha, \beta \in \mathbb{R}$  determine the frequency and the offset, respectively. We pick 500 points on a grid from the range  $[-\pi, \pi]$  and generate the corresponding targets. An example is depicted as the blue dots in Figure 3 (Left). We use a neural network with two hidden layers of width 1024 and leaky ReLU activation, train it with the Adam optimizer, and report the MSE on the

same 500 points during training. When comparing HL-Gauss and  $\ell_2$ , we try a wide range of step-size values in  $\{10^{-5}, 10^{-4}, \dots, 10^{-1}\}$  for each method and show the curve with the lowest area under the curve during 1000 iterations.

In the first part of the experiment, we fix  $\beta$  to 0 and vary the frequency  $\alpha \in \{1, 10, 20\}$ . The range of the target is  $[-1, 1]$ , and for HL-Gauss, we discretize the range  $[-1.5, 1.5]$  evenly to 100 bins and set  $\sigma$  to twice the resulting bin width. As shown in Figure 3 (Middle), for both losses training slows down with higher target frequencies and there is a clear difference between HL-Gauss and  $\ell_2$  in high frequencies. We have used a sufficiently wide range of step-size values to ensure this difference cannot be eliminated with a larger step-size. The predictions obtained with HL-Gauss and  $\ell_2$  for  $\alpha = 10$  are visualized in Figure 3 (Left).

We now fix  $\alpha$  to 10 and vary the offset  $\beta \in \{0, 1, 10\}$ . The range of the target is  $[-1 + \beta, 1 + \beta]$  and the support of the histogram in HL-Gauss is  $[-1.5, 11.5]$  regardless of the offset. The number of bins and  $\sigma$  are similar to the previous part. The learning curves are shown in Figure 3 (Right). The convergence rate of  $\ell_2$  deteriorates as the offset rises while the behavior of HL-Gauss is not affected.

Overall, the results in this synthetic experiment suggest that, compared to the  $\ell_2$  loss, HL-Gauss is especially helpful for convergence when the target has high frequency components or is far from zero.

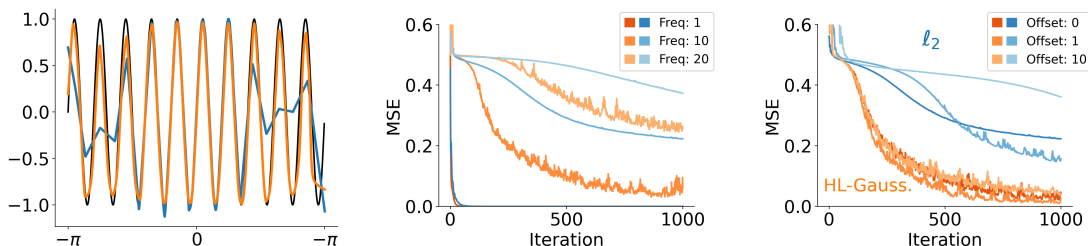


Figure 3: (Left) A synthetic task with frequency 10 and offset 0. The black curve shows the training data, and the orange and blue curves show the predictions obtained after training with HL-Gauss and  $\ell_2$ , respectively. (Middle) Learning curves for different frequencies in  $\{1, 10, 20\}$  and fixed offset 0. Orange and blue denote HL-Gauss and  $\ell_2$ , and brighter shades denote higher frequencies. Each curve is averaged over 5 runs with different random initializations. (Right) Learning curves for different offsets in  $\{0, 1, 10\}$  and fixed frequency 10. Brighter shades show higher offsets. Altogether, HL-Gauss trains remarkably faster than  $\ell_2$  when the target has high frequency or is far from zero.

## 4.2 Overview of Empirical Study

Now, we turn to the empirical study on real-world data. The purpose of these experiments is to first show proof of concept studies and then provide insights on the role of factors that differentiate HL from the  $\ell_2$  loss. The main goal is to understand when and why one would opt for HL. The data sets and the algorithms used in our experiments are described in Sections 4.3 and 4.4 respectively.

Section 4.5 compares HL-Gauss,  $\ell_2$ , and several baselines, each of which is designed to differ in a particular aspect from HL-Gauss or  $\ell_2$ . The comparison shows the importance of each of these factors. These experiments attempt to clarify (a) if HL-Gauss is a reasonable choice for a regression task, (b) whether the difference between the performance of HL-Gauss and  $\ell_2$  is merely because HL-Gauss is learning a flexible distribution, (c) if sensitivity of  $\ell_2$  to outliers can explain the difference the performance of the two losses, and (d) if this difference is the result of data augmentation in the output space when using HL-Gauss.

In Section 4.6, we investigate the impact of discretization bias on both HL-Gauss and HL-OneBin. Previously, we offered a theoretical discussion on the effect of the target distribution and the variance parameter on this bias in Section 3.3. The results in Section 4.6 show that this is indeed an important factor at play and can explain most of the difference between the behaviors of HL-OneBin and HL-Gauss. We further evaluate the models with different numbers of bins and variance parameter values to see if a bias-variance tradeoff is at work.

In reinforcement learning, learning the full distribution has been likened to auxiliary tasks that improve the performance by providing a better representation (Bellemare et al., 2017). Experiments in Section 4.7 test this hypothesis in a regression setting in three ways: (a) comparing the representations of a model that learns the distribution and one that learns the expected value, (b) evaluating a model with the main task of learning the expected value and the auxiliary task of learning the distribution, and (c) adding the auxiliary tasks of learning higher moments to a model that learns the expected value.

In Section 4.8 we compare the behavior of  $\ell_2$ , HL-OneBin, and HL-Gauss during the training to find if the difference in final performance is the result of different loss surfaces. This hypothesis is related to the gradient norms of different losses (described in Section 3.1) and has been posited by Imani and White (2018).

Finally, Section 4.9 measures the sensitivity of HL to target corruption and the sensitivity of the model to input perturbations. The first experiment tests to what extent the performance of a loss deteriorates in the presence of corrupted targets in the training data. The second experiment studies the sensitivity of a model’s output to input perturbation, which, in classification problems, has been closely associated with poor generalization.

### 4.3 Data Sets and Pre-Processing

Experiments are conducted on four regression data sets.

The **CT Position** data set is from CT images of patients (Graf et al., 2011), with 385 features and the target set to the relative location of the image.

The **Song Year** data set is a subset of The Million Song Dataset (Bertin-Mahieux et al., 2011), with 90 audio features for a song and a target corresponding to the release year.

The **Bike Sharing** data set (Fanaee-T and Gama, 2014), about hourly bike rentals for two years, has 16 features and the target set to the number of rented bikes.

The **Pole** data set (Olson et al., 2017) describes a telecommunication problem and has 49 features.

All data sets are complete, and there is no need for data imputation. All features are transformed to have zero mean and unit variance. We randomly split the data into train and test sets in each run. Root mean squared error (RMSE) and mean absolute error

(MAE) are reported over 5 runs, with standard errors. Other ways of processing these data sets may result in a higher performance. The goal of this study, however, is comparing different methods and testing hypotheses rather than achieving state-of-the-art results on these tasks. More details about these data sets are provided in Table 5 and Figure 14 in the appendix. We will show the results on CT Position data set in the main paper and defer the results on the other data sets to Appendix D.

#### 4.4 Algorithms

We compared several regression strategies, distribution learning approaches, and several HL variants. All the approaches—except for linear regression—use the same neural network, with differences only in the output layer. Unless specified otherwise, all networks using HL have 100 bins. The support of the histogram is chosen by the data set target range, and 10 bins are kept for padding on each side to minimize the effect of truncation. Hyperparameters for comparison algorithms are chosen according to the best Test MAE. Network architectures were chosen according to best Test MAE for  $\ell_2$ . More details about the architectures and the range of hyperparameters are provided in Appendix D.

**Linear Regression** is included as a baseline, using ordinary least squares with the inputs.

**Squared-error** ( $\ell_2$ ) is the neural network trained using the  $\ell_2$  loss. The targets are normalized to range  $[0, 1]$ , which was needed to improve stability and accuracy.

$\ell_2$ +**Noise** is the same as  $\ell_2$ , except Gaussian noise is added to the targets as a form of augmentation.

$\ell_2$ +**Clipping** is the same as  $\ell_2$ , but with gradient-norm clipping during training.

**Absolute-error** ( $\ell_1$ ) is the neural network trained using the  $\ell_1$  loss.

**MDN** is a Mixture Density Network (Bishop, 1994) that models the target distribution as a mixture of Gaussian distributions.

$\ell_2$ +**Softmax** uses a softmax-layer with  $\ell_2$  loss,  $\sum_{i=1}^k (f_i(\mathbf{x}_j)m_i - y_j)^2$  for bin centers  $m_i$ , with otherwise the same settings as HL-Gauss.

$\ell_2$ +**Softmax+Mean** uses a softmax-layer with  $\ell_2$  loss on the mean of the distribution,  $(\sum_{i=1}^k f_i(\mathbf{x}_j)m_i - y_j)^2$  for bin centers  $m_i$ , with otherwise the same settings as HL-Gauss.

**HL-OneBin** is HL with a Dirac delta target distribution.

**HL-Uniform** is HL with a target distribution that mixes between a delta distribution and the uniform distribution, with a weighting of  $\epsilon$  on the uniform and  $1 - \epsilon$  on the delta.

**HL-Gauss** is HL with a truncated Gaussian distribution as the target distribution. The parameter  $\sigma$  is set to the width of the bins. Importantly, we do not tune any of HL-Gauss’s hyperparameters to individual data sets.

#### 4.5 Overall Results

We first report the relative performance of different baselines compared to HL-Gauss in Table 1. The error rates of HL-Uniform are plotted separately in Figure 4 to show that, regardless of the choice of the parameter in the loss, a uniform target distribution cannot help performance. We focus on Test MAE as the evaluation criterion due to considerably lower variability in the results and its wide use in practice and present Train MAE, Train RMSE, and Test RMSE to provide a complete picture.

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lin Reg	6.073( $\pm 0.007$ )	8.209( $\pm 0.006$ )	6.170( $\pm 0.025$ )	8.341( $\pm 0.025$ )
$\ell_2$	0.140( $\pm 0.036$ )	0.183( $\pm 0.048$ )	0.176( $\pm 0.034$ )	0.267( $\pm 0.039$ )
$\ell_2$ +Noise	0.114( $\pm 0.011$ )	0.152( $\pm 0.015$ )	0.152( $\pm 0.010$ )	0.311( $\pm 0.076$ )
$\ell_2$ +Clip	0.111( $\pm 0.004$ )	0.143( $\pm 0.005$ )	0.148( $\pm 0.004$ )	0.231( $\pm 0.006$ )
$\ell_1$	0.121( $\pm 0.007$ )	0.164( $\pm 0.007$ )	0.162( $\pm 0.006$ )	0.389( $\pm 0.038$ )
MDN	0.114( $\pm 0.004$ )	0.152( $\pm 0.005$ )	0.153( $\pm 0.004$ )	0.308( $\pm 0.035$ )
$\ell_2$ +Softmax	0.065( $\pm 0.006$ )	0.094( $\pm 0.008$ )	0.105( $\pm 0.006$ )	0.301( $\pm 0.067$ )
$\ell_2$ +Softmax+Mean	0.115( $\pm 0.029$ )	0.261( $\pm 0.141$ )	0.122( $\pm 0.028$ )	0.302( $\pm 0.129$ )
HL-OneBin	0.309( $\pm 0.000$ )	0.364( $\pm 0.006$ )	0.335( $\pm 0.004$ )	0.660( $\pm 0.099$ )
HL-Gauss.	0.061( $\pm 0.006$ )	0.164( $\pm 0.090$ )	0.098( $\pm 0.005$ )	0.274( $\pm 0.090$ )

Table 1: Overall results on CT Scan. HL-Gauss achieved the lowest Test MAE. Among the other methods,  $\ell_2$ +softmax yielded an error rate close to that of HL-Gauss, and the rest of the baselines performed worse. There was a large gap between the performance of HL-OneBin and HL-Gauss.

The overall conclusion is that HL-Gauss does not harm performance and can often considerably improve performance over alternatives.

Another observation is that simply modeling the output distribution cannot achieve HL-Gauss’s performance. This is made clear by comparing HL-Gauss and MDN. MDN learns the output distribution, but as a mixture of Gaussian components rather than a histogram. In our experiments, MDN consistently underperformed HL-Gauss. Note that the error rate reported for MDN is the one observed after fixing numerical instabilities and tuning the number of components.

A related idea to learning the distribution explicitly is to use data augmentation in the target space as an implicit approach to minimizing divergences to distributions. We therefore also compared to directly modifying the labels and gradients, with  $\ell_2$ +Noise and  $\ell_2$ +Clipping. These approaches do perform slightly better than regression for some settings but do not achieve the same gains as HL-Gauss. The conclusion is that HL-Gauss’s performance cannot be attributed to data augmentation in the targets.

A well-known weakness of the  $\ell_2$  loss is its sensitivity to outliers. To find if the difference between the performance of the  $\ell_2$  loss and HL-Gauss can be explained by the presence of outliers, we can compare HL-Gauss with the  $\ell_1$  loss, which is robust to outliers. The model trained with  $\ell_1$  still underperformed HL-Gauss on three data sets, which suggests that the gap between the performance of HL-Gauss and the  $\ell_2$  loss is not merely due to the presence of outliers.

Choices of target distribution other than a Gaussian distribution are not effective. HL-OneBin and HL-Uniform appear to have fewer advantages, as shown in the results, and both can actually do worse than  $\ell_2$ . A uniform target distribution worsened the error on all data sets. An important artifact of label smoothing in HL-Uniform is that it biases the mean of the distribution. Since the mean of a uniform distribution is the center of the range, mixing the target distribution with a uniform distribution pulls the mean towards the center of the range. Recall from Section 3 that HL will suffer from large bias if the mean of the target distribution is far from the target.

Finally,  $\ell_2$ +softmax performed close to HL-Gauss on the CT Position data set and slightly worse on the other tasks. While this model does not estimate the target distribution by minimizing the KL-divergence, it still benefits from the softmax non-linearity in the output layer like HL-Gauss. The comparison between  $\ell_2$ +softmax and HL-Gauss shows that this softmax nonlinearity appears to be beneficial, but it cannot totally explain HL-Gauss’s performance. In our experiments on the three other data sets in Appendix F,  $\ell_2$ +softmax consistently underperformed HL-Gauss. Of additional note is that  $\ell_2$ +softmax+mean is slightly worse than  $\ell_2$ +softmax. The  $\ell_2$ +softmax+mean can be viewed as just adding the softmax nonlinearity to the NN and still having a scalar final output to predict the target, whereas  $\ell_2$ +softmax has a squared loss per bin and is more similar to HL-Gauss.

The comparisons in this section address several questions regarding HL-Gauss’s performance. HL-Gauss can often work better than  $\ell_2$ , and this rise in performance cannot be solely attributed to learning a flexible distribution, using data augmentation in the labels, being robust to outliers in the data set, or employing a softmax nonlinearity in the model.

#### 4.6 Bias and the Choice of Target Distribution

Characterizing the discretization bias in section 3 showed a link between the bias of the loss and the choice of  $\sigma$  in HL-Gauss. This parameter can also affect the bias from truncation as it controls the probability in the tails of the target distribution that exceed the support of the histogram. This section empirically investigates the effect of the target distribution on the bias of the loss. We first show how different values of  $\sigma$  result in different levels of bias. Then, we compare the target distributions of HL-Gauss and HL-OneBin with another target distribution which reduces the bias to zero. Finally, we evaluate HL-OneBin and HL-Gauss using different parameter values to find if these parameters control a bias-variance trade-off.

Weightings in HL-Gauss (which represent the area under the target distribution’s pdf in the range of each bin) are computed before training, and the histogram is trained to mimic these weightings. To obtain low bias, we want the mean of the trained histogram to be close to the original target. To analyze how close the loss is to this situation in practice, we plot the mean absolute error between the original targets in our data sets and

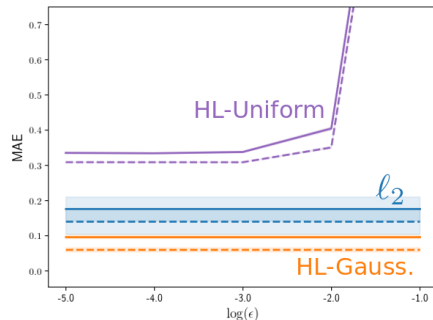


Figure 4: HL-Uniform results on CT Scan. Dotted and solid lines show train and test errors respectively. The parameter  $\epsilon$  is the weighting on the uniform distribution and raising it only impaired performance.

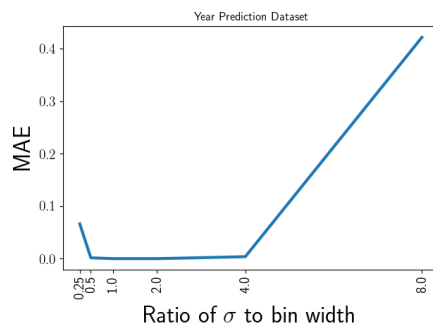


Figure 5: MAE between the means of HL targets and the original labels. It can be seen that extreme values of  $\sigma$  on either side biased the mean of the target distributions.

the mean of the histograms that perfectly match weightings obtained from those targets. Figure 5 shows this error for HL-Gauss with 100 bins and different choices of  $\sigma$ .

This simple analysis suggests that choosing a Gaussian target distribution with a carefully tuned variance parameter (rather than Dirac delta) can come with the extra benefit of reducing bias. To see to what extent the gap between HL-OneBin and HL-Gauss’s performance is because of this bias and to find if histogram losses in general are suffering from a large bias, we introduce a target distribution that reduces the discretization bias to zero (regardless of the number of bins) and compare it to HL-OneBin and HL-Gauss in the ordinary setting of 100 bins. As we showed in Section 3, if the target distribution  $q_y$  is chosen so that, for each sample, the expected value of the closest histogram density to it is exactly  $y$ , discretization bias is zero. Assume  $m_i$  is the last bin center before  $y$  (so  $y$  is somewhere between  $m_i$  and  $m_{i+1}$ ). The idea is to use a histogram density with probability  $1 - \frac{y-m_i}{w}$  in bin  $i$  and probability  $\frac{y-m_i}{w}$  in bin  $i + 1$  as the target distribution.<sup>2</sup> The expected value of this target distribution is always  $y$ , so this target distribution eliminates discretization bias. We call the histogram loss with this choice of target distribution *HL-Projected* and compare it with HL-OneBin and HL-Gauss on the four data sets.

Method	CT Scan	Song Year	Bike Sharing	Pole
HL-OneBin	0.335( $\pm 0.004$ )	5.906( $\pm 0.020$ )	26.689( $\pm 0.280$ )	1.264( $\pm 0.019$ )
HL-Projected	0.103( $\pm 0.003$ )	5.917( $\pm 0.019$ )	26.180( $\pm 0.348$ )	0.741( $\pm 0.018$ )
HL-Gauss.	0.098( $\pm 0.005$ )	5.903( $\pm 0.010$ )	25.525( $\pm 0.331$ )	0.714( $\pm 0.024$ )

Table 2: Discretization bias experiment. The reported numbers are Test MAE. HL-Projected achieved a Test MAE close to that of HL-Gauss and performed noticeably better than HL-OneBin. The gap between HL-Gauss and HL-Projected is small but appears on all the four data sets, and the gap is a bit bigger for Test RMSE (see Table 8).

While the difference between the performance of HL-Projected and HL-Gauss is small, it is consistent on the four data sets. The comparison shows that using this target distribution instead of HL-OneBin often results in a noticeable reduction in the errors, but the performance of HL-Gauss remains unbeaten. Although HL-Projected removes discretization bias, it still does not have HL-Gauss’s desirable property of punishing faraway predictions more severely. Overall, the bias of HL-Gauss with proper  $\sigma$  and padding is negligible, as discussed further in Appendix C.

The parameters in the loss, namely the number of bins and  $\sigma$ , can affect the bias. A hypothesis is that a good choice of parameters for HL can reduce overfitting and place the method in a sweet spot in a bias-variance trade-off. Two experiments were designed to find if there is a bias-variance trade-off at work. In the first experiment, we changed the number of bins while keeping the padding and target distribution  $\sigma$  fixed. The second experiment studies the effect of changing  $\sigma$  on the performance of HL-Gauss. Figure 6 (Left) evaluates HL-Gauss and HL-OneBin with different numbers of bins, and Figure 6 (Right) shows the effect of changing  $\sigma$  on HL-Gauss.

2. This is close to the projection operator by Bellemare et al. (2017), although that projection was introduced to fix the problem of disjoint support for discrete distributions. Rowland et al. (2018) also studied this projection operator and found its ability of preserving the expected value to be beneficial in reinforcement learning.

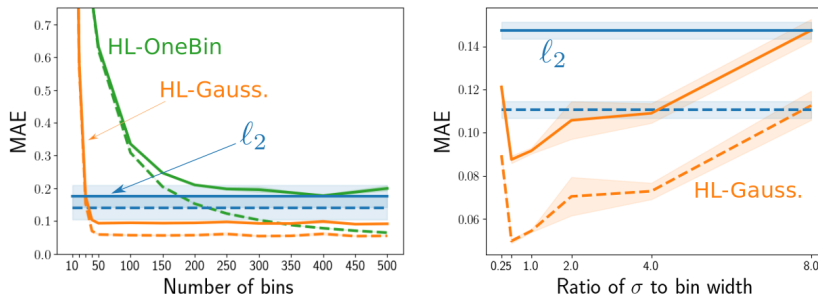


Figure 6: (Left) Changing the number of bins on CT Scan. Dotted and solid lines show train and test errors, respectively. The performance of  $\ell_2$  is shown as a horizontal line for comparison. A small number of bins resulted in high train and test errors, indicating high bias. A higher number of bins generally did not result in a rise in test error, with the exception of HL-OneBin’s test RMSE in Appendix F. (Right) Changing the parameter  $\sigma$  on CT Scan. Extreme values of  $\sigma$  resulted in bad performance. The error rates for train and test followed each other when changing this parameter.

Evaluating different numbers of bins shows that, although the quality of solution deteriorates when the bins are too few, there is no sign of overfitting in HL-Gauss with a higher number of bins. In the experiment on the  $\sigma$  parameter, we observe a v-shaped pattern. Increasing  $\sigma$  at first reduces both the train and test errors, and after a point, both errors begin to rise. Target distributions with higher  $\sigma$  have longer tails, so these high errors may be the result of truncation. There seems to be no evidence that the choice of  $\sigma$  caused overfitting, as both train and test error decreased and then increased together as we varied  $\sigma$ .

The analysis in this section shows the effect of the target distribution and the parameters of HL on the bias of the loss. Most of the gap between the performance of HL-Gauss and HL-OneBin can be explained by the high bias of HL-OneBin. Both the number of bins and  $\sigma$  can affect the bias of HL, but reducing this bias generally does not result in overfitting.

#### 4.7 Quality of Learned Representations

Learning a distribution, as opposed to a single statistic, provides a more difficult target—one that could require a better representation. The hypothesis is that among the functions  $f$  in the function class  $\mathcal{F}$ , there is a set of functions that can predict the targets almost equally well. To distinguish among these functions, a wider range of tasks can make it more likely to select the true function, or at least one that generalizes better. We conducted three experiments to test the hypothesis that an improved representation is learned.

First, we trained with HL-Gauss and  $\ell_2$  to obtain their representations. We tested (a) swapping the representations and re-learning only the last layer, (b) initializing with the other’s representation, and (c) using the same fixed random representation for both. For (a) and (c), the optimizations for both are convex, since the representation is fixed. If the challenge of predicting a distribution in HL results in a better representation, one

would expect the gap between  $\ell_2$  and HL-Gauss to go away when each one is trained on or initialized with the other’s representation.

	Loss	Default	Fixed	Initialized	Random
Train MAE	$\ell_2$	0.140( $\pm 0.036$ )	2.490( $\pm 0.094$ )	0.148( $\pm 0.013$ )	9.233( $\pm 0.157$ )
Train MAE	HL-Gauss.	0.061( $\pm 0.006$ )	0.153( $\pm 0.020$ )	0.063( $\pm 0.002$ )	2.604( $\pm 0.103$ )
Train RMSE	$\ell_2$	0.183( $\pm 0.048$ )	3.465( $\pm 0.158$ )	0.205( $\pm 0.022$ )	12.247( $\pm 0.172$ )
Train RMSE	HL-Gauss.	0.164( $\pm 0.090$ )	0.219( $\pm 0.027$ )	0.085( $\pm 0.004$ )	5.756( $\pm 0.232$ )
Test MAE	$\ell_2$	0.176( $\pm 0.034$ )	2.537( $\pm 0.089$ )	0.185( $\pm 0.013$ )	9.305( $\pm 0.191$ )
Test MAE	HL-Gauss.	0.098( $\pm 0.005$ )	0.187( $\pm 0.019$ )	0.101( $\pm 0.002$ )	2.727( $\pm 0.113$ )
Test RMSE	$\ell_2$	0.267( $\pm 0.039$ )	3.551( $\pm 0.143$ )	0.299( $\pm 0.025$ )	12.315( $\pm 0.216$ )
Test RMSE	HL-Gauss.	0.274( $\pm 0.090$ )	0.287( $\pm 0.024$ )	0.183( $\pm 0.005$ )	6.065( $\pm 0.234$ )

Table 3: Representation results on CT Scan. We tested (a) swapping the representations and re-learning on the last layer (**Fixed**), (b) initializing with the other’s representation (**Initialized**), (c) and using the same fixed random representation for both (**Random**) and only learning the last layer. Using HL-Gauss representation for  $\ell_2$  (first column, Fixed) caused a sudden spike in error, even though the last layer in  $\ell_2$  was re-trained. This suggests the representation is tuned to HL-Gauss. The representation did not even seem to give a boost in performance, as an initialization (second column, Initialization). Finally, even with the same random representation, where HL-Gauss cannot be said to improve the representation, HL-Gauss still obtained substantially better performance.

The results in Table 3 are surprisingly conclusive. They suggest that the performance gain is not due to better representations, and even under a random representation, HL-Gauss performs noticeably better than  $\ell_2$ .

In the second experiment, we tested a network called Multi-Task that predicted both the expected value (with the  $\ell_2$  loss) and the distribution (using HL-Gauss). The predicted distribution was not used for evaluation and was only present during the training as an auxiliary task to improve the representation. If predicting the extra information in a distribution is the reason for the superior performance of HL-Gauss, a regression network with the auxiliary task of predicting the distribution is expected to achieve a similar performance. Figure 7 (Left) shows the results on the four data sets.

It can be seen from these results that raising the coefficient for the auxiliary task does not improve the performance, and the new model generally has higher errors than HL-Gauss. The conclusion is that the extra information required for predicting the distribution cannot explain the gap between the  $\ell_2$  loss and HL-Gauss.

It might be argued that learning a histogram is not a suitable auxiliary task for regression, and the negative results above can be due to the incompatibility between predicting the mean with the  $\ell_2$  loss and predicting the distribution with HL-Gauss. We trained the  $\ell_2$  model with another auxiliary task in the third experiment. For the new task, predicting higher moments of the distribution, the target is raised to higher powers and the extra outputs try to estimate it by reducing the  $\ell_2$  loss. These auxiliary tasks of predicting higher moments are only used to improve the representation during training and are discarded at inference time. The results are shown in Figure 7 (Right).

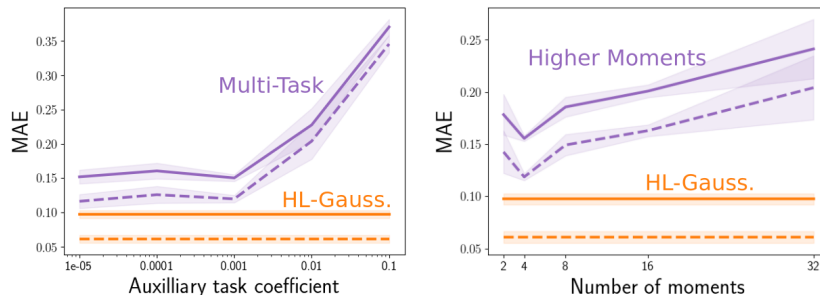


Figure 7: (Left) Multi-Task Network results on CT Scan. The loss function is the mean prediction’s squared error plus the distribution prediction’s KL-divergence multiplied by a coefficient. The horizontal axis shows the coefficient for KL-divergence. Dotted and solid lines show train and test errors, respectively. There was not a substantial drop in the error rate when increasing the coefficient in the loss, and the performance only became worse. (Right) Higher Moments Network results on CT Scan. The horizontal axis shows the number of moments (including the mean). There was no substantial decrease in error when predicting higher moments. With four moments, there was a slight reduction in error, but the new model still performed worse than HL-Gauss in terms of Test MAE.

There is little gain in predicting higher moments, and the new model still underperforms HL-Gauss. Like the previous experiment, predicting this extra information about the output distribution appears to have little benefit as an auxiliary task.

The experiments in this section consistently reject the hypothesis that the challenge of predicting the distribution is the main factor behind HL-Gauss’s performance. The performance cannot be achieved by exploiting HL-Gauss’s representation or by predicting the histogram or higher moments of the distribution as auxiliary tasks.

#### 4.8 Optimization Properties

In Section 3, we provided a bound on the gradient norm of HL that suggested ease of optimization with gradient descent. In this section, we show two experiments that study the optimization of the  $\ell_2$  loss and HL empirically.

First, we recorded the training errors and gradient norms during training to see if using HL-Gauss is beneficial for optimization. Each gradient norm value was normalized by the difference between the training loss observed at that point in the training of the model and the minimum possible value for the loss.<sup>3</sup> The goal was to find which model trains faster and has a smoother loss surface. Without the normalization of gradient norms, a simple scaling of the loss would shrink the fluctuations of gradient norms in the plot without making the optimization easier.

The results in Figure 8 suggest that HL-Gauss has more stable gradients than  $\ell_2$  and generally reduces the loss by a large amount in the first few epochs. As discussed in Section

3. The minimum value of the  $\ell_2$  loss is zero. HL, however, is the cross-entropy to the target distribution, so its minimum value is the entropy of the histogram that best estimates the target distribution.

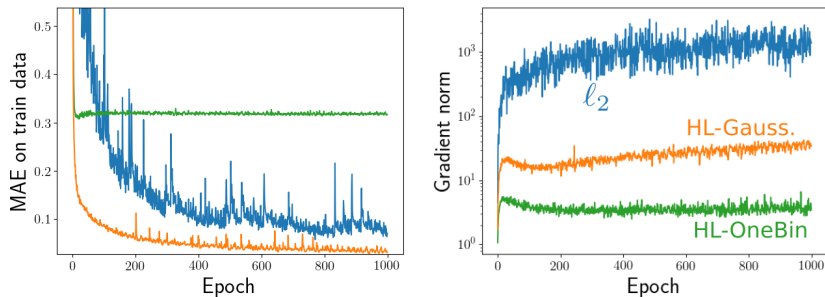


Figure 8: Training process on CT Scan. (Left) HL-Gauss reduced the train errors much faster than  $\ell_2$ . (Right) The gradient norm of  $\ell_2$  was highly varying through the training.

3, these properties can explain the performance of HL-Gauss. In comparison with HL-OneBin, however, HL-Gauss does not show a noticeable benefit in optimization.

The second experiment explores the role of  $\sigma$  in HL-Gauss. The benefit of the parameter  $\sigma$  in HL-Gauss can be twofold: (1) improving the final solution and (2) improving the optimization process. To compare the effect of these two properties, we design a network, called Annealing Network, that starts with a high  $\sigma$  and gradually reduces it to a small  $\sigma$  during training. The initial  $\sigma$  is 8 times the bin width, and during the first 20% of epochs the current value of  $\sigma$  is multiplied by a constant  $\tau$  at each epoch. The training continues without further annealing once 20% of the epochs are finished. The assumption is that the effect of  $\sigma$  on optimization shows up during training. Therefore, if the effect on optimization is substantial, the model that has a high  $\sigma$  earlier in training can benefit from it despite its small  $\sigma$  at the end. Results are shown in Figure 9.

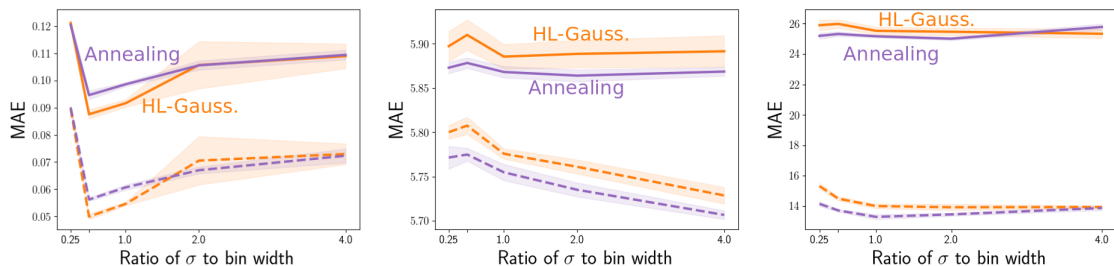


Figure 9: Annealing results on (Left) CT Scan, (Middle) Song Year, and (Right) Bike Sharing. Dotted and solid lines show train and test errors, respectively. The X-axis for Annealing shows the final value of  $\sigma$ . HL-Gauss keeps  $\sigma$  fixed to this value throughout training, while Annealing starts with a higher value of  $\sigma$  and reduces it to this value. There is some benefit in starting with a higher  $\sigma$  and reducing it on Song Year and Bike Sharing. See Appendix F for Pole and all RMSE results.

There was some gain in performance when the model started with a large value of  $\sigma$  and then reached a small value through annealing compared to training with a fixed small value of  $\sigma$ . So the parameter  $\sigma$  has some impact on the optimization.

The takeaway is that using HL instead of the  $\ell_2$  loss helps with optimization, which confirms the previous observation by Imani and White (2018). Further, the choice of the target distribution and the parameter in the loss have some effect on the optimization.

#### 4.9 Corrupted Targets and Input Perturbations

We tested the  $\ell_2$  loss,  $\ell_1$  loss, HL-OneBin, and HL-Gauss on data sets with corrupted targets. At each level, we replaced a ratio of the training targets with numbers sampled uniformly from the range of targets in the data set. Results in Figure 10 (Left) show that HL-Gauss and HL-OneBin are more robust to corrupted targets than  $\ell_2$  but not as robust as  $\ell_1$ . These results show a situation where the difference between the performance of HL and the  $\ell_2$  loss is more pronounced. Interestingly, HL can be worse than the  $\ell_1$  loss in the presence of corrupted targets.

We then compared the sensitivity of the outputs of models trained with the  $\ell_2$  loss,  $\ell_1$  loss, HL-OneBin, and HL-Gauss to input perturbations. The measure we used was the Frobenius norm of the Jacobian of the model’s output with respect to the input. Predictions in a regression problem are scalar values, so the Jacobian becomes a vector whose elements are the derivative of the output with respect to each input feature. A high value of this measure means that a slight perturbation in inputs will result in a drastic change in output. For a classification problem, there are theoretical and empirical results on the connection of this measure and generalization (Novak et al., 2018; Sokolić et al., 2017). Also, a representation that is less sensitive to input perturbations with this measure has been shown to improve the performance of classifiers (Rifai et al., 2011). Following Novak et al. (2018), we evaluated this measure at the test points and reported the average. Figure 10 (Right) summarizes the results.

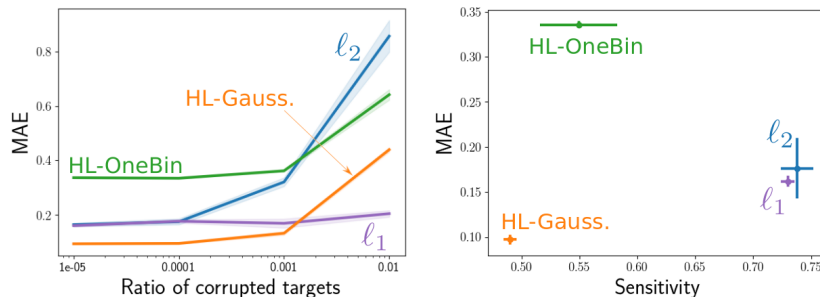


Figure 10: (Left) Corrupted targets on CT Scan. The lines show test errors. The  $\ell_1$  loss was robust to corrupted targets, and the  $\ell_2$  loss was affected the most. (Right) Sensitivity results on CT Scan. The horizontal axis shows the sensitivity of the model’s output to input perturbations (left means less sensitive), and the vertical axis shows the test error (lower is better). HL-Gauss showed less sensitivity and lower test errors than  $\ell_1$  and  $\ell_2$ .

The results indicate that predictions of the models that are trained with HL are less sensitive to input perturbations than the ones trained with  $\ell_1$  or  $\ell_2$ .

#### 4.10 Summary of the Empirical Study

In the empirical study, we observe that

1. HL-Gauss, even without per-dataset tuning, is a viable choice in regression and can often outperform the  $\ell_2$  loss across datasets (Section 4.5).
2. The softmax nonlinearity contributes to HL-Gauss’s performance, as shown by the results of  $\ell_2$ +Softmax, though it does not fully account for the improved performance under HL-Gauss. (Section 4.5).
3. The Gaussian target distribution reduces the discretization bias compared to HL-OneBin, and the remaining bias is bounded and negligible with proper  $\sigma$  and padding (Section 4.6).
4. The results are insensitive to the number of bins for  $k > 30$  and to  $\sigma$  within the interval  $[0.5, 4.0]$  times the bin width (Section 4.6).
5. A standard configuration of 100 bins with  $\sigma$  set to twice the bin width and padding of three times  $\sigma$  on each side performs well across all datasets without per-dataset tuning (Section 4.6).
6. Predicting the full distribution does not force better learned representations: HL-Gauss retains its advantages even with a fixed random representation (Section 4.7).
7. HL-Gauss produces more stable gradients and reduces training error faster than  $\ell_2$  (Section 4.8).
8. HL-Gauss is more robust than  $\ell_2$  to corrupted targets, though it is more sensitive to corrupted targets than  $\ell_1$  (Section 4.9).
9. HL-Gauss produces models whose outputs are less sensitive to input perturbations than both  $\ell_1$  and  $\ell_2$  (Section 4.9).

**Limitations of the study:** These conclusions are given the empirical set-up used in this work. The architecture, stepsize, and epochs was chosen based on  $\ell_2$  performance, and it is possible different conclusions could arise with per-loss hyperparameter tuning. The study also assumed knowledge of the range of the target; though this is typically a reasonable assumption, results may be different if the range had to be estimated (incrementally). Finally, applications of regression are broad, and the data sets in our empirical study and applications sections only cover a small set. Generalizing our findings to other applications such as depth estimation (Vasiljevic et al., 2019) or survival prediction (Haider et al., 2020) requires additional investigation. Similarly, emergence of new architectures and foundation models (Bommasani et al., 2021) calls for more extensive studies to ask whether our observed patterns extend to the training and fine-tuning of these models.

## 5. Applications

This section explores the applicability of HL-Gauss beyond the previous simple regression data sets. We evaluate several network architectures on time-series prediction tasks using both HL-Gauss and  $\ell_2$  loss. The network choices include linear models, long short-term memory networks (LSTMs, (Hochreiter and Schmidhuber, 1997b)), gated recurrent units

Architecture	Loss	ETTh1	ETTh2	ETTm1	ETTm2
Linear	$\ell_2$	$0.07105 \pm 0.001022$	$0.1624 \pm 0.002041$	$0.03402 \pm 0.001153$	$0.07743 \pm 0.0009791$
	HL-Gauss	$0.3826 \pm 0.003224$	$0.3389 \pm 0.003415$	$0.3243 \pm 0.00557$	$0.2569 \pm 0.00266$
LSTM	$\ell_2$	$0.775 \pm 0.03489$	$3.194 \pm 0.2182$	$0.3192 \pm 0.1191$	$0.5605 \pm 0.1366$
	HL-Gauss	<b><math>0.6575 \pm 0.04493</math></b>	<b><math>3.165 \pm 0.4313</math></b>	<b><math>0.2437 \pm 0.03809</math></b>	<b><math>0.3241 \pm 0.08616</math></b>
GRU	$\ell_2$	$0.5414 \pm 0.02092$	$1.208 \pm 0.3515$	$0.399 \pm 0.0299$	$0.3328 \pm 0.05124$
	HL-Gauss	$0.6733 \pm 0.0863$	<b><math>0.8228 \pm 0.4614</math></b>	<b><math>0.2468 \pm 0.02231</math></b>	<b><math>0.3325 \pm 0.1206</math></b>
Transformer	$\ell_2$	$0.533 \pm 0.02865$	$1.922 \pm 0.3305$	$0.2427 \pm 0.03834$	$0.2097 \pm 0.03613$
	HL-Gauss	$0.6389 \pm 0.1288$	$4.082 \pm 0.7923$	$0.2587 \pm 0.06254$	$0.3529 \pm 0.1232$

Table 4: Test MSE results on the ETD dataset. For each (architecture, loss, data set) tuple, the table shows the mean averaged over 5 independent seeds and 95% confidence interval. **Bold** text indicates better performance for HL-Gauss. HL-Gauss outperforms  $\ell_2$  for recurrent architectures (LSTM and GRU) but underperforms  $\ell_2$  for linear architecture, and mostly ties on the vanilla transformer.

(GRUs, (Chung et al., 2014)), and vanilla transformers (Vaswani et al., 2017). We then extend the experiments to state-value prediction for reinforcement learning, where we evaluate the performance of both losses on the Atari prediction task.

## 5.1 Time Series Prediction

We use the Electricity Transformer Dataset (ETD, (Zhou et al., 2021)) for the time series prediction experiments. The data set contains 6 features relating to the electricity loads, in addition to the oil temperature, which is the prediction target in all of our experiments. The data is collected from two different transformers and has two variants: hourly data sets, which were collected from each transformer at an hourly rate, and 15-minute data sets, where the data collection happened 4 times per hour. We consider the multi-step prediction task similar to previous work on this data set (Wu et al., 2021; Zeng et al., 2023), where given a history of length  $T$ , the goal is to predict the next  $k$  steps of the target (i.e., the oil temperature). In our experiment, we use  $T = k = 96$ .

We investigated the performance of four different network architectures typically used for time-series predictions. These architectures are LSTM (Hochreiter and Schmidhuber, 1997b), GRUs (Chung et al., 2014), Vanilla Transformer (Vaswani et al., 2017), and a simple linear model suggested by Zeng et al. (2023). For each architecture, we had two variations, one trained with HL-Gauss loss and one trained with  $\ell_2$  loss. For HL-Gauss, we divided the output range to 100 bins, set  $\sigma$  to twice this bin width, and extended the support with additional padding as described in Appendix E. The same appendix section provides more details on the hyperparameters and the experiment setup.

Table 4 shows the test MSE for all the architectures on the 4 data sets in ETD, and we show the test MAE results in Table 6 in Appendix E. The results suggest that the benefit of HL-Gauss for time-series prediction tasks is architecture-dependent. Recurrent architectures (LSTM and GRU) consistently benefit from HL-Gauss, achieving significantly lower test MSE on most datasets. In contrast, the linear architecture did not benefit from having HL-Gauss. One possible explanation for the architecture dependence is that, for

architectures with complex optimization dynamics such as LSTM and GRU, HL-Gauss helps stabilize the gradient dynamics (Section 3.1) and, as a result, improves their performance. On the other hand, simple models such as linear models are well-suited for direct  $\ell_2$  optimization and thus do not benefit from HL-Gauss.

## 5.2 State Value Prediction

A central problem in reinforcement learning (RL) is state-value prediction. An RL agent interacts with an environment modeled as a Markov decision process, and each action taken by the agent results in a transition, reward, and discounting or possible termination. The goal of state-value prediction is to predict the expected sum of discounted future rewards from a state.

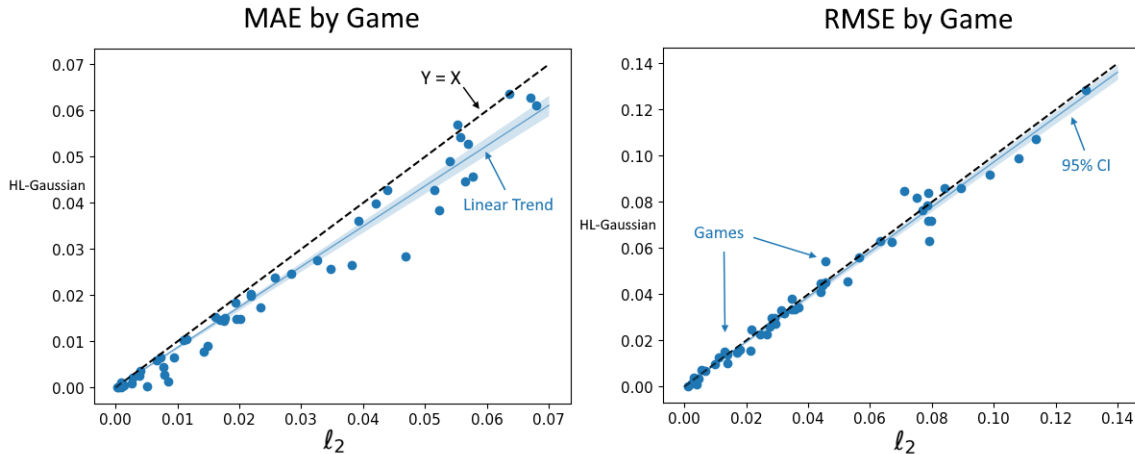


Figure 11: Atari results by game. Points below the dashed black line are games where HL-Gauss outperformed  $\ell_2$ . The estimated error ratio (blue line) is significant at 95% confidence for both metrics, indicating that HL-Gauss has a multiplicative advantage across games.

In our experiments, we predict state values for agents playing 58 different Atari 2600 games on the Arcade Learning Environment (Bellemare et al., 2013). In this setting, the state is observed through frames from the screen, and the reward is based on the game score obtained. We use actions from the Atari Prediction Benchmark from Javed et al. (2023). This benchmark contains at least 10 million actions for each game. The actions were produced by a pre-trained rainbow DQN agent from the ChainerRL model zoo (Fujita et al., 2021). While the benchmark paper explores the partially-observable setting with a single frame per observation, we use a stack of 4 frames to make it fully-observable. For HL-Gauss, the bin width is computed using Equation (15). Again, we divide the output range to 100 bins, set  $\sigma$  to twice this bin width, and then increase the bin width to add some padding as described in Appendix E. We use the first 5% of the episodes as the test set and train the models for one epoch on the remaining episodes. We exclude 3 games from our analysis in which the agent received no rewards (namely Tennis, Venture, and Montezuma Revenge).

Figure 11 shows the error of HL-Gauss against  $\ell_2$  across the 55 remaining games. We see that HL-Gauss results in lower errors for most games. To evaluate the statistical significance, we model the error ratio and test for a consistent multiplicative advantage across games. Performing a  $t$ -test on the log-transformed error ratios, we find moderate ( $p = 0.0294$ ) and strong evidence ( $p < 10^{-5}$ ) that HL-Gauss outperforms  $\ell_2$  across Atari games in Test RMSE and Test MAE, respectively.

## 6. Conclusion and Future Work

We analyzed a recent approach to regression, the histogram loss (HL) (Imani and White, 2018), which consists of the KL-divergence between a flexible histogram prediction and target distributions. Experiments on a variety of data sets and networks showed that this loss function often outperforms the oft-used  $\ell_2$  loss with a fixed set of recommended hyperparameter values and without the need for careful hyperparameter sweeps. The recommended rule of thumb is using 100 evenly sized bins, setting  $\sigma$  to twice the bin width, and then extending the range by three times  $\sigma$  on each side padding.

Besides providing a solution for regression, HL offers a testbed for analyzing some of the open questions in previous work. A question that we explored in depth was one of the hypotheses put forward by Bellemare et al. (2017): whether the performance gap between a model that learns the full distribution and one that only estimates the mean can be attributed to their representations, in a manner analogous to multi-task learning. Our experiments suggest that this is not the case in regression. Instead, the bound on the gradient norm of HL as well as its stability during training hint at a better-behaved loss surface as an important factor in effect. An additional factor in HL’s performance is the softmax nonlinearity that models the histogram distribution, as we found that minimizing the  $\ell_2$  loss between the target and the mean of a histogram distribution could yield an error rate close to that of HL.

Another question was the impact of the type of target distribution on performance. The method for constructing the target distribution has an effect on bias of the loss, and a Gaussian target distribution with a reasonable bandwidth can result in a lower bias than one-hot targets. However, it is also important to note that this is not the only benefit of a Gaussian target distribution, since even HL with an unbiased target distribution underperformed HL-Gauss.

Our results on the behavior of different models during optimization are still preliminary, and a more in-depth investigation is needed for a crisp conclusion. There is a growing literature on understanding loss surfaces of neural networks. Theoretical analyses are often insightful but require strong assumptions on the data or model (Soltanolkotabi et al., 2018; Choromanska et al., 2015; Jacot et al., 2018; Arora et al., 2018). An alternative line of research compares loss surfaces empirically by techniques like projection on lower dimensions or slightly perturbing the parameters in random directions, often to compare the flatness of the solutions (Hochreiter and Schmidhuber, 1997a; Keskar et al., 2017; Li et al., 2018). There are three important challenges in performing such comparison. First, intuitions about optimization in low-dimensional surfaces do not necessarily transfer to higher dimensions. Second, sometimes (like when comparing HL and  $\ell_2$  loss) the surfaces compared are of different dimensionalities, and this difference can be a confounding factor.

Third, a causal connection between the performance of the network and some measures of flatness is disputed, as solutions with similar performance but varying levels of flatness can be easily constructed (Dinh et al., 2017).

Despite the challenges above, investigating possible connections between performance gains in distribution learning and flatness is promising. Training regimes and optimizers that implicitly or explicitly enforce flatness have shown better final performance, an initial phase of fast learning, higher robustness to some degree of label noise, and less sensitivity to input perturbations (Keskar et al., 2017; Hoffer et al., 2017; Foret et al., 2021; Jiang et al., 2020; Novak et al., 2018). In our empirical study, we found HL-Gauss had these exact properties.

The experiment settings and baselines in this paper were focused on singling out and studying the points of difference between HL and the  $\ell_2$  loss. Further experiments can be designed to evaluate intersections of these factors. As a simple example, one could switch the  $\ell_2$  loss to the  $\ell_1$  loss in our  $\ell_2$ +softmax baseline. The new model can benefit from robustness to outliers and the nonlinearity in the softmax output layer.

In terms of applications, a future direction would be finding problem-specific or even adaptive discretization methods. Breaking down the target range to even bins might not be reasonable in some cases. For example, consider the task of predicting the number of views of an online video. The presence of viral videos that heavily stretch the range of targets in the data set will face a model designer with a trilemma: (1) choosing a wide support that covers the targets and finely discretizing it which results in a network with an enormous last layer, (2) lowering the number of bins and turning to a coarse discretization which loses information about small differences between the targets, or (3) truncating the support which requires drawing a line between informative data points and outliers and may also excessively limit the range of values the network can predict. An investigation into best practices or even automatic discretization methods could make HL easier to use in such problems.

## Acknowledgments

We would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chair Program, and Amii for research funding and the Digital Research Alliance of Canada for the computational resources. K.L. is supported by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA) through the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research. We also thank Nick Janßen for pointing out an issue in a previous version of the time-series experiments.

## Appendix A. Proofs

**Proposition 1 (Local Lipschitz constant for HL-Gauss)** *Assume  $\mathbf{x}, y$  are fixed, giving fixed coefficients  $c_i$  in HL-Gauss. Let  $h_i(\mathbf{x})$  be as in (2), defined by the parameters  $\mathbf{w} = \{\mathbf{w}_1, \dots, \mathbf{w}_k\}$  and  $\boldsymbol{\theta}$ , providing the predicted distribution  $h_{\mathbf{x}}$ . Assume for all  $i$  that  $\mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$  is locally  $l$ -Lipschitz continuous with respect to  $\boldsymbol{\theta}$ :*

$$\|\nabla_{\boldsymbol{\theta}}(\mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x}))\| \leq l. \quad (3)$$

Then the norm of the gradient for HL-Gauss, with respect to all the parameters in the network  $\{\boldsymbol{\theta}, \mathbf{w}\}$ , is bounded by

$$\|\nabla_{\boldsymbol{\theta}, \mathbf{w}} HL(q, h_{\mathbf{x}})\| \leq (l + \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\|) \sum_{i=1}^k |c_i - h_i(\mathbf{x})|. \quad (4)$$

**Proof** Let  $b_i = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathbf{w}_i$  and  $e_i = \exp(b_i)$ . Then, since  $h_j(\mathbf{x}) = \frac{e_j}{\sum_{l=1}^k e_l}$ , for  $j \neq i$

$$\frac{\partial}{\partial b_i} h_j(\mathbf{x}) = \frac{\partial}{\partial b_i} \frac{e_j}{\sum_{l=1}^k e_l} = -\frac{e_j}{\left(\sum_{l=1}^k e_l\right)^2} e_i = -h_j(\mathbf{x}) h_i(\mathbf{x})$$

For  $j = i$ , we get

$$\frac{\partial}{\partial b_i} h_j(\mathbf{x}) = \frac{e_i}{\sum_{l=1}^k e_l} - \frac{e_i}{\left(\sum_{l=1}^k e_l\right)^2} e_i = h_i(\mathbf{x}) [1 - h_i(\mathbf{x})]$$

Consider now the gradient of HL, with respect to  $b_i$

$$\begin{aligned} \frac{\partial}{\partial b_i} \sum_{j=1}^k c_j \log h_j(\mathbf{x}) &= \sum_{j=1}^k c_j \frac{1}{h_j(\mathbf{x})} h_j(\mathbf{x}) (1_{i=j} - h_i(\mathbf{x})) \\ &= \sum_{j=1}^k c_j (1_{i=j} - h_i(\mathbf{x})) = c_i - h_i(\mathbf{x}) \sum_{i=1}^k c_i = c_i - h_i(\mathbf{x}) \end{aligned}$$

Then

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}_i} \sum_{j=1}^k c_j \log h_i(\mathbf{x}) &= (c_i - h_i(\mathbf{x})) \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x}) \\ \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{j=1}^k c_j \log h_i(\mathbf{x}) &= \sum_{i=1}^k (c_i - h_i(\mathbf{x})) \nabla_{\mathbf{w}_i^\top} \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x}). \end{aligned}$$

The norm of the gradient of HL in Equation (1), with respect to  $\mathbf{w}$  which is composed of all the weights  $\mathbf{w}_i \in \mathbb{R}^k$  is

$$\left\| \frac{\partial}{\partial \mathbf{w}} \sum_{j=1}^k c_j \log h_j(\mathbf{x}) \right\| \leq \sum_{i=1}^k \left\| \frac{\partial}{\partial \mathbf{w}_i} \sum_{j=1}^k c_j \log h_j(\mathbf{x}) \right\|$$

$$= \sum_{i=1}^k \|(c_i - h_i(\mathbf{x}))\phi_{\theta}(\mathbf{x})\| \leq \sum_{i=1}^k |c_i - h_i(\mathbf{x})| \|\phi_{\theta}(\mathbf{x})\|$$

Similarly, the norm of the gradient with respect to  $\theta$  is

$$\begin{aligned} \left\| \frac{\partial}{\partial \theta} \sum_{j=1}^k c_j \log h_j(\mathbf{x}) \right\| &= \left\| \sum_{i=1}^k (c_i - h_i(\mathbf{x})) \nabla_{\theta} \mathbf{w}_i^{\top} \phi_{\theta}(\mathbf{x}) \right\| \\ &\leq \sum_{i=1}^k \left\| (c_i - h_i(\mathbf{x})) \nabla_{\theta} \mathbf{w}_i^{\top} \phi_{\theta}(\mathbf{x}) \right\| \leq \sum_{i=1}^k |c_i - h_i(\mathbf{x})| l \end{aligned}$$

Together, these bound the norm  $\|\nabla_{\theta, \mathbf{w}} HL(q, h_{\mathbf{x}})\|$ . ■

**Proposition 2 (Bias Characterization)** *The following statements hold:*

1. *If the form of  $h$  is not restricted, the minimizer of the KL-divergence is the result of smoothing  $p$  with  $q$ . Formally,  $\operatorname{argmin}_{h \in \mathcal{P}(\mathbb{R})} \mathbb{E}_{y \sim p}[D_{KL}(q_y || h)] = s$  where  $s(z) = \int p(y)q_y(z) dy$ .*
2. *The bias of  $s$  is the expected difference between the scalar target and the mean of the target distribution constructed from it. If the target distribution is Gaussian,  $s$  is unbiased. Formally,  $\operatorname{Bias}(s, p) = \mathbb{E}_{y \sim p}[\mathbb{E}_{z \sim q_y}[z] - y]$ .*
3. *If the form of  $h$  is restricted to a histogram with bins of equal widths, the minimizer of HL is  $h^*$ , where the probability in each bin is equal to the probability of  $s$  in the range of that bin. Formally,  $\operatorname{argmin}_{h \in \mathcal{P}(\mathbb{N})} \mathbb{E}_{y \sim p}[HL(q_y || h)] = h^*$  where  $h_i^* = \int_{l_i}^{l_i+w} s(z) dz$ .*
4. *If the target distribution is Gaussian, the bias of  $h^*$  is at most half of the bin width, and this bound is tight, i.e.  $|\operatorname{Bias}(h^*, p)| \leq w/2$ .*

**Proof** The integrals in this proof are over the whole real line unless stated otherwise. We use Fubini's theorem when we change the orders of integrals.

The function  $s$  in Statement 1 is a proper pdf since it is positive at each point and sums to one:

$$\int s(z) dz = \int \int p(y)q_y(z) dy dz = \int p(y) \left( \int q_y(z) dz \right) dy = \int p(y) dy = 1$$

Recall that HL is the cross-entropy between the predicted distribution and the target distribution. We denote the cross-entropy between  $q_y$  and  $h$  as  $H(q_y, h)$ . We now find the expected value of the cross-entropy under  $p$ , and the prediction distribution that minimizes it:

$$\begin{aligned} \mathbb{E}_{y \sim p}[H(q_y, h)] &= \int p(y) \int q_y(z) \log h(z) dz dy = \int \left( \int p(y)q_y(z) dy \right) \log h(z) dz \\ &= \int s(z) \log h(z) dz = H(s, h) \end{aligned}$$

So, as long as  $s$  is in our function class for the prediction distribution, the prediction distribution that minimizes the average cross-entropy is not  $p$ , but  $s$  (**Statement 1**). Formally

$$\operatorname{argmin}_{h \in \mathcal{P}(\mathbb{R})} \mathbb{E}_{y \sim p} [D_{KL}(q_y || h)] = \operatorname{argmin}_{h \in \mathcal{P}(\mathbb{R})} \mathbb{E}_{y \sim p} [H(q_y, h)] = \operatorname{argmin}_{h \in \mathcal{P}(\mathbb{R})} H(s, h) = s$$

In the case of the Dirac delta target distribution,  $s$  is the same as  $p$  and for a Gaussian target distribution,  $s$  is the result of Gaussian kernel smoothing on  $p$ .

If  $s$  is within our function class, although a Gaussian target distribution will change the minimizer from  $p$  to  $s$ , it will not bias the predicted mean, as we show below. Suppose  $g(\cdot | \mu, \sigma^2)$  is the pdf of a Gaussian distribution with mean  $\mu$  and variance  $\sigma^2$ . Let us write the bias of  $s$ :

$$\begin{aligned} \mathbb{E}_s[z] - \mathbb{E}_p[y] &= \int z s(z) dz - \mathbb{E}_p[y] = \int z \int p(y) q_y(z) dy dz - \mathbb{E}_p[y] \\ &= \int p(y) \int z q_y(z) dz dy - \mathbb{E}_p[y] = \mathbb{E}_{y \sim p} [\mathbb{E}_{z \sim q_y} [z] - y] \\ &= \int p(y) \int z g(z | y, \sigma^2) dz dy - \mathbb{E}_p[y] = \int p(y) y dy - \mathbb{E}_p[y] = 0 \end{aligned}$$

The second line above proves **Statement 2** and the third line shows the claim for Gaussian target distribution.

The prediction function  $h$  is restricted to a histogram in HL-Gauss. This restriction can bias the predictions in at least two ways. The first source of bias is due to the fact that  $s$  is truncated to match the support of  $h$ . We do not explore truncation here, and assume that the support of  $h$  is  $(-\infty, \infty)$ . The second source of bias is a result of using discrete bins for prediction which is present even in the case of infinite support. We call this bias *discretization bias* and quantify it.

We consider the case where all bins have equal width  $w$ . The center of bin  $i$  and the probability of a histogram distribution  $h$  in bin  $i$  are denoted by  $m_i$  and  $h_i$  respectively and  $\mathcal{S}_i := [m_i - \frac{w}{2}, m_i + \frac{w}{2})$  is the range of bin  $i$ .

Note that in this case the cross-entropy between  $h$  and  $s$  is  $\sum_i (\int_{\mathcal{S}_i} s(z) dz) \log(h_i)$ . Thus the histogram density that minimizes the cross-entropy to  $s$  is one where the probability in each bin is equal to the probability of  $s$  in the range of that bin (**Statement 3**). We call this density  $h^*$  and find the difference between the expected values of  $h^*$  and  $p$ .

$$\begin{aligned} \mathbb{E}_{h^*}[z] - \mathbb{E}_p[y] &= \sum_i m_i \int_{\mathcal{S}_i} s(z) dz - \int y p(y) dy \\ &= \sum_i m_i \int_{\mathcal{S}_i} \int p(y) q_y(z) dy dz - \int y p(y) dy \\ &= \int p(y) \sum_i m_i \int_{\mathcal{S}_i} q_y(z) dz dy - \int y p(y) dy \\ &= \int p(y) \left( \sum_i m_i \int_{\mathcal{S}_i} q_y(z) dz - y \right) dy \end{aligned}$$

The factor inside the parantheses is the difference between  $y$  and the expected value of the histogram density with the lowest KL-divergence from  $q_y$ . We will find this difference for the case of a Gaussian target distribution. This error depends on  $y$  and, once this error is known, the highest discretization bias can be found by choosing a distribution  $p$  that has all its probability on the value of  $y$  that results in the highest error. Also note that, if  $q_y$  is chosen so that the closest histogram density to it has the expected value of  $y$ , this error is zero and the whole discretization bias becomes zero. The proof below shows that discretization bias of HL-Gauss is bounded by  $\pm \frac{w}{2}$  and depends on the choice of the parameter  $\sigma$ .

We will denote with  $m_0$  the center of the bin that contains  $y$ . Other bin centers will be denoted by  $m_i$  where  $m_i := m_0 + iw$ . Also,  $\delta := (y - m_0)$ . Note that  $\delta \in [-\frac{w}{2}, \frac{w}{2})$ . Finally, CDF of a Gaussian with mean  $\mu$  and variance  $\sigma^2$  is denoted by  $\Phi_\mu$  and  $F_\mu(a, b) := \Phi_\mu(b) - \Phi_\mu(a)$ . (Subscripts to show the dependence of  $\Phi$  and  $F$  on  $\sigma$  are dropped for convenience.)

$$\begin{aligned} \sum_{i=-\infty}^{\infty} m_i \int_{\mathcal{S}_i} g(z|y, \sigma^2) dz - y &= \sum_{i=-\infty}^{\infty} (m_0 + iw) \int_{\mathcal{S}_i} g(z|y, \sigma^2) dz - y \\ &= m_0 \sum_{i=-\infty}^{\infty} \int_{\mathcal{S}_i} g(z|y, \sigma^2) dz + w \sum_{i=-\infty}^{\infty} i \int_{\mathcal{S}_i} g(z|y, \sigma^2) dz - y \end{aligned} \quad (7)$$

Recall that intervals  $\mathcal{S}_i$  are disjoint and  $\bigcup_{i=-\infty}^{\infty} \mathcal{S}_i = (-\infty, \infty)$ . So the series in the first term of (7) becomes one. Therefore

$$\begin{aligned} (7) &= (m_0 - y) + w \left( \sum_{i=-\infty}^{-1} i \int_{\mathcal{S}_i} g(z|y, \sigma^2) dz + \sum_{i=1}^{\infty} i \int_{\mathcal{S}_i} g(z|y, \sigma^2) dz \right) \\ &= -\delta + w \left( \sum_{i=-\infty}^{-1} i F_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2}) \right. \\ &\quad \left. + \sum_{i=1}^{\infty} i F_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2}) \right) \\ &= -\delta + w \left( \sum_{i=1}^{\infty} (-i) F_y(m_0 - iw - \frac{w}{2}, m_0 - iw + \frac{w}{2}) \right. \\ &\quad \left. + \sum_{i=1}^{\infty} i F_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2}) \right) \\ &= -\delta + w \left( \sum_{i=1}^{\infty} i (F_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2}) \right. \\ &\quad \left. - F_y(m_0 - iw - \frac{w}{2}, m_0 - iw + \frac{w}{2})) \right) \end{aligned} \quad (8)$$

Due to the symmetry of Gaussian distribution,  $F_\mu(a, b) = F_\mu(2\mu - b, 2\mu - a)$ . We define  $a_i := iw - \frac{w}{2}$  and  $b_i := iw + \frac{w}{2}$  and the series in (8) becomes

$$\begin{aligned}
 & \sum_{i=1}^{\infty} i(F_y(m_0 + a_i, m_0 + b_i) - F_y(m_0 - b_i, m_0 - a_i)) \\
 &= \sum_{i=1}^{\infty} i(F_y(m_0 + a_i, m_0 + b_i) - F_y(2y - m_0 + a_i, 2y - m_0 + b_i)) \\
 &\stackrel{\delta := y - m_0}{=} \sum_{i=1}^{\infty} i(F_y(y - \delta + a_i, y - \delta + b_i) - F_y(y + \delta + a_i, y + \delta + b_i)) \\
 &= \sum_{i=1}^{\infty} i(\Phi_y(y - \delta + b_i) - \Phi_y(y - \delta + a_i) + \Phi_y(y + \delta + a_i) - \Phi_y(y + \delta + b_i)) \\
 &= \sum_{i=1}^{\infty} i(-F_y(y + b_i - \delta, y + b_i + \delta) + F_y(y + a_i - \delta, y + a_i + \delta)) \\
 &= - \sum_{i=1}^{\infty} iF_y(y + b_i - \delta, y + b_i + \delta) + \sum_{i=1}^{\infty} iF_y(y + a_i - \delta, y + a_i + \delta) \tag{9}
 \end{aligned}$$

Since  $iw + \frac{w}{2} = (i + 1)w - \frac{w}{2}$ , we can replace  $b_i$  by  $a_{i+1}$  and have

$$\begin{aligned}
 (9) &= - \sum_{i=1}^{\infty} iF_y(y + a_{i+1} - \delta, y + a_{i+1} + \delta) + \sum_{i=1}^{\infty} iF_y(y + a_i - \delta, y + a_i + \delta) \\
 &= - \sum_{i=2}^{\infty} (i - 1)F_y(y + a_i - \delta, y + a_i + \delta) + \sum_{i=1}^{\infty} iF_y(y + a_i - \delta, y + a_i + \delta) \\
 &= \sum_{i=2}^{\infty} F_y(y + a_i - \delta, y + a_i + \delta) + F_y(y + a_1 - \delta, y + a_1 + \delta) \tag{10}
 \end{aligned}$$

$$\begin{aligned}
 &= \sum_{i=1}^{\infty} F_y(y + a_i - \delta, y + a_i + \delta) = \sum_{i=1}^{\infty} F_0(a_i - \delta, a_i + \delta) \\
 &= \sum_{i=1}^{\infty} F_0(iw - \frac{w}{2} - \delta, iw - \frac{w}{2} + \delta) \tag{11}
 \end{aligned}$$

where we used the method of differences to obtain (10). We can now plug (11) into (8) to have

$$(8) = -\delta + w \left( \sum_{i=1}^{\infty} F_0(iw - \frac{w}{2} - \delta, iw - \frac{w}{2} + \delta) \right) \tag{12}$$

The sum in (12) consists of the area under a Gaussian pdf in some intervals. Since  $\delta \in [-\frac{w}{2}, \frac{w}{2}]$  the sum in the second term can be at most 0.5 (when  $\delta = \frac{w}{2}$  and the intervals connect to each other to form the positive half of a Gaussian pdf) and at least  $-0.5$  (when  $\delta = -\frac{w}{2}$  and the interval bounds are reversed). So, given the bounds on  $\delta$  and the sum, we can see the overall error cannot exceed  $[-w, w]$ .

However, as long as  $\delta > 0$ , each term in the sum is positive and the sum is positive (because it is a part of a Gaussian pdf) and, when  $\delta < 0$  the terms in the sum become negative (because the integral bounds are flipped). So the two terms in the sum will have opposite signs and will not add to each other, and the error is bounded by  $[-\frac{w}{2}, \frac{w}{2}]$ . This is a tight bound since for example when  $\sigma \rightarrow 0$  (and the loss becomes HL-OneBin) and  $\delta = \pm \frac{w}{2}$  then  $h^*$  will have all its density on the bin that contains  $y$  and the error is  $m_0 - y = -\delta = \mp \frac{w}{2}$  (**Statement 4**). ■

**Proposition 3 (Bound on Prediction Error)** *Assuming that for a data point, the target distribution  $q_y$  and the model’s prediction distribution  $h_{\mathbf{x}}$  have supports bounded by the range  $[a, b]$ , the following bound holds:*

$$(\mathbb{E}_{q_y}[z] - \mathbb{E}_{h_{\mathbf{x}}}[z])^2 \leq 4 \max(|a|, |b|)^2 \min\left(\frac{1}{2}D_{KL}(q_y||h_{\mathbf{x}}), 1 - e^{-D_{KL}(q_y||h_{\mathbf{x}})}\right). \quad (6)$$

**Proof** We drop the subscript  $y$  for brevity.

$$(\mathbb{E}_q[z] - \mathbb{E}_{h_{\mathbf{x}}}[z])^2 = \left(\int_a^b (q(z) - h_{\mathbf{x}}(z))z \, dz\right)^2 \leq \left(\int_a^b |q(z) - h_{\mathbf{x}}(z)| \, dz \max(|a|, |b|)\right)^2 \quad (13)$$

The inequality is tight when  $a = -1, b = 1, q = \delta(-1), h_{\mathbf{x}} = \delta(1)$ . The integral on the right-hand side is twice the Total-Variation (TV) distance (Zhang, 2023, p. 437). Pinsker’s inequality and Bretagnolle–Huber inequality relate TV distance and KL-divergence in these two ways:

$$\frac{1}{2} \int_a^b |q(z) - h_{\mathbf{x}}(z)| \, dz \leq \sqrt{\frac{1}{2}D_{KL}(q||h_{\mathbf{x}})}, \quad \frac{1}{2} \int_a^b |q(z) - h_{\mathbf{x}}(z)| \, dz \leq \sqrt{1 - e^{-D_{KL}(q||h_{\mathbf{x}})}}$$

(Lattimore and Szepesvári, 2020, p. 192) and Canonne (2022) show proofs for these bounds and discuss when each bound is preferred. Using these bounds we have

$$\begin{aligned} (13) &= 4 \left(\frac{1}{2} \int_a^b |q(z) - h_{\mathbf{x}}(z)| \, dz\right)^2 (\max(|a|, |b|))^2 \\ &\leq 4 \max(|a|, |b|)^2 \min\left(\frac{1}{2}D_{KL}(q||h_{\mathbf{x}}), 1 - e^{-D_{KL}(q||h_{\mathbf{x}})}\right) \end{aligned}$$
■

## Appendix B. Prediction Bounds

The two quantities in the minimum are obtained using Pinsker’s inequality and Bretagnolle–Huber (BH) inequality and yield tighter bounds for small and large values of KL-divergence respectively (Lattimore and Szepesvári, 2020; Canonne, 2022). To better understand these terms and the overall bound, consider the example where  $a = -1, b = 1, q_y =$

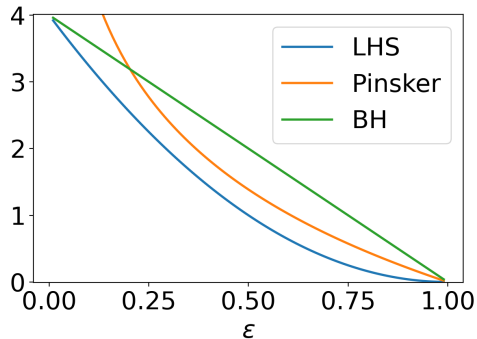


Figure 12: The behavior of the bound in Equation (6) on an example with Dirac delta functions. LHS shows the left-hand side, and the other two curves show the bounds obtained with Pinsker’s and BH inequality.

$\delta(-1), h_{\mathbf{x}} = \epsilon\delta(-1) + (1 - \epsilon)\delta(1)$  for  $\epsilon \in (0, 1)$ . As  $\epsilon$  rises, the two distributions become more similar to each other and both the squared error between their means (the left-hand side in Equation (6)) and the KL-divergence between the distributions decreases. Figure 12 shows the bounds obtained with Pinsker’s inequality and BH inequality (the first and the second term in the minimum) as well as the left-hand side. Both bounds decrease to zero as the two distributions approach each other. Pinsker’s inequality is preferred when the distributions are closer to each other. For small values of  $\epsilon$ , the bound using Pinsker’s inequality exceeds a trivial bound of  $(b - a)^2 = 4$ , while the bound with BH inequality never exceeds this value.

### Appendix C. Bias Simulation

When creating a discrete truncated Gaussian from a target  $y$ , bias is introduced as the mean of this distribution is typically not equal to  $y$ . The true mean of the histogram distribution,  $\hat{y}$ , can be found using the following equation:

$$\hat{y} = \sum_{i=1}^k c_i \left( \frac{\operatorname{erf}\left(\frac{c_i + \frac{w}{2} - y}{\sqrt{2}\sigma}\right) - \operatorname{erf}\left(\frac{c_i - \frac{w}{2} - y}{\sqrt{2}\sigma}\right)}{\operatorname{erf}\left(\frac{b - y}{\sqrt{2}\sigma}\right) - \operatorname{erf}\left(\frac{a - y}{\sqrt{2}\sigma}\right)} \right) \quad (14)$$

where the truncated normal distribution is contained within the interval  $(a, b)$ ,  $k$  is the number of bins,  $c_i$  is the value of the center of the  $i$ -th bin,  $w$  is the width of all bins and  $\sigma$  is the smoothness parameter of the truncated Gaussian.

We used simulation to approximate the bias as a function of  $\sigma$  and padding. We did this by generating simulated data, transforming it to a discretized truncated Gaussian distribution, and comparing the mean of this distribution to the original value. We first attempted to isolate the discretization bias by computing the total bias and using large padding, since the truncation bias vanishes as the support for the distribution grows to  $(-\infty, \infty)$ . By generating  $10^5$  equally spaced data points in the range  $[0, 1]$  divided into  $k = 100$  bins of equal width and varying  $\sigma$  we can plot the discretization bias as a function

of  $\sigma_w$ , defined as  $\sigma/(1/k)$ , and the relative position within a bin. We used padding of  $\psi = 100$ . Define  $\psi_\sigma$  as  $\psi/\sigma$ . The bin width is then adjusted as in the following equation:

$$w = \frac{y_{max} - y_{min}}{k - 2\sigma_w\psi_\sigma} \quad (15)$$

Figure 13 (left) displays the results. We define  $\delta$  as the difference between the sample and the center of the bin and the offset as  $\frac{\delta}{w}$ . The X-axis ( $\sigma_w$ ) has been log-transformed. As expected by the symmetry of the Gaussian distribution, we have zero bias for the center of the bin. The relationship with the bin offset converges to linear as  $\sigma_w \rightarrow 0$ , which corresponds to convergence towards HL-Onebin. The error also decreases to zero as  $\sigma_w \rightarrow \infty$ . Taking the mean absolute bias over all samples, we obtain Figure 13 (middle). The bias reaches a minimum at around  $\sigma_w \approx 1.35$ , where it becomes dominated by 64-bit floating point precision error.

We can also observe the total error for different values of  $\psi_\sigma$  in Figure 13 (right). Fixing  $\sigma_w = 2$ , the discretization error is negligible. Again, we see squared-exponential decay as the padding increases until  $\psi_\sigma = 8.5$ , after which the bias is dominated by floating point precision error. Overall, for  $\sigma_w > 0.8$  and  $\psi_\sigma > 6$ , we have bias less than  $10^{-7}$  which is negligible when using 32-bit floating point numbers. Empirically, the effect of these parameters on model learning has a greater impact on performance, but these graphs can be used to approximate the bias after scaling by the bin width.

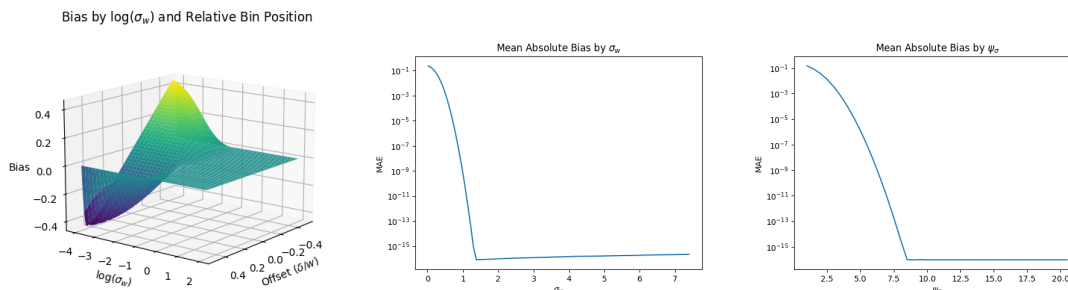


Figure 13: Bias simulation results. (Left) Discretization bias, (Middle) Mean Absolute Discretization Bias, (Right) Mean Absolute Truncation Bias.

## Appendix D. Empirical Study - Details

### D.1 Data Set details

Table 5 and Figure 14 show the details of the four regression data sets.

### D.2 Architectures and Hyperparameters

The architecture for CT Position is 385-192-192-192-192-1, for Song Year is 90-45-45-45-45-1 (4 hidden layers of size 45), for Bike Sharing is 16-64-64-64-64-1, and for Pole is 49-24-24-24-1. All hidden layers employ ReLU activation. Network architectures were chosen according to best Test MAE for  $\ell_2$ , with depth and width varied across 7 different values.

Data set	# train	# test	# feats	Y range
Song Year	412276	103069	90	[1922,2011]
CT Position	42800	10700	385	[0,100]
Bike Sharing	13911	3478	16	[0,1000]
Pole	12000	3000	49	[0,100]

Table 5: Overview of the data sets used in the experiments.

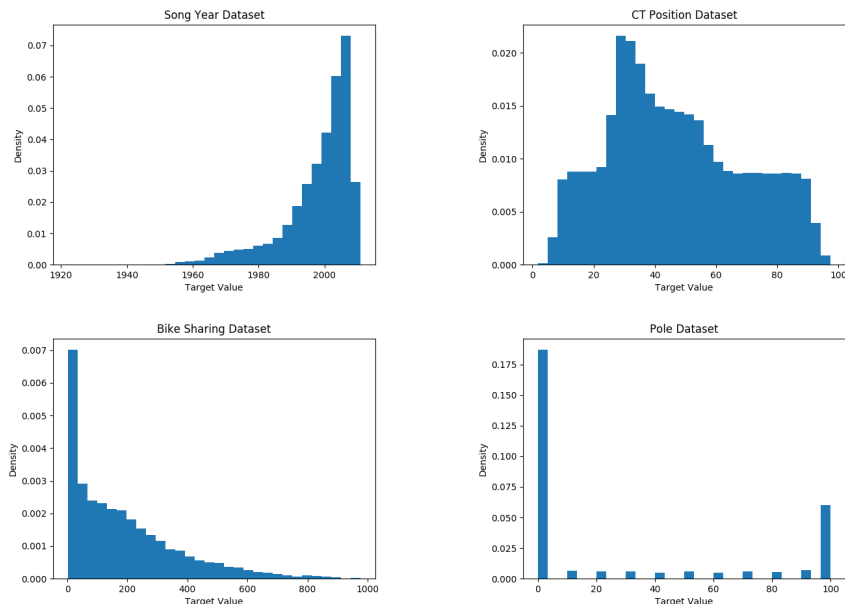


Figure 14: Histograms of the target values for the four data sets. Each plot shows a histogram with 30 bins, where the area of a bin is the ratio of targets in the data set that fall in that bin. Note that it is the total area of bins, and not the sum of heights, that equals one.

Some of the baselines required extra hyperparameters. The range of these hyperparameters are described below:

**$\ell_2$ +Noise** The standard deviation of the noise is selected from  $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ .

**$\ell_2$ +Clipping** The threshold for clipping is selected from  $\{0.01, 0.1, 1, 10\}$ .

**HL-Uniform**  $\epsilon \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ .

**MDN** The originally proposed model uses an exponential activation to model the standard deviations. However, inspired by Lakshminarayanan et al. (2017), we used softplus activation plus a small constant ( $10^{-2}$ ) to avoid numerical instability. We selected the number of components from  $\{2, 4, 8, 16, 32\}$ . Predictions are made by taking the mean of the mixture model given by the MDN.

Architecture	Loss	ETTh1	ETTh2	ETTm1	ETTm2
Linear	$\ell_2$	$0.1988 \pm 0.001382$	$0.3162 \pm 0.002306$	$0.1365 \pm 0.002068$	$0.2063 \pm 0.001591$
	HL-Gauss	$0.5548 \pm 0.003119$	$0.4779 \pm 0.003742$	$0.5303 \pm 0.006077$	$0.4249 \pm 0.003523$
LSTM	$\ell_2$	$0.7496 \pm 0.02461$	$1.662 \pm 0.06509$	$0.5056 \pm 0.1031$	$0.6145 \pm 0.08133$
	HL-Gauss	$0.7604 \pm 0.03374$	$1.664 \pm 0.1179$	<b><math>0.4009 \pm 0.06353</math></b>	<b><math>0.505 \pm 0.1101</math></b>
GRU	$\ell_2$	$0.6165 \pm 0.01295$	$0.9432 \pm 0.1581$	$0.5491 \pm 0.02453$	$0.4629 \pm 0.04164$
	HL-Gauss	$0.7948 \pm 0.05213$	<b><math>0.7682 \pm 0.2456</math></b>	<b><math>0.452 \pm 0.02874</math></b>	<b><math>0.4432 \pm 0.09381</math></b>
Transformer	$\ell_2$	$0.6804 \pm 0.01806$	$1.257 \pm 0.1399$	$0.4316 \pm 0.03694$	$0.3675 \pm 0.03441$
	HL-Gauss	$0.7532 \pm 0.08686$	$1.921 \pm 0.1787$	$0.4743 \pm 0.08024$	$0.5415 \pm 0.1324$

Table 6: Test MAE results on the ETD dataset. For each (architecture, loss, dataset) tuple, the table shows the mean averaged over 5 independent seeds and 95% confidence interval. The **bold** indicates better performance for HL-Gauss. HL-Gauss outperforms  $\ell_2$  for recurrent architectures (LSTM and GRU) but underperforms  $\ell_2$  for linear, and mostly ties on the vanilla transformer.

We used Scikit-learn (Pedregosa et al., 2011) for the implementations of Linear Regression, and Keras (Chollet et al., 2015) for the neural network models. All neural network models are trained with mini-batch size 256 using the Adam optimizer (Kingma and Ba, 2015) with a learning rate 1e-3 and the parameters are initialized according to the method suggested by LeCun et al. (1998). Dropout (Srivastava et al., 2014) with rate 0.05 is added to the input layer of all neural networks for CT Scan and Song Year tasks to avoid overfitting. We trained the networks for 1000 epochs on CT Position, 150 epochs on Song Year and 500 epochs on Bike Sharing and Pole. Dropout is disabled in the experiments in Section 4.8.

## Appendix E. Application - Details

### E.1 Time Series Prediction

In this section, we present the architectural details and hyperparameters used in the time-series prediction experiments.

The linear model extracts the target channel (oil temperature) and applies a single fully connected layer mapping the input sequence length to the prediction length, following the channel-independent design in Zeng et al. (2023). The LSTM and GRU models each stacks 2 recurrent layers, each with 256 hidden units, dropout of 0.1, and layer normalization. The recurrent block is then followed by a linear projection mapping to the prediction length. The Transformer uses 3 encoder blocks with model dimension 128, 8 attention heads, and a feed-forward dimension of 512, followed by the same projection layer. For each architecture, the base model generates a feature vector that is then either passed to a regression head or to HL-Gauss head.

We use the standard 12,4,4 month split (30 days per month) for training, validation, and test, following the convention in Informer (Zhou et al., 2021) and LTSF-Linear (Zeng et al., 2023). The input sequence length is 96 and the prediction horizon is 96 steps. The prediction task is univariate: all 7 input channels are used as features, and the target is the oil temperature (OT) column. All features are standardized to zero mean and unit variance

using statistics computed from the training split. We train all networks with a batch size of 32 using the AdamW optimizer with a fixed learning rate of  $10^{-4}$  and a weight decay  $10^{-4}$ . We set the maximum number of training epochs to 10 with early stopping based on the validation loss with patience 3, restoring the best weights.

For the HL-Gauss model, the bin width was determined as in Equation (15) where  $\sigma_w$  is the ratio of the  $\sigma$  parameter for the truncated Gaussian distribution to the bin width, and  $\psi_\sigma$  is the ratio of padding to bin width added to each side. We used  $k = 100$  bins with  $\sigma_w = 2$  and  $\psi_\sigma = 3$ . The bin widths are calculated using the minimum and maximum observation over the training dataset.

## E.2 Value Prediction

We first precomputed the values for each game. Then, we trained on the first 95% of the game iterations in the actions file and tested on the remaining 5% of the iterations. The samples were shuffled with a buffer size of 1000 within the training and testing sets. We scaled the targets to be in the range  $[0, 1]$ . The model backbone consists of 3 convolutional layers followed by a fully-connected layer. We used an ADAM optimizer with parameters  $\alpha = 10^{-3}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 10^{-7}$ . All models were implemented using Keras (Chollet et al. (2015)).

For HL-Gauss We used  $k = 100$  bins with  $\sigma_w = 2$  and  $\psi_\sigma = 4$  in Equation (15). The models were trained for 1-3 epochs, depending on the game.

## Appendix F. Other Results

This section provides the rest of the empirical results.

### F.1 Overall Results

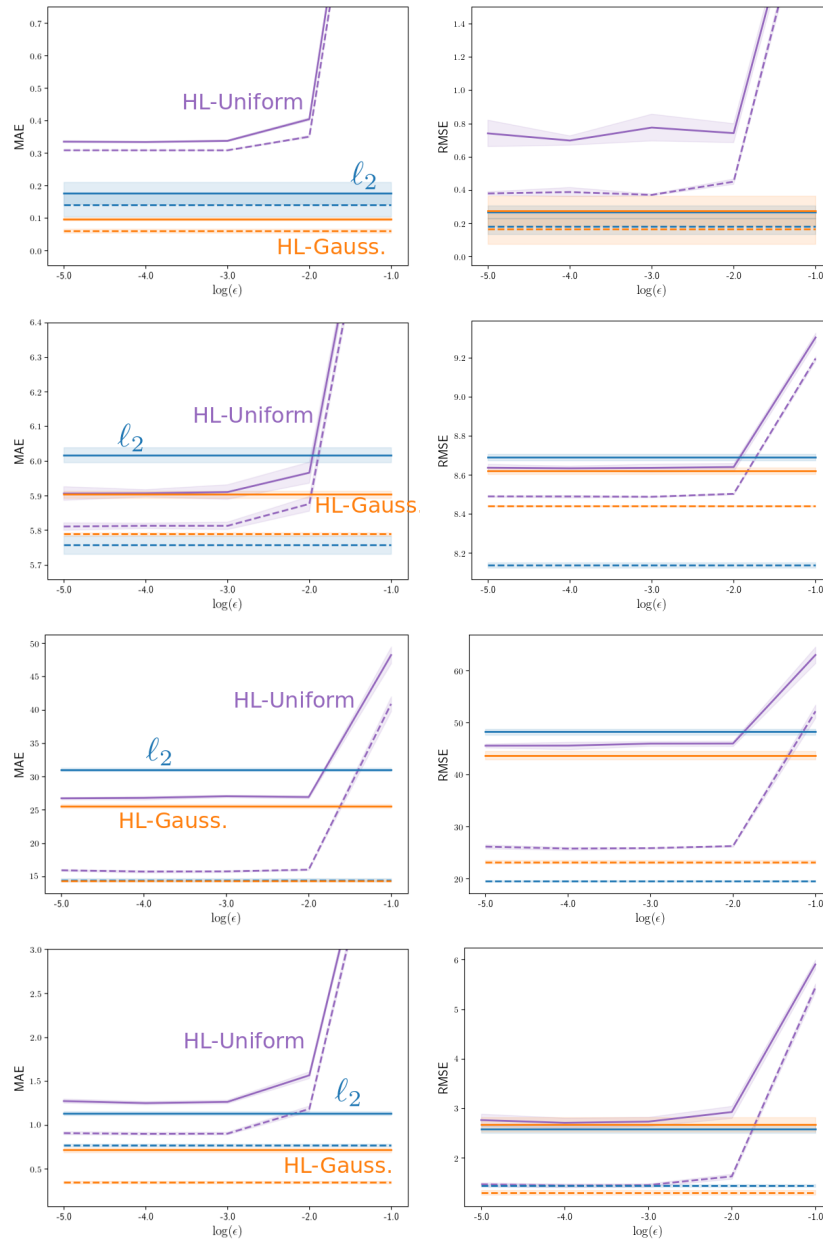


Figure 15: HL-Uniform results. The rows show CT Scan, Song Year, Bike, and Pole respectively. Dotted and solid lines show train and test errors respectively. The parameter  $\epsilon$  is the weighting on the uniform distribution and raising it only impaired performance.

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lin Reg	6.793( $\pm 0.003$ )	9.547( $\pm 0.003$ )	6.796( $\pm 0.007$ )	9.555( $\pm 0.014$ )
$\ell_2$	5.758( $\pm 0.026$ )	8.137( $\pm 0.014$ )	6.016( $\pm 0.022$ )	8.690( $\pm 0.015$ )
$\ell_2$ +Noise	5.697( $\pm 0.023$ )	8.124( $\pm 0.006$ )	5.959( $\pm 0.015$ )	8.682( $\pm 0.022$ )
$\ell_2$ +Clip	5.749( $\pm 0.032$ )	8.130( $\pm 0.012$ )	6.009( $\pm 0.032$ )	8.687( $\pm 0.015$ )
$\ell_1$	5.423( $\pm 0.010$ )	8.640( $\pm 0.016$ )	5.673( $\pm 0.006$ )	8.987( $\pm 0.025$ )
MDN	5.858( $\pm 0.020$ )	8.530( $\pm 0.008$ )	5.935( $\pm 0.025$ )	8.640( $\pm 0.020$ )
$\ell_2$ +Softmax	5.651( $\pm 0.015$ )	8.067( $\pm 0.009$ )	5.947( $\pm 0.012$ )	8.685( $\pm 0.012$ )
$\ell_2$ +Mean	5.863( $\pm 0.001$ )	8.434( $\pm 0.004$ )	5.995( $\pm 0.027$ )	8.724( $\pm 0.022$ )
HL-OneBin	5.810( $\pm 0.010$ )	8.487( $\pm 0.002$ )	5.906( $\pm 0.020$ )	8.636( $\pm 0.020$ )
HL-Gauss.	5.789( $\pm 0.004$ )	8.440( $\pm 0.004$ )	5.903( $\pm 0.010$ )	8.621( $\pm 0.016$ )

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lin Reg	106.047( $\pm 0.215$ )	141.854( $\pm 0.192$ )	105.788( $\pm 0.738$ )	141.617( $\pm 0.757$ )
$\ell_2$	14.453( $\pm 0.258$ )	19.487( $\pm 0.116$ )	31.029( $\pm 0.258$ )	48.205( $\pm 0.545$ )
$\ell_2$ +Noise	20.577( $\pm 0.270$ )	28.604( $\pm 0.325$ )	28.486( $\pm 0.278$ )	44.777( $\pm 0.747$ )
$\ell_2$ +Clip	13.221( $\pm 0.676$ )	18.011( $\pm 0.783$ )	29.404( $\pm 0.188$ )	46.309( $\pm 0.403$ )
$\ell_1$	12.764( $\pm 0.250$ )	21.456( $\pm 0.356$ )	27.550( $\pm 0.258$ )	44.717( $\pm 0.674$ )
MDN	15.406( $\pm 0.192$ )	27.593( $\pm 0.348$ )	26.268( $\pm 0.342$ )	43.679( $\pm 0.605$ )
$\ell_2$ +Softmax	11.991( $\pm 0.277$ )	16.799( $\pm 0.445$ )	27.827( $\pm 0.253$ )	45.825( $\pm 0.431$ )
$\ell_2$ +Mean	14.522( $\pm 0.353$ )	20.499( $\pm 0.518$ )	29.474( $\pm 0.527$ )	47.651( $\pm 0.990$ )
HL-OneBin	15.822( $\pm 0.198$ )	26.065( $\pm 0.335$ )	26.689( $\pm 0.280$ )	45.150( $\pm 0.857$ )
HL-Gauss.	14.335( $\pm 0.152$ )	23.178( $\pm 0.309$ )	25.525( $\pm 0.331$ )	43.671( $\pm 0.796$ )

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lin Reg	26.523( $\pm 0.018$ )	30.424( $\pm 0.010$ )	26.662( $\pm 0.066$ )	30.567( $\pm 0.038$ )
$\ell_2$	0.767( $\pm 0.016$ )	1.441( $\pm 0.026$ )	1.131( $\pm 0.024$ )	2.579( $\pm 0.073$ )
$\ell_2$ +Noise	0.700( $\pm 0.031$ )	1.406( $\pm 0.039$ )	1.062( $\pm 0.027$ )	2.529( $\pm 0.054$ )
$\ell_2$ +Clip	0.714( $\pm 0.018$ )	1.424( $\pm 0.024$ )	1.069( $\pm 0.017$ )	2.551( $\pm 0.103$ )
$\ell_1$	0.637( $\pm 0.013$ )	1.661( $\pm 0.029$ )	0.938( $\pm 0.029$ )	2.584( $\pm 0.034$ )
MDN	0.648( $\pm 0.026$ )	1.712( $\pm 0.078$ )	0.895( $\pm 0.013$ )	2.480( $\pm 0.095$ )
$\ell_2$ +Softmax	0.634( $\pm 0.017$ )	1.445( $\pm 0.019$ )	0.950( $\pm 0.016$ )	2.481( $\pm 0.055$ )
$\ell_2$ +Mean	0.613( $\pm 0.008$ )	1.452( $\pm 0.010$ )	0.986( $\pm 0.010$ )	2.773( $\pm 0.056$ )
HL-OneBin	0.899( $\pm 0.015$ )	1.442( $\pm 0.033$ )	1.264( $\pm 0.019$ )	2.760( $\pm 0.116$ )
HL-Gauss.	0.347( $\pm 0.018$ )	1.299( $\pm 0.049$ )	0.714( $\pm 0.024$ )	2.673( $\pm 0.141$ )

Table 7: Overall results on the other data sets. The tables show Song Year, Bike Sharing, and Pole from top to bottom. **Song Year:**  $\ell_1$  outperformed HL-Gauss in terms of Test MAE, there was not a substantial difference between the performance of HL-OneBin and HL-Gauss, and the rest of the methods had higher error rates than HL-Gauss. The differences in error rates were small on this data set and even Linear Regression worked well. **Bike Sharing:** HL-Gauss had the lowest Test MAE on this data set and  $\ell_2$  performed the worst among neural network methods. **Pole:** HL-Gauss achieved the lowest Test MAE. The other methods did not achieve a Test MAE close to that of HL-Gauss, and there was a noticeable gap between the performance of HL-OneBin and HL-Gauss.

**F.2 Bias and the Choice of Target Distribution**

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	0.309( $\pm 0.000$ )	0.364( $\pm 0.006$ )	0.335( $\pm 0.004$ )	0.660( $\pm 0.099$ )
HL-Projected	0.053( $\pm 0.001$ )	0.074( $\pm 0.002$ )	0.103( $\pm 0.003$ )	0.462( $\pm 0.065$ )
HL-Gauss.	0.061( $\pm 0.006$ )	0.164( $\pm 0.090$ )	0.098( $\pm 0.005$ )	0.274( $\pm 0.090$ )
Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	5.810( $\pm 0.010$ )	8.487( $\pm 0.002$ )	5.906( $\pm 0.020$ )	8.636( $\pm 0.020$ )
HL-Projected	5.815( $\pm 0.012$ )	8.477( $\pm 0.003$ )	5.917( $\pm 0.019$ )	8.629( $\pm 0.016$ )
HL-Gauss.	5.789( $\pm 0.004$ )	8.440( $\pm 0.004$ )	5.903( $\pm 0.010$ )	8.621( $\pm 0.016$ )
Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	15.822( $\pm 0.198$ )	26.065( $\pm 0.335$ )	26.689( $\pm 0.280$ )	45.150( $\pm 0.857$ )
HL-Projected	14.885( $\pm 0.169$ )	24.329( $\pm 0.329$ )	26.180( $\pm 0.348$ )	44.982( $\pm 0.845$ )
HL-Gauss.	14.335( $\pm 0.152$ )	23.178( $\pm 0.309$ )	25.525( $\pm 0.331$ )	43.671( $\pm 0.796$ )
Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	0.899( $\pm 0.015$ )	1.442( $\pm 0.033$ )	1.264( $\pm 0.019$ )	2.760( $\pm 0.116$ )
HL-Projected	0.364( $\pm 0.020$ )	1.340( $\pm 0.055$ )	0.741( $\pm 0.018$ )	2.753( $\pm 0.068$ )
HL-Gauss.	0.347( $\pm 0.018$ )	1.299( $\pm 0.049$ )	0.714( $\pm 0.024$ )	2.673( $\pm 0.141$ )

Table 8: Discretization bias experiment. The tables show CT-Scan, Song Year, Bike Sharing, and Pole from top to bottom. Overall, HL-Projected achieved a Test MAE close to that of HL-Gauss, and performed noticeably better than HL-OneBin.

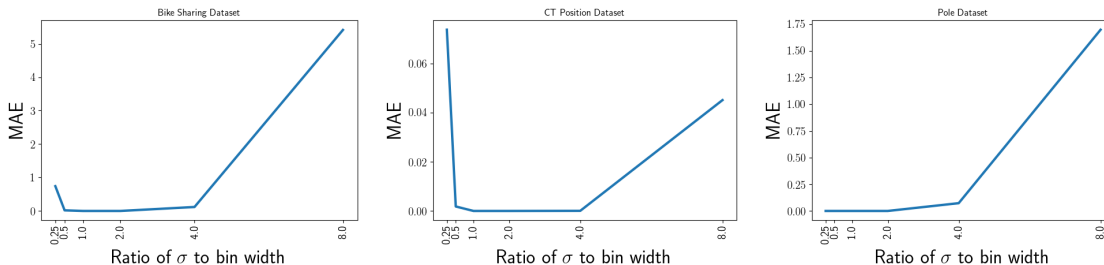


Figure 16: MAE between the means of HL targets and the original labels. It can be seen that extreme values of  $\sigma$  on either side biased the mean of the target distributions.

INVESTIGATING THE HISTOGRAM LOSS IN REGRESSION

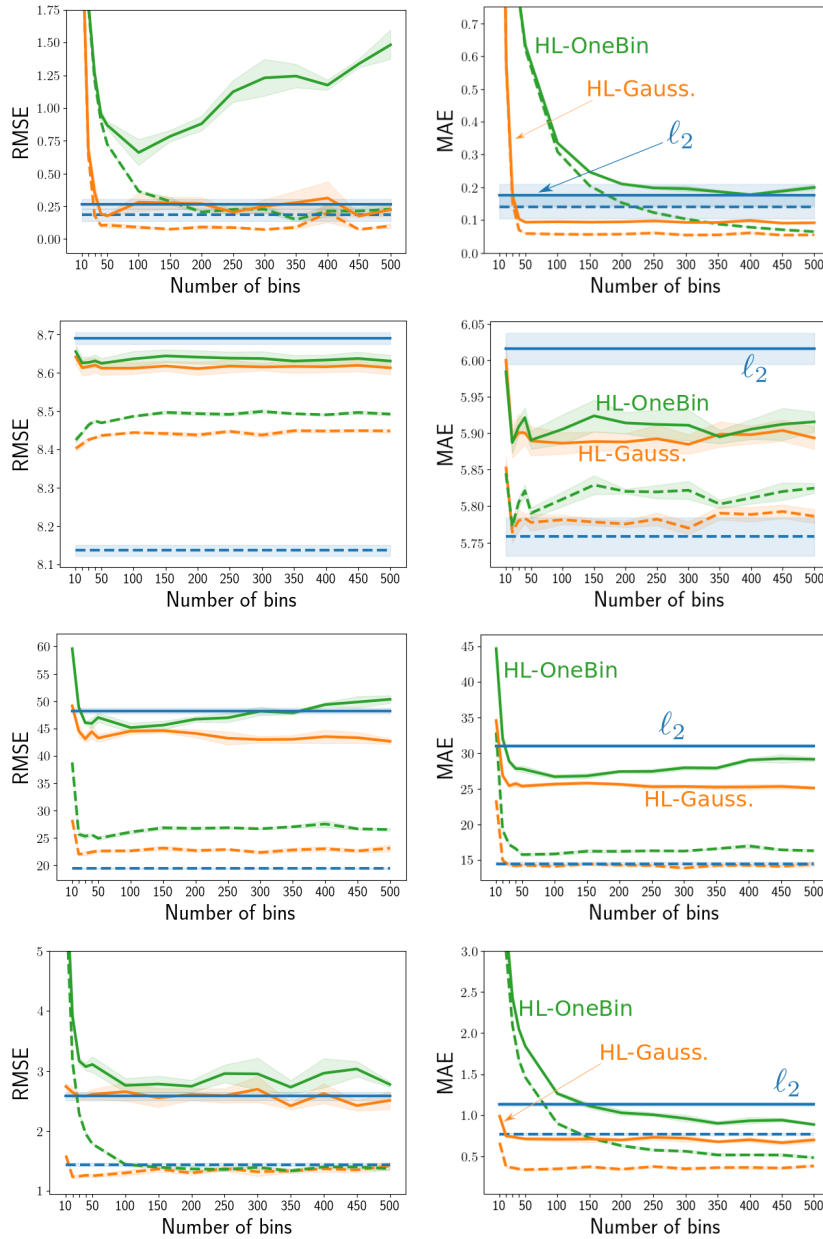


Figure 17: Changing the number of bins. The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. Dotted and solid lines show train and test errors respectively. A small number of bins resulted in high train and test errors, indicating high bias. A higher number of bins generally did not result in a rise in test error, with the exception of HL-OneBin’s test RMSE on CT Scan.

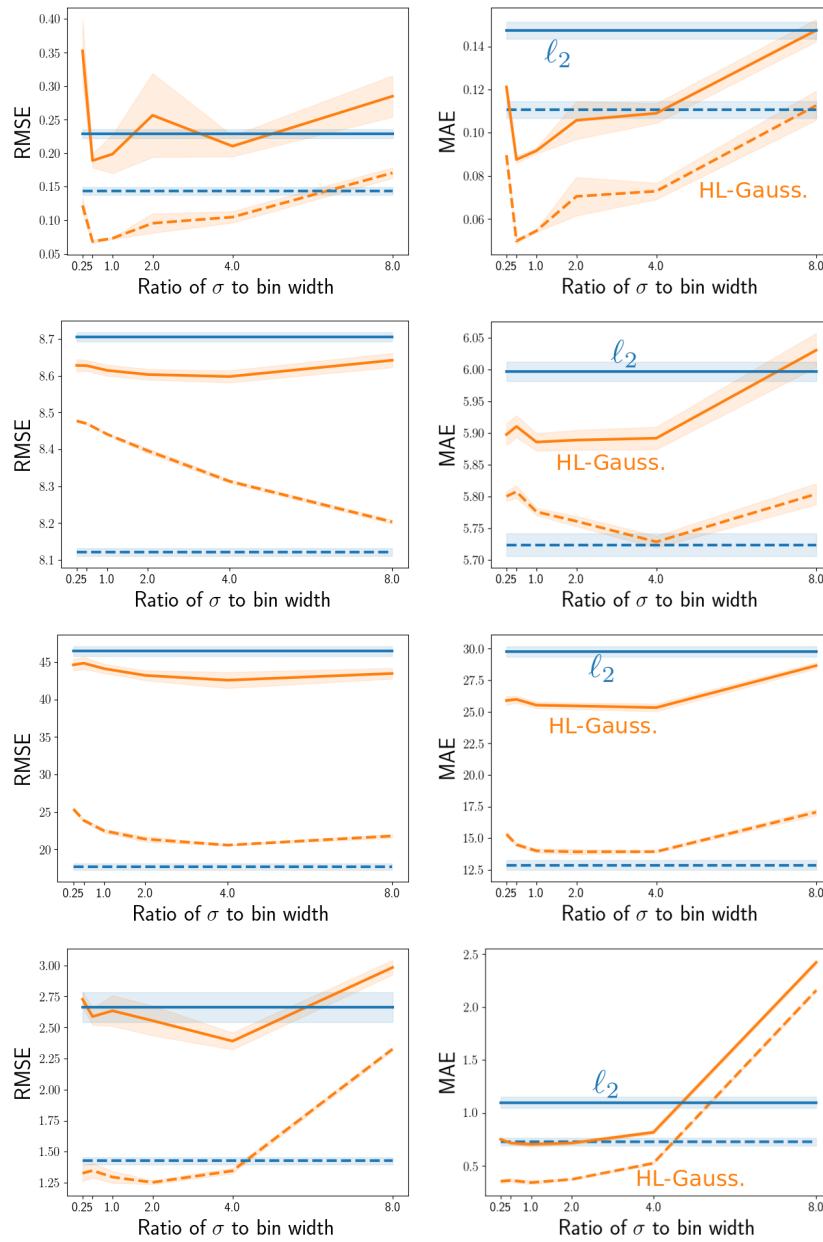


Figure 18: Changing the parameter  $\sigma$ . The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. Dotted and solid lines show train and test errors respectively. Generally, extreme values of  $\sigma$ , especially to the right, resulted in bad performance. The error rates for train and test followed each other when changing this parameter.

**F.3 Representation**

	Loss	Default	Fixed	Initialized	Random
Train MAE	$\ell_2$	5.758( $\pm 0.026$ )	6.268( $\pm 0.055$ )	5.743( $\pm 0.015$ )	8.060( $\pm 0.024$ )
Train MAE	HL-Gauss.	5.789( $\pm 0.004$ )	5.682( $\pm 0.014$ )	5.781( $\pm 0.010$ )	7.942( $\pm 0.021$ )
Train RMSE	$\ell_2$	8.137( $\pm 0.014$ )	8.797( $\pm 0.030$ )	8.131( $\pm 0.003$ )	10.730( $\pm 0.016$ )
Train RMSE	HL-Gauss.	8.440( $\pm 0.004$ )	8.130( $\pm 0.010$ )	8.396( $\pm 0.007$ )	10.691( $\pm 0.026$ )
Test MAE	$\ell_2$	6.016( $\pm 0.022$ )	6.338( $\pm 0.063$ )	5.997( $\pm 0.028$ )	8.069( $\pm 0.029$ )
Test MAE	HL-Gauss.	5.903( $\pm 0.010$ )	5.954( $\pm 0.011$ )	5.900( $\pm 0.016$ )	7.952( $\pm 0.017$ )
Test RMSE	$\ell_2$	8.690( $\pm 0.015$ )	8.901( $\pm 0.046$ )	8.666( $\pm 0.021$ )	10.748( $\pm 0.014$ )
Test RMSE	HL-Gauss.	8.621( $\pm 0.016$ )	8.723( $\pm 0.013$ )	8.603( $\pm 0.013$ )	10.712( $\pm 0.010$ )

	Loss	Default	Fixed	Initialized	Random
Train MAE	$\ell_2$	14.453( $\pm 0.258$ )	39.101( $\pm 1.054$ )	18.214( $\pm 1.154$ )	106.376( $\pm 0.569$ )
Train MAE	HL-Gauss.	14.335( $\pm 0.152$ )	30.171( $\pm 0.469$ )	12.571( $\pm 0.250$ )	99.251( $\pm 1.069$ )
Train RMSE	$\ell_2$	19.487( $\pm 0.116$ )	53.545( $\pm 1.264$ )	23.833( $\pm 1.137$ )	142.523( $\pm 1.052$ )
Train RMSE	HL-Gauss.	23.178( $\pm 0.309$ )	42.140( $\pm 0.664$ )	20.320( $\pm 0.518$ )	136.065( $\pm 1.283$ )
Test MAE	$\ell_2$	31.029( $\pm 0.258$ )	42.834( $\pm 0.983$ )	31.119( $\pm 0.906$ )	106.429( $\pm 0.982$ )
Test MAE	HL-Gauss.	25.525( $\pm 0.331$ )	37.696( $\pm 0.360$ )	25.470( $\pm 0.202$ )	99.311( $\pm 1.486$ )
Test RMSE	$\ell_2$	48.205( $\pm 0.545$ )	59.890( $\pm 0.674$ )	46.735( $\pm 0.387$ )	143.052( $\pm 1.472$ )
Test RMSE	HL-Gauss.	43.671( $\pm 0.796$ )	56.350( $\pm 0.525$ )	43.809( $\pm 0.593$ )	136.941( $\pm 1.639$ )

	Loss	Default	Fixed	Initialized	Random
Train MAE	$\ell_2$	0.767( $\pm 0.016$ )	4.218( $\pm 0.325$ )	1.109( $\pm 0.024$ )	28.936( $\pm 0.176$ )
Train MAE	HL-Gauss.	0.347( $\pm 0.018$ )	1.218( $\pm 0.052$ )	0.312( $\pm 0.010$ )	18.428( $\pm 0.969$ )
Train RMSE	$\ell_2$	1.441( $\pm 0.026$ )	6.184( $\pm 0.412$ )	1.836( $\pm 0.042$ )	33.642( $\pm 0.231$ )
Train RMSE	HL-Gauss.	1.299( $\pm 0.049$ )	2.899( $\pm 0.143$ )	1.212( $\pm 0.027$ )	26.795( $\pm 0.959$ )
Test MAE	$\ell_2$	1.131( $\pm 0.024$ )	4.396( $\pm 0.329$ )	1.437( $\pm 0.053$ )	29.221( $\pm 0.254$ )
Test MAE	HL-Gauss.	0.714( $\pm 0.024$ )	1.398( $\pm 0.021$ )	0.645( $\pm 0.009$ )	18.285( $\pm 1.062$ )
Test RMSE	$\ell_2$	2.579( $\pm 0.073$ )	6.423( $\pm 0.415$ )	2.882( $\pm 0.109$ )	33.944( $\pm 0.299$ )
Test RMSE	HL-Gauss.	2.673( $\pm 0.141$ )	3.441( $\pm 0.093$ )	2.373( $\pm 0.055$ )	26.621( $\pm 1.043$ )

Table 9: Representation results on the other data sets. The tables show Song Year, Bike Sharing, and Pole from top to bottom. We tested (a) swapping the representations and re-learning on the last layer (**Fixed**), (b) initializing with the other’s representation (**Initialized**), (c) and using the same fixed random representation for both (**Random**) and only learning the last layer. In general,  $\ell_2$  underperformed HL-Gauss in all the three settings. Using HL-Gauss’s fixed representation only made the performance of  $\ell_2$  worse. As an initialization, it did not result in a considerable improvement.

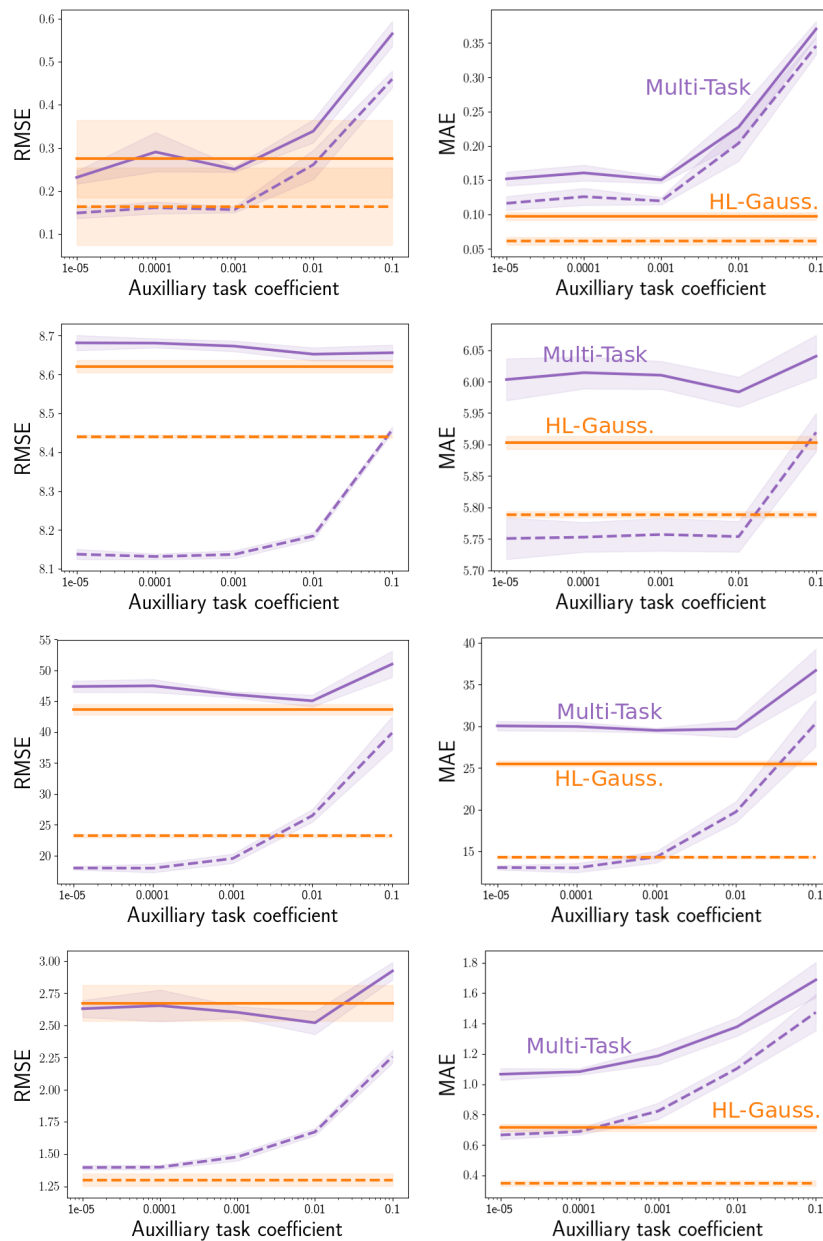


Figure 19: Multi-Task Network results. The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. The loss function is the mean prediction’s squared error plus the distribution prediction’s KL-divergence multiplied by a coefficient. The horizontal axis shows the coefficient for KL-divergence. Dotted and solid lines show train and test errors respectively. There was not a substantial drop in the error rate when increasing the coefficient in the loss and the performance only became worse.

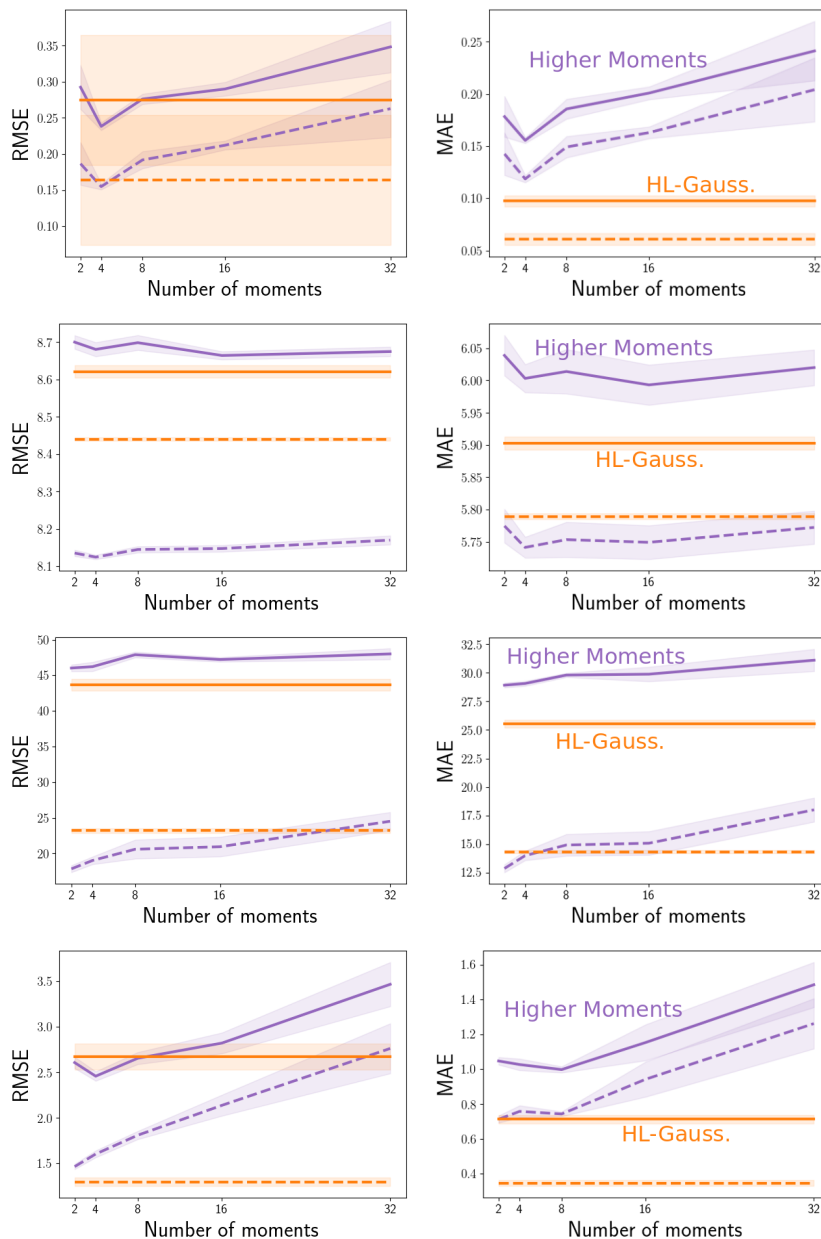


Figure 20: Higher Moments Network results. The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. The X-axis shows the number of moments (including the mean). Dotted and solid lines show train and test errors respectively. There was no substantial decrease in error when predicting higher moments. In some cases there was a slight reduction in error but the new model still performed worse than HL-Gauss in terms of Test MAE.

### F.4 Optimization Properties

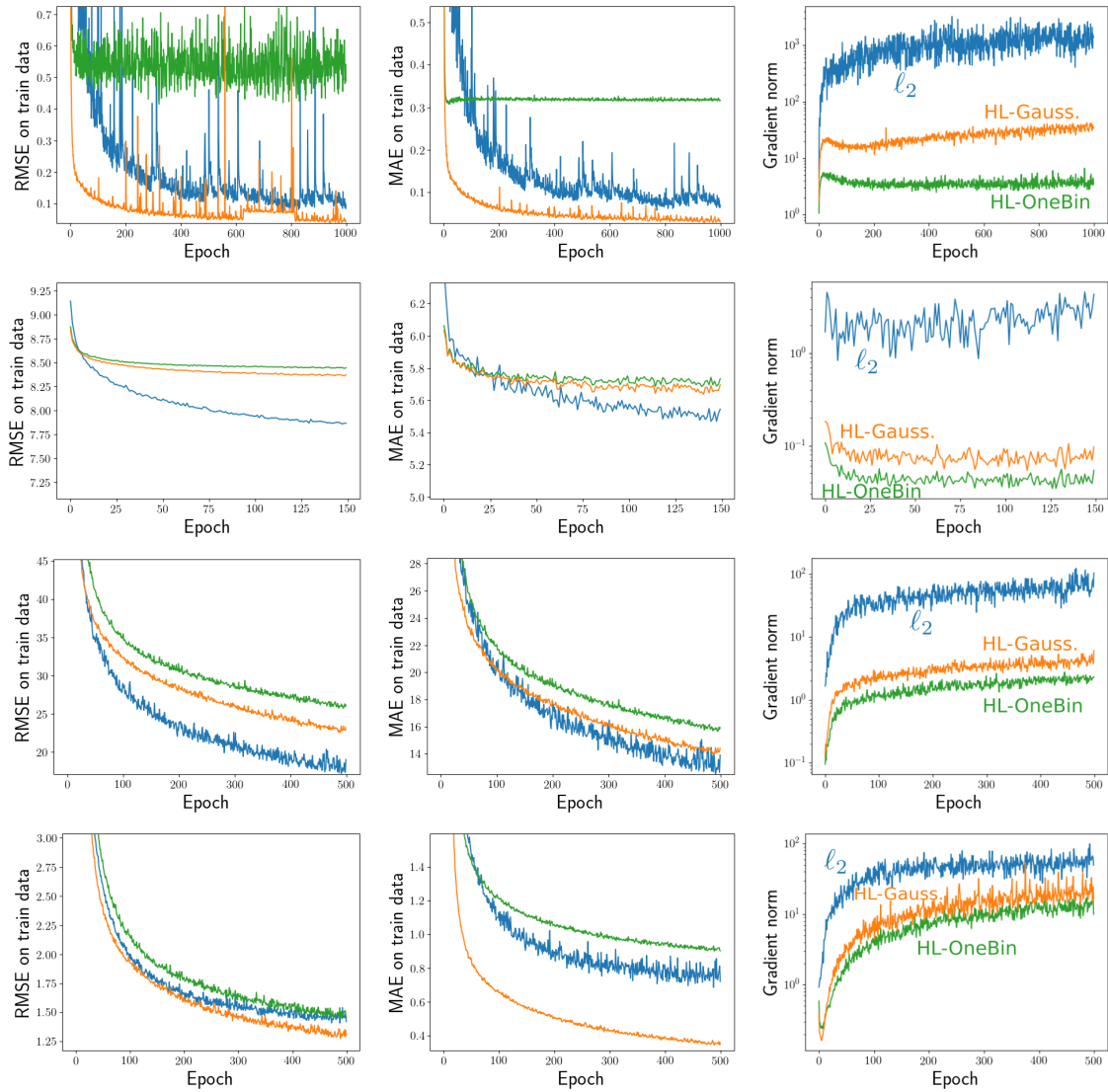


Figure 21: Training process. The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. A logarithmic scale is used in the Y-axis of the rightmost plot. In all cases, training with HL-Gauss resulted in initial fast reduction in train MAE compared to  $\ell_2$ . The gradient norm of  $\ell_2$  was highly varying through the training.

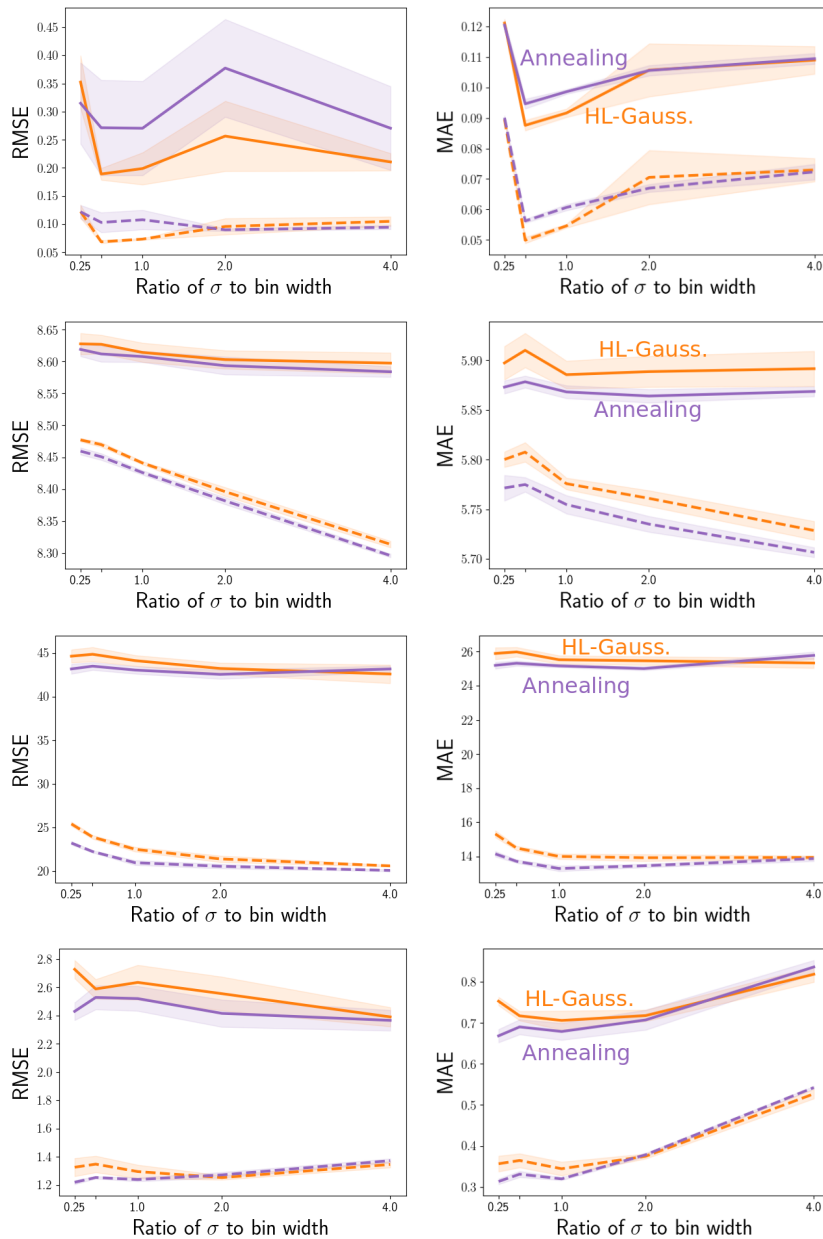


Figure 22: Annealing results. The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. Dotted and solid lines show train and test errors. On three data sets, there was a small benefit in starting with a higher  $\sigma$  and reducing it through the training.

F.5 Robustness to Corrupted Targets

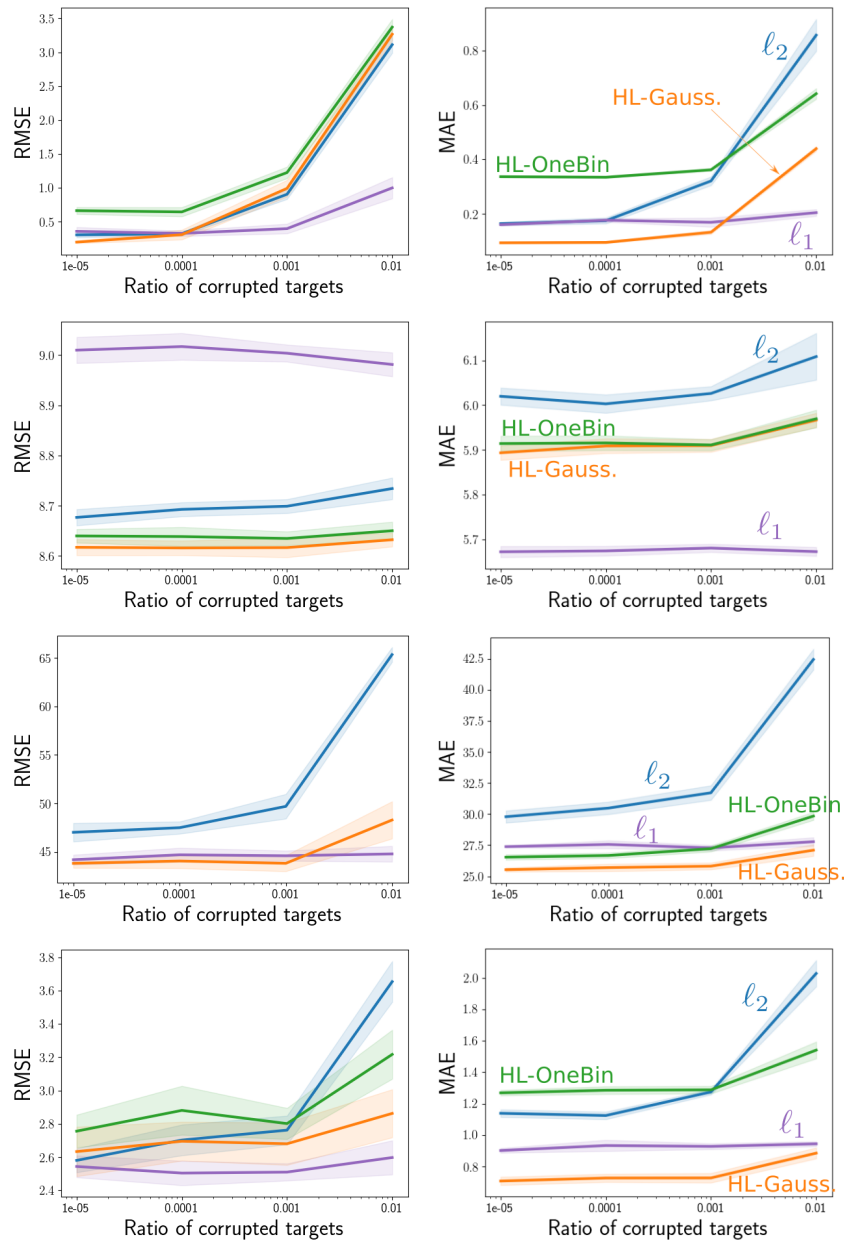


Figure 23: Corrupted targets results. The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. The lines show test errors. The  $l_1$  loss was robust to corrupted targets, and the  $l_2$  loss was affected the most.

F.6 Sensitivity to Input Perturbations

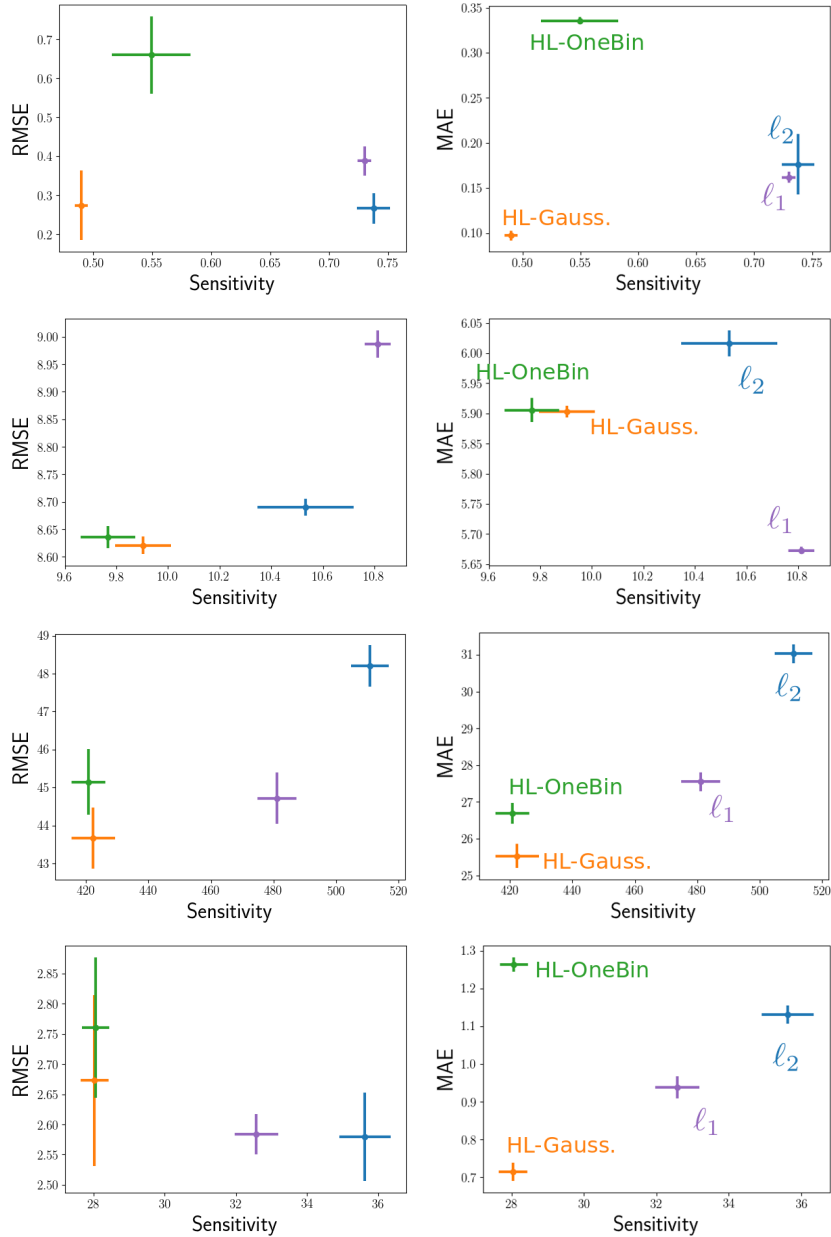


Figure 24: Sensitivity results. The rows show CT Scan, Song Year, Bike Sharing, and Pole respectively. The horizontal axis shows the sensitivity of the model’s output to input perturbations (left means less sensitive) and the vertical axis shows the test error (lower is better). HL-Gauss showed less sensitivity and lower test errors than  $l_1$  and  $l_2$ .

## References

- Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, 2018.
- Jonathan T Barron. A general and adaptive robust loss function. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.
- Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.
- Marc G Bellemare, Will Dabney, and Mark Rowland. *Distributional reinforcement learning*. 2023.
- Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Ismir*, 2011.
- Christopher M Bishop. Mixture density networks. *Technical Report*, 1994.
- Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv*, 2021.
- Clément L Canonne. A short note on an inequality between kl and tv. *arXiv*, 2022.
- Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *International Conference on Image Processing*, 1994.
- François Chollet et al. Keras, 2015.
- Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, 2015.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, 2018a.
- Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*, 2018b.

- Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 2017.
- Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2014.
- Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In *International Conference on Machine Learning*, 2024.
- Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.
- Yasuhiro Fujita, Prabhat Nagarajan, Toshiaki Kataoka, and Takahiro Ishikawa. Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research*, 2021.
- Bin-Bin Gao, Chao Xing, Chen-Wei Xie, Jianxin Wu, and Xin Geng. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 2017.
- Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2d image registration in ct images using radial image descriptors. *Medical Image Computing and Computer-Assisted Intervention*, 2011.
- Humza Haider, Bret Hoehn, Sarah Davis, and Russell Greiner. Effective ways to build and evaluate individual survival distributions. *Journal of Machine Learning Research*, 2020.
- Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, 2016.
- Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI conference on artificial intelligence*, 2018.
- Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 1997a.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997b.
- Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 2017.
- Peter J Huber. Robust statistics. In *International Encyclopedia of Statistical Science*. 2011.
- Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *International Conference on Machine Learning*, 2018.

- Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 2018.
- Khurram Javed, Haseeb Shah, Richard S. Sutton, and Martha White. Scalable real-time recurrent learning using columnar-constructive networks. *Journal of Machine Learning Research*, 2023.
- Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020.
- Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.
- Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.
- Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G Bellemare. On the generalization of representations in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2022.
- Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.
- Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*. 1998.
- Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 2018.
- Clare Lyle, Marc G Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2019.
- Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, 2021.
- Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, 2016.
- Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

- Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 2017.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.
- Cody Reading, Ali Harakeh, Julia Chae, and Steven L Waslander. Categorical depth distribution network for monocular 3d object detection. In *Conference on Computer Vision and Pattern Recognition*, 2021.
- Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*, 2011.
- Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *IEEE International Conference on Computer Vision Workshops*, 2015.
- Mark Rowland, Marc Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2018.
- Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 2017.
- Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 2018.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.
- Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. DIODE: A Dense Indoor and Outdoor DEpth Dataset. *arXiv*, 2019.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.
- Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? 2023.

Tong Zhang. *Mathematical analysis of machine learning algorithms*. Cambridge University Press, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI Conference on Artificial Intelligence*, 2021.