Distributional Losses for Regression

by

Ehsan Imani

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

In this thesis we introduce a new loss for regression, the Histogram Loss. There is some evidence that, in the problem of sequential decision making, estimating the full distribution of return offers a considerable gain in performance, even though only the mean of that distribution is used in decision making. A parallel line of research in classification has found that converting hard one-hot targets to soft targets, distributions that contain information about the relationship between classes or ambiguity in the label, can improve accuracy. These findings have given rise to questions about the underlying reasons that are still left unanswered. Our proposed loss function is influenced by these two ideas and involves learning the conditional distribution of the target variable by minimizing KL-divergence between a target distribution and a flexible histogram prediction. Experiments on four datasets show that the Histogram Loss often outperforms commonly used regression losses. We then design theoretical and empirical analyses to determine why and when this performance gain appears, and how different components of the loss contribute to it. Through this investigation we also provide additional insights about open questions and hypotheses posed in previous works.

Preface

Parts of this thesis have been published by Imani and White (2018). New results in the thesis are to be submitted as a journal paper.

To my family.

Be less curious about people and more curious about ideas.

– Marie Curie

Acknowledgements

First and foremost, I would like to thank my supervisor, Martha White, whose continuous support and feedback throughout this project helped me stay on course and make consistent progress. Other projects she advised me on were no less exciting and, altogether, working with her was a great opportunity for me to develop in both research and character. Second, I am thankful to my examining committee for taking the time to read this thesis and providing insightful comments and constructive criticism. Finally, I am grateful to the members of RLAI and AMII for creating such a positive and thriving atmosphere for research and collaboration.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A fundamental problem in machine learning considers predicting a continuous output variable from a set of input variables. Examples of applications range from predicting the price of a house to predicting the mortality of a disease in a region. Although solutions to this problem are often evaluated by simple measures like absolute error, the loss that is directly minimized can be different. Properties like ease of optimization, improved generalization, and robustness to outliers motivate designing surrogate losses or reductions between problems. This thesis introduces a new surrogate loss for this problem, and provides theoretical and empirical analysis regarding the properties of this loss and its performance in comparison with traditional approaches.

A core idea in the proposed loss function is learning a full distribution instead of its expected value. In regression, it is common to use the squared-error loss, or ℓ_2 loss. This corresponds to assuming that the continuous target variable Y is Gaussian distributed, conditioned on inputs $\mathbf{x} \in \mathbb{R}^d$: $Y \sim \mathcal{N}(\mu = f(\mathbf{x}), \sigma^2)$ for a fixed variance $\sigma^2 > 0$ and some function $f : \mathbb{R}^d \to \mathbb{R}$ on the inputs from a function class $\mathcal{F}$, such as a linear function $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$ for weights $\mathbf{w} \in \mathbb{R}^d$. The maximum likelihood function f for n samples $\{\mathbf{x}_i, y_i\}$, corresponds to minimizing the ℓ_2 loss

$$\min_{f \in \mathcal{F}} \sum_{j=1}^{n} (f(\mathbf{x}_j) - y_j)^2 \tag{1.1}$$

with prediction $f(\mathbf{x}) \approx \mathbb{E}[Y|\mathbf{x}]$.

Alternatively, one could consider learning a distribution over Y directly, and then taking the mean of that distribution—or other statistics—to provide a prediction. This additional difficulty seems hardly worth the effort, considering only the mean is required for prediction. However, the increased difficulty could beneficially prevent overfitting and promote generalization. Several previous studies have found benefit in learning distributions. Below we overview some recent works in Supervised Learning and Reinforcement Learning that motivate this idea.

In the problem of estimating the age of a person, labels are often whole numbers from 0 to 100. Deep Expectation of Apparent Age (DEX) (Rothe *et al.* 2015) treated this problem as multi-class classification, where the classes were the natural numbers of labels. This method parameterizes the distribution as a softmax output layer in a convolutional neural network. The network is trained with the commonly used negative log-likelihood loss in classification. The authors considered using the mode and the mean of the learned distribution, and the mean resulted in a lower error.

Another successful application of learning distributions is in Reinforcement Learning (RL). Value-based methods in RL estimate the expected return (sum of discounted future rewards). It has been known that these methods can be modified to estimate the full distribution of return rather than its expected value (Morimura *et al.* 2010; Morimura *et al.* 2012). A recent algorithm called *C51* (Bellemare, Dabney, *et al.* 2017) restricts the return distribution to a categorical distribution which is modeled with a softmax output layer in a neural network and shows considerable improvements over Q-learning (its well-known counterpart that directly estimates the expected value of return). Although C51's superior performance has made it a component of choice in various RL architectures (Barth-Maron *et al.* 2018; Gruslys *et al.* 2017; Hessel *et al.* 2017; Rowland *et al.* 2018), it is still unclear why it often outperforms Q-learning. Bellemare, Dabney, *et al.* (2017) conjectured several reasons for this phenomenon. For example, it has been hypothesized that the extra challenge of predicting the probability of different values in a categorical distribution can encourage a neural network to learn a better representation as in multi-task learning.

The second idea behind our loss function is using soft targets. This idea is popular in classification and the motivation is to inject prior knowledge or to improve generalization or optimization. Conventional classification models are trained with hard one-hot targets indicating to which class each data point belongs. Soft targets, in contrast, are labels whose values can range from zero to one and represent a probability distribution over classes. Here we describe some previous works on soft targets, and later in Chapter 2 we will draw links between our loss function and some of these works.

A simple modification of targets called Label Smoothing (LS) was introduced by (Szegedy *et al.* 2016) to prevent overfitting in neural networks. If a data point belongs to class k, raising its likelihood under the predicted distribution amounts to minimizing the cross-entropy between the prediction and a Kronecker delta distribution $q(y) \coloneqq \delta_{k,y}$. LS transforms this latter distribution to a mixture of p and a distribution u that does not the depend on the class:

$$q'(y) \coloneqq (1 - \epsilon)\delta_{k,y} + \epsilon u(k)$$

where ϵ is a hyper-parameter between 0 and 1 that controls the level of smoothing. The new distribution q' is then used for training. An alternative interpretation for this method is that, with probability ϵ, the data point's class is replaced with a class drawn from u. Szegedy *et al.* (2016) used a uniform distribution over classes as u.

Soft targets have also been successfully used for model compression. Sometimes achieving high performance on a task requires resource-consuming models (like deep convolutional neural networks) that cannot be implemented on a small device. For neural networks, a popular method to resolve this conflict is distillation (Hinton *et al.* 2015). In this approach, first a large "teacher" model is trained with little concern regarding resources. This model's output layer, a softmax function with a variable temperature parameter initially set to one, models the predicted distribution as a categorical distribution. Intuitively, a higher temperature makes the predicted distribution less confident. The temperature parameter is raised from one to a higher value *after* training the teacher, and the model's predictions on the training data are extracted. A smaller "student" network is then created with a temperature matching the teacher network. A mixture of the teacher's predictions and the original one-hot labels in the dataset are used to train the student network, and, finally, the student network's temperature is set back to one. It has been observed that a student network trained with this procedure, or even one trained with a small ratio of teacher's predicted distributions, outperforms one trained on the one-hot targets (Hinton *et al.* 2015). The claim is that soft targets capture the teacher model's knowledge about relationships among outcomes. For example, in image classification, if the correct label is "dog", the teacher is likely to assign a higher probability to "cat" than to "cabbage". This extra information is absent in one-hot targets.

Label Distribution Learning (LDL) (Geng 2016) is a paradigm that deals with ambiguity in labels. Unlike single-label learning (where a single label is provided for a data point) and multi-label learning (where each data point has a set of labels), in LDL, each data point is given a distribution over all possible labels. LDL paradigm gives rise to new models, as well as adaptations of single- or multi-label models. A successful approach focused on computer vision applications is Deep Label Distribution Learning (DLDL) (Gao *et al.* 2017) which uses a neural network with a softmax output to model the label distribution and trains it by reducing the KL-divergence between predictions and label distributions. Different methods were proposed by Gao *et al.* (2017) to construct label distributions from ambiguous labels in multi-label classification, semantic segmentation, age estimation, and pose estimation.

These successes and open questions underlie the work in this thesis. Our contributions are summarized as follows:

1. We propose a new loss for regression, called the *Histogram Loss* (HL). The targets are converted to a *target distribution*, and the KL-divergence taken between a histogram density and this target distribution. The choice of histogram density provides a relatively flexible *prediction distribution*, that nonetheless enables the KL-divergence to be computed

efficiently. The prediction is then the expected value of this histogram density.

2. A theoretical analysis of the HL is provided. First we show a bound on the gradient norm of the loss, which helps generalization and protects against abundance of abrupt changes in the gradient through the optimization. We then highlight a relationship between minimizing the HL and learning a policy in an entropy-regularized policy optimization framework. The bias of the loss is also characterized and bounded under a mild assumption regarding the support of the prediction distribution. Finally, we show that minimizing the HL reduces an upper bound on the difference between the predictions and the targets in the training data.

3. We compare the HL with several baselines on four regression datasets. In our experiments we find that the loss often improves and never harms performance. The baselines are designed to test effectiveness of various properties that distinguish the HL from the ℓ_2 loss. These properties include learning a flexible distribution, using a softmax nonlinearity in the output, robustness to outliers, and data augmentation in the labels.

4. We design other experiments to test hypotheses regarding the performance of the HL. We test

   (a) if the choice of target distribution has a major impact on the bias of the HL that can explain the difference in error rates,

   (b) if properties of the histogram prediction and target distribution give rise to a bias-variance trade-off,

   (c) if the challenge in the task of predicting a full distribution forces the model to learn a better representation,

   (d) if switching from the ℓ_2 loss to the HL helps gradient descent's search for a good solution,

   (e) if the HL is more robust to corrupted targets in the dataset, and

(f) if the HL finds a model whose output is less sensitive to input perturbations.

This thesis consists of 5 chapters. In Chapter 2 we introduce the Histogram Loss along with a discussion on the choice of target function. Chapters 3 and 4 provide extensive theoretical and empirical analyses of the proposed method. Finally, Chapter 5 concludes the thesis.

Chapter 2

Distributional Losses for Regression

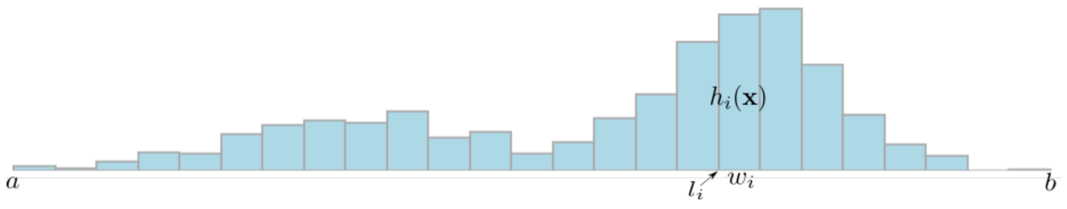
In this chapter, we introduce the Histogram Loss (HL), which generalizes beyond special cases of soft-target losses used in recent work (Gao *et al.* 2017; Norouzi *et al.* 2016; Szegedy *et al.* 2016). We first introduce the loss and how it can be used for regression. We then relate it to other objectives, including maximum likelihood for regression and other methods that learn distributions.

2.1 The Histogram Loss

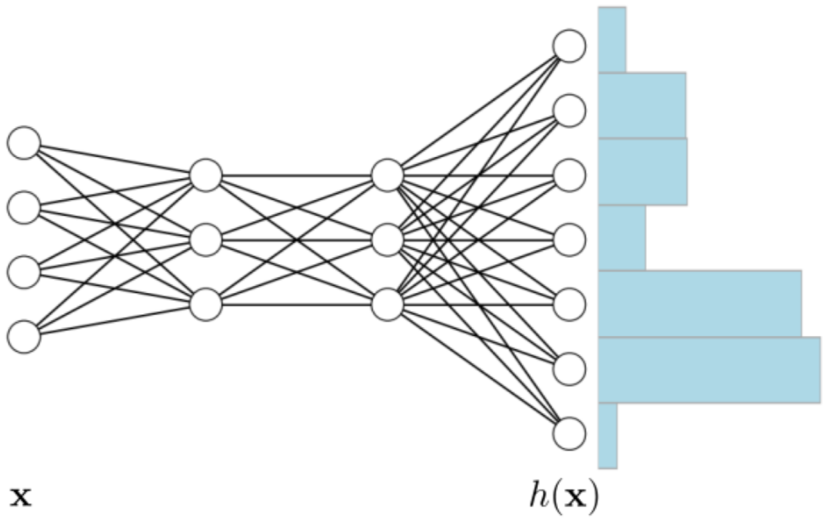
Consider predicting a continuous target Y with event space $\mathcal{Y}$, given inputs $\mathbf{x}$. Recall that directly predicting Y is possible by minimizing the squared error on samples $\{\mathbf{x}_j, y_j\}$ where input $\mathbf{x}_j \in \mathbb{R}^d$ is associated with target $y_j \in \mathbb{R}$. 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 pdf $q_j : \mathcal{Y} \to [0, 1]$. We would like to learn a parameterized *prediction distribution* $h_{\mathbf{x}} : \mathcal{Y} \to [0, 1]$, 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 probability density function. 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 dataset 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 k-dimensional vector $h(\mathbf{x})$ of the coefficients indicating the probability the target is in that bin, given $\mathbf{x}$. The density $h_{\mathbf{x}}$ corresponds to a (normalized) histogram, and has density values $h_i(\mathbf{x})/w_i$ per bin. A sample histogram is depicted in Figure 2.1a. 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 2.1b shows a sample neural network whose output represents a histogram.



(a)



(b)

Figure 2.1: (a) A sample histogram with 25 bins, and (b) a neural network with a softmax output layer that represents a histogram with seven bins.

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 $h_{\mathbf{x}}$:

$$\begin{aligned} -\int_a^b q(y)\log h_{\mathbf{x}}(y)dy &= -\sum_{i=1}^{k}\int_{l_i}^{l_i+w_i} q(y)\log\frac{h_i(\mathbf{x})}{w_i}dy \\ &= -\sum_{i=1}^{k}\log\frac{h_i(\mathbf{x})}{w_i}\underbrace{(F(l_i+w_i)-F(l_i))}_{c_i}. \end{aligned}$$

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

$$HL(q,h_{\mathbf{x}}) = -\sum_{i=1}^{k} c_i \log h_i(\mathbf{x}). \tag{2.1}$$

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. This contrasts previous work, which samples the KL for a subset of y values (Norouzi *et al.* 2016; Szegedy *et al.* 2016). 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 for each input. The method for constructing a target distribution from the target y_j in the dataset can be chosen upfront. Different choices simply result in different weightings in HL. Below, we consider some special cases that are of interest and highlight connections to previous work.

Truncated Gaussian on $Y|\mathbf{x}$ and HL-Gaussian. Consider a truncated Gaussian distribution, on support $[a, b]$, as the target distribution. The mean μ for this Gaussian is the datapoint y_j itself, with fixed variance σ^2. The pdf q is

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

where $Z = \frac{1}{2}(\operatorname{erf}\left(\frac{b-\mu}{\sqrt{2}\sigma}\right) - \operatorname{erf}\left(\frac{a-\mu}{\sqrt{2}\sigma}\right))$, and the HL has

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

This distribution enables smoothing over Y, through the variance parameter σ^2. We call this loss HL-Gaussian, defined by number of bins k and variance σ^2. Based on positive empirical performance, it will be the main HL loss that we advocate for and analyze.

Soft Targets and a Histogram Density on $Y|\mathbf{x}$. In classification, such as multinomial logistic regression, it is typical to assume $Y|\mathbf{x}$ is a categorical distribution, where Y is discrete. The goal is still to estimate $\mathbb{E}[Y|\mathbf{x}]$ and when training, hard 0-1 values for Y are used in the cross-entropy. Soft labels, instead of 0-1 labels, can be used by adding label noise (Norouzi *et al.* 2016; Pereyra *et al.* 2017; Szegedy *et al.* 2016). This can be seen as an instance of HL, but for discrete Y, where a categorical distribution is selected for the target distribution. Minimizing the cross-entropy to these soft-labels corresponds to trying to match such a smoothed target distribution, rather than the original 0-1 categorical distribution.

Such soft targets have also been considered for ordinal regression, particularly for age prediction (Gao *et al.* 2017; Rothe *et al.* 2018). The outputs are smoothed using radial basis function similarities to a set of bin centers. This procedure can be seen as selecting a histogram density for the target distribution, where the coefficients for each bin are determined by these radial basis function similarities. The resulting loss is similar to HL-Gaussian, with slightly different c_i, though introduced as data augmentation to smooth (ordinal) targets.

Dirac delta on $Y|\mathbf{x}$. Finally, we consider the relationship to maximum likelihood. For classification, Norouzi *et al.* (2016) and Szegedy *et al.* (2016) used a combination of maximum likelihood and a KL-divergence to a (uniform) distribution. Szegedy *et al.* (2016) add uniform noise to the labels and Norouzi *et al.* (2016) sample from an exponentiated reward distribution, with a temperature parameter, for structured prediction. Both consider only a finite set for Y, because they both address classification problems.

The relationship between KL-divergence and maximum likelihood can be extended to continuous Y. The connection is typically in terms of statistical consistency: the maximum likelihood estimator approaches the minimum of the KL-divergence to the true distribution, if the distributions are of the same parametric form (Wasserman 2004, Theorem 9.13). They can, however, be connected for finite samples with different distributions. Consider Gaussians centered around datapoints y_j, with arbitrarily small variances $\frac{1}{2}a^2$:

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

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

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

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

$$\operatorname*{argmin}_{h_1,\ldots,f_k} - \sum_{j=1}^{n} \log h_{i_j}(\mathbf{x}_j) = \operatorname*{argmin}_{h_1,\ldots,f_k} - \sum_{j=1}^{n} \log h_{\mathbf{x}_j}(y_j).$$

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**.

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 ϵ on the uniform distribution, the resulting loss **HL-Uniform** has $c_i = \epsilon$ for $i \neq i_j$, and $c_{i_j} = 1 - k\epsilon$.

Chapter 3

Theoretical Analysis

This chapter assembles our theoretical results on the Histogram Loss. The first section considers the behavior of gradient descent optimization on HL and the ℓ_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 the HL which suggests that the gradient varies less through the training and provides more reliable optimization steps. We also point out how this norm relates to the generalization performance of the model. The second section shows a similarity between minimizing the HL and entropy-regularized policy improvement based on recent work by Norouzi *et al.* (2016). The third section discusses the bias of the minimizer of the HL. The mean of the distribution of targets can be different from a coarse histogram prediction distribution that approximates it. We show that, when KL-divergence to data points with Gaussian target distributions 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 section shows that minimizing the 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 HL

Hardt *et al.* (2015) 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 the HL with a Gaussian distribution (HL-Gaussian) in fact promotes an improved bound on this norm, over both the ℓ_2 loss and the HL with all weight in one bin (HL-OneBin).

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

$$h_i(\mathbf{x}) = \frac{\exp(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathbf{w}_i)}{\sum_{j=1}^{k} \exp(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathbf{w}_j)} \tag{3.1}$$

for some function $\boldsymbol{\phi}_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{R}^k$ parameterized by a vector of parameters $\boldsymbol{\theta}$. For example, $\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$ 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-Gaussian) *Assume* $\mathbf{x}, y$ *are fixed, giving fixed coefficients* c_i *in HL-Gaussian. Let* $h_i(\mathbf{x})$ *be as in* (3.1)*, defined by the parameters* $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$ *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 w.r.t.* $\boldsymbol{\theta}$

$$\|\nabla_{\boldsymbol{\theta}}(\mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x}))\| \leq l \tag{3.2}$$

Then the norm of the gradient for HL-Gaussian, w.r.t. all the parameters in the network $\{\boldsymbol{\theta}, \mathbf{w}\}$*, is bounded by*

$$\|\nabla_{\boldsymbol{\theta},\mathbf{w}} HL(q, h_{\mathbf{x}})\| \leq (l + \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\|) \sum_{i=1}^{k} |c_i - h_i(\mathbf{x})| \tag{3.3}$$

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 the HL, w.r.t b_i

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

Then

$$\frac{\partial}{\partial \mathbf{w}_i} \sum_{j=1}^k c_i \log h_i(\mathbf{x}) = (c_i - h_i(\mathbf{x}))\, \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$$
$$\frac{\partial}{\partial \boldsymbol{\theta}} \sum_{j=1}^k c_i \log h_i(\mathbf{x}) = \sum_{i=1}^k (c_i - h_i(\mathbf{x}))\, \nabla \mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$$

where $J\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$ is the Jacobian of $\boldsymbol{\phi}_{\boldsymbol{\theta}}$.

The norm of the gradient of HL in Equation (2.1), w.r.t. $\mathbf{w}$ which is composed of all the weights $\mathbf{w}_i \in \mathbb{R}^k$ is

$$\left\| \frac{\partial}{\partial \mathbf{w}} \sum_{j=1}^k c_j \log h_j(\mathbf{x}) \right\| \leq \sum_{i=1}^k \left\| \frac{\partial}{\partial \mathbf{w}_i} \sum_{j=1}^k c_j \log h_j(\mathbf{x}) \right\|$$
$$= \sum_{i=1}^k \|(c_i - h_i(\mathbf{x}))\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\|$$

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

Similarly, the norm of the gradient w.r.t. $\boldsymbol{\theta}$ is

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

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

The results by Hardt *et al.* (2015) suggest it is beneficial for the local Lipschitz constant—or the norm of the gradient—to be small on each step. HL-Gaussian provides exactly this property. Besides the network architecture—which we are here assuming is chosen outside of our control—the HL-Gaussian gradient norm is proportional to $|c_i - h_i(\mathbf{x})|$. This number is guaranteed to be less than 1, but generally is 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 ℓ_2 loss and HL-OneBin. For the ℓ_2 loss, $(f(\mathbf{x}) - y)\begin{bmatrix}\nabla_{\boldsymbol{\theta}} \mathbf{w}^{\top} \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x}) \\ \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\end{bmatrix}$ is the gradient, giving gradient norm bound $(l + \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\|)|f(\mathbf{x}) - y|$. 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-Gaussian, but suffers from another problem. The Lipschitz constant l in Equation (3.2) 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

$\mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$ to get larger, to enable $h_i(\mathbf{x})$ to get close to 1.

3.2 Connection to Reinforcement Learning

The 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 then enable more efficient convergence to optimal. 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—make its current best prediction—and acting more randomly—with high-entropy. For continuous action set $\mathcal{Y}$, the goal is to minimize the following objective

$$\int_{\mathcal{X}} p_s(\mathbf{x}) \Big[-\tau H(h_{\mathbf{x}}) - \int_{\mathcal{Y}} h_{\mathbf{x}}(y) r(y, y_i) dy \Big] d\mathbf{x} \tag{3.4}$$

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 (3.4) 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 the HL and maximum-entropy reinforcement learning is that both are minimizing a divergence to this exponentiated distribution p. The 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 σ^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 $D_{KL}(h_{\mathbf{x}}||q)$ is upper-bounded by $D_{KL}(q||h_{\mathbf{x}})$ plus variance terms. The intuition, then, is that minimizing the 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 due to the chosen 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 distribution of targets.

Through this section, we assume that the model is flexible enough so that the predicted distribution for each input can be optimized independently. Therefore we consider one input and drop the subscripts that show dependence on $\mathbf{x}$. In the data, each input can be associated with one or multiple targets. More generally, we consider a *distribution of targets* with pdf $p : \mathcal{Y} \to [0, 1]$ for an input. Note that this is different from the target distribution that is used in the HL. Targets y_j are sampled 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-Gaussian, or a Dirac delta at y_j if we use HL-OneBin) which is then used to train the model with the 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.

We first define a function s:

$$s(z) := \int p(y) q_y(z) \, \mathrm{d}y$$

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

$$\begin{aligned}
\int s(z)\,\mathrm{d}z &= \int\int p(y)q_y(z)\,\mathrm{d}y\,\mathrm{d}z \\
&= \int p(y)(\int q_y(z)\,\mathrm{d}z)\,\mathrm{d}y \\
&= \int p(y)\,\mathrm{d}y \\
&= 1
\end{aligned}$$

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:

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

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. 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 $g(.|\mu, \sigma^2)$ is the pdf of a Gaussian distribution with mean μ and variance σ^2.

$$\begin{aligned}
\mathbb{E}_s[z] &= \int z s(z)\,\mathrm{d}z \\
&= \int z \int g(z|y, \sigma^2)p(y)\,\mathrm{d}y\,\mathrm{d}z \\
&= \int p(y) \int z g(z|y, \sigma^2)\,\mathrm{d}z\,\mathrm{d}y
\end{aligned}$$

$$
\begin{aligned}
&= \int p(y) y \,\mathrm{d}y \\
&= \mathbb{E}_p[y]
\end{aligned}
$$

However, the prediction function h is restricted to a histogram in HL-Gaussian. 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 h has a sufficiently wide support so that there is negligible probability in tails of s beyond this range. The second source of bias is a result of using discrete bins for prediction. We call this bias *discretization bias* and quantify it.

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. We call this density h^* and find the difference between the expected values of h^* and p.

We consider the case where all bins have equal width w. The center of bin i and the probability of 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.

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

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-Gaussian is bounded by $\pm\frac{w}{2}$ and depends on the choice of the parameter σ.

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

$$\begin{aligned}
&\sum_{i=-\infty}^{\infty} m_i \int_{\mathcal{S}_i} g(z|y,\sigma^2)\,\mathrm{d}z - y \\
&= \sum_{i=-\infty}^{\infty} (m_0 + iw) \int_{\mathcal{S}_i} g(z|y,\sigma^2)\,\mathrm{d}z - y \\
&= m_0 \sum_{i=-\infty}^{\infty} \int_{\mathcal{S}_i} g(z|y,\sigma^2)\,\mathrm{d}z + w \sum_{i=-\infty}^{\infty} i \int_{\mathcal{S}_i} g(z|y,\sigma^2)\,\mathrm{d}z - y \qquad (3.5)
\end{aligned}$$

Recall that intervals $\mathcal{S}_i$ are disjoint and $\bigcup\limits_{i=-\infty}^{\infty} \mathcal{S}_i = (-\infty, \infty)$. So the series in the first term of (3.5) becomes one. Therefore

$$\begin{aligned}
(3.5) =&(m_0 - y) + w\Bigg(\sum_{i=-\infty}^{-1} i \int_{\mathcal{S}_i} g(z|y,\sigma^2)\,\mathrm{d}z + \sum_{i=1}^{\infty} i \int_{\mathcal{S}_i} g(z|y,\sigma^2)\,\mathrm{d}z\Bigg) \\
=&-\delta + w\Bigg(\sum_{i=-\infty}^{-1} iF_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2}) \\
&+ \sum_{i=1}^{\infty} iF_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2})\Bigg) \\
=&-\delta + w\Bigg(\sum_{i=1}^{\infty} (-i)F_y(m_0 - iw - \frac{w}{2}, m_0 - iw + \frac{w}{2}) \\
&+ \sum_{i=1}^{\infty} iF_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2})\Bigg) \\
=&-\delta + w\Bigg(\sum_{i=1}^{\infty} i(F_y(m_0 + iw - \frac{w}{2}, m_0 + iw + \frac{w}{2}) \qquad (3.6) \\
&- F_y(m_0 - iw - \frac{w}{2}, m_0 - iw + \frac{w}{2}))\Bigg)
\end{aligned}$$

Due to the symmetry of Gaussian distribution, $F_\mu(a,b) = F_\mu(2\mu - b, 2\mu - a)$.

We define $a_i \coloneqq iw - \frac{w}{2}$ and $b_i \coloneqq iw + \frac{w}{2}$ and the series in (3.6) becomes

$$\begin{aligned}
&\sum_{i=1}^{\infty} i(F_y(m_0 + a_i, m_0 + b_i) - F_y(m_0 - b_i, m_0 - a_i)) \\
=&\sum_{i=1}^{\infty} i(F_y(m_0 + a_i, m_0 + b_i) - F_y(2y - m_0 + a_i, 2y - m_0 + b_i)) \\
\overset{\delta:=y-m_0}{=}&\sum_{i=1}^{\infty} i(F_y(y - \delta + a_i, y - \delta + b_i) - F_y(y + \delta + a_i, y + \delta + b_i)) \\
=&\sum_{i=1}^{\infty} i(\Phi_y(y - \delta + b_i) - \Phi_y(y - \delta + a_i) + \Phi_y(y + \delta + a_i) - \Phi_y(y + \delta + b_i)) \\
=&\sum_{i=1}^{\infty} i(-F_y(y + b_i - \delta, y + b_i + \delta) + F_y(y + a_i - \delta, y + a_i + \delta)) \\
=&-\sum_{i=1}^{\infty} iF_y(y + b_i - \delta, y + b_i + \delta) + \sum_{i=1}^{\infty} iF_y(y + a_i - \delta, y + a_i + \delta)
\end{aligned} \tag{3.7}$$

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

$$\begin{aligned}
(3.7) =&-\sum_{i=1}^{\infty} iF_y(y + a_{i+1} - \delta, y + a_{i+1} + \delta) + \sum_{i=1}^{\infty} iF_y(y + a_i - \delta, y + a_i + \delta) \\
=&-\sum_{i=2}^{\infty} (i-1)F_y(y + a_i - \delta, y + a_i + \delta) + \sum_{i=1}^{\infty} iF_y(y + a_i - \delta, y + a_i + \delta) \\
=&\sum_{i=2}^{\infty} F_y(y + a_i - \delta, y + a_i + \delta) + F_y(y + a_1 - \delta, y + a_1 + \delta) && (3.8) \\
=&\sum_{i=1}^{\infty} F_y(y + a_i - \delta, y + a_i + \delta) \\
=&\sum_{i=1}^{\infty} F_0(a_i - \delta, a_i + \delta) \\
=&\sum_{i=1}^{\infty} F_0(iw - \frac{w}{2} - \delta, iw - \frac{w}{2} + \delta) && (3.9)
\end{aligned}$$

where we used the method of differences to obtain (3.8). We can now plug (3.9) into (3.6) to have

$$(3.6) = -\delta + w\left(\sum_{i=1}^{\infty} F_0(iw - \frac{w}{2} - \delta, iw - \frac{w}{2} + \delta)\right) \tag{3.10}$$

The sum in (3.10) 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 δ 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 the worst case is when $\sigma \to 0$ (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 $m_0 - y = -\delta = \mp\frac{w}{2}$.

This analysis characterizes the bias of the mean of the distribution that minimizes HL-Gaussian 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. We empirically investigate this effect in Chapter 4.

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 2 *Assuming that for a data point, the target distribution q and* *the model's prediction distribution $h_{\mathbf{x}}$ have supports bounded by the range $[a, b]$,*

$$(\mathbb{E}[q] - \mathbb{E}[h_{\mathbf{x}}])^2 \leq \frac{(b-a)^4}{32} D_{KL}(q||h_{\mathbf{x}}) \tag{3.11}$$

Proof

$$\begin{aligned}
|\,\mathbb{E}[q] - \mathbb{E}[h_{\mathbf{x}}]| =& |\int_a^b (q(y) - h_{\mathbf{x}}(y)) y \,\mathrm{d}y| \\
=& |\int_a^b (q(y) - h_{\mathbf{x}}(y))(\frac{a+b}{2}) \,\mathrm{d}y + \int_a^b (q(y) - h_{\mathbf{x}}(y))(y - \frac{a+b}{2}) \,\mathrm{d}y| \\
=& |0 + \int_a^b (q(y) - h_{\mathbf{x}}(y))(y - \frac{a+b}{2}) \,\mathrm{d}y| \\
=& |\int_{-\frac{b-a}{2}}^{\frac{b-a}{2}} (q(y + \frac{a+b}{2}) - h_{\mathbf{x}}(y + \frac{a+b}{2})) y \,\mathrm{d}y| \\
=& |\int_{-\frac{b-a}{2}}^{0} (q(y + \frac{a+b}{2}) - h_{\mathbf{x}}(y + \frac{a+b}{2})) y \,\mathrm{d}y \\
&+ \int_0^{\frac{b-a}{2}} (q(y + \frac{a+b}{2}) - h_{\mathbf{x}}(y + \frac{a+b}{2})) y \,\mathrm{d}y| \\
=& | - \int_0^{\frac{b-a}{2}} (q(-y + \frac{a+b}{2}) - h_{\mathbf{x}}(-y + \frac{a+b}{2})) y \,\mathrm{d}y \\
&+ \int_0^{\frac{b-a}{2}} (q(y + \frac{a+b}{2}) - h_{\mathbf{x}}(y + \frac{a+b}{2})) y \,\mathrm{d}y| \\
=& |\int_0^{\frac{b-a}{2}} ((q(y + \frac{a+b}{2}) - h_{\mathbf{x}}(y + \frac{a+b}{2})) \\
&- (q(-y + \frac{a+b}{2}) - h_{\mathbf{x}}(-y + \frac{a+b}{2}))) y \,\mathrm{d}y|
\end{aligned} \tag{3.12}$$

where we have the difference between p and $h_{\mathbf{x}}$ at $y + \frac{a+b}{2}$ minus the difference between these two distributions at $-y + \frac{a+b}{2}$. Since this value is always multiplied by a positive number, we can replace it by twice the supremum of pointwise difference between p and $h_{\mathbf{x}}$ and have

$$\begin{aligned}
(3.12) \leq& 2(\sup_t |q(t) - h_{\mathbf{x}}(t)|) \int_0^{\frac{b-a}{2}} y \,\mathrm{d}y \\
=& (\sup_t |q(t) - h_{\mathbf{x}}(t)|) \frac{(b-a)^2}{4} \\
\leq& \frac{(b-a)^2}{4\sqrt{2}} \sqrt{D_{KL}(q||h_{\mathbf{x}})}
\end{aligned}$$

where we used Pinsker's inequality on the last step. ■

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

Chapter 4

Experiments

In this chapter, we investigate the utility of HL-Gaussian for regression, compared to using an ℓ_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 the HL.

4.1 Overview of Empirical Study

The purpose of our experiments is to first show proof of concept studies and then provide insights on the role of factors that differentiate the HL from the ℓ_2 loss. The main goal is to understand when and why one would opt for the HL. The datasets and the algorithms used in our experiments are described in Sections 4.2 and 4.3 respectively.

Section 4.4 compares HL-Gaussian, ℓ_2 and several baselines, each of which is designed to differ in a particular aspect from HL-Gaussian or ℓ_2. The comparison shows the importance of each of these factors. These experiments attempt to clarify (a) if HL-Gaussian is a reasonable choice for a regression task in practice, (b) whether the difference between the performance of HL-Gaussian and ℓ_2 is merely because HL-Gaussian is learning a flexible distribution, (c) if sensitivity of ℓ_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-Gaussian.

In Section 4.5, we investigate the impact of discretization bias on both

HL-Gaussian 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 4.5 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-Gaussian. We further evaluate the models with different numbers of bins and variance parameter 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, Dabney, *et al.* 2017). Experiments in section 4.6 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 4.7 we compare the behavior of ℓ_2, HL-OneBin, and HL-Gaussian 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, Sections 4.8 and 4.9 measure the sensitivity of the 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.

4.2 Datasets and Pre-Processing

Experiments are conducted on four large-scale regression datasets. We include an overview of the datasets in Table 4.1 and show a histogram of their targets in Figure 4.1.

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

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

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

All datasets 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 datasets 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.

Dataset	# 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 4.1: Overview of the datasets used in the experiments.

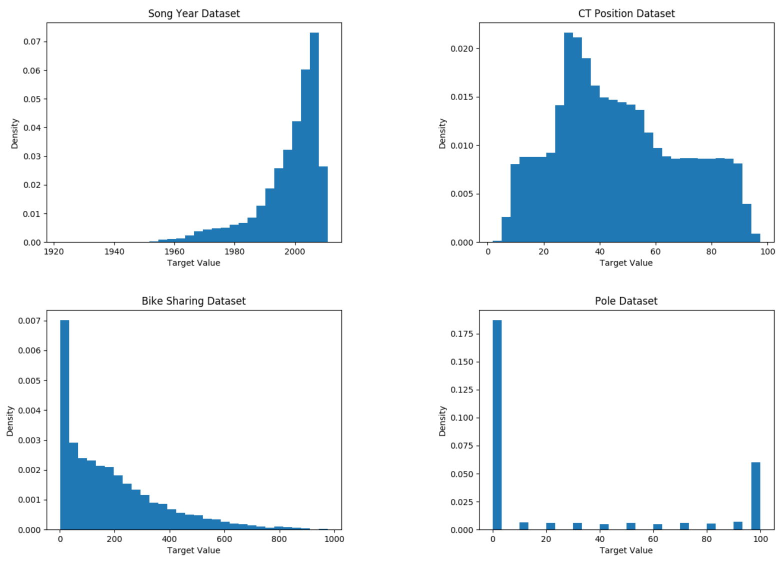


Figure 4.1: Histograms of the target values for the four datasets. Each plot shows a histogram with 30 bins, where the area of a bin is the ratio of targets in the dataset that fall in that bin.

4.3 Algorithms

We compared several regression strategies, distribution learning approaches and several variants of HL. All the approaches—except for Linear Regression—use the same neural network, with differences only in the output layer. The architecture 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, for CT Position is 385-192-192-192-192-1, and for Pole is 49-24-24-24-1. All units employ ReLU activation, except the last layer with linear activations. Unless specified otherwise, all networks using HL have 100 bins. The support of the histogram is chosen by the dataset target range and 10 bins are kept for padding on each side to minimize the effect of truncation. Meta-parameters for comparison algorithms are chosen according to best Test MAE. Network architectures were chosen according to best Test MAE for ℓ_2, with depth and width varied across 7 different values.

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

Squared-error ℓ_2 is the neural network trained using the ℓ_2 loss. The targets are normalized to range $[0, 1]$, which was needed to improve stability and accuracy.

Absolute-error ℓ_1 is the neural network trained using the ℓ_1 loss.

$\boldsymbol{\ell_2}$**+Noise** is the same as ℓ_2, except Gaussian noise is added to the targets as a form of augmentation. The standard deviation of the noise is selected from $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

$\boldsymbol{\ell_2}$**+Clipping** is the same as ℓ_2, but with gradient norm clipping during training. The threshold for clipping is selected from $\{0.01, 0.1, 1, 10\}$.

HL-OneBin is the HL, with Dirac delta target distribution.

HL-Uniform is the HL, with a target distribution that mixes between a delta distribution and the uniform distribution, with a weighting of ϵ on the uniform and $1-\epsilon$ on the delta, where $\epsilon \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

HL-Gaussian is the HL, with a truncated Gaussian distribution as the target distribution. The parameter σ is set to the width of the bins.

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

$\boldsymbol{\ell_2}$**+Softmax** uses a softmax-layer with ℓ_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-Gaussian.

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 J. Ba 2014) 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.

4.4 Overall Results

We first report the relative performance of different baselines compared to HL-Gaussian on the the four datasets in Tables 4.2 to 4.5. The error rates of HL-Uniform are plotted separately in Figures 4.2 to 4.5 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 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(±0.007)	8.209(±0.006)	6.170(±0.025)	8.341(±0.025)
ℓ_2	0.140(±0.036)	0.183(±0.048)	0.176(±0.034)	0.267(±0.039)
ℓ_2+Noise	0.114(±0.011)	0.152(±0.015)	0.152(±0.010)	0.311(±0.076)
ℓ_2+Clip	0.111(±0.004)	0.143(±0.005)	0.148(±0.004)	0.231(±0.006)
ℓ_1	0.121(±0.007)	0.164(±0.007)	0.162(±0.006)	0.389(±0.038)
MDN	0.114(±0.004)	0.152(±0.005)	0.153(±0.004)	0.308(±0.035)
ℓ_2+softmax	0.065(±0.006)	0.094(±0.008)	0.105(±0.006)	0.301(±0.067)
HL-OneBin	0.309(±0.000)	0.364(±0.006)	0.335(±0.004)	0.660(±0.099)
HL-Gauss.	0.061(±0.006)	0.164(±0.090)	0.098(±0.005)	0.274(±0.090)

Table 4.2: Overall results on CT Scan. HL-Gaussian achieved the lowest Test MAE. Among the other methods, ℓ_2+softmax yielded an error rate close to that of HL-Gaussian, and the rest of the baselines performed worse. There was a large gap between the performance of HL-OneBin and HL-Gaussian.

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lin Reg	6.793(±0.003)	9.547(±0.003)	6.796(±0.007)	9.555(±0.014)
ℓ_2	5.758(±0.026)	8.137(±0.014)	6.016(±0.022)	8.690(±0.015)
ℓ_2+Noise	5.697(±0.023)	8.124(±0.006)	5.959(±0.015)	8.682(±0.022)
ℓ_2+Clip	5.749(±0.032)	8.130(±0.012)	6.009(±0.032)	8.687(±0.015)
ℓ_1	5.423(±0.010)	8.640(±0.016)	5.673(±0.006)	8.987(±0.025)
MDN	5.858(±0.020)	8.530(±0.008)	5.935(±0.025)	8.640(±0.020)
ℓ_2+softmax	5.651(±0.015)	8.067(±0.009)	5.947(±0.012)	8.685(±0.012)
HL-OneBin	5.810(±0.010)	8.487(±0.002)	5.906(±0.020)	8.636(±0.020)
HL-Gauss.	5.789(±0.004)	8.440(±0.004)	5.903(±0.010)	8.621(±0.016)

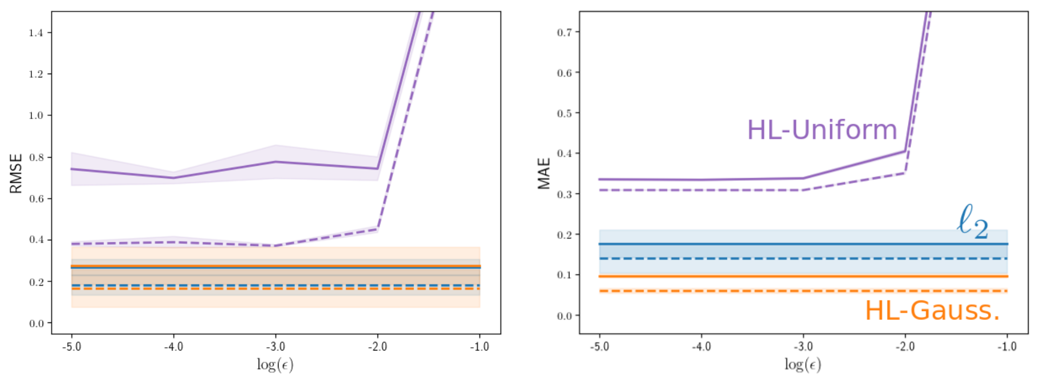
Table 4.3: Overall results on Song Year. On this dataset, ℓ_1 outperformed HL-Gaussian in terms of Test MAE, there was not a substantial difference between the performance of HL-OneBin and HL-Gaussian, and the rest of the methods had higher error rates than HL-Gaussian. The differences in error rates were small on this dataset and even Linear Regression worked well.

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lin Reg	106.047(±0.215)	141.854(±0.192)	105.788(±0.738)	141.617(±0.757)
ℓ_2	14.453(±0.258)	19.487(±0.116)	31.029(±0.258)	48.205(±0.545)
ℓ_2+Noise	20.577(±0.270)	28.604(±0.325)	28.486(±0.278)	44.777(±0.747)
ℓ_2+Clip	13.221(±0.676)	18.011(±0.783)	29.404(±0.188)	46.309(±0.403)
ℓ_1	12.764(±0.250)	21.456(±0.356)	27.550(±0.258)	44.717(±0.674)
MDN	15.406(±0.192)	27.593(±0.348)	26.268(±0.342)	43.679(±0.605)
ℓ_2+softmax	11.991(±0.277)	16.799(±0.445)	27.827(±0.253)	45.825(±0.431)
HL-OneBin	15.822(±0.198)	26.065(±0.335)	26.689(±0.280)	45.150(±0.857)
HL-Gauss.	14.335(±0.152)	23.178(±0.309)	25.525(±0.331)	43.671(±0.796)

Table 4.4: Overall results on Bike Sharing. HL-Gaussian had the lowest Test MAE on this dataset and ℓ_2 performed the worst among neural network methods.

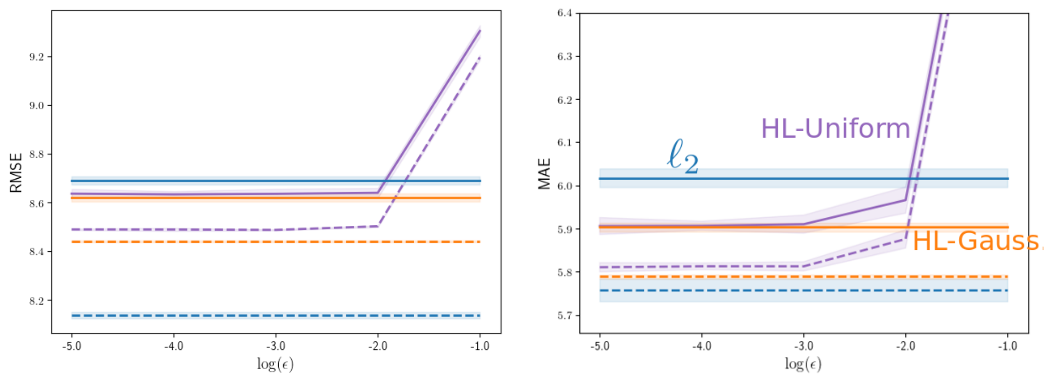
Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lin Reg	26.523(±0.018)	30.424(±0.010)	26.662(±0.066)	30.567(±0.038)
ℓ_2	0.767(±0.016)	1.441(±0.026)	1.131(±0.024)	2.579(±0.073)
ℓ_2+Noise	0.700(±0.031)	1.406(±0.039)	1.062(±0.027)	2.529(±0.054)
ℓ_2+Clip	0.714(±0.018)	1.424(±0.024)	1.069(±0.017)	2.551(±0.103)
ℓ_1	0.637(±0.013)	1.661(±0.029)	0.938(±0.029)	2.584(±0.034)
MDN	0.648(±0.026)	1.712(±0.078)	0.895(±0.013)	2.480(±0.095)
ℓ_2+softmax	0.634(±0.017)	1.445(±0.019)	0.950(±0.016)	2.481(±0.055)
HL-OneBin	0.899(±0.015)	1.442(±0.033)	1.264(±0.019)	2.760(±0.116)
HL-Gauss.	0.347(±0.018)	1.299(±0.049)	0.714(±0.024)	2.673(±0.141)

Table 4.5: Overall results on Pole. HL-Gaussian achieved the lowest Test MAE. The other methods did not achieve a Test MAE close to that of HL-Gaussian, and there was a noticeable gap between the performance of HL-OneBin and HL-Gaussian.



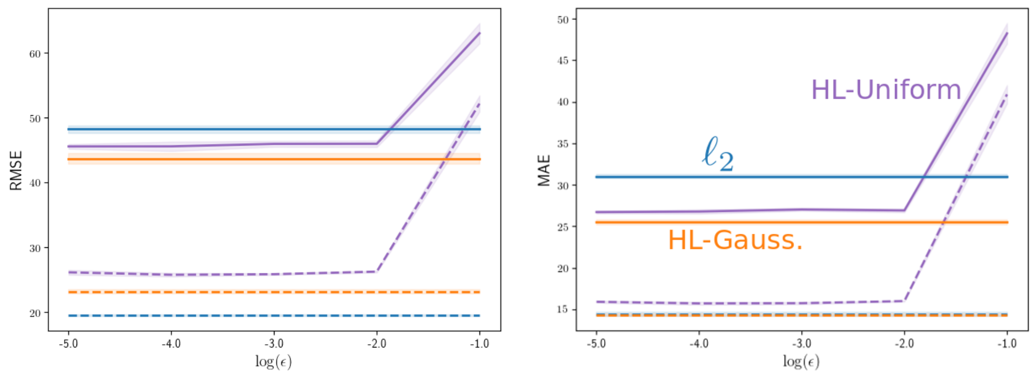
(a) RMSE (b) MAE

Figure 4.2: HL-Uniform results on CT Scan. Dotted and solid lines show train and test errors respectively. The parameter ϵ is the weighting on the uniform distribution and raising it only impaired performance.



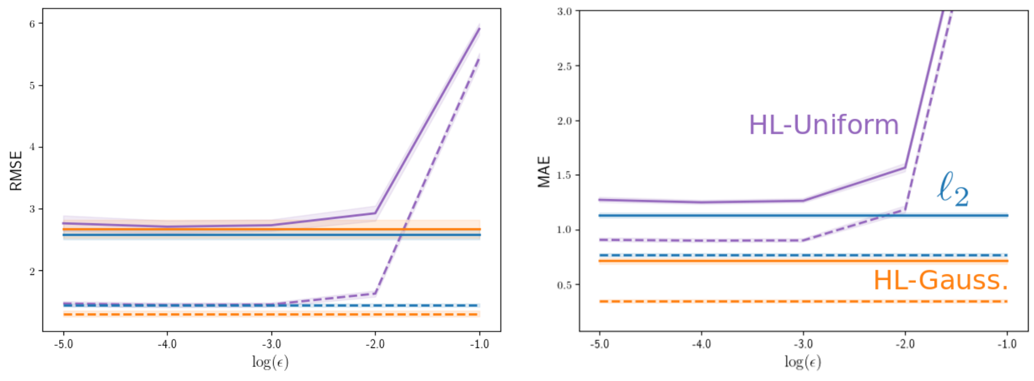
(a) RMSE (b) MAE

Figure 4.3: HL-Uniform results on Song Year. Dotted and solid lines show train and test errors respectively. The parameter ϵ is the weighting on the uniform distribution and raising it only impaired performance.



(a) RMSE (b) MAE

Figure 4.4: HL-Uniform results on Bike Sharing. Dotted and solid lines show train and test errors respectively. The parameter ϵ is the weighting on the uniform distribution and raising it only impaired performance.



(a) RMSE (b) MAE

Figure 4.5: HL-Uniform results on Pole. Dotted and solid lines show train and test errors respectively. The parameter ϵ is the weighting on the uniform distribution and raising it only impaired performance.

The overall conclusions are that the HL-Gaussian does not harm performance and can often considerably improve performance over alternatives.

Another observation is that modeling the output distribution is not the reason behind HL-Gaussian's performance. This is made clear by comparing HL-Gaussian 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-Gaussian. 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 as an implicit approach to minimizing divergences to distributions. We therefore also compared to directly modifying the labels and gradients, with ℓ_2-Noise and ℓ_2-Clipping. These model do perform slightly better than Regression for some settings, but do not achieve the same gains as HL-Gaussian. The conclusion is that HL-Gaussian's performance cannot be attributed to data augmentation in the targets.

A well-known weakness of the ℓ_2 loss is its sensitivity to outliers. To find if the difference between the performance of the ℓ_2 loss and HL-Gaussian can be explained by the presence of outliers in the dataset, we can compare HL-Gaussian with the ℓ_1 loss which is robust to outliers. The model trained with ℓ_1 still underperformed HL-Gaussian on three datasets, which suggests that the gap between the performance of HL-Gaussian and the ℓ_2 loss is not merely due to the presence of outliers in the dataset. Further experiments with other robust losses can provide a clearer conclusion.

Choices of target distribution other than a Gaussian distribution are not effective. HL-OneBin and HL-Uniform appear to have less advantages, as shown in the results, and both can actually do worse than ℓ_2. A uniform target distribution on all the datasets only worsened the error. 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.

Finally, ℓ_2+softmax performed close to HL-Gaussian on the CT Position dataset 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 nonlinearity in the output layer like HL-Gaussian. The comparison between ℓ_2+softmax and HL-Gaussian shows that this softmax nonlinearity appears to be beneficial but it cannot totally explain HL-Gaussian's performance.

The comparisons in this section address several questions regarding HL-Gaussian's performance. HL-Gaussian can often work better than ℓ_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 dataset, or employing a softmax nonlinearity in the model.

4.5 Bias and the Choice of Target Distribution

Characterizing the discretization bias in Chapter 3 showed a link between the bias of the loss and the choice of σ in HL-Gaussian. This parameter can also affect the bias from truncation as it controls the probability in the tails of the target distribution that exceeds 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 σ result in different levels of bias. Then, we compare the target distributions of HL-Gaussian and HL-OneBin with another target distribution which reduces the bias to zero. Finally, we evaluate HL-OneBin and HL-Gaussian using different parameter values to find if these parameters control a bias-variance trade-off.

Weightings in HL-Gaussian (which represent the area under of the target distribution's pdf in the range of each bin) are computed before the training, and the histogram is trained to mimic these weightings. To obtain a low training error, 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 datasets and the mean of the histograms that perfectly match weightings obtained from those targets. Figure 4.6 shows this error for HL-Gaussian with 100 bins and different choices of σ.

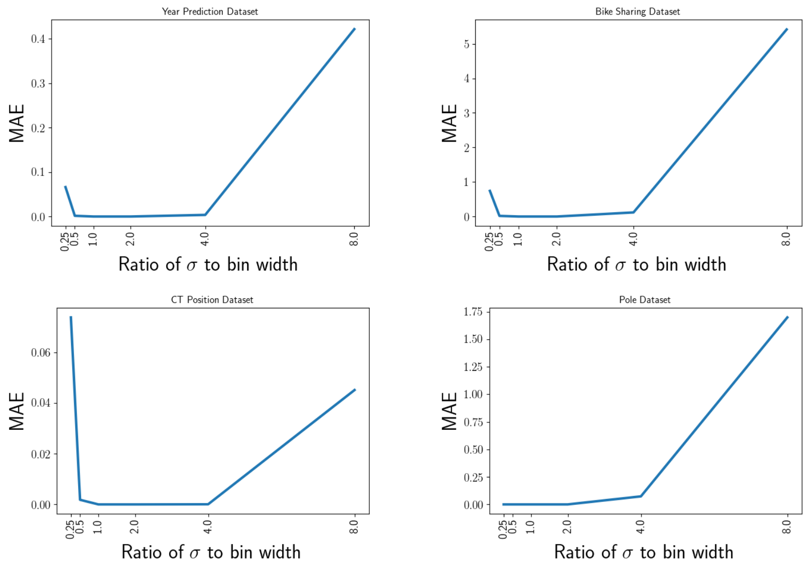


Figure 4.6: MAE between the means of HL targets and the original labels. It can be seen that extreme values of σ on either side biased the mean of the target distributions.

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-Gaussian'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-Gaussian in the ordinary setting of 100 bins. As we showed in Chapter 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.[1] The expected value

[1]This is close to the projection operator by Bellemare, Dabney, *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.

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-Gaussian on the four datasets.

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	0.309(±0.000)	0.364(±0.006)	0.335(±0.004)	0.660(±0.099)
HL-Gauss.	0.061(±0.006)	0.164(±0.090)	0.098(±0.005)	0.274(±0.090)
HL-Projected	0.053(±0.001)	0.074(±0.002)	0.103(±0.003)	0.462(±0.065)

Table 4.6: Discretization bias experiment on CT-Scan. HL-Projected achieved a Test MAE close to that of HL-Gaussian, and performed noticeably better than HL-OneBin.

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	5.810(±0.010)	8.487(±0.002)	5.906(±0.020)	8.636(±0.020)
HL-Gauss.	5.789(±0.004)	8.440(±0.004)	5.903(±0.010)	8.621(±0.016)
HL-Projected	5.815(±0.012)	8.477(±0.003)	5.917(±0.019)	8.629(±0.016)

Table 4.7: Discretization bias experiment on Song Year. While the Test MAE of HL-Gaussian was slightly lower than the other methods on average, there was no substantial difference between the performance of the the three methods.

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	15.822(±0.198)	26.065(±0.335)	26.689(±0.280)	45.150(±0.857)
HL-Gauss.	14.335(±0.152)	23.178(±0.309)	25.525(±0.331)	43.671(±0.796)
HL-Projected	14.885(±0.169)	24.329(±0.329)	26.180(±0.348)	44.982(±0.845)

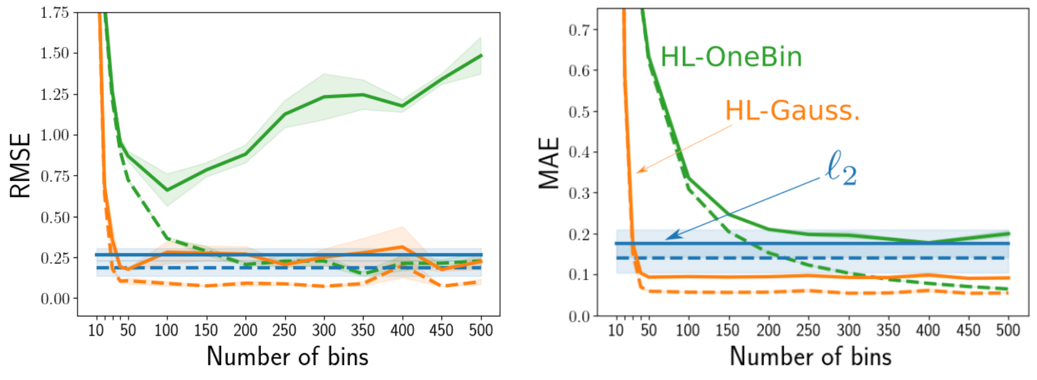
Table 4.8: Discretization bias experiment on Bike Sharing. HL-Projected performed better than HL-OneBin and worse than HL-Gaussian. We conducted a paired t-test on individual runs and found the difference between the Test MAE of HL-Projected and HL-Gaussian significant ($p < 0.05$).

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
HL-OneBin	0.899(±0.015)	1.442(±0.033)	1.264(±0.019)	2.760(±0.116)
HL-Gauss.	0.347(±0.018)	1.299(±0.049)	0.714(±0.024)	2.673(±0.141)
HL-Projected	0.364(±0.020)	1.340(±0.055)	0.741(±0.018)	2.753(±0.068)

Table 4.9: Discretization bias experiment on Pole. There is a noticeable difference between the error rates of HL-Projected and HL-OneBin. We conducted a paired t-test on individual runs and found the difference between the Test MAE of HL-Projected and HL-Gaussian significant ($p < 0.05$).

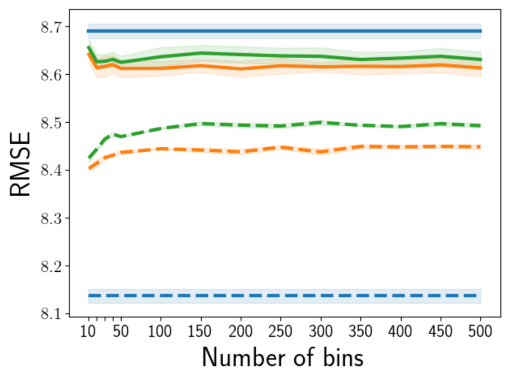
The comparison shows that using this target distribution instead of HL-OneBin often results in a noticeable reduction in the errors, but the performance of HL-Gaussian remains unbeaten. Average Test MAE for HL-Gaussian is generally lower than HL-Projected. Although HL-Projected removes discretization bias, it still does not have HL-Gaussian's desirable property of punishing faraway predictions more severely.

The parameters in the loss, namely the number of bins and σ, can affect the bias. A hypothesis is that a good choice of parameters for the 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 σ fixed. The second experiment studies the effect of changing σ on the performance of HL-Gaussian. Figures 4.7 to 4.10 evaluate HL-Gaussian and HL-OneBin with different numbers of bins, and Figures 4.11 to 4.14 show the effect of changing σ on HL-Gaussian.

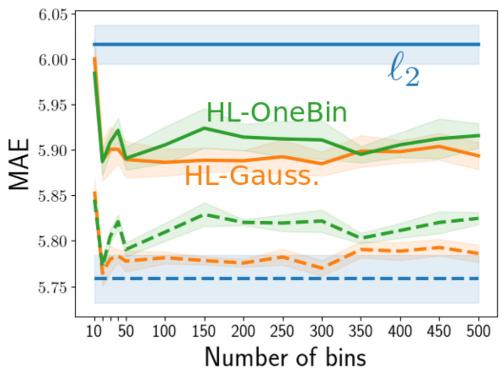


(a) RMSE (b) MAE

Figure 4.7: Changing the number of bins on CT Scan. 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.

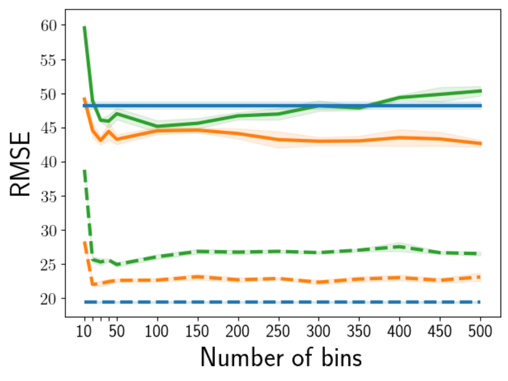


(a) RMSE

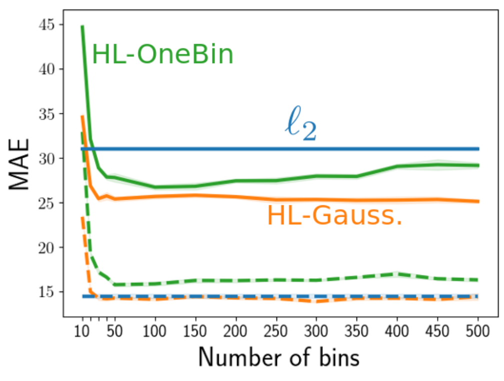


(b) MAE

Figure 4.8: Changing the number of bins on Song Year. Dotted and solid lines show train and test errors respectively. A small number of bins made the train and test MAE in both HL-Gaussian and HL-OneBin slightly higher. Increasing the number of bins did not worsen the test performance.

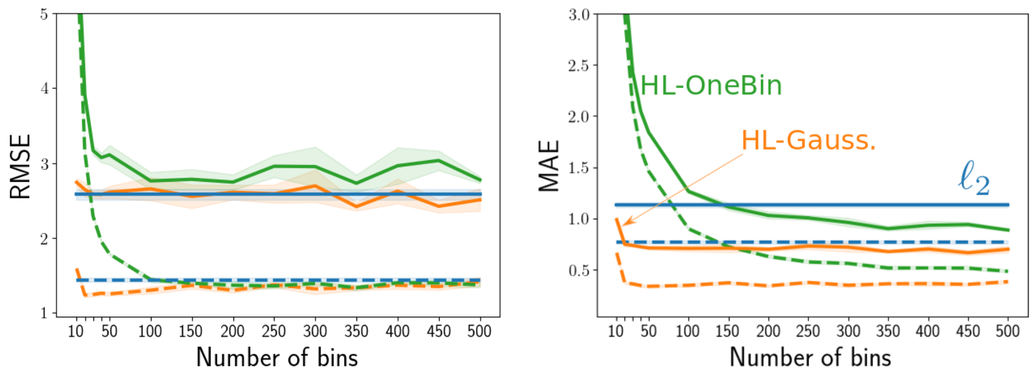


(a) RMSE



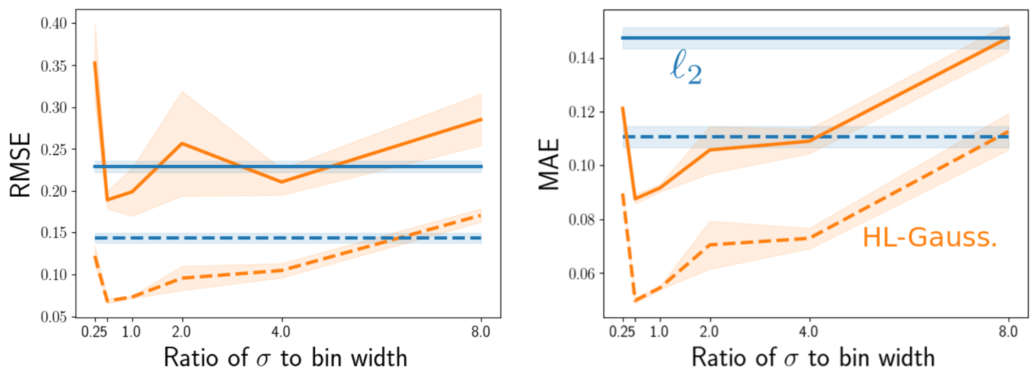
(b) MAE

Figure 4.9: Changing the number of bins on Bike Sharing. Dotted and solid lines show train and test errors respectively. Both train and test errors were high with a small number of bins. Increasing the number of bins had little affect on HL-OneBin's test erros and no effect on the performance of HL-Gaussian.



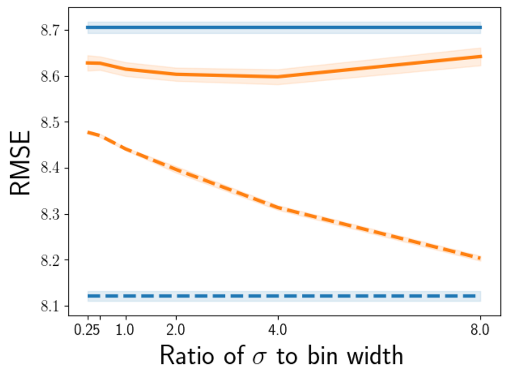
(a) RMSE (b) MAE

Figure 4.10: Changing the number of bins on Pole. Dotted and solid lines show train and test errors respectively. A small number of bins resulted in high train and test errors in both HL-Gaussian and HL-OneBin. High numbers of bins did not make the performance worse.

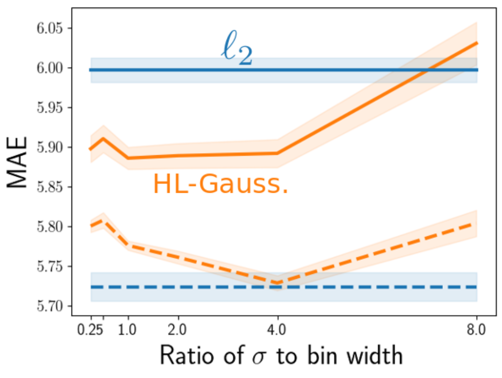


(a) RMSE (b) MAE

Figure 4.11: Changing the parameter σ on CT Scan. Dotted and solid lines show train and test errors respectively. Extreme values of σ resulted in bad performance. The error rates for train and test followed each other when changing this parameter.

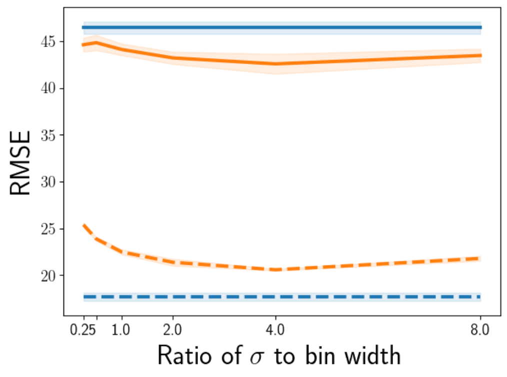


(a) RMSE

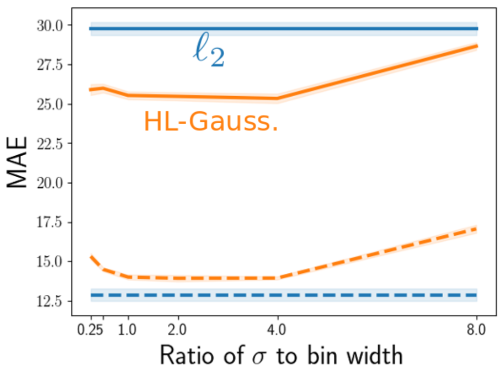


(b) MAE

Figure 4.12: Changing the parameter σ on Song Year. Dotted and solid lines show train and test errors respectively. On this dataset, with higher values of σ the gap between train and test errors kept increasing in terms of RMSE. This trend is less noticeable in Test MAE.

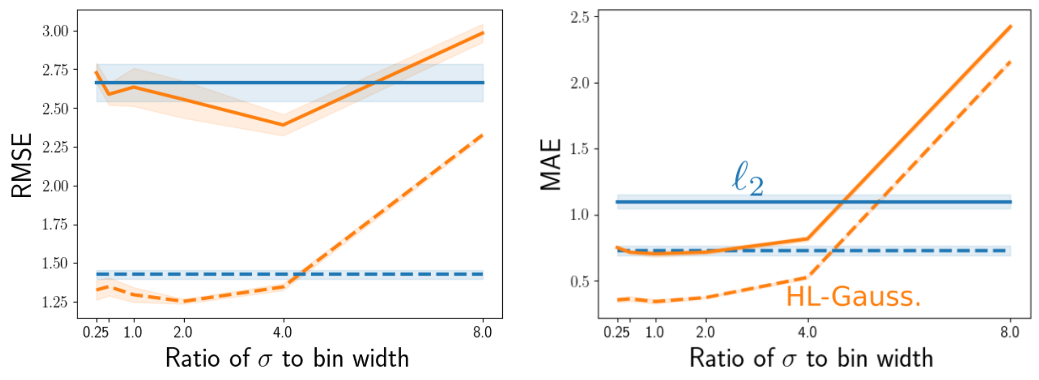


(a) RMSE



(b) MAE

Figure 4.13: Changing the parameter σ on Bike Sharing. Dotted and solid lines show train and test errors respectively. Changing σ on this dataset had little effect on the performance, and the difference between train and test errors remained constant.



(a) RMSE (b) MAE

Figure 4.14: Changing the parameter σ on Pole. Dotted and solid lines show train and test errors respectively. A high value of σ on this dataset resulted in considerably higher error rates. The difference between the train and Test MAE was similar across different values of σ. In terms of RMSE, small values of the parameter slightly increased this gap.

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-Gaussian with a higher number of bins. In the experiment on the σ parameter, we observe a v-shaped pattern previously shown by (Gao *et al.* 2017). Increasing σ at first reduces both the train and test errors and, after a point, both errors begin to rise. Target distributions with higher σ have heavier tails and these high errors can be the result of truncation. Still, if a bias-variance trade-off was present, changing this parameter would generally result in decreasing the train error while increasing the test error in regions of low-bias and high-variance. This trend rarely exists in the results.

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-Gaussian and HL-OneBin can be explained by the high bias of HL-OneBin. Both the number of bins and σ can affect the bias of the HL, but reducing this bias generally does not result in overfitting.

4.6 Representation

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 amongst 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 amongst 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-Gaussian and ℓ_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, (c) and 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 ℓ_2 and HL-Gaussian to go away when each one is trained on or initialized with the other's representation. Tables 4.10 to 4.13 show the results.

	Loss	Default	Fixed	Initialized	Random
Train MAE	ℓ_2	0.140(±0.036)	2.490(±0.094)	0.148(±0.013)	9.233(±0.157)
Train MAE	HL-Gauss.	0.061(±0.006)	0.153(±0.020)	0.063(±0.002)	2.604(±0.103)
Train RMSE	ℓ_2	0.183(±0.048)	3.465(±0.158)	0.205(±0.022)	12.247(±0.172)
Train RMSE	HL-Gauss.	0.164(±0.090)	0.219(±0.027)	0.085(±0.004)	5.756(±0.232)
Test MAE	ℓ_2	0.176(±0.034)	2.537(±0.089)	0.185(±0.013)	9.305(±0.191)
Test MAE	HL-Gauss.	0.098(±0.005)	0.187(±0.019)	0.101(±0.002)	2.727(±0.113)
Test RMSE	ℓ_2	0.267(±0.039)	3.551(±0.143)	0.299(±0.025)	12.315(±0.216)
Test RMSE	HL-Gauss.	0.274(±0.090)	0.287(±0.024)	0.183(±0.005)	6.065(±0.234)

Table 4.10: 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 the HL-Gaussian representation for ℓ_2 (first column, Fixed) caused a sudden spike in error, even though the last layer in ℓ_2 was re-trained. This suggests the representation is tuned to HL-Gaussian. 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-Gaussian cannot be said to improve the representation, HL-Gaussian still obtained substantially better performance.

	Loss	Default	Fixed	Initialized	Random
Train MAE	ℓ_2	5.758(±0.026)	6.268(±0.055)	5.743(±0.015)	8.060(±0.024)
Train MAE	HL-Gauss.	5.789(±0.004)	5.682(±0.014)	5.781(±0.010)	7.942(±0.021)
Train RMSE	ℓ_2	8.137(±0.014)	8.797(±0.030)	8.131(±0.003)	10.730(±0.016)
Train RMSE	HL-Gauss.	8.440(±0.004)	8.130(±0.010)	8.396(±0.007)	10.691(±0.026)
Test MAE	ℓ_2	6.016(±0.022)	6.338(±0.063)	5.997(±0.028)	8.069(±0.029)
Test MAE	HL-Gauss.	5.903(±0.010)	5.954(±0.011)	5.900(±0.016)	7.952(±0.017)
Test RMSE	ℓ_2	8.690(±0.015)	8.901(±0.046)	8.666(±0.021)	10.748(±0.014)
Test RMSE	HL-Gauss.	8.621(±0.016)	8.723(±0.013)	8.603(±0.013)	10.712(±0.010)

Table 4.11: Representation results on Song Year. 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. The differences were small on this datasets, but still ℓ_2 underperformed HL-Gaussian in all the three settings. Using HL-Gaussian's fixed representation only made the performance of ℓ_2 worse. As an initialization, it did not result in a considerable improvement.

	Loss	Default	Fixed	Initialized	Random
Train MAE	ℓ_2	14.453(±0.258)	39.101(±1.054)	18.214(±1.154)	106.376(±0.569)
Train MAE	HL-Gauss.	14.335(±0.152)	30.171(±0.469)	12.571(±0.250)	99.251(±1.069)
Train RMSE	ℓ_2	19.487(±0.116)	53.545(±1.264)	23.833(±1.137)	142.523(±1.052)
Train RMSE	HL-Gauss.	23.178(±0.309)	42.140(±0.664)	20.320(±0.518)	136.065(±1.283)
Test MAE	ℓ_2	31.029(±0.258)	42.834(±0.983)	31.119(±0.906)	106.429(±0.982)
Test MAE	HL-Gauss.	25.525(±0.331)	37.696(±0.360)	25.470(±0.202)	99.311(±1.486)
Test RMSE	ℓ_2	48.205(±0.545)	59.890(±0.674)	46.735(±0.387)	143.052(±1.472)
Test RMSE	HL-Gauss.	43.671(±0.796)	56.350(±0.525)	43.809(±0.593)	136.941(±1.639)

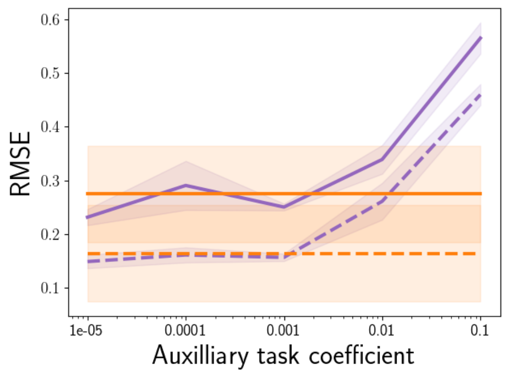
Table 4.12: Representation results on Bike Sharing. We tested (a) swapping the representations and re-learning on the last layer (**Fixed**), (b) initializing with the other's representation (**Initialized**), (c) and using the same fixed random representation for both (**Random**) and only learning the last layer. On this dataset, ℓ_2 underperformed HL-Gaussian in all the three settings. Using HL-Gaussian's representation only made the performance of ℓ_2 worse as shown in the Fixed and Initialized columns.

	Loss	Default	Fixed	Initialized	Random
Train MAE	ℓ_2	0.767(±0.016)	4.218(±0.325)	1.109(±0.024)	28.936(±0.176)
Train MAE	HL-Gauss.	0.347(±0.018)	1.218(±0.052)	0.312(±0.010)	18.428(±0.969)
Train RMSE	ℓ_2	1.441(±0.026)	6.184(±0.412)	1.836(±0.042)	33.642(±0.231)
Train RMSE	HL-Gauss.	1.299(±0.049)	2.899(±0.143)	1.212(±0.027)	26.795(±0.959)
Test MAE	ℓ_2	1.131(±0.024)	4.396(±0.329)	1.437(±0.053)	29.221(±0.254)
Test MAE	HL-Gauss.	0.714(±0.024)	1.398(±0.021)	0.645(±0.009)	18.285(±1.062)
Test RMSE	ℓ_2	2.579(±0.073)	6.423(±0.415)	2.882(±0.109)	33.944(±0.299)
Test RMSE	HL-Gauss.	2.673(±0.141)	3.441(±0.093)	2.373(±0.055)	26.621(±1.043)

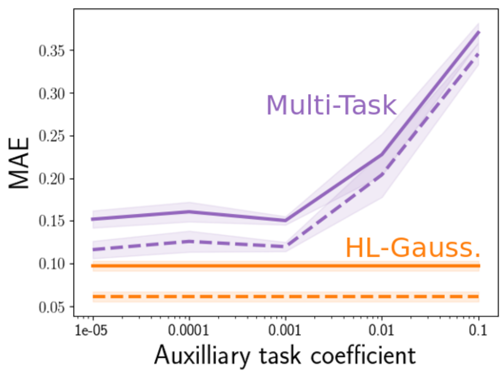
Table 4.13: Representation results on Pole. We tested (a) swapping the representations and re-learning on the last layer (**Fixed**), (b) initializing with the other's representation (**Initialized**), (c) and using the same fixed random representation for both (**Random**) and only learning the last layer. On this dataset, ℓ_2 underperformed HL-Gaussian in all the three settings. Using HL-Gaussian's representation only made the performance of ℓ_2 worse as shown in the Fixed and Initialized columns.

The results above are surprisingly conclusive. Using the representation from HL-Gaussian does not improve performance of ℓ_2, and even under a random representation, HL-Gaussian performs noticeably better than ℓ_2.

In the second experiment, we tested a network that predicted both the expected value (with the ℓ_2 loss) and the distribution (using HL-Gaussian). 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-Gaussian, a regression network with the auxiliary task of predicting the distribution is expected to achieve a similar performance. Figures 4.15 to 4.18 show the results on the four datasets.

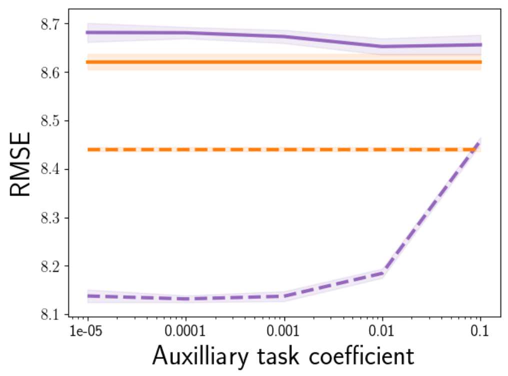


(a) RMSE

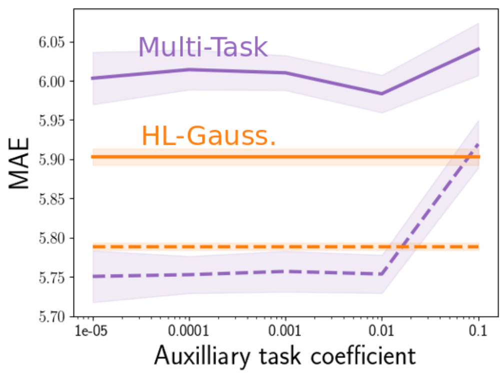


(b) MAE

Figure 4.15: 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.

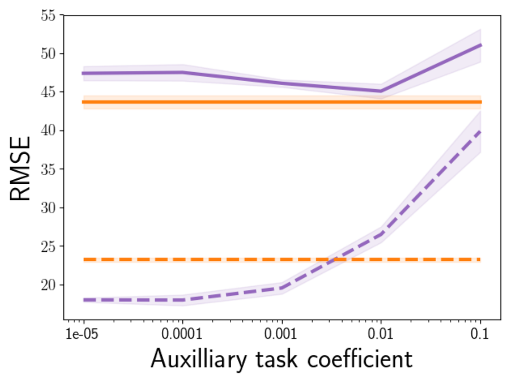


(a) RMSE

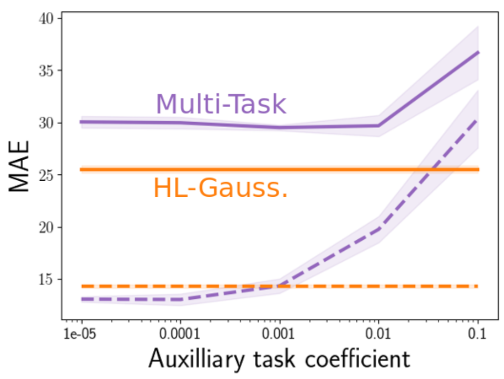


(b) MAE

Figure 4.16: Multi-Task Network results on Song Year. 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.

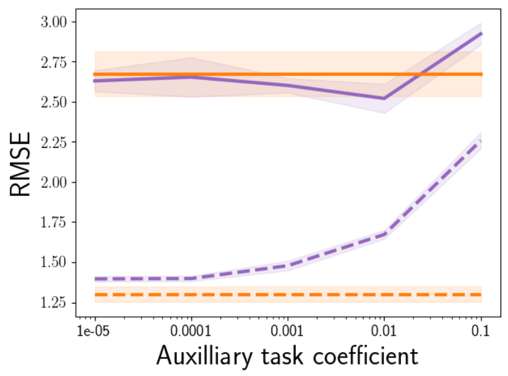


(a) RMSE

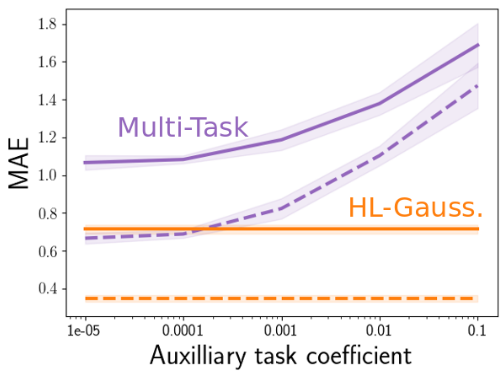


(b) MAE

Figure 4.17: Multi-Task Network results on Bike Sharing. 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.



(a) RMSE

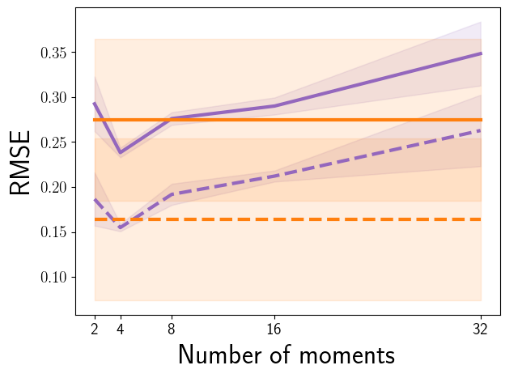


(b) MAE

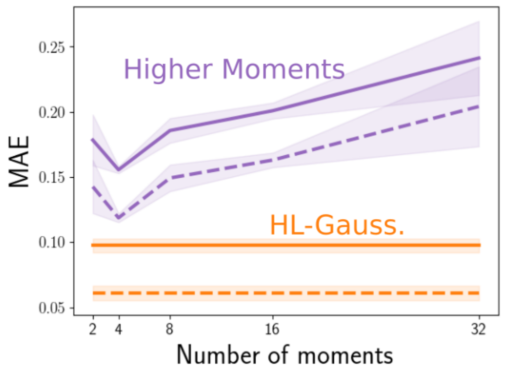
Figure 4.18: Multi-Task Network results on Pole. The loss function is the mean prediction's squared error plus the distribution prediction's KL-divergence multiplied by a coefficient. The horizontal axis shows the coefficient for KL-divergence. Dotted and solid lines show train and test errors respectively. There was not a substantial drop in the error rate when increasing the coefficient in the loss and the performance only became worse.

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-Gaussian. The conclusion is that the extra information required for predicting the distribution cannot explain the gap between the ℓ_2 loss and HL-Gaussian.

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 ℓ_2 loss and predicting the distribution with HL-Gaussian. We trained the ℓ_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 ℓ_2 loss. The results are shown in Figures 4.19 to 4.22.

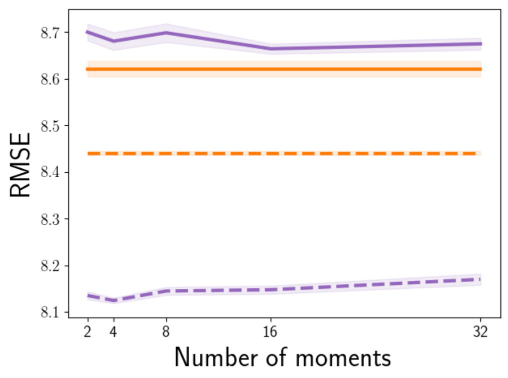


(a) RMSE

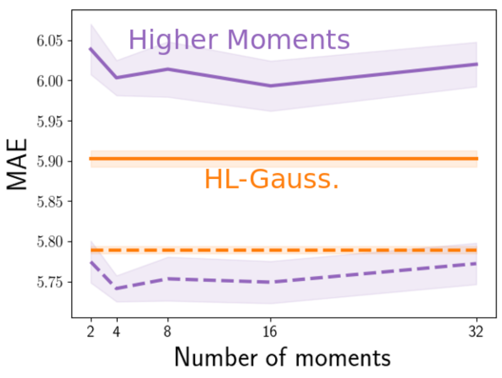


(b) MAE

Figure 4.19: Higher Moments Network results on CT Scan. The horizontal 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. With four moments, there was a slight reduction in error but the new model still performed worse than HL-Gaussian in terms of Test MAE.

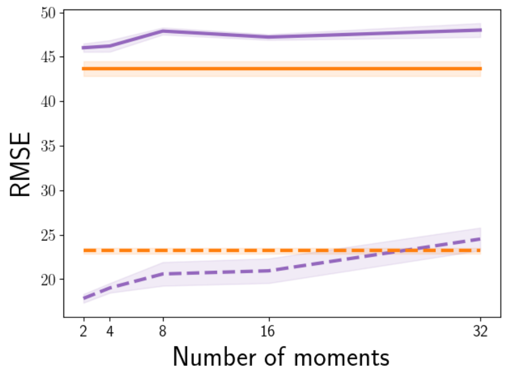


(a) RMSE

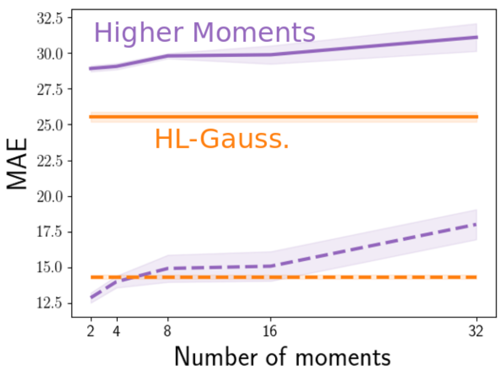


(b) MAE

Figure 4.20: Higher Moments Network results on Song Year. The horizontal 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 on this dataset. The new model consistently performed worse than HL-Gaussian.

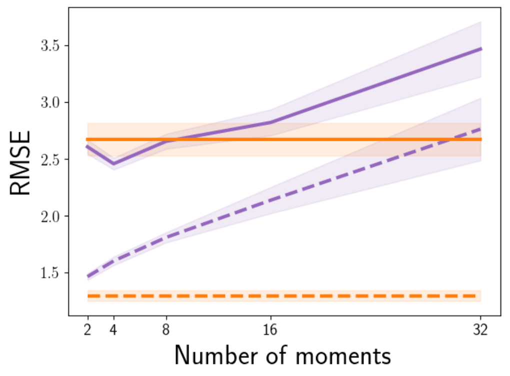


(a) RMSE

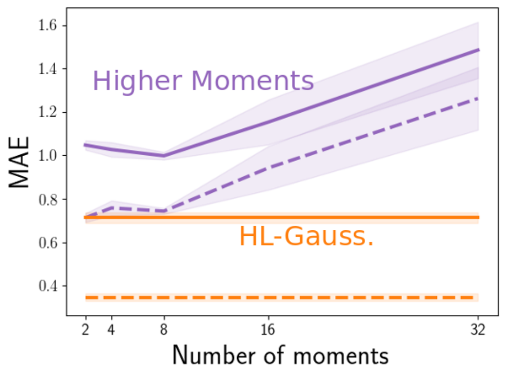


(b) MAE

Figure 4.21: Higher Moments Network results on Bike Sharing. The horizontal 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 on this dataset. The new model consistently performed worse than HL-Gaussian.



(a) RMSE



(b) MAE

Figure 4.22: Higher Moments Network results on Pole. The horizontal 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 on this dataset. There was a slight reduction in error but the new model still performed worse than HL-Gaussian in terms of Test MAE.

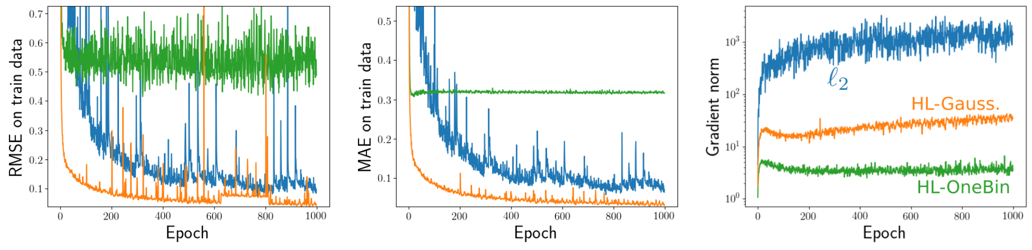
There is little gain in predicting higher moments and the new model still underperforms HL-Gaussian. Like the previous experiment, predicting this extra information about the output distribution does not appear beneficial as an auxiliary task.

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

4.7 Optimization Properties

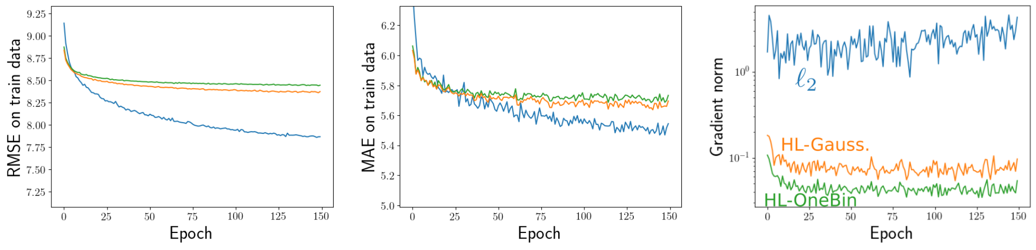
In Chapter 3 we provided a bound on the gradient norm of the HL that suggested ease of optimization with gradient descent. In this section we show two experiments that study the optimization of the ℓ_2 loss and HL empirically.

First, we recorded the training errors and gradient norms during training without dropout to see if using HL-Gaussian 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.[2] 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. Figures 4.23 to 4.26 show the results.



(a) RMSE (b) MAE (c) Gradient norm

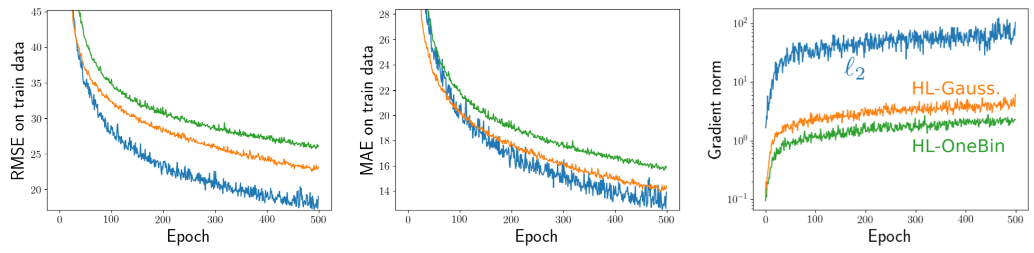
Figure 4.23: Training process on CT Scan. A logarithmic scale is used in the Y axis of the rightmost plot. HL-Gaussian reduced the train errors considerably faster than ℓ_2. The gradient norm of ℓ_2 was highly varying through the training.



(a) RMSE (b) MAE (c) Gradient norm

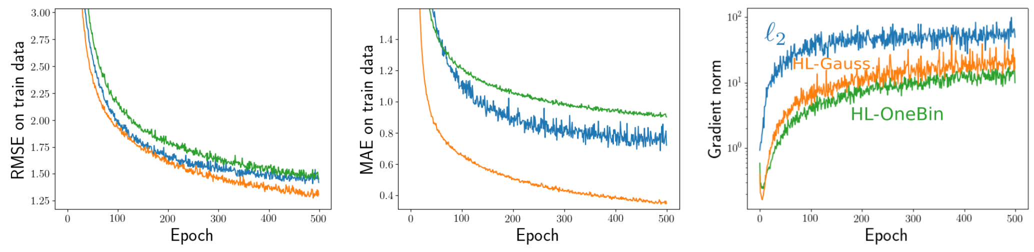
Figure 4.24: Training process results on Song Year. A logarithmic scale is used in the Y axis of the rightmost plot. HL-Gaussian reduced the train errors faster than ℓ_2 although it settled on a worse training errors at the end. The gradient norm of ℓ_2 was highly varying through the training.

[2] The minimum value of the ℓ_2 loss is zero. HL, however, is the cross-entropy to the target distribution and its minimum value is the entropy of the histogram that best estimates the target distribution.



(a) RMSE (b) MAE (c) Gradient norm

Figure 4.25: Training process results on Bike Sharing. A logarithmic scale is used in the Y axis of the rightmost plot. HL-Gaussian reduced the train errors faster than ℓ_2 early in the training and slower than ℓ_2 after around 100 epochs. The gradient norm of ℓ_2 was highly varying through the training.

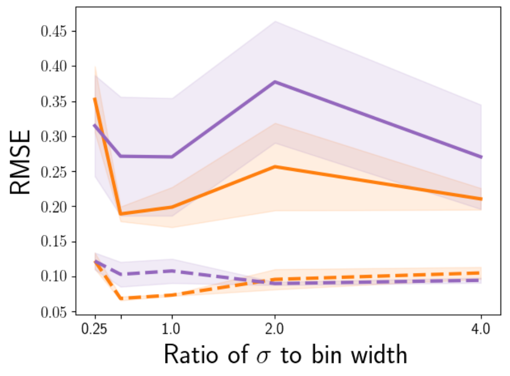


(a) RMSE (b) MAE (c) Gradient norm

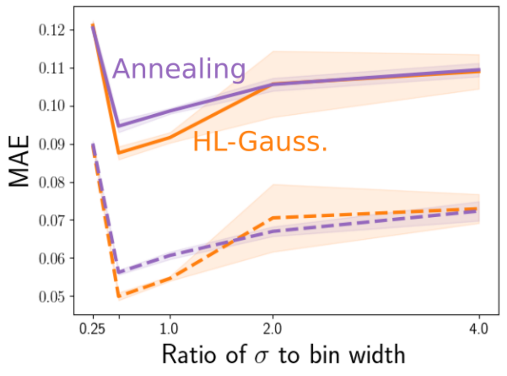
Figure 4.26: Training process results on Pole. A logarithmic scale is used in the Y axis of the rightmost plot. HL-Gaussian reduced the train errors considerably faster than ℓ_2. The gradient norm of ℓ_2 was highly varying through the training.

The results above show that HL-Gaussian shows more stable gradients than ℓ_2 and generally needs a smaller number of steps for optimization. As discussed in Chapter 3, these properties can explain the performance of HL-Gaussian. In comparison with HL-OneBin, however, HL-Gaussian does not show a noticeable benefit in optimization.

The second experiment explores the role of σ in HL-Gaussian. The benefit of the parameter σ in HL-Gaussian can be twofold: (1) improving the final solution and (2) improving optimization process. To compare the effect of these two properties, we design a network, called Annealing Network, that starts with a high σ and gradually reduces it to a small σ during the training. The initial σ is 8 times the bin width, and during the first 20% of epochs the current value of σ is multiplied by a constant τ at each epoch. The training continues without further annealing once 20% of the epochs are finished. The assumption is that the effect of σ on optimization shows up during the training, unlike the effect on final solution. Therefore, if the effect on optimization is substantial, the model that has a high σ earlier in training can benefit from it despite its small σ at the end. Results are shown in Figures 4.27 to 4.30.

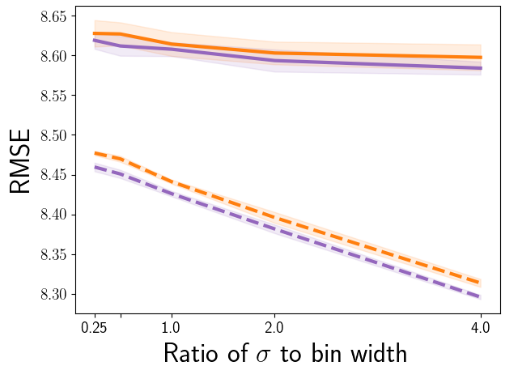


(a) RMSE

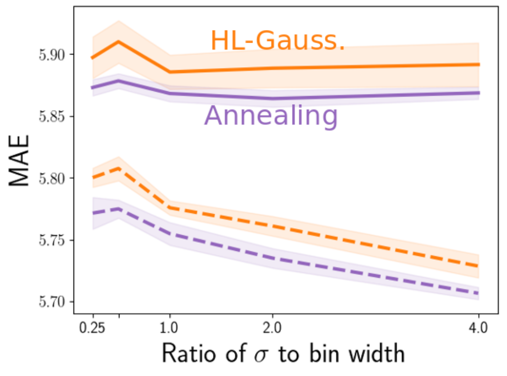


(b) MAE

Figure 4.27: Annealing results on CT Scan. Dotted and solid lines show train and test errors respectively. There was no noticeable benefit in starting with a higher σ and reducing it though the training.

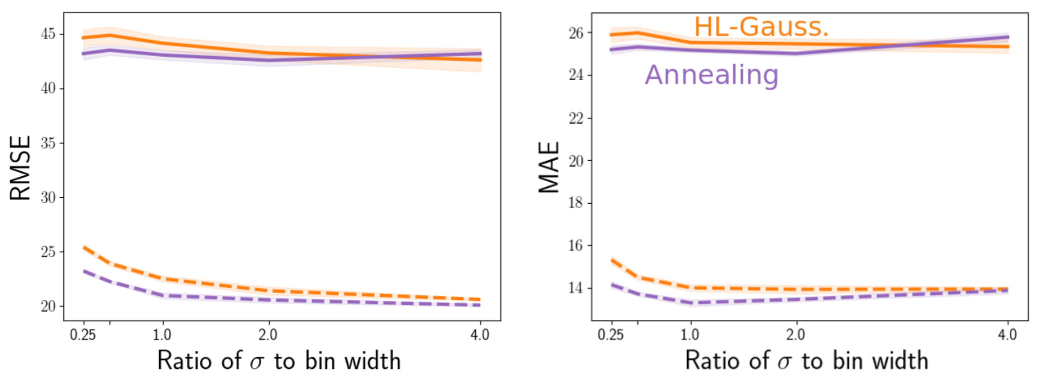


(a) RMSE



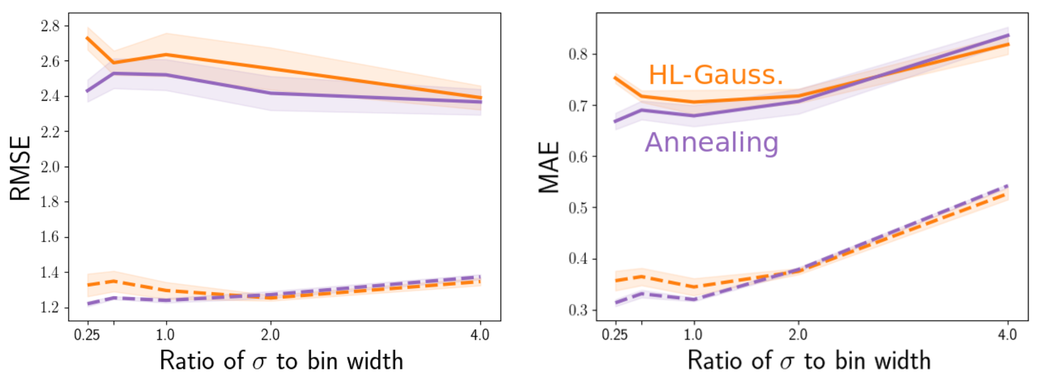
(b) MAE

Figure 4.28: Annealing results on Song Year. There was a small benefit in starting with a higher σ and reducing it though the training in terms of MAE.



(a) RMSE (b) MAE

Figure 4.29: Annealing results on Bike Sharing. There was no noticeable benefit in starting with a higher σ and reducing it though the training.



(a) RMSE (b) MAE

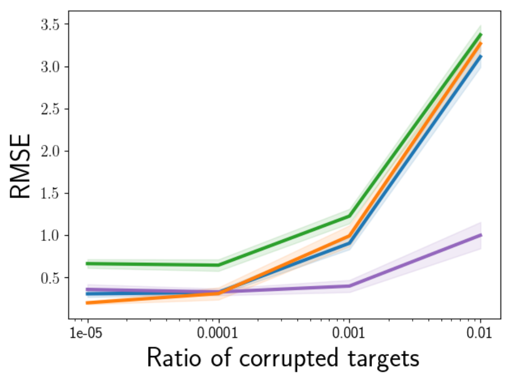
Figure 4.30: Annealing results on Pole. There was no noticeable benefit in starting with a higher σ and reducing it though the training.

Generally, there was little difference between the performance of a model that started with a small value of σ and one that reached that value through annealing. So the parameter σ does not seem to have a considerable impact on the optimization.

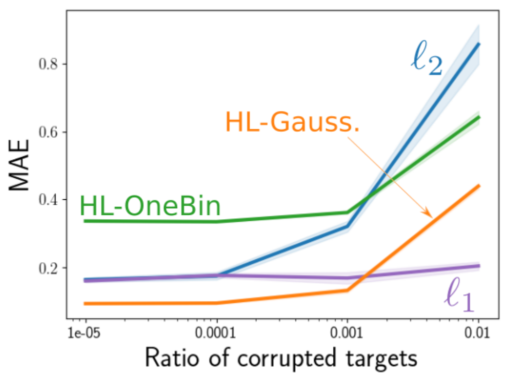
The results in this section show that using the HL instead of the ℓ_2 loss helps with optimization, which confirms the previous observation by Imani and White (2018). However, the choice of the target distribution and the parameter in the loss do not have a major effect on the optimization.

4.8 Robustness to Corrupted Targets

We tested ℓ_2 loss, ℓ_1 loss, HL-OneBin, and HL-Gaussian on datasets with corrupted targets. At each level, we replaced a ratio of training targets with numbers sampled uniformly from the range of targets in the dataset. Results in Figures 4.31 to 4.34 show that HL-Gaussian and HL-OneBin are more robust to corrupted targets than ℓ_2 but not as robust as ℓ_1.

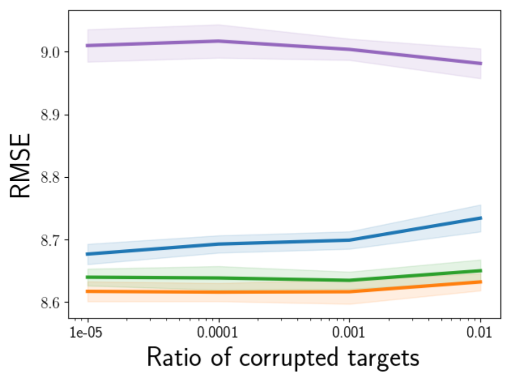


(a) RMSE

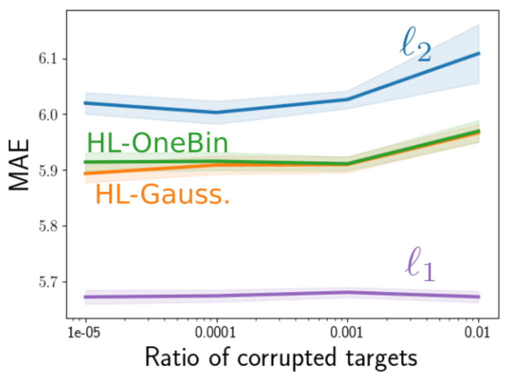


(b) MAE

Figure 4.31: Corrupted targets on CT Scan. The lines show test errors. The ℓ_1 loss was robust to corrupted targets, and the ℓ_2 loss was affected the most.

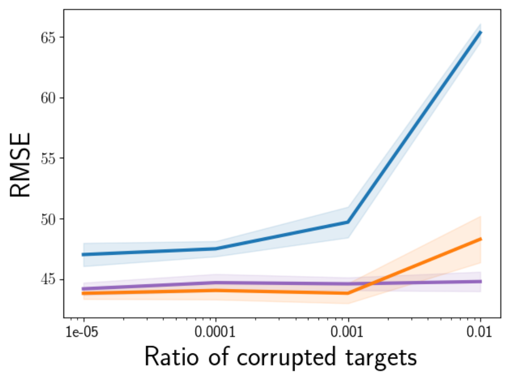


(a) RMSE

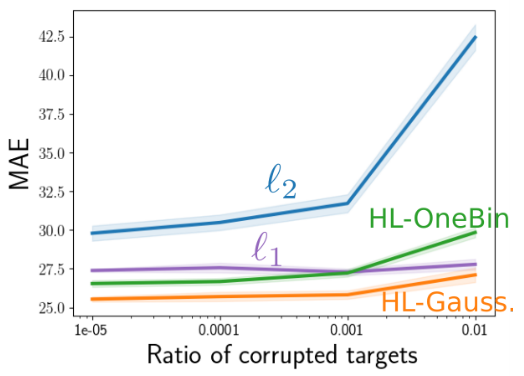


(b) MAE

Figure 4.32: Corrupted targets on Song Year. The ℓ_1 loss was robust to corrupted targets, and the ℓ_2 loss was affected the most.

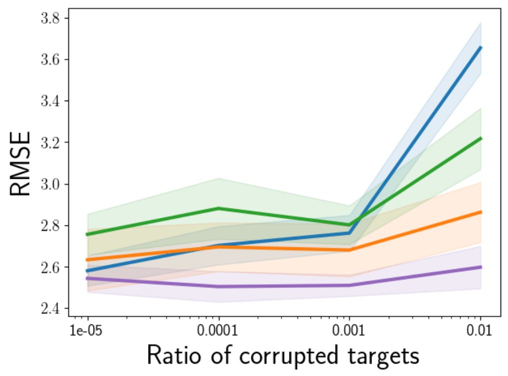


(a) RMSE

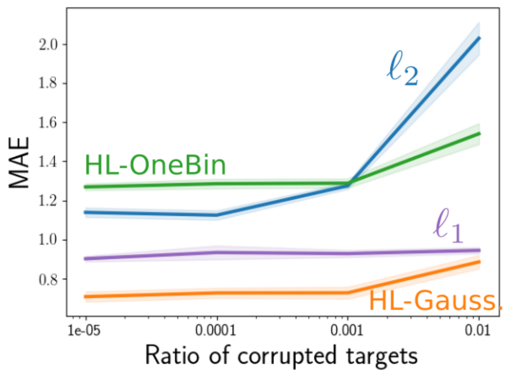


(b) MAE

Figure 4.33: Corrupted targets on Bike Sharing. The ℓ_1 loss was robust to corrupted targets, and the ℓ_2 loss was affected the most.



(a) RMSE



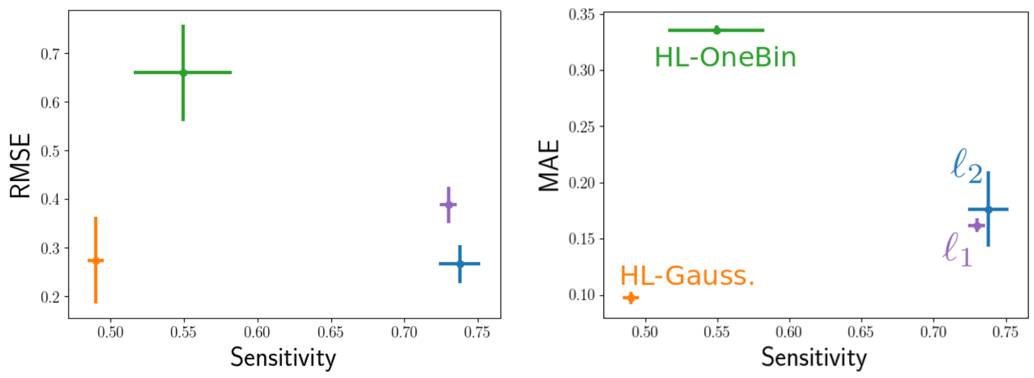
(b) MAE

Figure 4.34: Corrupted targets on Pole. The ℓ_1 loss was robust to corrupted targets, and the ℓ_2 loss was affected the most.

These results show a situation where the difference between the performance of the HL and the ℓ_2 loss is more pronounced. Interestingly, the HL can work worse than the ℓ_1 loss in the presence of corrupted targets. This analysis is exploratory, and further studies on the effect of corrupted targets in training can lead to a better understanding the differences between these loss functions.

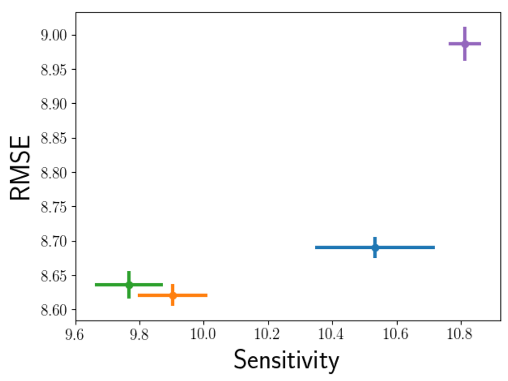
4.9 Sensitivity to Input Perturbations

We compared the sensitivity of the outputs of models trained with ℓ_2 loss, ℓ_1 loss, HL-OneBin, and HL-Gaussian to input perturbations. The measure we used was the Frobenius norm of the Jacobian of the model's output w.r.t. the input. Predictions in a regression problem are scalar values, so the Jacobian becomes a vector whose elements are the derivative of the output w.r.t. 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.* n.d.). 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. Figures 4.35 to 4.38 summarize the results.

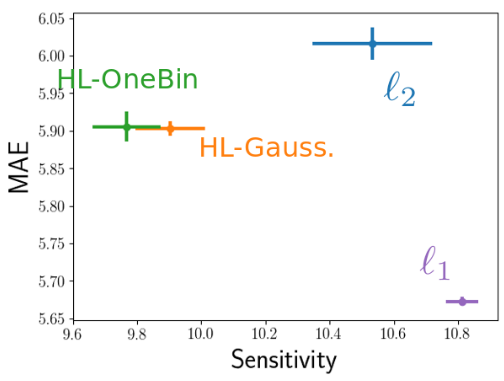


(a) RMSE (b) MAE

Figure 4.35: 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-Gaussian showed less sensitivity and lower test errors than ℓ_1 and ℓ_2.

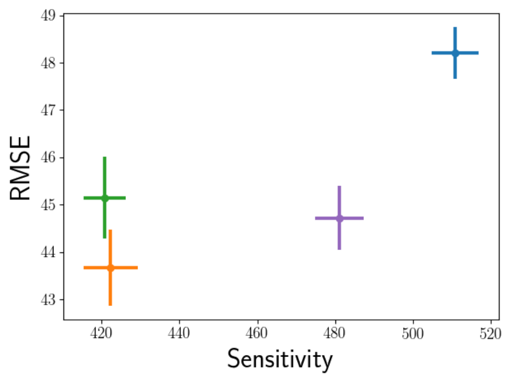


(a) RMSE

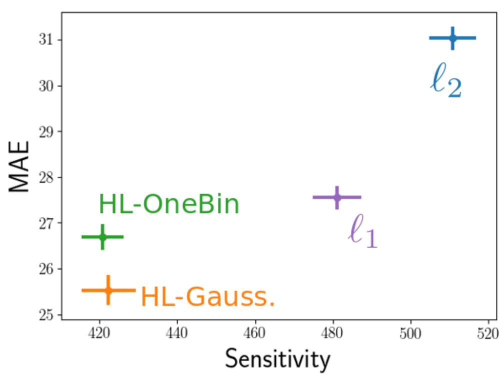


(b) MAE

Figure 4.36: Sensitivity results on Song Year. 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-Gaussian showed less sensitivity than ℓ_1 and ℓ_2, while it had a higher Test MAE than ℓ_1.

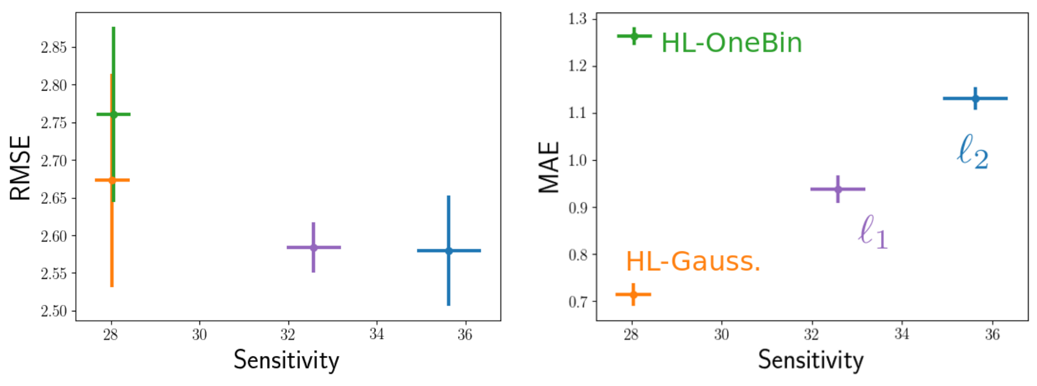


(a) RMSE



(b) MAE

Figure 4.37: Sensitivity results on Bike Sharing. 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-Gaussian showed less sensitivity and lower test errors than ℓ_1 and ℓ_2.



(a) RMSE (b) MAE

Figure 4.38: Sensitivity results on Pole. 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-Gaussian showed less sensitivity and lower test errors than ℓ_1 and ℓ_2.

The results show a recurring pattern that predictions of the models that are trained with HL are less sensitive to input perturbations than the ones trained with ℓ_1 or ℓ_2. Further analysis is required to know why this difference in sensitivity happens and to what extent it impacts the performance in a regression setting.

4.10 Summary of Empirical Study

Through this section, we tested several hypotheses about the performance of the HL. We found that

1. HL-Gaussian is a viable choice in regression and can often outperform the ℓ_2 loss,
2. learning a flexible distribution is not why HL-Gaussian outperforms the ℓ_2 loss,
3. data augmentation in the labels cannot explain the difference between the performance of HL-Gaussian and the ℓ_2 loss,
4. robustness to outliers is not the reason why HL-Gaussian performs better,

5. the softmax nonlinearity in HL-Gaussian is beneficial but it cannot wholly explain its superior performance,
6. the choice of target distribution affects the bias of the loss and HL-OneBin can suffer from a high bias,
7. there is not a considerable bias-variance trade-off controlled by the number of bins and the σ parameter,
8. predicting the full distribution does not force HL-Gaussian's model to learn a better representation,
9. using HL-Gaussian instead of the ℓ_2 loss helps gradient descent's search for a good solution,
10. the HL is more robust to corrupted targets in the dataset than the ℓ_2 loss and less robust than the ℓ_1 loss, and
11. the HL finds a model whose output is less sensitive to input perturbations than both the ℓ_2 loss and the ℓ_1 loss.

Chapter 5

Conclusion

This chapter summarizes the contributions in the thesis and highlights some future directions for research on this topic.

5.1 Contributions

Our main contribution is proposing a new loss for regression, called the Histogram Loss, which consists of the KL-divergence between a flexible histogram prediction and target distributions. Experiments on four datasets showed that this loss function does not harm performance and, in fact, it often outperforms the oft-used ℓ_2 loss without a need for careful hyper-parameter tuning.

Besides providing a solution for regression, the Histogram Loss offers a testbed for analyzing some of the open questions in previous works. A question that we explored in depth was one of the hypotheses put forward by Bellemare, Dabney, *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 the 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 the HL's performance is the softmax nonlinearity that models the histogram distribution as we found that a minimizing the ℓ_2 loss between the target and the mean of a histogram distribution could yield an error rate close to that of the HL.

Another question was the impact of label distribution on performance, previously mentioned in Deep Label Disribution Learning (Gao *et al.* 2017). They found that, in age estimation, their particular choice of label distribution can result in a considerably lower error compared to both one-hot targets of DEX and label smoothing. We pointed out the similarity between the label distribution of DLDL and a special case of HL-Gaussian in Chapter 2, and later revealed an undiscussed factor that can explain most of this gap. The choice of label 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 the HL with an unbiased target distribution underperformed HL-Gaussian.

5.2 Future Work

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 data or model (Choromanska *et al.* 2015; Soltanolkotabi *et al.* 2018). There is an alternative line of research that compares loss surfaces empirically by techniques like projection on lower dimensions or slightly perturbing the parameters in random directions (Ahmed *et al.* 2018; Li *et al.* 2018). There are two important challenges in performing such comparison. First, intuitions about optimization in low-dimensional surfaces do not necessarily transform to higher dimensions. Second, sometimes (as in our setting) the surfaces compared are of different dimensionalities, and this difference can be a confounding factor.

The experiment settings and baselines in this thesis were focused on singling out and studying the points of difference between the HL and the ℓ_2 loss. Further experiments can be designed to evaluate intersections of these factors. As a simple example, one could switch the ℓ_2 loss to the ℓ_1 loss in our

ℓ_2+softmax baseline. The new model can benefit from robustness to outliers and the nonlinearity in the softmax output layer.

Another topic for future work considers the hyper-parameters of the HL. Most of our experiments are done with the default settings of 100 bins and a σ parameter equal to the bin width. Exploratory results on the effect of these parameters showed that higher performance can be achieved by tuning them. An interesting future work would be finding problem-specific or even adaptive values for the hyper-parameters in the loss. Two other unexplored directions are the support of the histogram and the discretization method. We chose the range of targets in the dataset plus a padding of 10 bins on each side to realize our assumption that the target distribution has negligible probability beyond the support of the histogram, and discretized the range evenly. This choice might not be reasonable in some cases. For example, consider the task of predicting the number of views of an online video. Presence of viral videos that heavily stretch the range of targets in the dataset 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, and (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. Finding a reasonable support and the best way to descretize it are left for future work.

References

[1] Z. Ahmed, N. L. Roux, M. Norouzi, and D. Schuurmans, “Understanding the impact of entropy in policy learning,” *arXiv preprint arXiv:1811.11214*, 2018. 63

[2] Anonymous, “Visualizing the loss landscape of neural nets,” *International Conference on Learning Representations*, 2018. [Online]. Available: `https://openreview.net/forum?id=HkmaTz-0W`.

[3] L. J. Ba and R. Caruana, “Do Deep Nets Really Need to be Deep?” In *Advances in Neural Information Processing Systems*, 2013.

[4] J. T. Barron, “A More General Robust Loss Function.,” *arXiv*, 2017.

[5] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, A. Muldal, N. Heess, and T. Lillicrap, “Distributed distributional deterministic policy gradients,” *arXiv preprint arXiv:1804.08617*, 2018. 2

[6] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab, “Robust optimization for deep regression,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2830–2838.

[7] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” *arXiv preprint arXiv:1707.06887*, 2017. 2, 27, 37, 62

[8] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, “The cramer distance as a solution to biased wasserstein gradients,” *arXiv preprint arXiv:1705.10743*, 2017.

[9] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset.,” in *Ismir*, vol. 2, 2011, p. 10. 28

[10] C. M. Bishop, “Mixture density networks,” *Technical Report*, 1994. 30

[11] R. Caruana, “Multitask learning,” in *Learning to learn*, Springer, 1998, pp. 95–133.

[12] F. Chollet *et al.*, *Keras*, `https://github.com/fchollet/keras`, 2015. 30

[13] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in *Artificial Intelligence and Statistics*, 2015, pp. 192–204. 63

[14] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,” in *Advances in neural information processing systems*, 2014, pp. 2933–2941.

[15] K. De Brabanter, K. Pelckmans, J. De Brabanter, M. Debruyne, J. A. K. Suykens, M. Hubert, and B. De Moor, “Robustness of Kernel Based Regression: A Comparison of Iterative Weighting Schemes,” in *International Conference on Artificial Neural Networks*, 2009.

[16] A. Dubey, O. Gupta, R. Raskar, I. Rahwan, and N. Naik, “Regularizing prediction entropy enhances deep learning with limited data,”

[17] H. Fanaee-T and J. Gama, “Event labeling combining ensemble detectors and background knowledge,” *Progress in Artificial Intelligence*, vol. 2, no. 2-3, pp. 113–127, 2014. 28

[18] C. D. Freeman and J. Bruna, “Topology and geometry of half-rectified network optimization,” *arXiv preprint arXiv:1611.01540*, 2016.

[19] B.-B. Gao, C. Xing, C.-W. Xie, J. Wu, and X. Geng, “Deep label distribution learning with label ambiguity,” *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2825–2838, 2017. 4, 7, 10, 43, 63

[20] X. Geng, “Label distribution learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1734–1748, 2016. 4

[21] A. Ghosh, H. Kumar, and P. Sastry, “Robust loss functions under label noise for deep neural networks.,” in *AAAI Conference on Artificial Intelligence*, 2017.

[22] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[23] F. Graf, H.-P. Kriegel, M. Schubert, S. Pölsterl, and A. Cavallaro, “2d image registration in ct images using radial image descriptors,” *Medical Image Computing and Computer-Assisted Intervention*, pp. 607–614, 2011. 28

[24] A. Gruslys, M. G. Azar, M. G. Bellemare, and R. Munos, “The reactor: A sample-efficient actor-critic architecture,” *arXiv preprint arXiv:1704.04651*, 2017. 2

[25] M. Hardt, B. Recht, and Y. Singer, “Train faster, generalize better: Stability of stochastic gradient descent,” *arXiv preprint arXiv:1509.01240*, 2015. 14, 16

[26] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, “Rainbow: Combining improvements in deep reinforcement learning,” *arXiv preprint arXiv:1710.02298*, 2017. 2

[27] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015. 3, 4

[28] P. J. Huber, "Robust statistics," in *International Encyclopedia of Statistical Science*, Springer, 2011, pp. 1248–1251.

[29] E. Imani and M. White, "Improving regression performance with distributional losses," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018, pp. 2162–2171. [Online]. Available: `http://proceedings.mlr.press/v80/imani18a.html`. iii, 27, 55

[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 30

[31] D. E. Knuth, *The Art of Computer Programming, Volume I: Fundamental Algorithms*. Addison-Wesley, 1968.

[32] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6405–6416. 30

[33] J. Langford, R. Oliveira, and B. Zadrozny, "Predicting Conditional Quantiles via Reduction to Classification.," in *Conference on Uncertainty in Artificial Intelligence*, 2006.

[34] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, Springer, 1998, pp. 9–50. 30

[35] S. Lee, S. P. S. Prakash, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra, "Stochastic multiple choice learning for training diverse deep ensembles," in *Advances in Neural Information Processing Systems*, 2016, pp. 2119–2127.

[36] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Advances in Neural Information Processing Systems*, 2018, pp. 6389–6399. 63

[37] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing by virtual adversarial examples," in *International Conference on Learning Representations*, 2016.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[39] T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka, "Nonparametric return density estimation for reinforcement learning," in *27th international conference on machine learning (ICML)*, 2010, pp. 21–25. 2

[40] T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka, "Parametric return density estimation for reinforcement learning," *arXiv preprint arXiv:1203.3497*, 2012. 2

[41] M. Norouzi, S. Bengio, N. Jaitly, M. Schuster, Y. Wu, D. Schuurmans, *et al.*, "Reward augmented maximum likelihood for neural structured prediction," in *Advances In Neural Information Processing Systems*, 2016, pp. 1723–1731. 7, 9–11, 13, 17, 18

[42] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: An empirical study," *arXiv preprint arXiv:1802.08760*, 2018. 58

[43] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "Pmlb: A large benchmark suite for machine learning evaluation and comparison," *BioData Mining*, vol. 10, no. 1, p. 36, Dec. 2017, ISSN: 1756-0381. DOI: `10.1186/s13040-017-0154-4`. [Online]. Available: `https://doi.org/10.1186/s13040-017-0154-4`. 28

[44] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[45] 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*, vol. 12, pp. 2825–2830, 2011. 30

[46] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017. 10

[47] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, Omnipress, 2011, pp. 833–840. 58

[48] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for Thin Deep Nets," in *International Conference on Learning Representations*, 2014.

[49] R. Rothe, R. Timofte, and L. Van Gool, "Dex: Deep expectation of apparent age from a single image," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 10–15. 2

[50] ——, "Deep expectation of real and apparent age from a single image without facial landmarks," *International Journal of Computer Vision*, vol. 126, no. 2, pp. 144–157, 2018. 10

[51] M. Rowland, M. G. Bellemare, W. Dabney, R. Munos, and Y. W. Teh, "An analysis of categorical distributional reinforcement learning," *arXiv preprint arXiv:1802.08163*, 2018. 2, 37

[52] C. Rupprecht, I. Laina, M. Baust, F. Tombari, G. D. Hager, and N. Navab, “Learning in an uncertain world: Representing ambiguity through multiple hypotheses,” *arXiv preprint arXiv:1612.00197*, 2016.

[53] W. Shen, Y. Guo, Y. Wang, K. Zhao, B. Wang, and A. Yuille, “Deep regression forests for age estimation,” *arXiv preprint arXiv:1712.07195*, 2017.

[54] W. Shen, K. Zhao, Y. Guo, and A. Yuille, “Label distribution learning forests,” *arXiv preprint arXiv:1702.06086*, 2017.

[55] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” in *Advances in Neural Information Processing Systems*, 2015, pp. 3483–3491.

[56] J. Sokolić, R. Giryes, G. Sapiro, and M. R. Rodrigues, “Robust large margin deep neural networks,” *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4265–4280, 58

[57] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, “Theoretical insights into the optimization landscape of over-parameterized shallow neural networks,” *IEEE Transactions on Information Theory*, 2018. 63

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. 30

[59] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* 2018.

[60] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826. 3, 7, 9–11

[61] Y. Tang and R. R. Salakhutdinov, “Learning stochastic feedforward neural networks,” in *Advances in Neural Information Processing Systems*, 2013, pp. 530–538.

[62] G. Urban, K. J. Geras, S. E. Kahou, Ö. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, “Do Deep Convolutional Nets Really Need to be Deep and Convolutional?” In *International Conference on Machine Learning*, 2016.

[63] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004. 11

[64] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, “DisturbLabel: Regularizing CNN on the Loss Layer,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016.