Improving Regression Performance with Distributional Losses

Ehsan Imani [1] Martha White [1]

Abstract

There is growing evidence that converting targets to soft targets in supervised learning can provide considerable gains in performance. Much of this work has considered classification, converting hard zero-one values to soft labels—such as by adding label noise, incorporating label ambiguity or using distillation. In parallel, there is some evidence from a regression setting in reinforcement learning that learning distributions can improve performance. In this work, we investigate the reasons for this improvement, in a regression setting. We introduce a novel distributional regression loss, and similarly find it significantly improves prediction accuracy. We investigate several common hypotheses, around reducing overfitting and improved representations. We instead find evidence for an alternative hypothesis: this loss is easier to optimize, with better behaved gradients, resulting in improved generalization. We provide theoretical support for this alternative hypothesis, by characterizing the norm of the gradients of this loss.

1. Introduction

The choice of problem formulation for regression has a large impact on prediction performance on new data—generalization performance. There is an extensive literature on problem formulations to promote generalization, including robust losses (Huber, 2011; Ghosh et al., 2017; Barron, 2017); proxy losses and reductions between problems (Langford et al., 2006); the addition of regularization to impose constraints or preferences on the solution; the addition of label noise (Szegedy et al., 2016); and even ensuring multiple tasks are learned simultaneously, rather than separately, as in multi-task learning (Caruana, 1998). There is typically a goal in mind—such as classification accuracy or absolute error for regression—but those losses are not necessarily directly minimized.

In recent years, there has been a particular focus on learning representations with neural networks that generalize better. With fixed representations, the loss or problem formulation can only have so much impact, because the learned function is a linear function of inputs. With (deep) neural networks, however, the performance can vary widely, based even on simple modifications such as the initialization (Glorot & Bengio, 2010). Particularly in classification, modifying the outputs can significantly improve performance. An extensive empirical study on classification and age prediction (Gao et al., 2017), under label ambiguity, showed that data augmentation on the label side—putting a distribution over an ambiguous label—significantly improved test accuracy, validated also by other work on age estimation (Rothe et al., 2018). Work on model compression (Ba & Caruana, 2013; Urban et al., 2016) and distillation (Hinton et al., 2015) highlight that a smaller student model can be trained to capture the generalization ability of a larger teacher model. In general, there is a growing literature on data augmentation and label smoothing, that advocates for reduced overfitting and improved generalization from modifying the outputs (Norouzi et al., 2016; Szegedy et al., 2016; Xie et al., 2016; Miyato et al., 2016; Pereyra et al., 2017) and in reinforcement learning where learning distributional outputs, rather than means, improves performance (Bellemare et al., 2017).

There has been some work—though considerably less—towards understanding the impact of the properties of the loss that promote effective optimization. There is a recent insight that minimizing training time increases generalization performance (Hardt et al., 2015), motivating the design of losses that can be more easily optimized. Though not the focus in data augmentation, there have been some insights about loss properties. Gao et al. (2017) showed that their data augmentation approach provided a faster convergence rate (see their Figure 8). Pereyra et al. (2017) showed that label smoothing and their regularizer penalizing confident predictions for classification provided smoother gradient norms than without regularization. Bellemare et al. (2017) hypothesized that the properties of the KL-divergence could have improved learning performance, in a reinforcement learning setting. These papers hint at something deeper occurring with the loss, and motivate investigation into not just the conversion of the problem but into the loss itself.

[1]Department of Computing Science, University of Alberta, Edmonton. Correspondence to: Martha White <whitem@ualberta.ca>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

In this work, we show that the properties of the loss have a significant effect, and better explain the resulting increase in performance than preventing overfitting. We first propose a new loss for regression, called a 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. This modification could be seen as converting the problem to a more difficult (multi-task) problem—from one output, to multiple values to represent the distribution—that promotes generalization in the learner and reduces overfitting. We show that instead of this hypothesis, the (optimization) properties of the HL seem to be the key factor in the resulting improved accuracy. We provide a series of empirical results to support this hypothesis. We also characterize the norm of the gradient of the HL which directly relates to sample complexity (Hardt et al., 2015). The bounds on the variability of the gradient help explain the positive empirical performance of the HL, and further motivate the use of this loss as an alternative for the standard loss for regression.

2. Distributional Losses for Regression

In this section, we introduce the Histogram Loss (HL), which generalizes beyond special cases of soft-target losses used in recent work (Norouzi et al., 2016; Szegedy et al., 2016; Gao et al., 2017). 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. Learning means and distributions

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, 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

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

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, as motivated above, the increased difficulty could beneficially prevent overfitting and promote generalization.

There are many options for learning conditional distributions, $p(y|\mathbf{x})$ even when only considering those that use neural networks (Bishop, 1994; Tang & Salakhutdinov, 2013; Rothe et al., 2015; Bellemare et al., 2017). The goal of this work, however, is not to provide another method to learn distributions. Rather, the goal is to benefit from inducing a distribution over Y, even if that distribution will subsequently not be used, other than for computing a mean prediction. In our experiments, we will compare to an approach that learns distributions, but only to evaluate regression performance.

2.2. The Histogram Loss

Consider predicting a continuous target Y with event space $\mathcal{Y}$, given inputs $\mathbf{x}$. Instead of directly predicting Y, we select a *target distribution* on $Y|\mathbf{x}$. This target distribution is selected upfront, by us, rather than being learned. Suppose the target distribution has support $[a, b]$, pdf p, and cdf F. We would like to learn a parameterized *prediction distribution* $q_\mathbf{x} : \mathcal{Y} \to [0, 1]$, conditioned on $\mathbf{x}$, by minimizing a KL-divergence to p. For any p, however, this may be expensive. Further, depending on the parameterization of the prediction distribution, this may also be potentially non-convex in those parameters.

We propose to restrict the prediction distribution $q_\mathbf{x}$ to be a *histogram density*. Assume $[a, b]$ has been uniformly partitioned into k bins, of width w_i, and let function $f : \mathcal{X} \to [0, 1]^k$ provide k-dimensional vector $f(\mathbf{x})$ of the coefficients indicating the probability the target is in that bin, given $\mathbf{x}$. The density $q_\mathbf{x}$ corresponds to a (normalized) histogram, and has density values $f_i(\mathbf{x})/w_i$ per bin. The KL-divergence between p and $q_\mathbf{x}$ is

$$D_{KL}(p||q_\mathbf{x}) = h(p, q_\mathbf{x}) - h(p).$$

The second term is the differential entropy—the extension of entropy to continuous random variables. Because the second term only depends on p, the aim is to minimize the first term: the cross-entropy between p and $q_\mathbf{x}$. This loss simplifies, due to the form on $q_\mathbf{x}$:

$$\begin{aligned} h(p, q_\mathbf{x}) &= -\int_a^b p(y) \log q_\mathbf{x}(y) dy \\ &= -\sum_{i=1}^{k} \int_{l_i}^{l_i + w_i} p(y) \log \frac{f_i(\mathbf{x})}{w_i} dy \\ &= -\sum_{i=1}^{k} \log \frac{f_i(\mathbf{x})}{w_i} \underbrace{(F(l_i + w_i) - F(l_i))}_{p_i}. \end{aligned}$$

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

$$HL(p, q_\mathbf{x}) = -\sum_{i=1}^{k} p_i \log f_i(\mathbf{x}). \tag{2}$$

This loss has several useful properties. One important property is that it is convex in $f_i(\mathbf{x})$; even if the loss is not convex in all network parameters, it is at least convex on the last layer. The other three benefits are due to restricting the form of the predicted distribution $q_{\mathbf{x}}$ to be a histogram density. First, the divergence to the full distribution p 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 p is flexible, as long as its CDF can be evaluated for each bin. The weighting $p_i = F(l_i + w_i) - F(l_i)$ can be computed offline once for each sample, making it inexpensive to query repeatedly for each sample during training. Third, different distributional choices simply result in different weightings in the cross-entropy. This simplicity facilitates interpreting the impact of changing the distributional assumptions on Y.

2.3. Target distributions and related objectives

Below, we consider some special cases for p 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 p is

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

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

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

This distribution enables significant 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; Szegedy et al., 2016; Pereyra et al., 2017). 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, again motivated as label smoothing, 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 p_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 $p(y) = \delta_{a,j}(y)$ for each sample. Define function $p_{i,j} : [0, \infty) \to [0, 1]$ as $p_{i,j}(a) = \int_{l_i}^{l_i+w_i} \delta_{a,j}(y)dy$. For each y_j, as $a \to 0$, $p_{i,j}(a) \to 1$ if $y_j \in [l_i, l_i + w_i]$ and $p_{i,j}(a) \to 0$ otherwise. So, for i_j s.t. $y_j \in [l_{i_j}, l_{i_j} + w_i]$,

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

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

$$\operatorname*{argmin}_{f_1,\ldots,f_k} -\sum_{j=1}^{n} \log f_{i_j}(\mathbf{x}_j) = \operatorname*{argmin}_{f_1,\ldots,f_k} -\sum_{j=1}^{n} \log q_{\mathbf{x}_j}(y_j).$$

Such a delta distribution on $Y|\mathbf{x}$ results in one coefficient p_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, $p(y)$ 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 $p_i = \epsilon$ for $i \neq i_j$, and $p_{i_j} = 1 - k\epsilon$.

3. Optimization properties of the HL

There are at least two motivations for this loss, in terms of promoting the search for effective solutions. The first is the stability of gradients, promoting stable gradient descent. The second is a connection to learning optimal policies in reinforcement learning. Both provide some insight that the properties of the HL, during optimization, promote better generalization performance.

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

$$f_i(\mathbf{x}) = \frac{\exp(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathbf{w}_i)}{\sum_{j=1}^k \exp(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathbf{w}_j)} \tag{3}$$

for some function $\boldsymbol{\phi}_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{R}^k$ 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 p_i in HL-Gaussian. Let $f_i(\mathbf{x})$ be as in* (3)*, defined by the parameters $\mathbf{w} = \{\mathbf{w}_1, \dots, \mathbf{w}_k\}$ and $\boldsymbol{\theta}$, providing the predicted distribution $q_{\mathbf{x}}$. Assume for all i that $\mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$ is locally l-Lipschitz continuous w.r.t $\boldsymbol{\theta}$*

$$\|\nabla_{\boldsymbol{\theta}}(\mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x}))\| \le l \tag{4}$$

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

$$\|\nabla_{\boldsymbol{\theta},\mathbf{w}} HL(p, q_{\mathbf{x}})\| \le (l + \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\|) \sum_{i=1}^k |p_i - f_i(\mathbf{x})| \tag{5}$$

Proof. First consider the gradient of the HL, with explicit details on these computations in Appendix A

$$\frac{\partial}{\partial \mathbf{w}_i} \sum_{j=1}^k p_j \log f_j(\mathbf{x}) = (p_i - f_i(\mathbf{x}))\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})$$

The norm of the gradient of HL in Equation (2), w.r.t. $\mathbf{w}$ which is composed of all the weights $\mathbf{w}_i \in \mathbb{R}^k$ is

$$\begin{aligned}\Big\|\frac{\partial}{\partial \mathbf{w}} \sum_{j=1}^k p_j \log f_j(\mathbf{x})\Big\| &\le \sum_{i=1}^k \Big\|\frac{\partial}{\partial \mathbf{w}_i} \sum_{j=1}^k p_j \log f_j(\mathbf{x})\Big\| \\ &= \sum_{i=1}^k \|(p_i - f_i(\mathbf{x}))\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\| \\ &\le \sum_{i=1}^k |p_i - f_i(\mathbf{x})| \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\|\end{aligned}$$

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

$$\begin{aligned}\Big\|\frac{\partial}{\partial \boldsymbol{\theta}} \sum_{j=1}^k p_j \log f_j(\mathbf{x})\Big\| &= \Big\|\sum_{i=1}^k (p_i - f_i(\mathbf{x}))\nabla_{\boldsymbol{\theta}} \mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\Big\| \\ &\le \sum_{i=1}^k \Big\|(p_i - f_i(\mathbf{x}))\nabla_{\boldsymbol{\theta}} \mathbf{w}_i^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{x})\Big\| \\ &\le \sum_{i=1}^k |p_i - f_i(\mathbf{x})| l\end{aligned}$$

Together, these bound the norm $\|\nabla_{\boldsymbol{\theta},\mathbf{w}} HL(p, q_{\mathbf{x}})\|$. □

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 $|p_i - f_i(\mathbf{x})|$. This number is guaranteed to be less than 1, but generally is likely to be even smaller, especially if $f_i(\mathbf{x})$ reasonably accurately predicts p_i. Further, the gradients should push the weights to stay within a range specified by p_i, rather than preferring to push some to be very small—close to 0—and others to be close to 1. For example, if $f_i(\mathbf{x})$ starts relatively uniform, then the objective does not encourage predictions $f_i(\mathbf{x})$ to get smaller than p_i. If p_i are non-negligible, this keeps $f_i(\mathbf{x})$ 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 $\sum_{i=1}^k |p_i - f_i(\mathbf{x})|$, can be much larger, even if y is normalized between $[0, 1]$, and can vary significantly 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 (4) will likely be larger, because p_i is frequently zero and so pushes $f_i(\mathbf{x})$ 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 $f_i(\mathbf{x})$ to get close to 1.

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

$$\int_{\mathcal{X}} p_s(\mathbf{x})\Big[-\tau h(q_{\mathbf{x}}) - \int_{\mathcal{Y}} q_{\mathbf{x}}(y) r(y, y_i) dy\Big] d\mathbf{x} \qquad (6)$$

where $\tau > 0$; p_s is a distribution over states $\mathbf{x}$; $q_{\mathbf{x}}$ is the policy or distribution over actions for a given $\mathbf{x}$; 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 (6) corresponds to minimizing the KL-divergence across $\mathbf{x}$ between $q_{\mathbf{x}}$ and the exponentiated payoff distribution $p(y) = \frac{1}{Z}\exp(r(y, y_i)/\tau)$ where $Z = \int \exp(r(y, y_i)/\tau)$, because

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

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 $D_{KL}(p||q_{\mathbf{x}})$ instead of $D_{KL}(q_{\mathbf{x}}||p)$. For example, Gaussian target distribution with variance σ^2 corresponds to minimizing $D_{KL}(p||q_{\mathbf{x}})$ 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 $D_{KL}(q_{\mathbf{x}}||p)$ is upper-bounded by $D_{KL}(p||q_{\mathbf{x}})$ plus variance terms. The intuition, then, is that minimizing the HL is promoting an efficient search for an optimal (prediction) policy.

4. Experiments

In this section, we investigate the utility of the HL-Gaussian for regression, compared to using an ℓ_2 loss. We particularly investigate why the modification to this distributional loss improves performance, designing experiments to test if it is due to (a) the utility of learning distributions or smoothed targets, (b) a bias-variance trade-off from bin size or variance in the HL-Gaussian, (c) an improved representation, (d) nonlinearity introduced by the HL and (e) improved optimization properties of the loss.

Datasets and pre-processing. All features are transformed to have zero mean and unit variance. We randomly split the data into train and test sets in each run.
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 & Gama, 2014), about hourly bike rentals for two years, has 16 features and target set to the number of rented bikes.

Root mean squared error (RMSE) and mean absolute error (MAE) are reported over 5 runs, with standard errors. We include both errors and objective values, on train and test, to provide a more complete picture of the causes of differences between the losses. For space, we only include in-depth results on CT Position in the main body. We summarize the overall conclusions on all three datasets below, and include the tables for Song Year and Bike Sharing in Appendix C and more dataset information in Appendix B.

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 and for CT Position is 385-192-192-192-192-1. All units employ ReLU activation, except the last layer with linear activations. Unless specified otherwise, all networks using HL have 100 bins. 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 with final choices being neither biggest nor smallest.

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 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 $\{0.01, 0.1, 1, 10\}$.

Method	Train objective	Train MAE	Train RMSE	Test objective	Test MAE	Test RMSE
Linear Reg.	6738.719 ($\pm$10.024)	607.277 ($\pm$0.706)	820.896 ($\pm$0.610)	6957.086 ($\pm$41.419)	616.992 ($\pm$2.461)	834.077 ($\pm$2.485)
ℓ_2	0.002 ($\pm$0.001)	15.624 ($\pm$3.353)	20.774 ($\pm$4.713)	0.001 ($\pm$0.000)	19.110 ($\pm$3.034)	29.512 ($\pm$3.622)
HL-Gaussian	146.521 ($\pm$0.045)	5.266 ($\pm$0.155)	7.097 ($\pm$0.169)	147.300 ($\pm$0.102)	**8.992** ($\pm$0.235)	**19.980** ($\pm$2.169)
ℓ_1	0.152 ($\pm$0.002)	12.084 ($\pm$0.665)	16.369 ($\pm$0.651)	0.161 ($\pm$0.006)	16.180 ($\pm$0.606)	38.884 ($\pm$3.760)
ℓ_2+Noise	0.000 ($\pm$0.000)	11.398 ($\pm$1.108)	15.184 ($\pm$1.466)	0.001 ($\pm$0.001)	15.233 ($\pm$1.038)	31.077 ($\pm$7.616)
ℓ_2+Clipping	0.000 ($\pm$0.000)	11.090 ($\pm$0.382)	14.331 ($\pm$0.450)	0.001 ($\pm$0.000)	14.795 ($\pm$0.362)	23.052 ($\pm$0.614)
HL-OneBin	7.387 ($\pm$0.185)	24.623 ($\pm$0.055)	33.732 ($\pm$3.072)	59.141 ($\pm$1.653)	28.001 ($\pm$0.322)	63.290 ($\pm$5.249)
HL-Uniform	7.525 ($\pm$0.169)	24.603 ($\pm$0.016)	31.257 ($\pm$1.600)	58.553 ($\pm$1.356)	28.012 ($\pm$0.392)	71.088 ($\pm$7.304)
MDN	-366.062 ($\pm$0.225)	14.398 ($\pm$1.604)	22.977 ($\pm$3.759)	-365.004 ($\pm$0.543)	17.950 ($\pm$1.514)	28.355 ($\pm$2.781)
ℓ_2+Softmax	0.000 ($\pm$0.000)	9.183 ($\pm$3.784)	12.969 ($\pm$5.313)	0.001 ($\pm$0.000)	12.720 ($\pm$3.609)	22.383 ($\pm$4.525)

Table 1. Performance on CT Position dataset. All the numbers are multiplied by 10^2.

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 variance σ^2 is set to the radius 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.
ℓ_2+Softmax use a softmax-layer with ℓ_2 loss, $\sum_{i=1}^{k}(f_i(\mathbf{x}_j)c_i - y_j)^2$ for bin centers c_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 & 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 to avoid overfitting. We trained the networks for 1000 epochs on CT Position, 150 epochs on Song Year and 500 epochs on Bike Sharing.

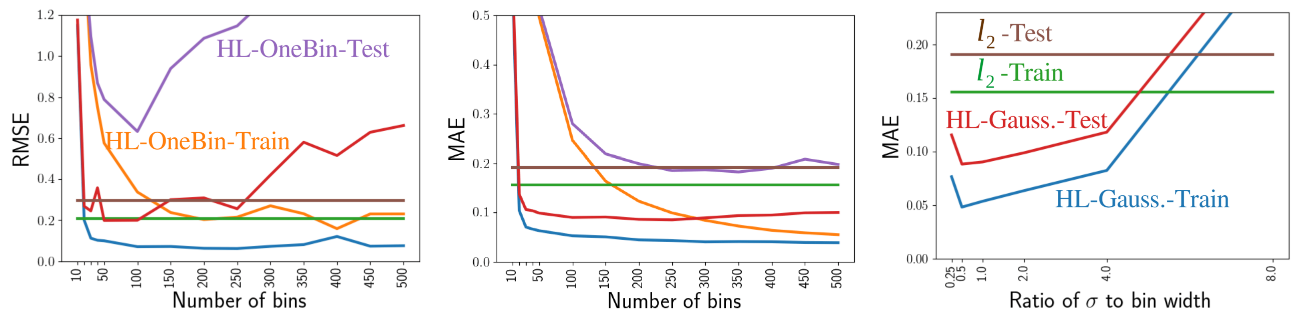
Overall performance and conclusions (Tables 1, 4, 6). We first report the relative performance of all these models, on the CT Position dataset (Table 1) and, in Appendix C, the Song Year dataset (Table 4) and Bike Sharing dataset (Table 6). The overall conclusions are that the HL-Gaussian never harms performance—slightly improving performance on the Song Year dataset—and otherwise can significantly improve performance over alternatives—on both the CT Position and Bike Sharing datasets. We only report the full set of algorithms for CT Position, and more in-depth experiments understanding the result on that domain.

Learning other distributions is not effective (Table 1). HL-Gaussian improves performance, but the other distribution-learning approaches appear to have little advantages, as shown in Table 1. HL-OneBin and HL-Uniform can actually do worse than Regression. MDN provides only minor gains over Regression. Interestingly, it has been shown MDN suffers from numerical instabilities, making training difficult (Oord et al., 2016; Rupprecht et al., 2016).

A related idea to learning the distribution explicitly is to use data augmentation, through label smoothing. We therefore also compared to directly modifying the labels and gradients, with ℓ_2-Noise and ℓ_2-Clipping. These models do perform slightly better than Regression for some settings, but do not achieve the same gains as HL-Gaussian.

The bias-variance trade-off in the loss definition is not significantly impacting performance (Figure 1). If one fixes the possible range of the output variable, the distribution becomes more and more expressive as the number of bins increases. The model could have a higher chance of overfitting in this situation. Reducing the number of bins, on the other hand, introduces discretization error and increases the bias. Further, the entropy parameter σ^2 introduces a bias-variance trade-off, making the target distribution more uniform as entropy increases—likely resulting in lower variance—but also washing out the signal—incurring high bias. The selection of these parameters, therefore, may provide a opportunity to influence this bias-variance trade-off, and improve performance by essentially optimizing the loss for a problem. The ability for the user to select these parameters could explain some of the performance gains in recent results (Gao et al., 2017; Bellemare et al., 2017), compared to standard losses that cannot be tuned.

We tested the impact of varying the number of bins, and the entropy σ^2 for HL-Gaussian. We found that these parameters, especially the entropy, can have an impact on performance, but that the results were much more robust to changing these parameters than might be expected (reported



(a) RMSE with changing bins (b) MAE with changing bins (c) MAE with changing σ

Figure 1. Investigating the bias-variance trade-offs for HL-Gaussian, on the CT Position dataset. Both training and testing performance are reported, with Regression included as a baseline, which does not vary with bins or σ. For a wide range of both bins and σ, HL-Gaussian improves performance over Regression, emphasizing that this performance improvement is not due to carefully setting these additional parameters in the loss. For MAE, for both bins and σ, the training and testing error have similar shapes, suggesting that there is no significant effect of overfitting. An interesting point is that according to the RMSE, however, there does appear to be some overfitting. Overfitting is typically stated agnostic to the loss, but here MAE and RMSE show different trends. We hypothesize that this result occurs because RMSE is magnifying errors that are slightly larger, even though in terms of absolute error, the performance only slightly degrades.

in more depth in Figure 1). It does not seem to be the case, therefore, that the tuning of these hyperparameters is the primary explanation for the improved performance.

The learned representation is not better (Table 2). 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 your 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 than an improved representation is learned. We first 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. The results in Table 2, 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 significantly better than ℓ_2. This suggests that HL-Gaussian is not causing a more useful or more general representation to be learned, as otherwise ℓ_2 should be able to take advantage of that representation.

The softmax nonlinearity is not the main cause (Table 1). The HL-Gaussian can be seen as a generalized linear model, where a small amount of non-linearity is introduced from the transfer. The level of nonlinearity is similar to that in the cross-entropy loss, and the effect should be small because each transformed output $\mathbf{w}_i^\top \boldsymbol{\phi}(\mathbf{x})$ has to predict a probability value. This contrasts with an alternative way to use a softmax layer—which we call ℓ_2+Softmax—which gets to tune the softmax layer to directly predict y given $\mathbf{x}$. Such a layer has additional parameters to predict one target (100 additional parameters, for 100 bins). This contrast the HL-Gaussian, which has also 100 bins but has to predict 100 targets instead of just one target.[1]

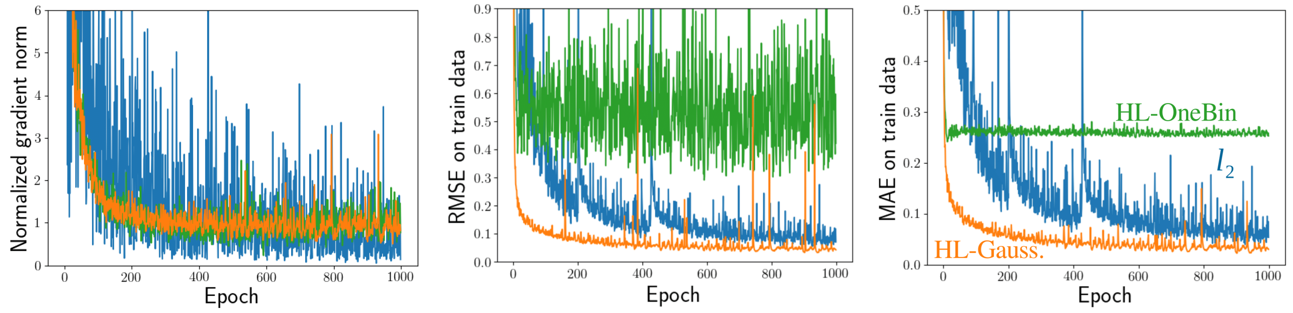
Despite the differences between the role of the softmax in HL-Gaussian and ℓ_2+Softmax, we provide this comparison to provide some insight into potential nonlinearities introduced by the loss. The result in Table 1 shows that this softmax layer can improve performance (to 12.720), but not as significant as HL-Gaussian (8.992). This is particularly intriguing, because as mentioned above, ℓ_2+Softmax can much more flexibly tune the nonlinear softmax layer. The ability to outperform ℓ_2+softmax-layer emphasizes that there are properties of the HL causing improvements beyond the use of the softmax.

HL-Gaussian trains fast (Figure 4). We trained ℓ_2, HL-OneBin, and HL-Gaussian on the CT Position dataset with no dropout to find the role of the loss function on the rate of convergence. We also computed the norm the gradient w.r.t. the parameters of the last layer after each epoch, and normalized the gradient norms of each model by their median to compare their variability. As shown in Figure 4, HL-Gaussian has significantly better behaved gradients, than ℓ_2. Correspondingly, it converges significantly faster and more smoothly. The other two methods that more carefully controlled gradients—ℓ_2-Noise and ℓ_2-Clip—provided the next best gains to HL-Gaussian.

[1]It is possible that having 100 extra parameters in the last layer makes it possible to benefit from randomness, over the ℓ_2. We ran experiments enabling the ℓ_2 to have 100 outputs, each predicting the target but with different initial weights. Even selecting the best of the 100 outputs *on the test data* only slightly improved performance, with a test MAE of 18.421.

	Loss	Default	Fixed	Initialized	Random
Train MAE	Regression	15.624 (±3.353)	288.667 (±13.344)	24.814 (±5.917)	923.335 (±15.665)
	HL-Gaussian	5.266 (±0.155)	16.890 (±2.026)	5.971 (±0.103)	247.851 (±9.686)
Train RMSE	Regression	20.774 (±4.713)	399.664 (±17.364)	34.185 (±8.868)	1224.689 (±17.199)
	HL-Gaussian	7.097 (±0.169)	24.744 (±2.470)	7.834 (±0.130)	555.212 (±22.567)
Test MAE	Regression	19.110 (±3.034)	291.980 (±13.587)	28.310 (±6.018)	930.481 (±19.094)
	HL-Gaussian	8.992 (±0.235)	20.296 (±1.832)	10.089 (±0.087)	260.863 (±10.751)
Test RMSE	Regression	29.512 (±3.622)	403.070 (±17.237)	44.418 (±12.183)	1231.502 (±21.574)
	HL-Gaussian	19.980 (±2.169)	31.288 (±2.145)	23.161 (±3.682)	589.087 (±22.525)

Table 2. Representation experiment results on CT Position dataset. All the numbers are multiplied by 10^2. 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. We highlight the Test MAE, though the other rows have similar trends. Using the HL-Gaussian representation for Regression (first column, Fixed) causes a sudden spike in error, even though the last layer in Regression is re-trained. This suggests the representation is tuned to HL-Gaussian. The representation does 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 obtains significantly better performance, solely from optimizing the last layer with a different loss than Regression.



(a) Normalized gradient norm during training (b) RMSE on training data, vs. epochs (c) MAE on training data, vs. epochs

Figure 2. The norm of the gradient and error values for the training on the CT Position dataset, for three training objectives. The behaviour for the testing error is similar to the training error, and so is included in Appendix C.1. The gradient norms are normalized by the median value, where the median norms are 0.0014 for Regression, 0.0640 for HL-OneBin, and 0.0305 for HL-Gaussian. The median norm for HL-OneBin is an order of magnitude larger than HL-Gaussian, so though it is not variable, it is consistently larger. Because targets are normalized, the median norm for ℓ_2 is actually lower, but it has significantly more variability.

5. Conclusion

We introduced a novel loss for regression, called the Histogram Loss (HL), that explicitly constructs a distribution over targets to predict, rather than directly estimating the mean of the target conditioned on inputs. The loss involves minimizing the KL-divergence between a predicted distribution and this target distribution. To make this loss efficient to compute, without significantly reducing modeling power, we restrict the class of approximation densities to histogram densities. We highlight that for a particular setting of the HL—with a target Gaussian distribution—the norm of the gradient does not grow large or vary widely. Combined with recent results that show reducing training steps for stochastic gradient results in improved generalization provide some theoretical justification for why we observe such strong performance of HL-Gaussian in practice. We conduct a series of experiments to identify this gain, with evidence that the main role is not due to overfitting or an improved representation, but rather due to the fact that the HL can be optimized in a smaller number of steps, with smoother gradients.

The introduction of the HL provides several avenues to improve our choice of loss function. One direction is to more explicitly take advantage of the specification of the target distribution. In this work, we considered this loss only for a fixed set of bins, widths and variance parameter σ for the target distribution. To be more agnostic to these choices, we demonstrated performance across possible parameter settings. However, these parameters could be determined using meta-parameter optimization strategies, such as cross validation, or even learning strategies with particular objectives for these parameters. The key property to make the HL easy to specify and optimize was the use of a histogram to predict the target; the derivation does not prevent also optimizing the bins centers, widths and variances.

Overall, this work provides some unification of recent results using soft targets, through the introduction of the HL. We hope for it to facilitate discussion and development on the design of losses that promote learning, and direct further investigation into the importance of the optimization properties of these losses.

Acknowledgments

We would like to thank Alberta Innovates for funding AMII (the Alberta Machine Intelligence Institute) and this research.

References

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

Barron, J. T. A More General Robust Loss Function. *arXiv*, 2017.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. The million song dataset. In *Ismir*, volume 2, pp. 10, 2011.

Bishop, C. M. Mixture density networks. *Technical Report*, 1994.

Caruana, R. Multitask learning. In *Learning to learn*, pp. 95–133. Springer, 1998.

Chollet, F. et al. Keras. `https://github.com/fchollet/keras`, 2015.

Fanaee-T, H. and Gama, J. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2-3):113–127, 2014.

Gao, B.-B., Xing, C., Xie, C.-W., Wu, J., and Geng, X. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 26(6):2825–2838, 2017.

Ghosh, A., Kumar, H., and Sastry, P. Robust loss functions under label noise for deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2017.

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

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

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Huber, P. J. Robust statistics. In *International Encyclopedia of Statistical Science*, pp. 1248–1251. Springer, 2011.

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

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

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

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

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

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., Schuurmans, D., et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pp. 1723–1731, 2016.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

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

Rothe, R., Timofte, R., and Van Gool, L. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, 126(2):144–157, 2018.

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

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

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

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

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

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

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

A. Explicit gradient computations

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

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

For $j = i$, we get

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

Consider now the gradient of the HL, w.r.t b_i

$$\frac{\partial}{\partial b_i} \sum_{j=1}^k p_j \log f_j(\mathbf{x}) = \sum_{j=1}^k p_j \frac{1}{f_j(\mathbf{x})} f_j(\mathbf{x})(1_{i=j} - f_i(\mathbf{x}))$$
$$= \sum_{j=1}^k p_j (1_{i=j} - f_i(\mathbf{x}))$$
$$= p_i - f_i(\mathbf{x}) \sum_{i=1}^k p_i$$
$$= p_i - f_i(\mathbf{x})$$

Then

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

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

Dataset	# train	# test	# feats	Y range
Song Year	463715	51630	90	[1922,2011]
CT Position	42800	10700	385	[0,100]
Bike Sharing	13911	3478	16	[0,1000]

Table 3. Overview of the datasets used in the experiments.

B. Dataset details.

We include an overview of the datasets in Table 3. We additionally show a histogram of their targets, in Figure 3.

C. Additional experiments

We provide overall performance results for the two other datasets. We include the learning curves on test data for CT Position, corresponding to Figure 4 in the main text.

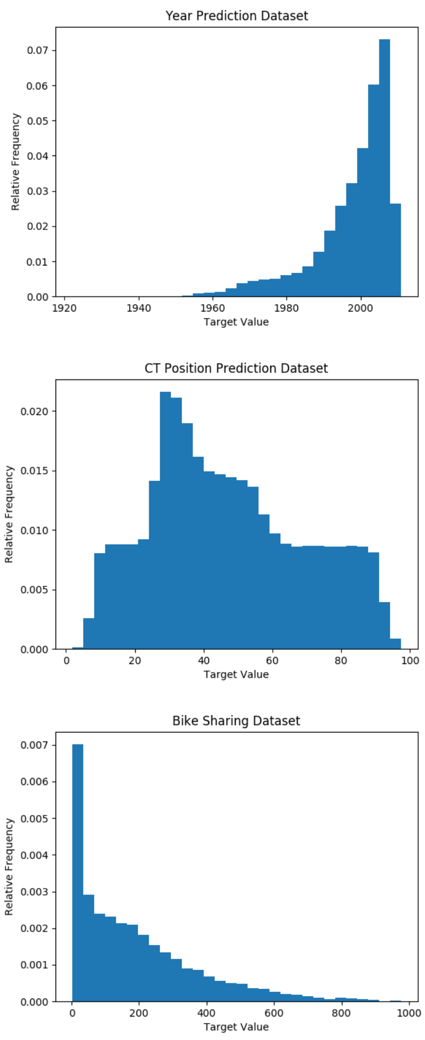


Figure 3. Histogram of the target values for the three datasets.

C.1. Test Learning Curves for CT Position dataset

We include additional graphs for the variability in the RMSE and MAE, for the three objectives HL-Gaussian, HL-OneBin and ℓ_2, for test data in Figure 4.

C.2. Experiments on Song Year dataset

For the Song Year dataset, we include results both for random train-test splits and report results for the fixed train/test split recommended by the authors of the dataset to avoid the effect of an artist having songs in both the train and test sets.

For this dataset, both HL-Gaussian and HL-OneBin outperform ℓ_2 only slightly, and perform similarly to each other. The ℓ_2 loss with a nonlinear softmax layer also performs about the same, suggesting the main (small) gain for this dataset is from this nonlinearity. This further suggests that the ℓ_2 is likely a suitable loss for this problem, and there

Method	Train objective	Train MAE	Train RMSE	Test objective	Test MAE	Test RMSE
Linear Reg.	9114.456 (±6.524)	679.285 (±0.264)	954.696 (±0.342)	9129.131 (±26.215)	679.646 (±0.741)	955.461 (±1.370)
ℓ_2	0.900 (±0.001)	575.997 (±2.171)	813.868 (±0.825)	0.955 (±0.004)	602.393 (±2.026)	869.569 (±1.923)
HL-Gaussian	320.475 (±0.039)	580.305 (±0.723)	846.072 (±0.282)	320.260 (±0.052)	591.304 (±1.413)	862.656 (±1.683)
HL-OneBin	304.411 (±0.063)	581.230 (±0.967)	848.745 (±0.284)	304.827 (±0.091)	590.823 (±1.589)	863.475 (±1.621)
ℓ_2+Softmax	0.895 (±0.003)	569.797 (±3.613)	809.113 (±1.679)	0.962 (±0.003)	600.607 (±3.064)	872.765 (±1.477)

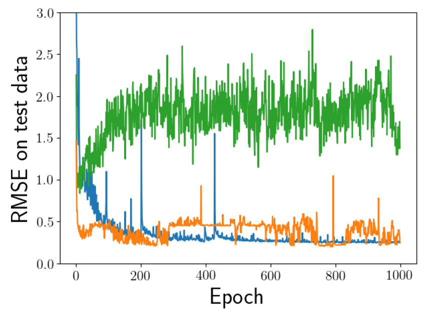
Table 4. Performance on the Song Year dataset. All the numbers are multiplied by 10^2.

Method	Train objective	Train MAE	Train RMSE	Test objective	Test MAE	Test RMSE
Linear Reg.	9125.643	679.557	955.282	9044.316	680.050	951.016
ℓ_2	0.904	574.110	817.002	0.997	610.344	888.808
HL-Gaussian	320.334	576.914	844.565	322.028	598.490	875.857
HL-OneBin	304.389	581.066	848.344	307.988	601.927	879.877

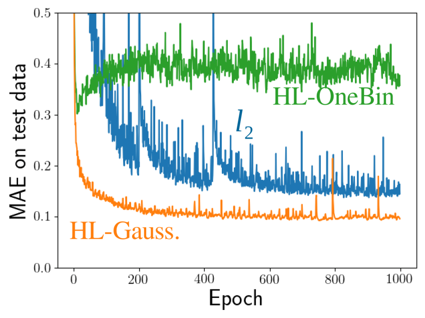
Table 5. Performance on the Song Year dataset with the authors' suggested train and test splits. All the numbers are multiplied by 10^2.

Method	Train objective	Train MAE	Train RMSE	Test objective	Test MAE	Test RMSE
ℓ_2	0.04 (±0.00)	1343.51 (±41.46)	1865.55 (±60.99)	0.22 (±0.00)	2899.84 (±52.44)	4601.21 (±80.54)
HL-Gaussian	212.43 (±1.12)	1667.68 (±30.12)	2645.08 (±41.10)	254.60 (±1.01)	2495.21 (±13.28)	4182.06 (±57.35)
HL-OneBin	169.62 (±3.53)	1768.68 (±39.17)	2871.14 (±56.85)	287.13 (±5.80)	2607.38 (±15.94)	4373.20 (±53.96)
ℓ_2+Softmax	0.07 (±0.01)	1750.31 (±127.58)	2463.93 (±150.76)	0.22 (±0.00)	2886.93 (±70.80)	4544.68 (±41.25)

Table 6. Performance on the Bike Sharing dataset. All the numbers are multiplied by 10^2.



(a) RMSE on testing data, vs. epochs



(b) MAE on testing data, vs. epochs

Figure 4. The error values for the testing data on the CT Position dataset, for three training objectives.

is little to gain for switching to the HL. There is a slightly larger gain for HL-Gaussian in Table 5 for the training/test split suggested by the authors of this data, but still not nearly as large as CT Position or Bike Sharing.

C.3. Experiments on Bike Sharing dataset

We provide a comparison of performance on the Bike Sharing dataset in Table 6. We used early stopping to avoid overfitting, because on this dataset, dropout was ineffective.

The network for Bike Sharing uses four hidden layers of width 64, but we additionally tested a network architecture with four hidden layers of width 512. For this wider network, ℓ_2 was able to get better final TEST MAE and Test RMSE performance of 2402.67 and 4014.31 respectively. However, performance was quite a bit more variable during learning—likely due to the overparameterization. Future work is to better understand the effect of different network architectures on the performance of the different losses.