Mitigating Class-Boundary Label Uncertainty to Reduce Both Model Bias and Variance

MATTHEW ALMEIDA, YONG ZHUANG, and WEI DING, University of Massachusetts Boston
SCOTT E. CROUTER, University of Tennesee Knoxville
PING CHEN, University of Massachusetts Boston

The study of model bias and variance with respect to decision boundaries is critically important in supervised learning and artificial intelligence. There is generally a tradeoff between the two, as fine-tuning of the decision boundary of a classification model to accommodate more boundary training samples (i.e., higher model complexity) may improve training accuracy (i.e., lower bias) but hurt generalization against unseen data (i.e., higher variance). By focusing on just classification boundary fine-tuning and model complexity, it is difficult to reduce both bias and variance. To overcome this dilemma, we take a different perspective and investigate a new approach to handle inaccuracy and uncertainty in the training data labels, which are inevitable in many applications where labels are conceptual entities and labeling is performed by human annotators. The process of classification can be undermined by uncertainty in the labels of the training data; extending a boundary to accommodate an inaccurately labeled point will increase both bias and variance. Our novel method can reduce both bias and variance by estimating the pointwise label uncertainty of the training set and accordingly adjusting the training sample weights such that those samples with high uncertainty are weighted down and those with low uncertainty are weighted up. In this way, uncertain samples have a smaller contribution to the objective function of the model's learning algorithm and exert less pull on the decision boundary. In a real-world physical activity recognition case study, the data present many labeling challenges, and we show that this new approach improves model performance and reduces model variance.

CCS Concepts: • **Computing methodologies → Knowledge representation and reasoning**; *Neural networks*;

Additional Key Words and Phrases: Bias and variance, label uncertainty, neural networks

ACM Reference format:
Matthew Almeida, Yong Zhuang, Wei Ding, Scott E. Crouter, and Ping Chen. 2021. Mitigating Class-Boundary Label Uncertainty to Reduce Both Model Bias and Variance. *ACM Trans. Knowl. Discov. Data* 15, 2, Article 27 (March 2021), 18 pages.
https://doi.org/10.1145/3429447

Funding for this research was provided by NIH grants 1R01HD083431-01A1: Novel Approaches for Predicting Unstructured Short Periods of Physical Activities in Youth and R21HL093407-01A1: Novel Techniques for the Assessment of Physical Activity in Children. This research was also partially supported by the Oracle grant to the College of Science and Mathematics at the University of Massachusetts Boston.
Authors' addresses: M. Almeida, Y. Zhuang, W. Ding, and P. Chen, University of Massachusetts Boston, 100 Morrissey Boulevard, Boston, MA 02125; emails: {malmeida, yong}@cs.umb.edu, {Wei.Ding, ping.chen}@umb.edu; S. E. Crouter, University of Tennesee Knoxville, Knoxville, TN 37996; email: scrouter@utk.edu.

1556-4681/2021/03-ART27 $15.00
https://doi.org/10.1145/3429447

1 INTRODUCTION

One of the most important tasks in modern machine learning is that of supervised classification [36], whereby a training set $\mathbf{X}$, with associated class labels y, is used to minimize the value of an objective function $\mathcal{L}(\mathbf{X}, \theta)$ with respect to the data $\mathbf{X}$ and model parameters θ, such that the trained model is able to reliably assign labels to new, unseen data.

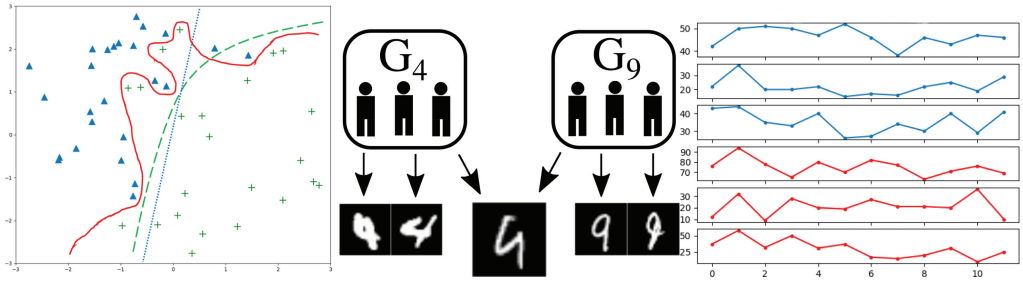
Training a model that will generalize to unseen data is a fundamental challenge in supervised learning, and is subject to the bias-variance dilemma [17, 44]. To lower bias, the model needs to be adapted such that the decision boundary is permitted to contort to accommodate more boundary training samples and improve training accuracy, which results in a more complex classification model (see Figure 1(a)). However, in this process, noisy or uncertain points may also be accommodated, which could harm the generalization and make predictions less accurate against a test set. On the other hand, a less-complex, higher-bias model is relatively simple and may exhibit improved generalization (i.e., have a lower variance). One issue with reducing both model bias and variance lies in the trustworthiness of a sample: Ideally, an informative sample should be wholly accommodated and a noisy sample should be discounted or discarded completely.

Understanding the nature of uncertainty has long been an active topic in artificial intelligence (AI) research. We take a data-centered view and consider that noise and uncertainty comes from four sources:

- *Value noise*: Value noise exists in collected values due to imprecise collection procedure and measurement tools, which is stochastic noise or aleatoric (inherent to the problem) uncertainty discussed in literature [10].
- *Feature uncertainty*: Feature uncertainty arises from the erroneous inclusion or exclusion of important features. If discriminative features are left out of an analysis, the data are projected into a lower-dimensional subspace where class representations may overlap. If spurious features are included in an analysis, especially a large number of them, the curse of dimensionality will make the data sparse and learning more difficult.
- *Distribution uncertainty*: The number of samples is too small or data are biased, and so does not accurately reflect the true data distribution. This uncertainty is often called deterministic noise or epistemic (due to the modeling process) uncertainty in literature [10].
- *Label uncertainty*: Class labels are often conceptual entities, and labeling is performed by human annotators. Even with carefully designed labels and experienced annotators uncertainty can arise due to different interpretation of labels and samples.

This categorization of noise and uncertainty solely serves to inform our approach; a comprehensive discussion of noise and uncertainty goes beyond the scope of this article. Here, we focus on a particular type of label uncertainty: that which is tied to the representation of the samples rather than errors which are stochastic in nature (labels that are flipped with a certain probability by a random process). As we will illustrate in the following sections, better understanding and handling of label uncertainty can contribute to the reduction of both model bias and variance.

While label uncertainty can be revealed through multiple inconsistent labels if more than one annotator is used, in practice a sample is often annotated only once due to the prohibitive labeling cost. Using a single label for both high- and low-confidence samples obscures the label uncertainty. In this article, we will consider this specific type of heteroscedastic (data-dependent) label uncertainty and provide a method to mitigate its effects on the decision boundary of the classification model. Our main idea is to use a k-nearest-neighbor-based entropy measure to estimate the degree to which a point's label is uncertain: the label of a point surrounded by points of another class should be treated as less trustworthy—and the point's ability to move the decision



(a) Decision Boundaries in 2D classification (b) Generating Processes for MNIST digits (c) Example data from activity recognition

Fig. 1. Three examples illustrating label uncertainty. (a) Three decision boundaries drawn by different classification models. The green (dashed) model is the true decision boundary, but it is impossible to know. The red (solid) model has adjusted to accommodate all samples and results in an overfitting low-bias but high-variance model. The blue (dotted) model is an underfitting low-variance but high-bias model. Uncertainty in the label of points near the decision boundary could make a large difference when determining the boundary. (b) Represents the generating processes for "people writing 4s" (G_4) and "people writing 9s" (G_9). The central picture will be labeled either 4 or 9, but could be generated by either process. (c) Shows two training examples from an activity recognition dataset (collected on a hip-mounted triaxial accelerometer). The top three time series (in blue) make up a walking sample, and the bottom three (in red) make up a running sample. Such labeled samples are very difficult to differentiate.

boundary reduced—as the position of its representation is suspicious. Points in a highly heterogeneous neighborhood are similarly weighted down, as they are in an area of the representation space that is very chaotic; it is possible the points from two different classes are being projected down an unavailable data axis into the same region of the feature space.

To be more specific, our approach to improve both model performance and generalization is to identify points in the dataset that are likely to be "noisy" or "misleading" to the model (terms that will be defined formally in Section 3) due to labeling uncertainty and automatically adjust their corresponding sample weights. In this way, data samples with uncertain labels will not contribute to the loss function to the same extent as informative data samples during training. We derive a sample weight for each point, using a function that calculates a pointwise score based on the entropy and distances of samples within each point's neighborhood, such that a point in a homogeneous neighborhood, with many neighbors of the same class as itself, will be weighted up, and points in heterogeneous neighborhoods will be weighted down as potentially noisy or mislabeled. The main contributions of this article are the following:

- —A novel definition of what is meant by an informative training sample with respect to its paired label. Instead of the existing approaches focusing on model fine-tuning—normally subject to the bias-variance tradeoff—we directly attack the core of the problem: how to measure the trustworthiness of a sample so we can decide whether a decision boundary should be adjusted to accommodate it accordingly. In this way, we can improve generalization performance while reducing model variance.
- —A new k-nearest-neighbor-based framework for estimating label uncertainty point by point and mapping it to sample weights for use in model training.
- —When performing experiments, we configure our training environment such that our runs are deterministic for each random seed. In this way, we can be sure that the variation in performance is due to the variable we intend to observe, the sample weighting, not from the randomness of learning models.

This article is organized as follows. In the next section, we discuss related work. We then present our k-nearest-neighbor-based label uncertainty measure and describe how to map those values to sample weights. We then describe our experimental procedure and results, and give final concluding remarks.

2 RELATED WORK

The tradeoff between bias and variance has been studied from different perspectives. Geman, et al. introduced the dilemma in terms of neural networks (NNs) in [17], showing that increasing variance with model capacity makes NN models require an "unrealistic" number of training examples that they could have not have foreseen becoming realistic with today's data collection and storage. Goodfellow et al. presented an updated discussion in [20]. We examine the bias–variance relationship in settings where labeling errors are data-dependent, meaning that the probability of a given example i having an incorrect label is dependent on the example's representation, $\mathbf{x_i}$. This setting could arise in human activity recognition, for instance, when trying to classify walking and running: examples along a model's decision boundary separating the two classes are much more likely to be mislabeled than examples far from it. (Imagine a model that separated only on a person's average speed over a window of activity; examples near the speed representing the split point between the classes likely have much more uncertainty to their labels than very high- or low-speed examples.)

In [31], Liu and Tao took an approach to handling label noise using importance-based sample reweighting. They worked specifically on 2-class label noise, where there is certain probability ρ_{-1} that an example of class -1 will have its label flipped to $+1$, and a probability ρ_{+1} that an example of class $+1$ will have its label flipped to -1. They used a density ratio estimation method to perform the reweighting. This work and that of Scott et al. [5, 42, 43] require estimation of the noise rates ρ_{-1} and ρ_{+1}. The authors of [33] offer optimization methods to avoid issues due to binary label noise and give steps to estimate noise from corrupted data.

In [37], the authors introduce a method called Rank Pruning to treat noise in labels, whereby they train one or more classifiers and then prune the training set of likely false positives and false negatives based on the confidence rankings of the trained models; they then retrain a new model based on the cleaned training set. With our method, we are concerned both with points that are actively misleading to the model (and should be removed entirely, as in the case of label corruption as in [37]) and with classes with definitions that are more ambiguous, such as those in activity recognition; in these cases, removing points with low model confidence could result in a loss of useful information.

Natarajan, et al. [35] also addressed binary classification with class-specific label noise, offering approaches to modify surrogate loss functions robust to it.

A popular avenue of current research studies a form of stochastic label noise that is assumed to be as bad as possible for the model, when creating adversarial examples [12, 13, 21, 22].

In [40], the authors use sample weighting to improve the performance of deep learning models. They develop a method to calculate sample weights, for example, by learning a weighting during optimization and modifying the weights at each step based on how they reduce loss with respect to mini-batches drawn from a high-quality validation set. This method can help with label noise because instances with incorrect labels should presumably have very poor agreement with the validation set and therefore be weighted down. In our case, where we are trying to improve generalization along decision boundaries where we expect labeling problems, it would be difficult to obtain a truly clean validation set. Additionally, the authors analyze the method only on stochastic noise settings, where there is a uniform probability of label flipping or a certain probability with which labels from any class are flipped to a "background" class, representing the case when

human annotators miss a positive example. Other recent methods [19, 26] also assume stochastic label noise or corruption that is not data-dependent.

We use a k-nearest-neighbor-based method to calculate our pointwise uncertainty scores, by comparing the local self-information of the class of each point $\mathbf{x_i}$ with the local self-information of the other classes in the dataset, weighted by the distances from each point to the other $k-1$ points.

k-nearest-neighbor methods are commonly used in estimators of differential entropy and mutual information over continuous random variables [3, 11, 15, 16, 46, 48]. A popular method from [27] uses the volume of open d-dimensional balls around each point, with a radius of the distance from the point to its kth neighbor, to estimate the pointwise local densities for the available samples, and then uses those volumes to compute an estimate of the global entropy.

We work in the discrete case, analyzing the local self-information over a finite set of classes, but find the k-nearest-neighbor approach to entropy estimation valuable because it allows us to look at the local label entropy by class from a pointwise perspective. This gives us a measure of surprise to find the point's label at its location in the representation space, which we combine with the sparsity of the neighborhood and the local entropy of all classes to assign a score from which we can derive a sample weight.

In our work, we consider the multi-class case where there is uncertainty in the labeling. We will show how to reduce bias and variance together in the following sections.

3 OVERVIEW OF MODEL BIAS AND VARIANCE AND CONNECTION TO LABEL UNCERTAINTY

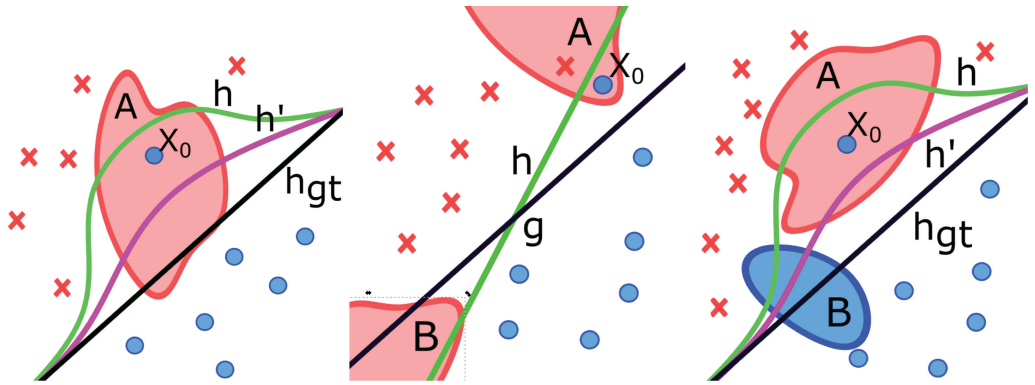
It is clear that label uncertainty impacts the trustworthiness of a sample, which, in turn, determines how much the sample should be accommodated when a decision boundary is produced as finding an optimal decision boundary can reduce both bias and variance. In this section, we will discuss how we define label uncertainty, and then analyze its connection to model bias and variance.

3.1 Uncertain and Informative Data Points

In this article, we make many references to the ideas of uncertain and informative data points. We use these terms relative to the ground truth of the classification problem and a hypothetical "ideal model" or true labeling function. Any set of training data can be viewed as a set of draws from some unknown joint distribution, with each class representing a marginal distribution over the feature space. Each vector describing those samples is a single point in the feature space.

We consider the case where the data collection and labeling process is complete and cannot be revisited. Of course, if more features could be added to each point's representation, the neighborhoods of the points would be changed, and, consequently, the uncertainty estimation would be different as well. If adding a feature to a point maps it into a homogeneous region instead of a heterogeneous one, treating it as a more certain point is a reasonable approach.

A model is then a function $g(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{N}$, which maps input vectors to class labels, integers that denote which class—which marginal distribution—a data vector was most likely generated from. It may be that a particular data point, say, the MNIST [28] 9 with its top left open enough to resemble a 4 (as illustrated in Figure 1(b)), could have been generated by the process of people writing 9s with probability p and the process of people writing 4s with probability $(1-p)$. (For the purposes of this discussion, we assume no one writing any of the other digits could ever produce the sample.) When we have a label (in an ideal setting), we have the correct answer for which class marginal distribution a sample was generated from—but we do not know whether or not that class's generating process was the one most likely to generate a sample at that point in the feature space.



(a) A case where downweighting examples results in lower generalization error

(b) A case where low model capacity causes fitting noisy points to be beneficial

(c) A more detailed case where fitting noisy points helps in some areas and hurts others

Fig. 2. Three examples illustrating the behavior along decision boundaries. Each picture depicts the boundary between two classes, the red Xs and blue circles. The lines represent model decision boundaries and the shaded regions A and B represent areas that are colored for the most likely class to be generated in that region. The red area is most likely to generate a red X, and the blue is most likely to generate a blue circle. X_0 is a blue point in a red region, and is therefore a noisy point. (a) Three decision boundaries drawn by different classification models. h accommodates noisy point X_0, and so any future points generated in the large region underneath h will be classified incorrectly. h' was trained with X_0 weighted down, so the erroneous region is smaller. h_{gt} was trained with no noise, and is optimal (but imperfect). (b) Here, the downweighting process fails because of the low model capacity. h accommodates a noisy point that model g does not, but the mistake helps because all of region B is moved to the correct side. (c) A more complex case where the accommodation of a noisy point causes misclassifications in A but correct classifications in B.

In the cases where the same point in the feature space could be drawn from more than one class marginal distribution—there is some overlap—an ideal model, a model with perfect knowledge of the probability with which each process will generate a sample at each point in the space, cannot have perfect accuracy. The best that a model can do with such data is to predict the most likely generating class marginal distribution at each point in the domain. If the example digit is generated by the process producing 9s with probability .7 and by the process producing 4s with probability .3, the best possible model can only predict 9 for that image, and be incorrect 30% of the time.

This leads us to a formal definition of informative and uncertain data points: a data point is "informative" if it is of the class most likely to generate a data point at its location in the feature space. It is "uncertain" if it was generated by any other class.

Definition: Let $\mathbf{X}$ be a dataset composed of n d-dimensional training samples $\mathbf{x_i} \in \mathbf{X}$, and let $f(\mathbf{x_i}) : \mathbb{R}^d \rightarrow \mathbb{R}^C$, where C is the number of classes in the dataset, be a function taking any input point to the distribution over classes representing the probability that class c generated sample $\mathbf{x_i}$. $\sum_c f(\mathbf{x_i})_c = 1$. Let $y \in \mathbb{R}^n$ give the observed labels for each $\mathbf{x_i}$. f in this formulation can be seen as the ground-truth labeling function because it contains all possible knowledge of the labeling behavior of the problem.

Then, we say that a sample $\mathbf{x_i}$ is an *informative* point when $y(\mathbf{x_i}) = \max[f(\mathbf{x_i})]$. In other cases, $\mathbf{x_i}$ is an *uncertain* point.

3.2 Model Bias and Variance

When we refer to model bias and variance, we refer to the bias and variance terms in the decomposition of the expected out-of-sample (generalization) error of a classifier, as introduced in [17] and re-presented in many canonical texts, including [23] and [1]. A good term-by-term explanation is available in [47].

Take a setting where we have a problem domain $\mathcal{D}$, which consists of a dataset, $\mathbf{X}_{\mathcal{D}}$, and associated labels given by a true labeling function $f(\mathbf{X}_{\mathcal{D}})$, which encapsulates an element of data-dependent label noise, as above: For a given point $\mathbf{x_i} \in \mathbf{X}_{\mathcal{D}}$, $f(\mathbf{x_i})$ is vector-valued, giving the probability distribution of observing a particular label $y_i \in \{1...C\}$ at $\mathbf{x_i}$. $|f(\mathbf{x_i})| = C, \sum_c f(\mathbf{x_i})_c = 1$.

We can think of the objects subscripted with $\mathcal{D}$ as being population-level; let $(\mathbf{X}, \mathbf{y}) \sim (\mathcal{D}; f)$ refer to drawing a particular $\mathbf{X} \subset \mathbf{X}_{\mathcal{D}}$ and $\mathbf{y} \sim f(\mathbf{X}_{\mathcal{D}})$ from $\mathcal{D}$. This yields an observed set of individual points $\mathbf{x_i}$ and associated labels y_i, with each y_i drawn from the distribution $f(\mathbf{x_i})$. Model bias and variance are decomposed from the expected generalization error taken over such draws. Let g be a model function that yields predicted class $g(\mathbf{x_i})$. The typical decomposition of the expected prediction error of a model trained on a single draw into bias and variance is expressed as follows, using the squared error loss function [1]:

$$\mathbb{E}_{pred}[g(\mathbf{X})] = \mathbb{E}_{\mathbf{X}}[(g(\mathbf{x}) - \mathbf{y})^2], \tag{1}$$

where $\mathbf{y}$ represents the labels $\{y_i\}$ drawn from the distributions $f(\mathbf{x_i})$, and we write the expected error over potential observed datasets from domain $\mathcal{D}$ as

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\text{pred}}[g(\mathbf{X})]\right] &= \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\mathbf{X}}[(g(\mathbf{X}) - \mathbf{y})^2]\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}[(g(\mathbf{X}) - \mathbf{y})^2]\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}[g(\mathbf{X})^2] - 2\mathbb{E}_{\mathcal{D}}[g(\mathbf{X})]\mathbf{y} + y^2\right]. \end{aligned} \tag{2}$$

Abu-Mostafa et al. [1] observe that $\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(\mathbf{x})]$ is an "average function" over trained models and denote it $\bar{g}$, and then derive the model bias and variance:

$$\mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\text{pred}}(g)\right] = \mathbb{E}_{\mathbf{X}}\left[\mathbb{E}_{\mathcal{D}}[g(\mathbf{X})^2] - 2\bar{g}(\mathbf{X})\mathbf{y} + \mathbf{y}^2\right]. \tag{3}$$

Adding in terms summing to 0, $-\bar{g}(\mathbf{X})^2 + \bar{g}(\mathbf{X})^2$, we have

$$\begin{aligned} &\mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\text{pred}}(g)\right] \\ &= \mathbb{E}_{\mathbf{x}}[\underbrace{\mathbb{E}_{\mathcal{D}}[g(\mathbf{X})^2] - \bar{g}(\mathbf{X})^2}_{\mathbb{E}_{\mathcal{D}}[(g(\mathbf{X}) - \bar{g}(\mathbf{X})^2]} + \underbrace{\bar{g}(\mathbf{X})^2 - 2\bar{g}(\mathbf{X})\mathbf{y} + \mathbf{y}^2)}_{(\bar{g}(\mathbf{X}) - \mathbf{y})^2}], \end{aligned} \tag{4}$$

where $(\bar{g}(\mathbf{X}) - \mathbf{y})^2$ is the bias and $\mathbb{E}_{\mathcal{D}}[(g(\mathbf{X}) - \bar{g}(\mathbf{X}))^2]$ is the variance. Extension to loss functions beyond squared error and a detailed analysis of systemic and variance effects are available in [25].

3.3 Bias, Variance, and Label Noise

The above shows a bias–variance decomposition for squared error in a setting where $\mathbf{y}$ is considered the absolute truth, not a particular draw from a set of data-dependent label distributions $f(\mathbf{X})$. Here, we examine how label noise impacts the bias and variance of models.

Take a model function $h(\mathbf{X})$ trained on data with noisy labels, $(\mathbf{X}, \mathbf{y}) \sim (\mathcal{D}; f)$. Let y_{gt} be the ground-truth label vector, the draw from $f(\mathbf{X})$ for which each point is assigned its most probable label: $y_{gt_i} = \arg\max_c f(\mathbf{x_i})$, and let $h_{gt}(\mathbf{X})$ be a model trained on y_{gt} of the same hypothesis class as h.

We can split the observed data $\mathbf{X}$ into two subsets, the informative points $\mathbf{X}_{\text{info}}$ and noisy points $\mathbf{X}_{\text{noisy}}$. $\mathbf{X}_{\text{info}}$ and $\mathbf{X}_{\text{noisy}}$ are disjoint and their intersection includes all examples in $\mathbf{X}$. Assume

$\mathbf{X}_{\text{noisy}} \neq \emptyset$. In the case where we have a deterministic training process and infinite model capacity in the hypothesis class, $h_{gt}(\mathbf{X})$ will be Bayes optimal but $h(\mathbf{X})$ will be sub-optimal—it will incorrectly predict each point in X_{noisy}, and its bias relative to h_{gt} will be higher.

If we were able to identify which $\mathbf{x_i}$ were in $\mathbf{X}_{\text{noisy}}$ via some process with perfect confidence, we could remove that subset from the training set and remove the effect of the label noise. Since we assume that any such attempt to identify noisy points would have its own uncertainty, or that the labels themselves might not be perfectly orthogonal, we instead weight down those examples that we suspect are noisy. A model $h'(\mathbf{X})$ trained with the noisy $\mathbf{y}$ vector but with the noisy points downweighted will have a decision boundary between that of h and that of h_{gt}. In this case, since potential h' decision boundaries do not reach to the noisy points to the same extent the h models do, the h' models will have lower variance than the h models—the models with the noisy points weighted down have better bias and variance.

Of course, the above does not hold in general, even if we stipulate that the points in $\mathbf{X}_{\text{noisy}}$ can be reliably identified. We assume above that we have deterministic training and infinite capacity—it is easy to imagine a case where a linear decision boundary is pivoted to accommodate a noisy point and the boundary on the far side of the pivot changes the prediction associated with a new area of the feature space from incorrect to correct; the incorrect accommodation of the point in this case would improve the model's performance. Additionally, improved variance does not guarantee a better expected generalization error in all cases [25]; a higher-variance model can have a lower variance effect on the expected prediction error than a lower-variance model. Mislabeled points could also drag the decision boundary over regions that were previously being predicted incorrectly (possibly even because no data had been observed there), improving the expected prediction error over the whole domain.

With that said, many modern machine learning methods—especially NNs—have enormous model capacity; two-layer NNs can approximate arbitrary functions in the infinite-width limit [30]. We expect that when working with data from real-world distributions, when we weight down points with high label uncertainty, we will obtain models with improved decision boundaries in practice.

4 ESTIMATING POINTWISE LABEL UNCERTAINTY

4.1 Requirements for the Uncertainty Estimation Function

Of course, identifying which points are uncertain and informative using the above definition (Section 3.1) would require knowledge of the generating processes or other information that could be difficult or impossible to obtain. Instead, we estimate which points are likely to be uncertain by examining each sample's neighborhood within the available dataset, defined by a parameter $k \in \mathbb{N}$, the neighborhood size (by the number of neighbors, including the point itself). We define a scoring function to assign a value to each point based on the entropy of observed classes within its neighborhood and the relative sparsity of the neighborhood, with the intention that the value is indicative of the uncertainty of the point's label. The score should have the following properties:

(1) A data example should have a score of 0 when all $k - 1$ neighbors are of the same class as the example.
(2) Examples in highly heterogeneous neighborhoods (i.e., neighborhoods with a high number of classes present) should have higher scores than points in homogeneous neighborhoods consisting of mostly their own class, but lower scores than points in homogeneous neighborhoods consisting of points of mostly another class.
(3) Examples in relatively dense neighborhoods should have higher scores than points in relatively sparse neighborhoods, with label composition held constant.

The intuition for these requirements follows from our goal to use only the information contained in the dataset, i.e., the neighborhood of each sample, to estimate its label uncertainty. If all other points in a given point's neighborhood are of the same class as the point, we choose to trust its label. Its neighborhood score should be 0, indicating no uncertainty. If a point is in a dense region of the feature space and its neighbors are all of another class, we should be highly suspicious of its label being potentially incorrect. The second and third requirements follow from how we think of noisy regions. Points with highly diverse labels in their neighborhood—especially in a dense neighborhood—are more likely to not be of the class most likely to generate a sample at that point in the domain. The presence of many classes in the same neighborhood indicates that several class processes could generate samples in that region and that the model should put less weight on such samples when drawing the decision boundary. Performance gain from adjusting to accommodate those points is unlikely to generalize because the region is chaotic. Both informative and uncertain samples near class boundaries will have nonzero scores, as they will have neighbors with different labels.

4.2 Incorporating Neighborhood Uncertainty Scores into the Loss Function for Classification

After a neighborhood is analyzed from the view of label uncertainty, we need to take a further step to perform classification. Uncertainty scores are converted into sample weights via a logistic mapping function and incorporated into the objective function optimized during model training: Let $\mathcal{L}(\mathbf{X}, \Theta)$ denote the objective function without sample weighting, where $\mathbf{X}$ is the set of all data points $(\mathbf{x_i}, y_i), i \in [0, N)$, and Θ represents the model parameters. If $\mathbf{b}$ is the length-n vector containing the neighborhood scores for each $(\mathbf{x_i}, y_i) \in \mathbf{X}$, and $g(\cdot)$ is the logistic mapping function taking neighborhood scores to sample weights, then our objective function becomes

$$\mathcal{L}^*(\mathbf{X}, \Theta) = \frac{1}{N} \sum_{i=1}^{N} g(b_i) \mathcal{L}(\mathbf{x_i}, \Theta). \tag{5}$$

4.3 Calculation of Neighborhood Uncertainty Scores

We calculate the score for a sample $\mathbf{x_i}$ with label y_i as follows:

$$b_{\mathbf{x_i}} = \frac{-C * (\frac{k_{y_i}}{k} \log \frac{k_{y_i}}{k} * \frac{k_{y_i}}{\sum \mathbf{d}_{\mathbf{x_i}}})}{-\sum_{j=1}^{C} (\frac{k_j}{k} \log \frac{k_j}{k} * \frac{k_j}{\sum \mathbf{d_j}})}, \tag{6}$$

where C is, as before, the number of classes in the dataset, k is the number of neighbors that we consider for each sample, k_{y_i} is the number of neighbors with the same label as $\mathbf{x_i}$, and k_j is the number of neighbors with class label y_j. $\mathbf{d}_{\mathbf{x_i}}$ is a distance vector that stores the normalized distances to the k_{y_i} neighbors with the same class label as $\mathbf{x_i}$. The normalization is performed by setting the distance to $\mathbf{x_i}$'s nearest neighbor to be 1, and scaling the distances to the other neighbors based on that value. The terms $\frac{k_{y_i}}{\sum \mathbf{d}_{\mathbf{x_i}}}$ in the numerator and $\frac{k_j}{\sum \mathbf{d_j}}$ in the denominator are included to weight the class self-information in the formula by the average inverse distance to the neighbors of that class (to reduce the influence of far-away points). The denominator of this formula equals 0 if all neighbors of the sample $\mathbf{x_i}$ have label y_i. In these cases, we define the value of b_{x_i} to equal 0.

For a point $\mathbf{x_i}$, this calculation compares the entropy of labels in $\mathbf{x_i}$'s class, y_i, to the expected entropy of labels in the neighborhood. This meets our established requirements:

(1) When all $k - 1$ neighbors have the same label, the entropy in the denominator of equation (1) is 0, and $b_{\mathbf{x_i}}$ is defined to be 0.

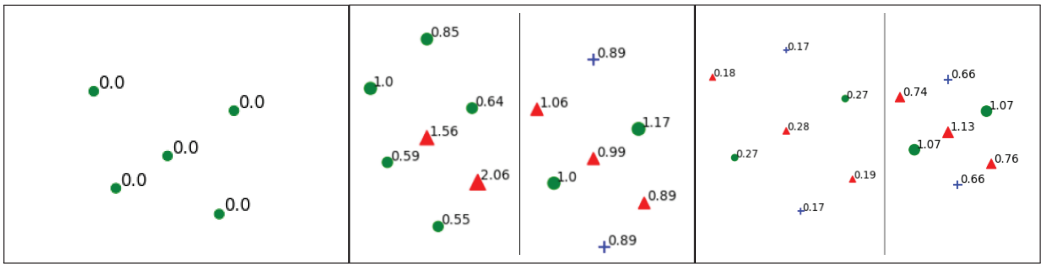


Fig. 3. Three example figures demonstrating neighborhood scores, calculated in the two-dimensional (2D) plane with Euclidean distance. Each corresponds to one of the requirements we have of a scoring function. The far left figure shows a neighborhood where each point has the same label; all are assigned a score of 0. The central figure depicts two distinct neighborhoods with identical spacing but different label distributions. The neighborhood with a more even mix of labels represented has a more even distribution of scores. The right figure shows two copies of the same neighborhood, with the relative spacing held constant but the distances between points increased by a constant factor. (Spacing increased 4x for score calculation, and pictured at 2x for readability.)

(2) In a neighborhood that is highly heterogeneous, the total number of points with each label is similar. (If one class label had many more points than the others, the neighborhood would not be highly heterogeneous.) Therefore, the entropy term for the label of point $\mathbf{x_i}$, $\frac{k_{y_i}}{k} \log \frac{k_{y_i}}{k}$, is close to the expected entropy over all labels, represented in the denominator of equation (1). Additionally, inverse average distances are similar over all classes in such regions, so the neighborhood score is close to 1 for each point in the neighborhood, giving us the desired effect.

(3) By weighting the terms corresponding to each class j by the inverse average distance from $\mathbf{x_i}$ to its neighbors in class j, we reduce the effect of sparsely represented classes in the neighborhood and increase that of denser classes.

Example values can be found in Figure 3.

4.4 Mapping Neighborhood Scores to Sample Weights for Classification

There are potentially many ways to map neighborhood scores to sample weights. The neighborhood scoring function has a minimum value of 0 (for a point in a fully homogeneous neighborhood).

The logistic function (and especially the sigmoid function, a special case of the logistic) is commonly used in machine learning as a "squashing" function to ensure output values fall in a certain range [18]; we use a negative logistic function to transfer neighborhood scores to the desired range for sample weights, because it allows us to generate high weights (>1.0) on points with a low neighborhood score and low weights (<1.0) on points with a high score. This is exactly the behavior we want—points with low uncertainty are weighted up and those with high uncertainty are weighted down. See the Section 5 for further examination of this relationship.

The hyperparameters that define this function do need to be tuned based on data. We find that logistic functions of the form similar to the following fit our requirements:

$$g(b_i) = \frac{\gamma}{1 + e^{-\alpha(-b_{x_i} + \beta)}} + \eta. \tag{7}$$

Here, β controls the value of neighborhood score that the logistic function is centered on. We find empirically that the median of the nonzero scores is a good initial value for this parameter,

and tends not to needlessly downweight useful samples by considering too many of them to be uncertain. α controls the steepness of the logistic curve, the "hardness" of the threshold that separates an informative point that is upweighted from a uncertain point that is downweighted. γ and η take the score values and map them to values in the range $[\eta, \eta + \gamma]$, such that the low values of scores (near 0) are mapped to nearly $\eta + \gamma$, and high values are mapped to η.

We find empirically that the sample weights should fall in a range from $\approx$0.25 to $\approx$2.0; allowing weights to go to 0 effectively shrinks the available dataset, reducing performance. Upweighting samples beyond 2.0 tends to overfit those samples too much when applied to our dataset.

5 EXPERIMENTS

5.1 Case Study 1: Classifications of a Real-World Physical Activity Dataset

Objective and accurate measurement of physical activity is a critical requirement for a better understanding of the relationship between sedentary behaviors, physical activity, and health [7, 34]. We evaluate our method on a physical activity recognition dataset collected from hip-mounted, triaxial accelerometers from a cohort of 184 child participants. There were 98 male subjects from ages 8 to 15 and 86 female subjects from ages 8 to 14. Each subject was observed for a period of lying rest with median 17 minutes (maximum 30 minutes), and median 4 minutes for each other activity (maximum 10 minutes). Researchers observed each activity and recorded the activity performed and the start and end times of each bout, so the data feature ground-truth segmentation. We split each bout into discrete 12-second windows of activity described with the output of a single triaxial accelerometer running at 1 Hz, resulting in 36 features per sample. We have 11,543 samples, and calculate the neighborhood scores using $k = 5$ and cosine similarity.

Like many real-world applications, the labeling process is difficult and comes with significant uncertainty. Labeling is performed based on both in-person and video observations, and classes are often difficult to distinguish. There are five classes in our analysis: sedentary, light household and games, moderate–vigorous household and sports, walking, and running, and they are superclasses of the full label set, which consists of Computer Games, Reading, Light Cleaning, Sweeping, Brisk Track Walking, Slow Track Walking, Track Running, Walking Course, Playing Catch, Wall Ball, and Workout Video. Even among the superclasses, there are typically samples from different classes that appear to be very similar (e.g., walking across the house during a "light household" sample and walking across a basketball court during a "sports" sample). Using the even more fine-grained labeling approach would introduce more noise and drastically reduce the amount of data available per class, making it impractical.

5.1.1 Grid Search Validation for Neighborhood Scoring Function. In our first set of experiments, we aim to validate our scoring function, and show that the samples with high scores are in fact the uncertain samples and that weighting them down improves performance. To do this, we calculated the neighborhood score b_{x_i} for each sample in the training set, and assigned those samples to groups. First, we put all samples x_i for which $b_{x_i} = 0$ into group $G0$; a score of 0 indicates that a point's entire neighborhood is from the same class as itself. This is the zero-uncertainty group: Our method considers points in fully homogeneous regions to have no label uncertainty. We then take all the remaining points, sort them by score, and divide them in half such that we have two more groups: $G1$, those points with low but nonzero scores, and $G2$, those points with the highest scores. Points that are close to decision boundaries but are not uncertain (by the definition in Section 3.1) should have low, nonzero scores, and points that are misleading should have high scores, so we aim to separate points around class boundaries into informative and uncertain points with this split. We perform this process to test if there is an advantage to downweighting, leaving the same, and upweighting the three groups split by estimated uncertainty; we need to make sure

that our intuition to upweight samples with low uncertainty and downweight samples with high uncertainty holds in practice. We refer to experiments performed on data split this way as neighborhood or NB-weighted experiments. Of course, no test examples are ever used in the calculation of the uncertainty scores; inclusion of those examples would leak information about the test set into the training process whenever a test example was in the neighborhood of a training example.

For comparison, we create a second split into three groups (such that the sizes of the groups correspond to the sizes of the NB-split groups) but assign samples to groups randomly. We refer to experiments with these splits as having been run with "random assignment" groups. We do this to make sure that the results we observe are due to our method and not to chance. By running the whole suite of experiments a second time on randomly split groups, we can observe the distribution of results from the random assignments to see what variations in performance we should expect due to randomness. We can then compare our NB-weighted results to make sure they are significant.

We perform a grid search to evaluate each possible combination of sample weight assignment and groups, using five discrete weights chosen to cover a range of weighting options but not leave large gaps: 0.25, 0.6, 1.0, 1.5, and 2.0. These values are chosen as proxies for the following possible ways to adjust the sample weights for a group: strongly downweight, somewhat downweight, no adjustment, somewhat upweight, and strongly upweight.

This experiment is intended to show two main points: (1) that the score values capture useful information about the data's feature space, and (2) that our interpretation of the score values is consistent with observed performance differences in the grid search; i.e., weighting up the zero-score group (those points we identify as having no label uncertainty) and weighting down the high-score group (those points we identify as having uncertain labels) outperformed other sample weight-group assignments. The five weights were assigned to the three sample groups, for both assignment schemes, in all possible combinations. Each combination was run 10 times and the results (as measured against a fixed held-out test set) were averaged, for a total of 2,500 model runs ($5^3 = 125$ total combinations of weight assignments, $*10$ runs per assignment, $*2$ experimental conditions per combination—assignment = 2,500 runs).

Training was performed using the Keras library [6] and the Theano backend using single-threaded CPU computation only. This step was taken to remove nondeterminism introduced by multithreaded CPU context switching and cuDNN. With these settings, a run with fixed sample weights and a random seed is deterministic, and will finish training with the exact same result each time. A total of 10 random seeds were generated once at the beginning of the experiment, and the same 10 seeds were used for each combination of sample weight assignments to reduce the effect of particular combinations having stronger performance due to a lucky set of initializations within the weight space. We use a simple two-layer Multi-Layer Perceptron architecture to keep running time reasonable.

The results of this process are shown in Table 1. Reported figures represent a change in model performance when using various weighting configurations compared to a baseline model that was trained with no sample weighting (all weights = 1.0). The performance of the baseline model was 83.4%, averaged over 10 runs. While there is no discernible pattern in the random results, as we would expect, there is a clear pattern in the results when k-nearest-neighbor-based weighting is used: Performance is strong when the most uncertain points (group 2) are weighted down (G2 weighted down in all five of the top combinations), and performance is weak when the uncertain points are weighted up (G2 weighted up for all five of the bottom combinations). All five of the best neighborhood score combinations are better than the best one when weights are randomly assigned; all five of the worst weight combinations perform worse than the worst run under the random setting. This validates our interpretation of the neighborhood scoring function—weighting

Table 1. Results of Grid Search

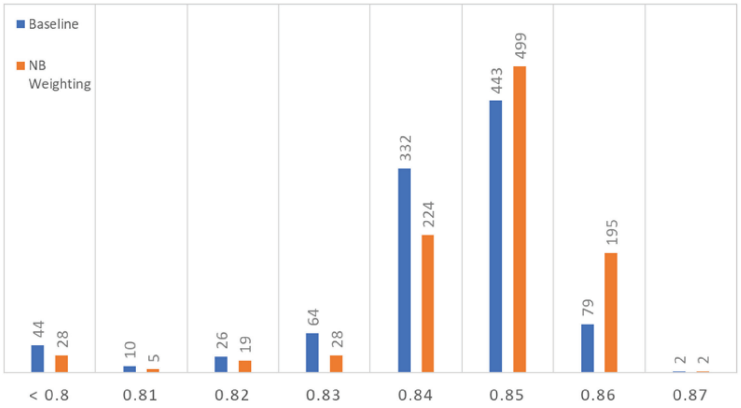
Top 5 Weight Combinations by Average Improvement over Baseline (%)					
Group Assignments	G0 / G1 / G2	Average over Baseline	Group Assignments	G0 / G1 / G2	Average over Baseline
NB Score	1.5 / 0.6 / 0.25	+0.92	Random	2.0 / 0.6 / 1.0	+0.74
NB Score	2.0 / 0.6 / 0.25	+0.91	Random	1.5 / 0.25 / 0.25	+0.71
NB Score	0.25 / 1.0 / 0.6	+0.85	Random	0.6 / 0.25 / 0.25	+0.69
NB Score	0.25 / 0.6 / 0.25	+0.84	Random	1.5 / 0.25 / 0.6	+0.68
NB Score	0.6 / 1.5 / 0.25	+0.77	Random	1.5 / 0.25 / 2.0	+0.68
Bottom 5 Weight Combinations by Average Performance Reduction from Baseline (%)					
Group Assignments	G0 / G1 / G2	Average under Baseline	Group Assignments	G0 / G1 / G2	Average under Baseline
NB Score	0.6 / 0.6 / 2.0	−3.7	Random	0.6 / 0.6 / 0.6	−0.82
NB Score	0.25 / 2.0 / 2.0	−3.7	Random	0.25 / 2.0 / 0.6	−0.86
NB Score	0.25 / 0.6 / 2.0	−4.5	Random	1.5 / 2.0 / 1.5	−0.87
NB Score	0.25 / 0.25 / 1.5	−5.4	Random	0.25 / 1.5 / 1.5	−1.1
NB Score	0.25 / 0.25 / 2.0	−8.6	Random	0.6 / 0.25 / 1.5	−1.9

down the points with highest scores improves model performance over baseline, and weighting up high-scoring points increases model focus on points with uncertain labels and decreases accuracy.

5.1.2 Evaluating k-Nearest-Neighbor Weighting against Baseline for Activity Recognition. Our second round of experiments takes the best *k*-nearest-neighbor weighting model with weights (1.5 / 0.6 / 0.25), and uses equation (3) to create a continuous mapping function from scores to weights, and then measures the performance of models trained with these weights against baseline models (where all sample weights = 1.0) more thoroughly. We choose 1,000 random seeds from integers in the interval [0, 100,000) and run an NB-weighted model and a baseline model for each one, under the same CPU-based calculation conditions as the models from the grid search, so that any difference in performance is directly attributable to the difference in the weighting scheme. This yields 1,000 NB-weighted models and 1,000 baseline models. We summarize the results in Figure 4. The histogram on the left of Figure 4 shows a distribution of trained models by performance, with better-performing models on the right. The absolute counts are provided in the table to the right of the histogram. We can see that the models using *k*-nearest-neighbor weighting have both improved performance on average (are further right) and have lower variance (are more clustered in the histogram). The NB-weighted models are on average +0.534% better than the baseline models by accuracy, and have greatly reduced variance $\sigma^2 = 2.53$ and standard deviation $\sigma = 1.59\%$, as compared to the baseline models' variance $\sigma^2 = 4.71$ and standard deviation $\sigma = 2.17\%$. Note that this is the variance of the model results, and is a different mathematical quantity from the model variance defined in Section 3; that variance is measured over models each trained on a different dataset sampled from a particular data domain. The variance calculated here is a variance over the model training process using a single dataset.

5.2 Case Study 2: High-Dimensional Experiments in Music Genre Recognition

To evaluate the behavior of the method on high-dimensional data and to further validate our method on a dataset from an alternative data domain, we perform additional experiments on a music genre recognition dataset called the Free Music Archive (FMA) [9]. We use the curated FMA small version, which consists of 8,000 song clips of 30 seconds each. There are eight balanced



Accuracy (%)	Baseline Models	NB Weighted Models
≤ 80	44	28
81	10	5
82	26	19
83	64	28
84	332	224
85	443	499
86	79	195
87	2	2

Fig. 4. Histogram of k-nearest-neighbor weighting versus baseline. After validating the approach with the grid search, we take the best sample weighting combination (1.5 / 0.6 / 0.25) and use equation (2) to create a continuous mapping of scores to weights, such that the continuous mapping follows the same weight pattern (low scores mapped to weight 1.5, high to 0.25). With $\gamma = 1.25$, $\alpha = 4$, $\beta = 1.13$, and $\eta = 0.25$, we run 1,000 pairs of models, each pair being one with neighborhood weighting and one with all weights set to 1.0, and both using the same random seed—model pairs see the same examples in the same order throughout training, and start from identical weight initializations. The models trained with k-nearest-neighbor weighting are better (clustered further right), on average, and exhibit lower variance over 1,000 runs. (Baseline models have $\sigma^2 = 4.71$ and $\sigma = 2.17\%$ and NB-weighted models have $\sigma^2 = 2.53$ and $\sigma = 1.59\%$.)

classes, each with 1,000 songs: Hip-Hop, Pop, Folk, Experimental, Rock, International, Electronic, and Instrumental. Music genres are similar to activities in that the labels have a similar gray area; one Rock song could be more Hip-Hop than Folk, and another more Folk than Hip-Hop. It is also reasonable to think that different, reasonable annotators could disagree on labels in this domain—a song being labeled Electronic versus Experimental, for example.

We import the songs in Python as NumPy [38] arrays using the librosa library [32], and resample them to 22,500 Hz (making each song an array of length $30 * 22,500 = 675,000$) and use dynamic time warping [2] with the Python FastDTW library [41] to calculate approximated pairwise distances between songs. We then use equation (6) to calculate the neighborhood scores. We translate them into sample weights using equation (7) with γ set to 1.75 and η set to 0.25, putting the scores in the range [0.25, 2.0], and setting β to 2.09903, the median value of the nonzero scores as recommended in Section 4.4.

Following [4], we use a 1D DenseNet-style model [24] to predict the genres. We use the NumPy representations of the songs to compute the mel spectrogram [8] of each song, a representation popular in deep audio processing (such as speech recognition [14]). Essentially, this method splits up the song into contiguous windows and performs a Fourier transform on each, and then maps those frequency magnitudes onto a scale (the *mel* scale) to normalize the pitch difference of the sounds based on human perception. We perform this preprocessing using the librosa library. We read in each song at a sample rate of 44,100 and split each one into 39 spectrograms of size (128, 128), 128 timesteps each containing a mel-scaled Fourier transform of length 128 giving us 20 spectrograms per song, and then we use a 50% stride to generate 19 more (the second half of the first discrete spectrogram is combined with the first half of the second to form a new example, and so on).

The train–validation–test split is provided by the dataset for reproducibility: 6,400 songs are train, 800 are validation, and 800 are test. This gives us 249,600 training examples and 31,200 each in validation and test. We run 25 models of each type, baseline and NB-weighted, and report the results in Figure 5. We use a small version of the DenseNet architecture, with two dense blocks

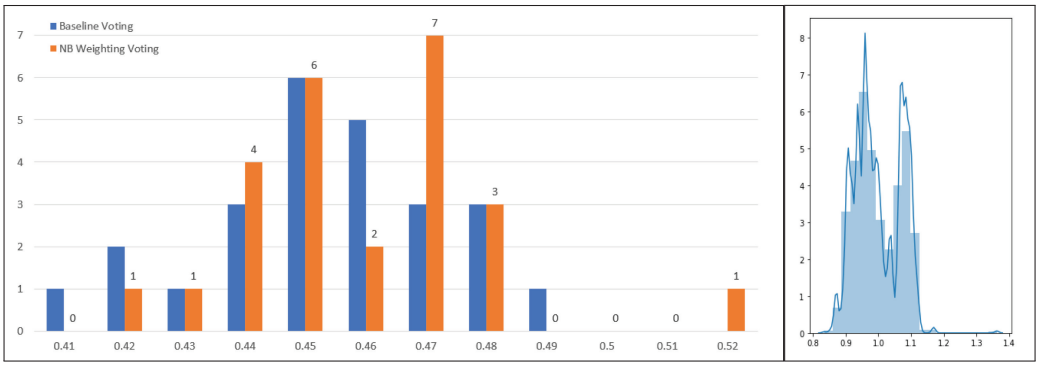


Fig. 5. Histogram of k-nearest-neighbor weighting versus baseline for 1D DenseNet models on the FMA dataset. The left histogram compares the accuracy of the models run with and without NB-scoring. The mean accuracy values are 45.36% for the baseline models and 45.88% for the NB-weighted models. The variance of the baseline distribution is 4.07, while the variance of the NB-weighted models is 3.78. The histogram on the right pictures the distribution of the sample weights used during training. The overwhelming majority of weights are between 0.9 and 1.1.

and two transition blocks alternating, for two reasons: (1) because we want the models relatively small so they can be trained quickly, and (2) because we observe empirically that increasing the model size has rapidly diminishing returns when training on these data. We use a batch size of 32 and a max learning rate of 0.01, which we vary in an increasing-then-decreasing fashion using a 1cycle policy as in [45]. We use L2 regularization (0.001) but no dropout, and do use batch normalization on the feature axis (the magnitude of the mel-scaled frequencies). We allow for early stopping and save the best model based on validation loss, but use a high patience (25 epochs). One-dimensional convolutional filters are set to size 3. Following [4], we generate a prediction for each spectrogram that came from each song using our deep learning model, and then assign final song-level predictions by choosing the genre with the highest representation among the spectrogram-level predictions.

We can see in Figure 5 that while the NB-weighted models are shifted slightly right of baseline, the distributions are very close. We observe slightly better performance, with mean accuracy values of 45.36% for the baseline models and 45.88% for the NB-weighted models. The variance is reduced as well: The baseline distribution has a variance of 4.07, while the distribution of NB-weighted models has a variance of 3.78, an improvement of 7%. We believe that the small effect is due to the curse of dimensionality that affects all neighborhood-based methods: As dimensionality increases, all data points become far away from each other. Here, this manifests itself as the calculated sample weights being tightly distributed around 1.0. The histogram of the sample weights is pictured in Figure 5, right.

The state of the art on this dataset is significantly higher than the performance we list here in Figure 5. The best performance we have seen is [4], with 69.8%, but they use a complex support vector machine (SVM)-on-top-of-DenseNet model and use significant data augmentation by pitch shifting the spectrograms. Our performance is in line with that seen in [29] (51.2%). [39] also achieve 56.8% using a transfer-learning convolutional NN.

6 CONCLUSION AND FUTURE WORK

With an eye on the bias–variance dilemma, we formulated a k-nearest-neighbor-based method to estimate pointwise uncertainty in labeling and mitigate its effects by weighting down the samples

in areas of the feature space with high density and label entropy. By working on the fundamental issue of the bias–variance dilemma (i.e., whether a decision boundary should accommodate a sample point according to the trustworthiness of that sample), we show improved model bias and variance in a real-world application. Using an NN architecture to classify accelerometer data for activity recognition, we improve performance in a real-world domain where accurate, consistent labeling is very difficult. We further validate our method on a higher-dimensional music genre recognition dataset using a different model type. In future work, we hope to improve the method we use for the estimation of label uncertainty, with the aim of obviating the need to calculate a distance matrix with k-nearest neighbors.

REFERENCES

[1] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. 2012. *Learning from Data*. AMLBook.

[2] Donald J. Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *Proceedings of the KDD Workshop*, Vol. 10. 359–370.

[3] Thomas B. Berrett, Richard J. Samworth, and Ming Yuan. 2016. Efficient multivariate entropy estimation via k-nearest neighbour distances. *The Annals of Statistics* 47, 1 (2016), 288–318. DOI: 10.1214/18-AOS1688

[4] Wenhao Bian, Jie Wang, Bojin Zhuang, Jiankui Yang, Shaojun Wang, and Jing Xiao. 2019. Audio-based music classification with DenseNet and data augmentation. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*. Springer, 56–65.

[5] Gilles Blanchard, Gyemin Lee, and Clayton Scott. 2010. Semi-supervised novelty detection. *Journal of Machine Learning Research* 11, Nov (2010), 2973–3009.

[6] François Chollet et al. 2015. Keras. Retrieved from https://keras.io.

[7] Scott E. Crouter, Jennifer I. Flynn, and David R. Bassett Jr. 2015. Estimating physical activity in youth using a wrist accelerometer. *Medicine and Science in Sports and Exercise* 47, 5 (2015), 944.

[8] Steven Davis and Paul Mermelstein. 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28, 4 (1980), 357–366.

[9] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. 2017. FMA: A dataset for music analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*. DOI: https://arxiv.org/abs/1612.01840.

[10] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural Safety* 31, 2 (2009), 105–112.

[11] Wei Ding, Tom Stepinski, and J. Salazar. 2009. Discovery of geospatial discriminating patterns from remote sensing datasets. In *Proceedings of the 2009 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics.

[12] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2018. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning* 107, 3 (2018), 481–508.

[13] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. 2016. Robustness of classifiers: From adversarial to random noise. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. ACM, 1632–1640.

[14] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. 2005. Comparative evaluation of various MFCC implementations on the speaker verification task. In *Proceedings of the 10th International Conference on Speech and Computer*.

[15] Weihao Gao, Sewoong Oh, and Pramod Viswanath. 2017. Density functional estimators with k-nearest neighbor bandwidths. In *Proceedings of 2017 IEEE International Symposium on Information Theory*. IEEE, 1351–1355.

[16] Weihao Gao, Sewoong Oh, and Pramod Viswanath. 2018. Demystifying fixed k-nearest neighbor information estimators. *IEEE Transactions on Information Theory* 64, 8 (2018), 5629–5661.

[17] Stuart Geman, Elie Bienenstock, and René Doursat. 1992. Neural networks and the bias/variance dilemma. *Neural Computation* 4, 1 (1992), 1–58.

[18] Neil Gershenfeld. 1999. *The Nature of Mathematical Modeling*. Cambridge University Press, New York, NY.

[19] Jacob Goldberger and Ehud Ben-Reuven. 2016. Training deep neural-networks using a noise adaptation layer. In *Proceedings of the 5th International Conference on Learning Representations*.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. Retrieved from http://www.deeplearningbook.org.

[21] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

[22] Shixiang Gu and Luca Rigazio. 2015. Towards deep neural network architectures robust to adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15), San Diego, CA, USA, May 7-9, 2015*, Yoshua Bengio and Yann LeCun (Eds.). https://dblp.org/rec/journals/corr/GuR14.bib.

[23] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. 2005. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer* 27, 2 (2005), 83–85.

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4700–4708.

[25] Gareth M. James. 2003. Variance and bias for general loss functions. *Machine Learning* 51, 2 (2003), 115–135.

[26] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Regularizing very deep neural networks on corrupted labels. In *Proceedings of the 35th International Conference on Machine Learning*.

[27] L. F. Kozachenko and Nikolai N. Leonenko. 1987. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii* 23, 2 (1987), 9–16.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of IEEE* 86, 11 (1998), 2278–2324.

[29] Donmoon Lee, Jaejun Lee, Jeongsoo Park, and Kyogu Lee. 2019. Enhancing music features by knowledge transfer from user-item log data. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 386–390.

[30] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks* 6, 6 (1993), 861–867.

[31] Tongliang Liu and Dacheng Tao. 2016. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 3 (2016), 447–461.

[32] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference*.

[33] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. 2015. Learning from corrupted binary labels via class-probability estimation. In *Proceedings of the International Conference on Machine Learning*. 125–134.

[34] Yang Mu, Henry Z. Lo, Wei Ding, Kevin Amaral, and Scott E. Crouter. 2014. Bipart: Learning block structure for activity detection. *IEEE Transactions on Knowledge and Data Engineering* 26, 10 (2014), 2397–2409.

[35] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *Proceedings of the Advances in Neural Information Processing Systems*, J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc.

[36] Andrew Ng. 2017. The State of Artificial Intelligence. Retrieved May 14, 2018 from https://youtu.be/NKpuX_yzdYs.

[37] Curtis G. Northcutt, Tailin Wu, and Isaac L. Chuang. 2017. Learning with confident examples: Rank pruning for robust classification with noisy labels. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence (UAI'17), Sydney, Australia, August 11-15, 2017*, Gal Elidan, Kristian Kersting, and Alexander T. Ihler. AUAI Press. http://auai.org/uai2017/proceedings/papers/35.pdf.

[38] Travis E. Oliphant. 2006. *A Guide to NumPy*. Vol. 1. Trelgol Publishing.

[39] Jiyoung Park, Jongpil Lee, Jangyeon Park, Jung-Woo Ha, and Juhan Nam. 2018. Representation learning of music using artist labels. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR'18), Paris, France, September 23-27, 2018*, Emilia Gómez, Xiao Hu, Eric Humphrey, and Emmanouil Benetos (Eds.). 717–724. http://ismir2018.ircam.fr/doc/pdfs/168_Paper.pdf.

[40] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. *arXiv:1803.09050*.

[41] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[42] Clayton Scott. 2015. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*. 838–846.

[43] Clayton Scott, Gilles Blanchard, and Gregory Handy. 2013. Classification with asymmetric label noise: Consistency and maximal denoising. In *Proccedings of the Conference on Learning Theory*. 489–511.

[44] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

[45] Leslie N. Smith. 2018. *A Disciplined Approach to Neural Network Hyper-Parameters: Part 1—Learning Rate, Batch Size, Momentum, and Weight Decay*. Technical Report 5510-026. US Naval Research Laboratory.

[46] Tom Stepinski, Wei Ding, , and R. Vilalta. 2012. Detecting impact craters in planetary images using machine learning. In *Intelligent Data Analysis for Real-Life Applications: Theory and Practice*. IGI Global, 146–159.

[47] Sethu Vijayakumar. 2007. The Bias-Variance Tradeoff (PDF). Retrieved from http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf.

[48] Dawei Wang, Wei Ding, Kui Yu, Xindong Wu, Ping Chen, David L. Small, and Shafiqul Islam. 2013. Towards long-lead forecasting of extreme flood events: A data mining framework for precipitation cluster precursors identification. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1285–1293.

Received July 2019; revised July 2020; accepted October 2020