On the Robustness of Monte Carlo Dropout Trained with Noisy Labels

Purvi Goel
Facebook
purvigoel@fb.com

Li Chen
Facebook
lichen66@fb.com

Abstract

The memorization effect of deep learning hinders its performance to effectively generalize on test set when learning with noisy labels. Prior study has discovered that epistemic uncertainty techniques are robust when trained with noisy labels compared with neural networks without uncertainty estimation. They obtain prolonged memorization effect and better generalization performance under the adversarial setting of noisy labels. Due to its superior performance amongst other selected epistemic uncertainty methods under noisy labels, we focus on Monte Carlo Dropout (MCDropout) and investigate why it is robust when trained with noisy labels. Through empirical studies on datasets MNIST, CIFAR-10, Animal-10n, we deep dive into three aspects of MCDropout under noisy label setting: 1. efficacy: understanding the learning behavior and test accuracy of MCDropout when training set contains artificially generated or naturally embedded label noise; 2. representation volatility: studying the responsiveness of neurons by examining the mean and standard deviation on each neuron's activation; 3. network sparsity: investigating the network support of MCDropout in comparison with deterministic neural networks. Our findings suggest that MCDropout further sparsifies and regularizes the deterministic neural networks and thus provides higher robustness against noisy labels.

1. Introduction

Neural networks exhibit state-of-the-art performance on many learning tasks, such as classification and segmentation. However, training these networks requires an abundance of carefully labeled data; networks tend to overfit quickly to noise in training labels, which makes their application to noisy real-world problems less effective. Expert-labeled data is expensive and time-consuming to collect; label noise is common in less carefully crafted datasets due to measurement inaccuracies, human error, etc.

Nonetheless the latter type of data, albeit noisy, is more readily available. One recent strategy shown to perform well on datasets containing significant amounts of label noise is augmenting the neural network with an uncertainty estimation method like Monte Carlo Dropout [5]. These uncertainty estimation models display a delayed memorization effect of noisy training labels, and can generalize better to clean test data. Augmenting models with Monte Carlo Dropout shows a slower degradation of classification performance, consistent on benchmark datasets like MNIST and CIFAR-10[5]. In addition to its resilient performance against noisy training labels, MCDropout adds no training overhead and only adds minimal cost to inference time.

The robustness property and low-computational cost of MCDropout indicate it as an effective and practical solution against noisy labels. In this paper, our goal is to not only determine whether MCDropout performs consistently better in these noisy-label situations, but also provide an in-depth analysis for *why* it performs better. We present an investigation into the performance and representation learned by a model augmented with MCDropout. We first evaluate the accuracy of MCDropout models in comparison with deterministic neural networks on datasets like MNIST, CIFAR-10, and Animal-10n with artificially injected noisy labels. Second, we measure neuron responsiveness in each layer, to better explore the differences between representations learned by certainty and MCDropout models. Finally we study network sparsity and find that the sparsity property offered by MCDropout models contribute to robustness against noisy training labels. To our knowledge, our work provides the first detailed analysis of MCDropout in the setting of noisy labels.

The rest of the paper is organized as follows. In Section 2, we provide the background information on the noisy label setting, label noise taxonomies, MCDropout and related work. In Section 3, we describe our study directions including measuring efficacy, neuron responsiveness via volatility and network sparsity. In Section 4, we demonstrate the effectiveness of MCDropout on empirical datasets such as MNIST, CIFAR10 with artificially corrupted training labels and Animal-10n a real-world dataset containing annotation noise. We further analyze the neuron responsiveness and network sparsity by MCDropout in comparison with deterministic networks. Finally in Section 5, we discuss optimal

placement for MCDropout on a neural network and conclude our paper.

2. Preliminaries

In this section, we present the problem statement, the preliminaries on label noise, Monte Carlo Dropout and related work.

We consider a fully supervised learning problem in image classification, where the images and its associated labels in the training set, denoted by $\mathcal{T}_{train} := \{(X_i, Y_i)\}_{i=1}^{n}$, with n denoting the total number of training samples and all the pairs $\{(X_i, Y_i)\}_{i=1}^{n}$ sampled i.i.d from a joint distribution $F_{X,Y}$. However instead of observing all the correctly annotated labels, we observe the training data $\mathcal{T}_{tr} := \{(X_i, \hat{Y}_i)\}_{i=1}^{n}$, where given by a probabilistic process, $\hat{Y}_i$ deviates from Y_i. Our exploitation task is to learn a robust classifier on $\mathcal{T}_{tr} = \{(X_i, \hat{Y}_i)\}_{i=1}^{n}$ containing noisy labels such that the classification efficacy on incoming test image X can best predict the unknown label Y.

Across this paper, we refer to the deterministic neural network without uncertainty estimation as *certainty model* or *deterministic model* interchangeably. We refer to the neural network augmented with MCDropout layers as the *MC-Dropout model*.

2.1. Label Noise Taxonomies

There are several categorizations of noise labels. One commonly used categorization depends on whether or not the noisy label depends on the features. If the noisy label generation process is conditionally independent of the features, then a noise transition matrix $T_{c\times c}$, where c is the number of classes, is sufficient to describe the label noise generation process. Each entry in $T_{ij} = p_{ij}$ is a probability such that the true label will be changed into a noisy label with probability p_{ij}. If the observed label is different from the true label with a uniform probability, then the noise is considered to be label-independent and this noise is called considered symmetric or uniform noise. If the observed label is changed from the true label with probabilities depending on the original ground truth, then the noise is label-dependent and called asymmetric noise. On the other hand, if the corruption process depends on the features and labels, the label noise is called instance-dependent. A more recent study proposes a new but practical assumption within instance-dependent label noise, defined as part-dependent label noise, where the noise depends partially on an instance [27].

Another perspective on label noise is via uncertainty characterization [5]. The noisy label generation process is probabilistic and random. Naturally uncertainty characterization comes into play. From the notion of deep learning uncertainty, the noise in the labels can be considered a type of aleatoric uncertainty, a measurement of the intrinsic and irreducible uncertainty within the data. Within aleatoric uncertainty, homoscedastic uncertainty is constant across the input while heteroscedastic uncertainty is dependent on the input. Hence if the noise transition matrix is a uniform or symmetric one, then the label noise can be considered homoscedastic; if it is label-dependent, then the label noise can be considered heteroscedastic. In recent noise simulation schemes, label noise is applied on samples that are more likely to be mislabeled given by pre-learned model [1]. We consider such type of noise as epistemic uncertainty, a term that describes uncertainty induced by models.

2.2. Monte Carlo Dropout

The deep learning uncertainty perspective to characterize label noise inspires us to study label noise via deep learning uncertainty estimation techniques. Chen et al [5] proposed using epistemic uncertainty estimation methods when learning with noisy labels. Comparing Monte Carlo Dropout, Bootstrap [16], Bayesian CNN upon Bayes by Backprop [4] and certainty neural networks trained in noisy label settings, the authors discovered that Monte Carlo Dropout (MC-Dropout) had a prolonged memorization effect and possessed the best classification performance on test set.

We also included Figure 1 as our motivational example here. Hence in this paper, we laser-focus on the study of why MCDropout possesses robustness against noisy labels in comparison with certainty models.

The core idea of MCDropout is to enable dropout regularization at both training and test time. With multiple forward passes at inference time, the prediction is not deterministic and can be used to estimate the posterior distribution. As a result, MCDropout offers Bayesian interpretation. First proposed in [8], the authors established the theoretical framework of MCDropout as approximate Bayesian inference and proved MCDropout minimises the Kullback–Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process. More formally, let d_l denotes dropout at the l-th layer of a neural network, where $d_l \sim Bernoulli(p)$. Then at inference time, with K forward passes, we obtained a distribution of K logits and predictions per test data, where we can compute the expected value, standard deviation, variation ratio and entropy to assess uncertainty.

2.3. Related Work on Deep Learning with Noisy Labels

While there has not been much work on applying epistemic uncertainty methods to address noisy labels, an abundant of research has been done in deep learning noisy labels ranging from loss function adjustment, robust architecture design, data processing, data filtering and so on. Authors in [10, 29, 26, 19] devised robust loss function to achieve a smaller risk for unseen clean data when learning with noisy

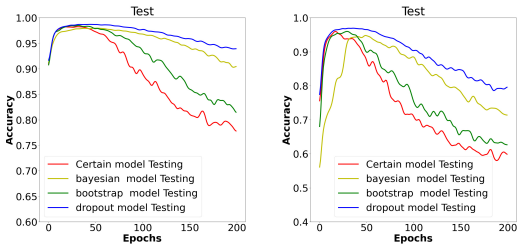

Figure 1. (Left): MNIST test accuracy when training labels contain 15% noise. (Right): MNIST test accuracy when training labels contain 35% noise. Our previous study suggests that MCDropout has the best classification performance among a few other uncertainty estimation methods. Further MCDropout does not increase training time per epoch and has relatively cheap inference cost. Hence in this paper, we focus on investigating the robustness of MCDropout when training with noisy labels.

labels. Sample selection techniques as to filter the clean labels for training and removing the noisy labels have been proposed in [14, 11, 28, 21]. Sample selection and label correction for spatial computing is studied in [6]. Devising loss to estimate noise transition matrix and correct the labels are studied in [23, 12, 2]. Semi-supervised learning is another field of techniques on noisy labels, where the noisy labelled data are treated as unlabeled and clean labelled data are as labeled [22, 7, 18].

3. Investigation

Our goal is to analyze the representations produced by MCDropout models, particularly in comparison with certainty models, trained in the presence of noisy labels. Similar to the definitions presented in Bau et al [3], we use the term *representation* to describe the outputs of a particular layer in a model. More specifically: which channels of the layer have been activated for various data inputs? How strongly have these channels been activated? What is the variation in a specific channel's possible activations? Comparing the representations lends insight and intuition as to why one model may perform better than another one. Essentially, we investigate why MCDropout performs better than a certainty model by comparing the different representations of the two models respectively.

Again following the vocabulary used in Bau et al [3], we refer to feature maps as the output of every layer in the network–the aggregate of the feature maps makes up the network's learned representation. We refer to a neuron as a specific channel of the feature map. In this paper, we use the term *activation gamut* to refer to all the possible values that a particular neuron can produce. We can approximate the activation gamut as the set of a neuron's activation values for each image in a dataset.

We compare the classification efficacy, neuron responsiveness, and network sparsity by the two models respectively. To understand how two models have learned and encoded information differently, we evaluate trained models on test set and cache neuron activations from each layer, where we derive statistics such as mean and standard deviations on each neuron with respect to data samples from the test set.

3.1. Measuring Efficacy

We first train MCDropout and certainty models on training data with noisy labels and evaluate their accuracy on a cleanly labeled test set. We present the learning behaviors during training and testing over epochs.

3.2. Measuring Responsiveness

Next we compare neuron responsiveness measured by volatility in the two models. We define volatility as the standard deviation of a neuron's activations over a dataset; if a neuron is capable of producing vastly different activation values for different input images, the neuron's activation gamut would possess high standard deviation, indicating a highly responsive neuron.

To compute the activation gamut of a neuron, we first cache the feature maps u_j^i , post-ReLu, produced by the i-th neuron on the j-th test set image. We find the mean activation value, a_i^j for the feature map u_j^i. In other words, for a feature map with n rows and m columns,

$$a_i^j = \frac{\sum_{r=0}^{n}\sum_{c=0}^{m} u_i^j(r,c)}{nm}.$$

Per neuron, this results in j values which compose its activation gamut A_i

$$A_i = \{a_i^0, a_i^1 ... a_i^j\}.$$

We can perform statistical analysis on these gamuts and aggregate them per-layer, such as finding the mean activation value V_l of all I neurons in the l-th network layer:

$$V_l = \frac{\sum_{i=0}^{I} mean(A_i)}{I}.$$

We also find the average gamut standard deviation S_l for all I neurons in the l-th network layer

$$S_l = \frac{\sum_{i=0}^{I} std(A_i)}{I}.$$

We would observe the activation gamut of a volatile neuron to possess a higher standard deviation than that of a non-volatile neuron. The activation gamut of a volatile neuron may also include extremes, showing a higher maximum activation than a non-volatile neuron.

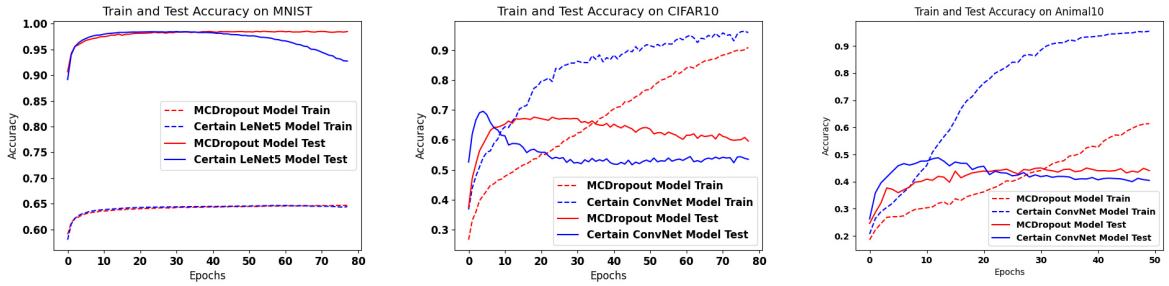

Figure 2. We show the train and test accuracies per epoch for (Left) the certain and MCDropout LeNet5 models on MNIST; (middle) the certain and MCDropout ConvNet models on CIFAR10; (right) the certain and MCDropout ConvNet models on Animal-10n. All of these models are trained in a noisy-label setting with 35 percent noise. In all these cases, the MCDropout model has a better validation accuracy than the certain model; the certain model generally overfits to training set noise.

3.3. Measuring Sparsity

Along with research directions into network uncertainty and robustness, neural network sparsity has become a subject of interest for many machine learning researchers [9, 13, 25]. Sparse neural networks are desirable because they require less computation at test time, demand less memory [9], and are less likely to overfit to training data [20]. In the context of our investigation, the tendency for sparse neural networks to overfit more slowly to training data can allow them to avoid memorizing noisy training labels. We can evaluate network sparsity on a per-neuron level: which neurons never or rarely activate, for any and all test samples, and how common are these neurons throughout the entire model? Network sparsity can be defined as the subset of neurons output a value that is always zero [9], or very close: these neurons do not affect the final predictions in any significant manner. The larger the subset of neurons with this property, the more sparse the network's learned representation is. Because neural networks can easily overfit to noise in training labels [5], we are interested in the observed property of sparse models to overfit more slowly. With fewer tunable parameters available, sparse models have fewer degrees of freedom to overfit to noise.

4. Results

We study the classification efficacy, neuron responsiveness, and network sparsity of MCDropout and certainty model on three benchmark classification datasets: MNIST, CIFAR10, and Animal-10n [24]. We use two different architectures: LeNet5 [17] and ConvNet, a CNN architecture with 4 convolutional layers followed by 3 fully connected layers. To maximize the effect of MCDropout, we use an all-layer MCDropout architecture where each layer in the certainty model is augmented with MCDropout. Our investigation compares the findings for the original certainty model and its augmented all-layer MCDropout model.

We train both models on noisy training labels. Because we are evaluating these models on a classification task, mislabeled data simply means that training samples labeled with the incorrect class. We use a uniform noise simulation scheme to add noise to 35% of our training labels: in this scheme, each corrupted label has an equal chance of 35% being mislabeled as any of the other classes. Once training is complete, we run our trained models on clean test data and compute all the neurons' individual activation maps after the application of an activation function. All of our chosen architectures use ReLu as their activation.

4.1. Classification Efficacy

We compare the performance of the certainty model and MCDropout models trained on noisy data and plot their training and accuracy curves over time 2. We see an emerging trend consistent across all models and datasets. The certain model overfits to the noise in the training data and results in a similar or higher final training accuracy than the MCDropout model. However, the MCDropout model consistently produces a higher validation accuracy.

Next we investigate why MCDropout outperforms certainty by analyzing the representations learned by both models. Consider the results on MNIST shown in Figure 2, left. Given that the training accuracies of the certainty and MCDropout model are quite similar after 100 epochs, both models are clearly learning *something*. However given the vastly different test accuracies between the two models–the uncertain models undoubtedly generalize better to the test dataset–the models are representing information *differently*.

4.2. Neuron Responsiveness Measured by Volatility

Next, we compare the volatility of neurons in uncertain and certain models. We cache each neuron's activation map for every image in the test set. We find the mean activation value for each feature map. To measure volatility, we compare two statistics per layer: the standard deviation of the layer's mean activation values and the mean of the layer's mean activation values.

Metric	Model	conv0	conv1	fc1	fc2	fc3
Activation STD	Certain MNIST	0.215	0.5367	2.386	1.335	1.5733
	MCDropout MNIST	**0.0646**	**0.1085**	**0.4207**	**0.2144**	**0.9182**
Activation Mean	Certain MNIST	1.009	1.3936	1.4381	0.8383	-0.0324
	MCDropout MNIST	**0.2443**	**0.2041**	**0.1567**	**0.1208**	**-0.498**
Unresponsive neurons	Certain MNIST	0.0	0.0	0.0916	0.0	0.0
	MCDropout MNIST	**0.1666**	**0.25**	**0.5083**	**0.2023**	0.0

Table 1. Quantitative metrics for our MNIST experiment, with the same table layout as described in Table 2. For all layers, the certain model's activations posses higher means and standard deviations, while the uncertain model has more relatively unresponsive neurons and lower activation values. Notice how several of the layers in the certain model have *no* relatively unresponsive neurons.

Metric	Model	conv0	conv1	conv2	conv3	fc1	fc2	fc3
Activation STD	Certain ConvNet	0.0602	0.0343	0.1715	0.1279	7.3578	11.8364	6.5248
	MCDropout ConvNet	**0.047**	**0.0123**	**0.0708**	**0.0871**	**4.3155**	**9.2449**	**4.480**
Activation Mean	Certain ConvNet	0.0818	0.04378	0.238	0.106	1.6077	5.038	0.075
	MCDropout ConvNet	**0.060**	**0.0149**	**0.091**	**0.0616**	**0.4894**	**3.340**	**-2.424**
Unresponsive neurons	Certain ConvNet	0.4166	0.4583	0.4323	0.4414	0.5449	**0.2031**	0.0
	MCDropout ConvNet	**0.4583**	**0.71875**	**0.7083**	**0.6172**	**0.7851**	0.0781	0.0

Table 2. Quantitative metrics for our CIFAR10 experiment. We show standard deviations of the mean activation value per neuron (top), the average of the mean activation value per neuron (middle), and a ratio of the ConvNet layer's neurons that are relatively unresponsive (bottom). We accumulate metrics across all test samples. In general, the certain model's layer activations have higher standard deviation and mean (suggesting greater volatility). The uncertain model is less volatile and has more unresponsive neurons (suggesting sparsity).

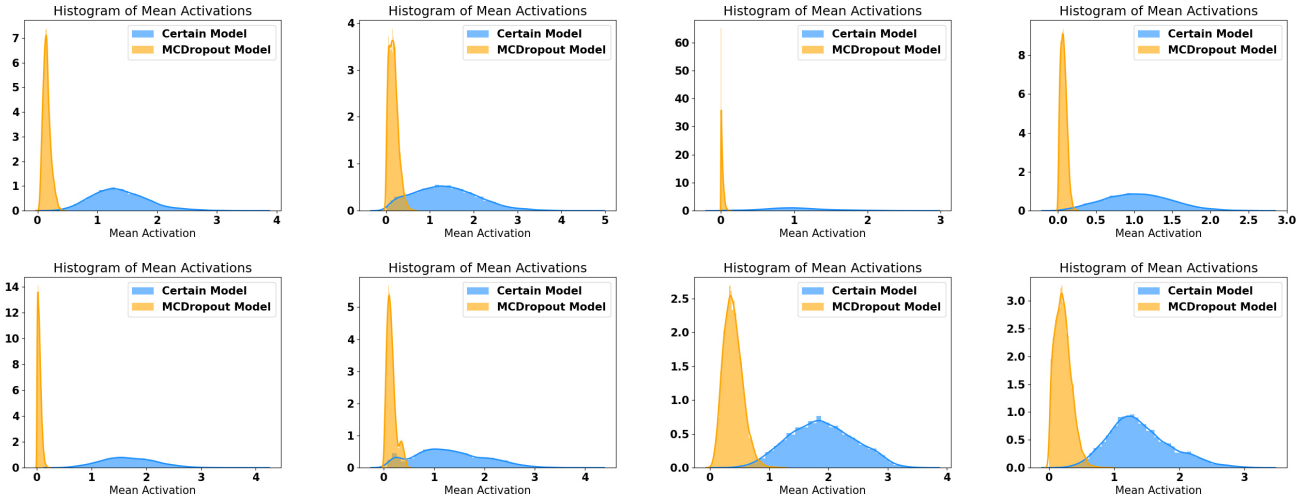

Figure 3. Histograms for our MNIST experiment, with the same set-up described in Figure 4. We show 8 neurons from the second convolution layer. Several neurons from the uncertain model (orange) are unresponsive to all inputs: notice how many of the uncertain activation distributions are centered around 0, with shorter tails. This suggests that the uncertain model is sparser. We include a quantitative summary of activation values over the entire test dataset in Table 1

We show the results of this investigation for each dataset in Table 1 for MNIST, Table 2 for CIFAR10 and Table 3 for Animal-10n. With the exception of a few layers, neurons in certainty models activate more strongly and with more variance: the mean activation and standard deviation is higher.

4.3. Network Sparsity

We can compare the sparsity via analysis of the neurons' individual feature maps. Sparser models contain more neurons whose feature maps have values close to some constant **c**, usually 0, no matter the input sample from the test set.

For qualitative evaluation, we visualize the post-activation feature map of individual neuron for a given sample of testing data. We can visually compare how many neurons seem near constant or activate in only small patches of the map. We show heatmaps from various layers in Figure 6 (MNIST), Figure 7 (CIFAR10), and Figure 8 (Animal-10n). In all cases, activation maps from MCDropout mod-

Metric	Model	conv0	conv1	conv2	conv3	fc1	fc2	fc3
Activation STD	Certain ConvNet	0.2037	0.0202	0.0596	0.0304	2.077	5.071	5.695
	MCDropout ConvNet	**0.0191**	**0.0132**	**0.0277**	**0.0301**	**1.6326**	**4.032**	**3.8372**
Activation Mean	Certain ConvNet	0.0217	0.0146	0.0599	0.0191	0.4833	3.017	-2.256
	MCDropout ConvNet	**0.0172**	**0.0086**	**0.0265**	**0.0148**	**0.210**	**1.534**	**-1.9296**
Unresponsive neurons	Certain ConvNet	0.625	**0.6354**	0.4583	**0.6367**	0.4804	0.1094	0.0
	MCDropout ConvNet	**0.6666**	**0.7708**	**0.7448**	0.4687	**0.6679**	**0.5**	0.0

Table 3. Quantitative metrics for our Animal-10n experiment, with the same table layout as described in Table 2. While less exaggerated than results shown on MNIST in Table 1 and CIFAR in Table 2, we see a similar trend. For the majority of layers, the certain model is more volatile, with a larger gamut of possible activation values and higher overall magnitude of activations, while the uncertain model has more relatively unresponsive neurons and a sparser representation.

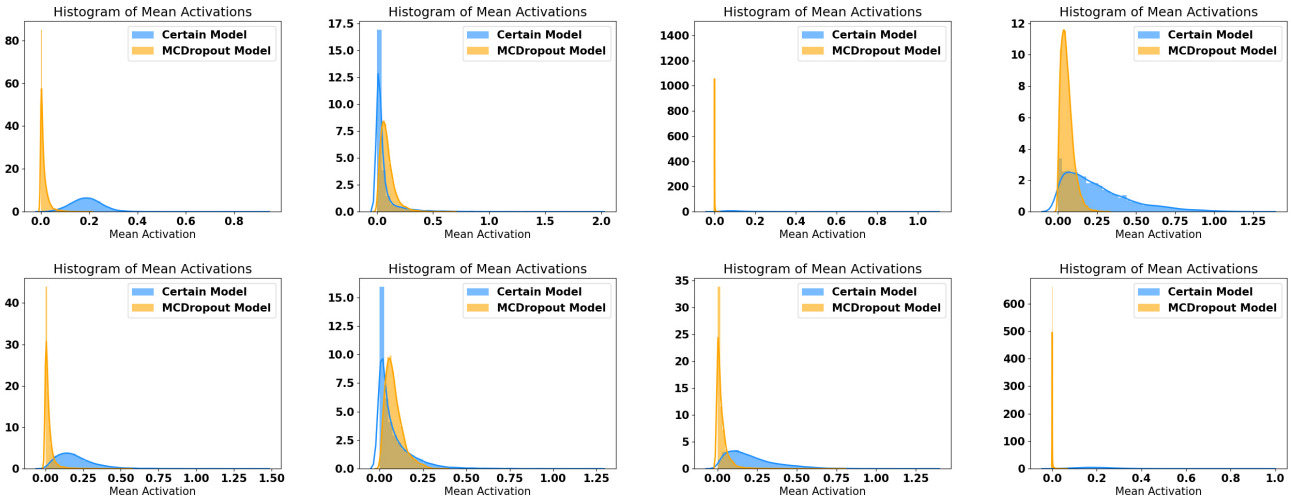

Figure 4. Histograms for our CIFAR10 experiment. We gather the mean activation values per neuron for each image in the CIFAR10 test set, for both certain and MCDropout models. We show 8 neurons from the 1st convolution layer; for each neuron, we plot a histogram of the mean activation values. Mean activations from the certain model are in blue; uncertain model are in orange. Notice how the range of the uncertain model's distributions are smaller: uncertain neurons produce a smaller gamut of possible mean activations. The uncertain model's distributions tend to be centered on or near 0. This suggests several of the uncertain model's neurons produce no activation no matter the input image. We choose 8 neurons to fit in page requirements; we include a quantitative summary of these metrics in Table 2.

els have spatially sparse activations: when they do activate, it is in tight, localized regions, and large patches of each activation map remain inactivated. In addition, several of the MCDropout activation maps show very little activation at all. We also calculate the mean activation value per neuron per image in each experiment's test dataset. We plot per-neuron histograms of these mean activation values for the certain and MCDropout models in Figures 3 (MNIST), 4 (CIFAR10), and 5 (Animal-10n). We can then compare the mean and support of the resulting activation distributions: in many cases, the distributions from MCDropout models possess smaller supports and are centered more closely around a mean activation value of 0.0. This provides an intuitive understanding of why MCDropout is more robust against noisy labels: the neurons that may be influenced by noisy labels in the certainty model are not activated as strongly in MCDropout models. MCDropout layers provide regularization against these "corrupted" neurons.

These qualitative traits show that the MCDropout model's learned representation is sparser. For a quantitative analysis, we can count how many neurons are "relatively unresponsive" based on their gamut of possible activations for all the test images. Neurons that rarely activate–that is, the mean of their activations for all images on the test set falls below some epsilon threshold–are tallied in the final row of Tables 1 (MNIST), 2 (CIFAR10), and 3 (Animal-10n). We report these numbers as the ratio of "relatively unresponsive neurons" to the total number of neurons in the layer. The results show that the major of the MCDropout models' layers have more dead neurons than corresponding layers in the certain model does. This indicates that the uncertain model has learned a more sparse representation.

5. Discussion

We have compared the representations, on a per-neuron level, of MCDropout models and certainty models when

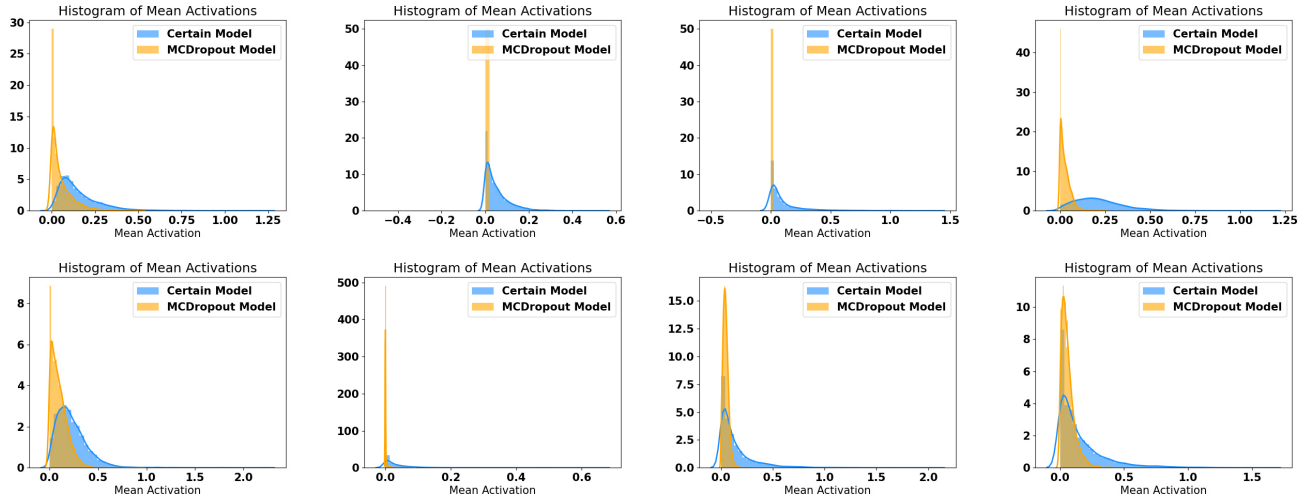

Figure 5. Histograms for our Animal-10n experiment, with the same set-up as Figure 4. We show neurons from the third convolution layer. Similar to Figure 4, several neurons from the uncertain model are unresponsive (result in mean activations near 0) to all inputs. This suggests that the uncertain model is sparser. We include a quantitative summary of activation values over the entire dataset in Table 3

.

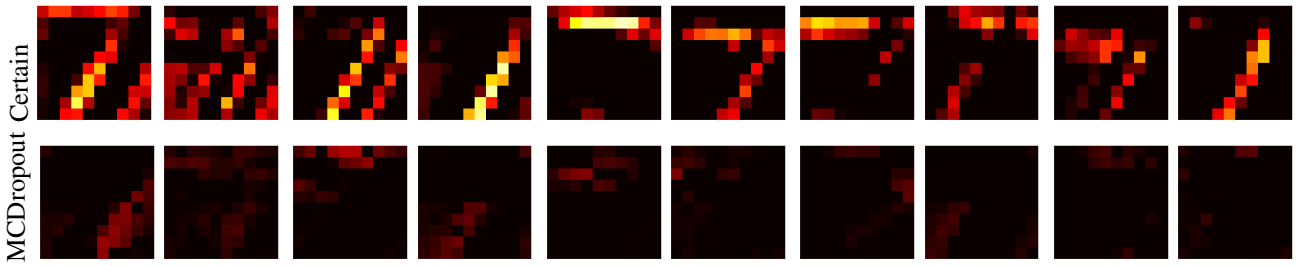

Figure 6. Activation maps for our MNIST experiment. We show the activations from 10 neurons on a random image from MNIST's test set, from the certain LeNet5's second convolution layer (top) and the MCDropout model's second convolution layer. Brighter values indicate higher activation values. The chosen layer contains 16 neurons total; for each model, we choose the 10 neurons with the highest mean activation value. The certain model activates much more strongly, while the MCDropout model's activations are more muted, and spatially sparse.

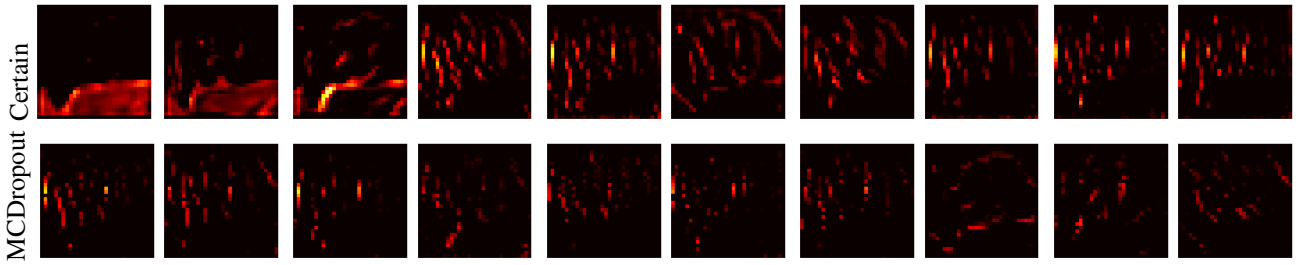

Figure 7. Activation maps for our CIFAR10 experiment. We show the activations from 10 neurons on a random image from CIFAR10's test set, from the certain ConvNet's first convolution layer (top) and the MCDropout model's first convolution layer. The chosen layer contains 48 neurons total; for each model, we choose the 10 neurons with the highest mean activation value. The brighter the value at a pixel, the higher the activation at that point. We see on that MCDropout has smaller overall activations: there are no bright yellow areas in the bottom row's feature maps, while we do for several maps in the top row. In addition, the activations are more spatially sparse: many of the displayed MCDropout model's featuremaps have entire regions with no activations. This is not as noticeable in the certain model.

trained with noisy labels. MCDropout's representation is less volatile but more sparse, a justification for its greater effectiveness and generalization in noisy-label scenarios. MCDropout provides regularization so that neurons are not overly influenced by the noisy labels; these neurons are not regularized at test time, contributing to robustness against

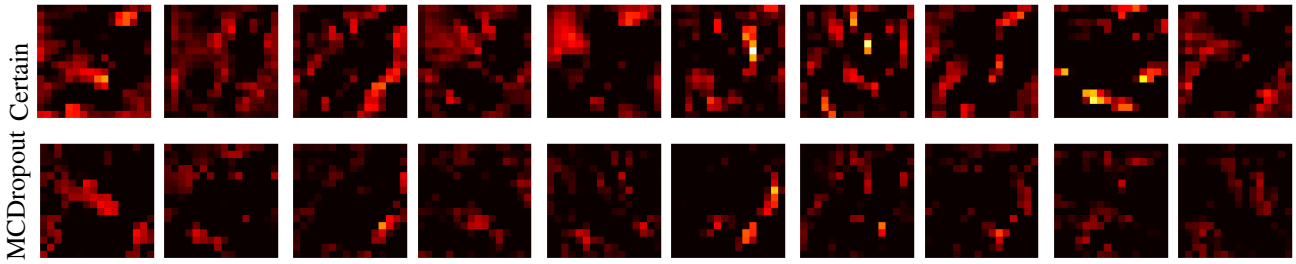

Figure 8. Activation maps for our Animal-10n experiment. We show the activations from 10 neurons on a random image from Animal-10n's test set, from the certain ConvNet's third convolution layer (top) and the MCDropout model's third convolution layer. Brighter values indicate higher activation values. The chosen layer contains 196 neurons total; for each model, we choose the 10 neurons with the highest mean activation value. While less obvious than the comparison in Figure 7 and Figure 6, the certain model still shows higher and less spatially sparse activations than the MCDropout model.

noisy labels. With fewer free parameters to over-explain training label noise, MCDropout models forge representations that are less capable of overfitting to noisy labels.

Our goal was not to build state-of-the-art models for the presented datasets or find the best network to deal with noisy labels. Rather, we investigate interpretable metrics and observations from the model responses that identify why MCDropout outperforms certainty models.

In our experiments, we primarily analyzed all-layered MCDropout for the purpose of maximizing the MCDropout effect. However we acknowledge there are other different configurations of uncertainty placement. As seen in Figure 9, we further analyze different MCDropout placement configurations on MNIST dataset and discover that MCDropout on all layers possesses the best test classification accuracy when training with noisy labels. Such behavior is consistent with the theoretical establishment that all-layered MCDropout best approximates Bayesian neural network. While other configurations such as converting only convolutional layers, internal layers, final layers, etc., to MCDropout layers still outperform the certainty model, the best-performing model benefits from the most number of MCDropout layers. We believe research directions on an optimal trade off between classification performance and MCDropout layer placement is critical for noisy-label training with constraints on memory or inference time.

We also comment on Dropout in comparison with MCDropout in the setting of noisy labels. While both remove a subset of neurons at random during training, at inference time, Dropout is deterministic but MCDropout estimates a posterior predictive distribution that is informed by the network–and perhaps affected by network properties that we have investigated, such as volatility and sparsity. MCDropout is able to offer extra information such as uncertainty and confidence via variation ratio, entropy, standard deviation to better detect samples with noisy or clean labels [5, 15]. Furthermore, during inference time, MCDropout conducts an ensemble of K forward passes. The ensemble procedure may contribute to less volatility but more unresponsive neurons, and thus additional robustness, compared with just Dropout during noisy label training. We plan to conduct thorough ablation studies to compare regularization, sparsity, and robustness between MCDropout and Dropout when training with noisy labels.

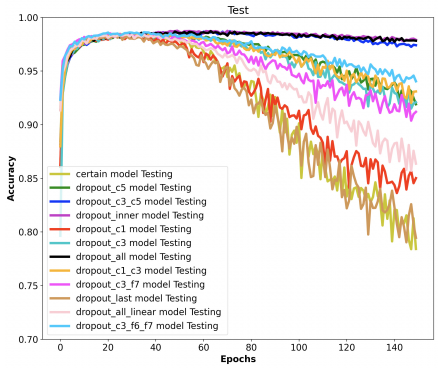

Figure 9. We further analyze different MCDropout placement configurations and discover that MCDropout on all layers (black line) possesses the optimal results against noisy labels, particularly compared to the certain model (yellow line).

We hope this investigation helps us ask and answer more questions. It would be interesting to see if our observations about volatility and sparsity hold for other uncertainty estimation or ensemble methods like Bootstrap [16] or Bayes by Backprop [4]. These observations may also help us tune MCDropout-related hyperparameters, such as the best locations to place MCDropout layers in a model architecture.

6. Acknowledgements

We thank the anonymous reviewers for their thoughtful suggestions. We thank Ilknur Kabul, Theofanis Karaletsos, Forough Arabshahi, Adly Templeton, Ousmane Dia, Karen Chen, and Sahar Karimi for helpful discussions. We thank Jessica Ai, Audrey Flower, Beliz Gokkaya, Neamah Hussein, and Ehsan Emamjomeh-Zadeh for developing the uncertainty estimation techniques used in this study.

References

[1] Görkem Algan and Ilkay Ulusoy. Label noise types and their effects on deep learning. *arXiv preprint arXiv:2003.10471*, 2020. 2

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019. 3

[3] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 3

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1613–1622. JMLR.org, 2015. 2, 8

[5] Li Chen, Purvi Goel, and Ilknur Kabul. Uncertainty estimation methods in the presence of noisy labels. *Advances in Neural Information Processing Systems, Women in Machine Learning Workshop*, 2021. 1, 2, 4, 8

[6] Li Chen, David Yang, Purvi Goel, and Ilknur Kabul. Robust deep learning with active noise cancellation for spatial computing. *arXiv preprint arXiv:2011.08341*, 2020. 3

[7] Yifan Ding, Liqiang Wang, Deliang Fan, and Boqing Gong. A semi-supervised two-stage approach to learning from noisy labels. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1215–1224. IEEE, 2018. 3

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016. 2

[9] T. Gale, E. Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *ArXiv*, abs/1902.09574, 2019. 4

[10] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. 2

[11] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *arXiv preprint arXiv:1804.06872*, 2018. 3

[12] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. *arXiv preprint arXiv:1802.05300*, 2018. 3

[13] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, 01 2021. 4

[14] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018. 3

[15] Jan M Köhler, Maximilian Autenrieth, and William H Beluch. Uncertainty based detection and relabeling of noisy image labels. In *CVPR Workshops*, pages 33–37, 2019. 8

[16] Felix Laumann and Kumar Shridhar. Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference, 06 2018. 2, 8

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[18] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020. 3

[19] Yueming Lyu and Ivor W Tsang. Curriculum loss: Robust learning and generalization against label corruption. *arXiv preprint arXiv:1905.10045*, 2019. 2

[20] R. Ma and L. Niu. A survey of sparse-learning methods for deep neural networks. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 647–650, 2018. 4

[21] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". *arXiv preprint arXiv:1706.02613*, 2017. 3

[22] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. Self: Learning to filter noisy labels with self-ensembling. *arXiv preprint arXiv:1910.01842*, 2019. 3

[23] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017. 3

[24] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. SELFIE: Refurbishing unclean samples for robust deep learning. In *ICML*, 2019. 4

[25] S. Srinivas, A. Subramanya, and R. V. Babu. Training sparse neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 455–462, 2017. 4

[26] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 322–330, 2019. 2

[27] Xiaobo Xia, Tongliang Liu, Bo Han, Nannan Wang, Mingming Gong, Haifeng Liu, Gang Niu, Dacheng Tao, and Masashi Sugiyama. Part-dependent label noise: Towards instance-dependent label noise. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[28] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173. PMLR, 2019. 3

[29] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018. 2