Beyond Point Estimate: Inferring Ensemble Prediction Variation from Neuron Activation Strength in Recommender Systems

Zhe Chen, Yuyan Wang, Dong Lin, Derek Zhiyuan Cheng, Lichan Hong, Ed H. Chi, Claire Cui
Google, Inc.
{chenzhe,yuyanw,dongl,zcheng,lichan,edchi,claire}@google.com

ABSTRACT

Despite deep neural network (DNN)'s impressive prediction performance in various domains, it is well known now that a set of DNN models trained with the same model specification and the exact same training data could produce very different prediction results. People have relied on the state-of-the-art ensemble method to estimate prediction uncertainty. However, ensembles are expensive to train and serve for web-scale traffic systems.

In this paper, we seek to advance the understanding of prediction variation estimated by the ensemble method. Through empirical experiments on two widely used benchmark datasets Movielens and Criteo in recommender systems, we observe that prediction variations come from various randomness sources, including training data shuffling, and random initialization. When we add more randomness sources to ensemble members, we see higher prediction variations among these ensemble members, and more accurate mean prediction. Moreover, we propose to infer prediction variation from neuron activation strength and demonstrate its strong prediction power. Our approach provides a simple way for prediction variation estimation, and opens up new opportunities for future work in many interesting areas (e.g., model-based reinforcement learning) without relying on serving expensive ensemble models.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Ensemble methods**; **Uncertainty quantification**.

KEYWORDS

Ensemble, Neural Networks, Neuron Activation, Prediction Uncertainty, Recommender Systems

ACM Reference Format:
Zhe Chen, Yuyan Wang, Dong Lin, Derek Zhiyuan Cheng, Lichan Hong, Ed H. Chi, Claire Cui. 2021. Beyond Point Estimate: Inferring Ensemble Prediction Variation from Neuron Activation Strength in Recommender Systems. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441770

WSDM '21, March 8–12, 2021, Virtual Event, Israel.
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8297-7/21/03.
https://doi.org/10.1145/3437963.3441770

1 INTRODUCTION

Deep neural networks (DNNs) have gained widespread adoption in recent years across many domains. Despite their impressive performance in various applications, most DNNs today only generate point predictions. And it is well known that a set of DNN models trained with the same model specification and the same data can produce very different predictions [26, 40, 49]. Researchers realize that point predictions do not tell the whole story and raise questions about whether DNNs predictions can be trusted [22, 44].

Thus, a growing number of researches are looking into measuring prediction uncertainty for DNNs. Ensemble method is a state-of-the-art benchmark for prediction uncertainty estimation to consolidate agreements among the ensemble members and produce better point predictions [6, 11, 26, 40]. Many researches focus on whether these point predictions are well-calibrated on either in-distribution or out-of-distribution (OOD) data [16, 27, 30, 40]. However, there exist prediction disagreements in the ensemble, which we call ***prediction variation***. For example, different models in the ensemble often yield different prediction results even on the same input example.

Ensembles provide us with a good approximation of prediction variation, but they are computationally expensive as they require training multiple copies of the same model. At inference time, they require computing predictions on every example for every ensemble member, which can be infeasible for real-time large-scale machine learning systems. Researchers have proposed various techniques to improve the efficiency of ensembles, such as SnapshotEnsembling [21] and BatchEnsemble [49]. However, as far as we know, none of these works studies whether we can infer prediction variation from neuron activation strength collected from the DNN directly, without running predictions on the same data multiple times. Here, we use ***neuron activation strength*** to indicate DNN's neuron output strength, e.g., the neuron output after activation.

We hypothesize that neuron activation strength could be directly used to infer prediction variation. Our intuition is based on neuroscience's Long-Term Potentiation (LTP) process [37] which states that connections between neurons become stronger with more frequent activation. LTP is considered as one of the underlying mechanisms for learning and memorization. If we imagine deep networks learn like the brain, some groups of neurons will be more frequently and/or strongly activated, i.e. strengthened neurons. During learning, these strengthened neurons represent where the network has learned or memorized better.

Our Goal — In this paper, we aim to advance the understanding of prediction variations estimated by ensemble models, and look into the predictive power of neuron activation strength on prediction variations. To the best of our knowledge, we are the first to conduct comprehensive studies on prediction variations from different ensemble models, and we are also the first to demonstrate that we are able to infer prediction variation from neuron activation strength.

Challenges — We face the following challenges:

Variation Quantification — There are a variety of prediction problems. For example, predicting the target rating for a user on a given movie could be a regression task or a classification task by dividing the movie ratings into multiple buckets. There is no standard way to quantify prediction variation for such a variety of tasks.

Variation Sources — Training a set of models with the same model specification and the same data could produce very different results. The prediction disagreements are inherently caused by the nonconvex nature of DNN models in which multiple local minima exist. In addition, different randomness sources, including random initialization of DNN parameters, random shuffling of training data, sub-sampling of training data, and even the hardware itself, could lead to disagreements. It is often hard to identify the contribution of each randomness source to prediction variation.

Neuron Activation Strength — We hypothesize that neuron activation strength has prediction power to infer a deep network's prediction variations. However, it is not straightforward to demonstrate this prediction power for different prediction problems and randomness sources.

Our Approach — In this paper, we investigate sources for prediction variation, and by controlling the randomness sources explicitly, we demonstrate that neuron activation strength has strong prediction power to infer ensemble prediction variations in different randomness-controlled settings. We demonstrate our findings on two popular benchmark datasets, MovieLens and Criteo.

First, we quantify prediction variation across different target tasks, including regression, binary and multi-class classification tasks. We use standard deviation of the ensemble predictions to quantify prediction variation for regression and binary classification tasks, and KL divergence based score to quantify prediction distribution disagreement for multi-class classification tasks.

Second, we identify and examine three variation sources of randomness (data shuffling, weight initialization, and data re-sampling). By explicitly controlling randomness sources, we study their contributions to the performance of mean prediction and prediction variations. Our results show that every variation source exhibits a non-negligible and different contribution towards the total prediction variation. Ensemble mean predictions are often better calibrated and have lower variances [13, 26, 46], but in this paper, we focus on studying prediction variations among the individual ensemble members. We observe that when we include more variation sources to the ensemble models, the ensemble members are more diverse and different from each other. Moreover, the ensemble's prediction mean tends to be more accurate and the prediction variations among the ensemble members are higher.

Finally, we demonstrate that neuron activation strength has strong prediction power in estimating prediction variation of ensemble models. With the activation strength information obtained from a single DNN, we can add an auxiliary task to estimate prediction variation directly. Our experiment results show that our activation strength based method estimates prediction variation fairly well as a regression task. The average R^2 on MovieLens is 0.43 and 0.51 with different task definitions, and on Criteo is 0.78. Our method is especially good at detecting the lowest and highest variation bucket examples, on average with 0.92 AUC score for the lowest bucket and 0.89 AUC score for the highest bucket on both datasets. Our approach is complementary and orthogonal to many other resource-saving or single model prediction variation estimation techniques, as it doesn't alter the target task's optimization objectives and process.

Applications — Prediction variation quantification is a fundamental problem and our activation strength based approach opens up new opportunities for a lot of interesting applications. For example, in model-based reinforcement learning, prediction variation has to be quantified for exploration [7, 51]. In curriculum learning [1], prediction variation can be used as a way for estimating example difficulty. In medical domain, prediction variation can be used to capture significant variability in patient-specific predictions [12]. Our proposed activation strength based method provides a simple and principled way to serve prediction variation estimate by deploying an auxiliary task, instead of using an expensive ensemble model, during inference time.

Contributions — Our contributions are four fold:

- Framework for prediction variation estimation using activation strength in Section 3.
- Formal quantification on prediction variation for various target tasks in Section 4.
- Prediction variation understanding by explicitly controlling various randomness sources in Section 5.
- Empirical experiments to demonstrate strong predictive power from neuron activation strength to estimate ensemble prediction variation in Section 6.

We cover the related work in Section 2, and conclude with a discussion of future work in Section 7.

2 RELATED WORK

In machine learning literature, researchers mostly focus on two types of uncertainty: aleatoric uncertainty and epistemic uncertainty [10]. Aleatoric uncertainty is due to the stochastic variability inherent in the data generating process [29]. Aleatoric uncertainty corresponds to *data uncertainty*, which describes uncertainty for a given outcome due to incomplete information [24]. Epistemic uncertainty is due to our lack of knowledge about the data generating mechanism [29], and corresponds to *model uncertainty*, which can be viewed as uncertainty regarding the true function underlying the observed process [4]. In this paper, we focus on studying model uncertainty, especially prediction variations or disagreements.

There has been extensive research on methodologies for estimating model uncertainty and discussions on their comparisons [40]. Principled approaches include Bayesian approaches [20, 31, 33, 36] and ensemble-based approaches [26]. Bayesian methods provide a mathematically grounded framework to model uncertainty, through learning the deep neural network as Gaussian processes [36], or learning approximate posterior distributions for all or some weights of the network [5, 25]. Ensemble methods [26], on the other hand, is a conceptually simpler way to estimate model uncertainty. There are multiple ways to create ensembles of neural networks: bagging [6], Jackknife [34], random initialization, or random shuffling of training examples. The resulting ensemble of neural networks contains some diversity, and the variation of their predictions can be used as an estimate of model uncertainty. A lot of research work results are based on the ensemble method [3, 7, 9, 28, 40]. For example, [9] demonstrates promising results uses deep ensembles for diagnosis and referral in retinal disease. [7] proposes a new algorithm for model-based reinforcement learning by incorporating uncertainty via ensemble. In this paper, we use ensemble as the ground truth to produce prediction variations in different scenarios (i.e., different randomness settings).

Estimating model uncertainty through Bayesian modeling or ensemble usually incurs significant computation cost. For example, Bayesian neural networks that perform variational learning on the full network [5] significantly increase the training and serving cost. The cost for ensemble methods scales by the number of models in the ensemble, which can be prohibitive for practical use. To this end, researchers have proposed various techniques to reduce the cost for Bayesian modelling and ensemble methods. For example, single-model approaches are proposed to quantify model uncertainty by modifying the output layer [30, 48], deriving tractable posteriors from last layer output only [42, 45], or constructing pseudo-ensembles that can be solved and estimated analytically [32]. Our proposed method in this paper can also be viewed as a single-model approach for model uncertainty estimation. However, we do not impose any Bayesian assumptions on the network or any distributional assumptions on the ensembles. Instead, we build an empirical model to learn the association between activation strength and model uncertainty, and use it to estimate model uncertainty for new examples, which offers a relatively simple, robust and computationally efficient way to estimate prediction variation from a single model.

3 VARIATION ESTIMATION FRAMEWORK

Similar to the work on model uncertainty estimation [38, 39, 47], we build two components for the prediction variation estimation framework: target task, and variation estimation task, as shown in Figure 1. Before discussing the two tasks in detail, we first introduce the two experiment datasets that we use throughout this paper.

3.1 Datasets

Our studies are based on two datasets: MovieLens and Criteo.

MovieLens — The MovieLens [1] contains 1M movie ratings from 6000 users on 4000 movies. This data also contains user related features and movie related features.

Criteo — The Criteo Display Advertising challenge [2] is a binary classification task to predict click-through rate (clicked event's label is 1, otherwise 0). The Criteo data consists of around 40M examples with 13 numerical and 26 categorical features.

3.2 Target Task

The target task is defined by the original prediction problem, such as the rating prediction task on MovieLens, and the click-through

[1]http://files.grouplens.org/datasets/movielens/ml-1m-README.txt

[2]https://www.kaggle.com/c/criteo-display-ad-challenge

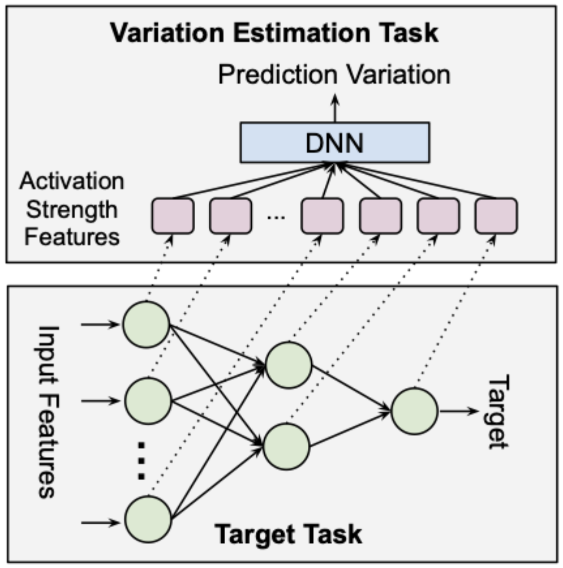

Figure 1: Our framework for prediction variation estimation using activation strength.

prediction task on Criteo. In this paper, we focus on the multi-layer perceptron architecture (MLP), with ReLU as the activation function. Furthermore, we define three target tasks as follows.

MovieLens Regression (MovieLens-R) — The target task takes in user-related features (i.e., id, gender, age, and occupation) and movie-related features (i.e., id, title and genres), and predicts movie rating as a regression task. The movie ratings are integers from 1 to 5. We use mean squared error (MSE) as the loss function. MSE is a standard metric for evaluating the performance of rating prediction in recommenders [2, 19, 43]. For example, [2] used 100 million anonymous movie ratings and reported their Root Mean Squared Error (RMSE) performance on a test dataset as 0.95.

Each model trains for 20 epochs with early-stopping. Similar to the neural collaborative filtering framework [17], we use fully connected ReLU based MLP model for the rating prediction task. We set the fully connected neuron layer sizes to be [50, 20, 10] with batch size of 1024, using the Adam optimizer [23]. We set the user id and item id embedding size to 8 [17]. All input features are categorical in MovieLens[3]. The user age feature was divided into 7 categories, and thus we set the user age embedding size to 3. We also set the user occupation embedding size to 5.

MovieLens Classification (MovieLens-C) — Similar to the MovieLens regression task, we predict movie ratings as 5 integer values from 1 to 5 and model this problem as a multi-class classification task with softmax cross entropy as the loss function.

We experiment with temperature scaling values T =[0.1, 0.2, 0.5, 1, 2, 5, 10] with batch size of 1024 to make sure predictions are well calibrated [16]. We pick $T = 0.2$ which gives the best Brier score[4] while achieving similar accuracy compared to other settings.

Criteo — This target task uses a set of numerical and categorical features to predict the click-through rate. The label for the task

[3]http://files.grouplens.org/datasets/movielens/ml-1m-README.txt

[4]https://en.wikipedia.org/wiki/Brier_score

is either 0 or 1 representing whether an ad is clicked or not. We model this problem as a binary classification task with sigmoid cross entropy loss function. The trained model outputs a float between 0 and 1 representing the predicted click-through probability.

We use the same model setting as described in [40], except for ReLU layer sizes. In the beginning, we set ReLU layer sizes to be [2572, 1454, 1596] as in [40], but found that only around 80 neurons are activated at least once on a 10k sample data. As a result, in this paper, we use ReLU layer sizes of [50, 20, 10] with a batch size of 1024 and set the $beta\_1$ for the Adam optimizer [23] to be 0.97, and we find the prediction performance is similar to the model with much larger ReLU layer sizes. Each model is trained for 1 epoch.

3.3 Variation Estimation Task

The variation estimation task uses neuron activation strength to estimate prediction variation for each input example. Neurons at different layers could yield values of different scales. Thus, we define **neruon activation strength** as the normalized neuron output to represent how strong a neuron output is. To be more specific, given a target task model and for each neuron, we collect the neuron raw output values for all the training examples to obtain the distribution mean and standard deviation. Then we are able to conduct Z-score normalization on the neuron raw outputs to serve as neuron activation strength.

As shown in Figure 1, we build a neural network model taking the neuron activation strength features to estimate prediction variation. In our current setup, we collect activation strength features from all the neurons in the target task. We concatenate all the activation features and feed to a DNN for prediction variation estimation. During training time, we use ensemble to estimate prediction variation as the ground-truth label, which are used to train the variation estimation task model. We formally define prediction variation in Section 4. During the inference time, we directly output the estimated prediction variation using activation strength as an auxiliary task. We find that it is possible to identify important neurons and reduce the number of features. Due to the space limit, we will not discuss feature reduction in this paper. The detailed setup of the variation estimation task is discussed in Section 6.

4 PREDICTION VARIATION QUANTIFICATION

In this section, we formally quantify prediction variation in different problem settings. Ensemble is one state-of-the-art benchmark for prediction uncertainty estimation [6, 11, 26, 40]. We use ensemble to estimate model prediction variation, that is how much disagreement there is among ensemble model predictions.

Given the same training data and model configuration, we train an ensemble of N models $M = \{m\}$, where M is the set of models, and N is the ensemble size. Let $\{x\}$ be the testing data, and x represents the feature vector. Each of the trained model $m \in M$ makes a prediction on an example $x \in \{x\}$ as $y'_m(x)$.

For *regression and binary classification* tasks, the model output is a float value, thus we define prediction variation as *value prediction variation* based on the standard deviation of the predicted float values across different model members in the ensemble. For *multi-class classification* tasks, the model output is a probability distribution over different categories. We define prediction variation as *distribution prediction variation* based on the KL-disagreement or generalized Jensen-Shannon divergence [26] on the predicted probability distributions. Now we define prediction variation formally.

Definition 4.1. (Value Prediction Variation) Given an example x that represents the feature vector, we define its prediction variation $PV(x)$ to be the standard deviation of predictions from the set of models M as $PV(x) = \sqrt{\frac{\sum_{m \in M}(y'_m(x) - \bar{y}(x))^2}{|M|-1}}$, where $\bar{y}(x) = \frac{1}{|M|}\sum_{m \in M} y'_m(x)$

Definition 4.2. (Distribution Prediction Variation [26]) Given the example x that represents the feature vector, let the prediction distribution for the example x be $p(y|x)$. We define prediction variation $PV(x)$ to be the sum of the Kullback-Leibler (KL) divergence from the prediction distribution of each model $m \in M$ to the mean prediction distribution of the ensemble. $PV(x) = \sum_{m=1}^{M} KL(p_m(y|x)||p_E(y|x))$ where $p_E(y|x) = M^{-1}\sum_m p(y|x)$ is the mean prediction of the ensemble.

5 PREDICTION VARIATION SOURCES

In this section, we diagnose the prediction variation sources, and we are interested in seeing their effects on the total prediction variation by controlling each randomness source.

There are many sources contributing to prediction variation. Random initialization of DNN parameters contributes randomness to model predictions. Randomness can also come from training data shuffling or sub-sampling. In addition, asynchronous or distributed training could lead to training order randomness. More surprisingly, we observe the hardware itself contributes to the model prediction variation: we find that by fixing all the other settings, training a model on different CPUs might produce different models.

In this paper, we consider three types of randomness sources:

1. Shuffle (S) — Whether randomly shuffles input data, i.e., randomizes input data order.

2. RandInit (R) — Whether randomly initializes model parameters, including DNN weights and embeddings. We can fix the initialization by setting a global random seed in Tensorflow.

3. Jackknife (J) — Whether randomly sampling input data by applying Jackknife sub-sampling. Similar to delete-1 Jackknife [34], we split data into N Jackknife sub-samples and each ensemble member randomly leaves one Jackknife sample out. We use 100 models for each ensemble as discussed in Appendix A.1, and we split the data into 100 unique Jackknife sub-samples. Another popular data sampling method is bootstrap [50]. For this work, we pick Jackknife due to its simplicity to implement.

We set up the prediction variation randomness control experiments as follows: First, if incorporating Jackknife randomness, we obtain the Jackknife sub-samples; otherwise, we use all the training data. Second, if incorporating Shuffle randomness, we shuffle the training data; otherwise we do not. Finally, if incorporating RandInit randomness, we randomly initialize all the parameters without a fixed global seed for all the ensemble members; otherwise

Randomness Settings	MovieLens-R MSE	ACC	PV Mean	PV Std	PV Coeff	MovieLens-C ACC	PV Mean	PV Std	Criteo AUC	PV Mean	PV Std	PV Coeff
(R0) None	0.7980	0.4483	0.0000	0.0000	0.00%	0.4635	0.0000	0.0000	0.7956	0.0000	0.0000	0.00%
(R1) R	0.7569	0.4570	0.1948	0.0692	5.83%	0.4818	4.4486	2.5744	0.7991	0.0300	0.0162	16.3%
(R2) S	0.7671	0.4473	0.1433	0.0509	4.36%	0.4746	2.7379	1.6717	0.7999	0.0359	0.0179	20.7%
(R3) R+S	0.7479	0.4521	0.1936	0.0649	5.87%	0.4829	4.4637	2.5053	0.7999	0.0358	0.0181	20.4%
(R4) J	0.7718	0.4464	0.1494	0.0638	4.54%	0.4745	2.7522	1.9771	**0.8003**	0.0394	0.0198	22.9%
(R5) R+J	0.7489	0.4522	**0.2035**	**0.0701**	**6.14%**	0.4829	4.7250	2.6514	0.8002	0.0396	0.0199	22.9%
(R6) S+J	0.7640	0.4486	0.1560	0.0571	4.74%	0.4766	3.1739	1.9803	0.8002	**0.0409**	0.0200	23.5%
(R7) R+S+J	**0.7457**	**0.4528**	0.2013	0.0667	6.08%	**0.4838**	**4.7332**	**2.6723**	0.8000	0.0407	**0.0202**	**23.6%**

Table 1: Ensemble's prediction accuracy and prediction variation (PV) on 8 randomness settings. PV coefficient (coeff) shows the average ratio of PV to the ensemble mean prediction over all the testing examples.

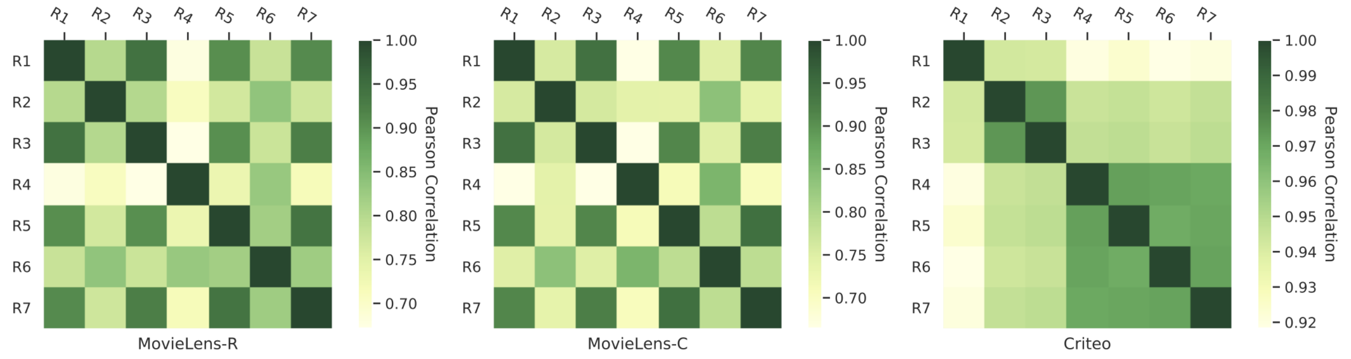

Figure 2: Pearson correlation of prediction variations for ensembles with different randomness settings.

we use a fixed global seed.[5] We use an ensemble of 100 models for each of the randomness settings: as discussed in Appendix A.1, 93% of prediction variation in the ensemble of 1000 models can be captured with size 100 ensemble.

On each dataset of MovieLens and Criteo, we randomly split the data into training and testing. On MovieLens we split the 1M data into 60% for training, and 40% for testing. On Criteo, same to [40], we use 37M data for training, 4.4M for validation, and 4.4M for testing. We obtain the prediction variation estimation on all the testing examples for further analysis.

Randomness Source Comparison — Table 1 shows the accuracy and prediction variation statistics for different combinations of the three randomness sources. For each type of randomness combination (e.g., (R3) R+S means using RandInit and Shuffle only), we train 100 models of the same setting, and obtain the ensemble mean prediction for accuracy evaluation and report the prediction variations. For accuracy evaluation, we report Mean Squared Error (MSE) and accuracy (ACC) for MovieLens-R, ACC for MovieLens-C, and AUC score for Criteo. We obtain ACC for MovieLens-R by rounding the ratings to the closest integers. For prediction variation metrics, we report prediction variation (PV) mean and standard deviation, and PV coefficient (coeff). We obtain the PV coefficient for each example x as PV(x) divided by the ensemble mean prediction.

From the tables, we can see that each type of the randomness sources exhibits a non-negligible and different contribution towards the total prediction variations. Ensemble models with more different randomness sources tend to produce more accurate aggregated mean prediction with higher prediction variations. On all three target tasks, the R7 setting appears to exhibit the best or close to best accuracy score, and its prediction variations are also the highest or close to the highest among all the randomness settings. It also seems that different target tasks or datasets are sensitive to different types of randomness sources. We observe that the Criteo data is more sensitive to Shuffle and Jackknife randomness while the MovieLens data is more sensitive to RandInit. We verify that by fixing all the randomness sources, we do not observe any prediction variation in the R0 settings. The PV Mean and PV std is always 0 for R0. According to PV coefficient, we notice that MovieLens shows around 5-6% of prediction sway from the mean prediction while Criteo has around 20% of prediction sway.

Randomness Source Correlations — Under each randomness setting, we are able to obtain the prediction variation for each example. Figure 2 shows the Pearson correlation of the prediction variations of all the examples between each pair of the randomness settings. The randomness setting can be found at Table 1. We do not consider R0 because it eliminates all the randomness in the model and PV is always 0.

As we can see from Figures 2, the prediction variation correlation patterns on MovieLens is quite different from Criteo. For example, while the lowest Pearson correlation score on MovieLens is around 0.7, all the randomness settings on Criteo are quite correlated with the lowest Pearson correlation score to be around 0.92.

Regression vs Classification — On MovieLens, we are able to predict ratings as regression or classification. Table 1 shows that the prediction accuracy is higher when we predict ratings as a classification task than as a regression task almost for all the randomness settings, as classification optimizes for the accuracy metric directly.

[5]When RandInit is enabled, we use a fixed set of 100 random seeds.

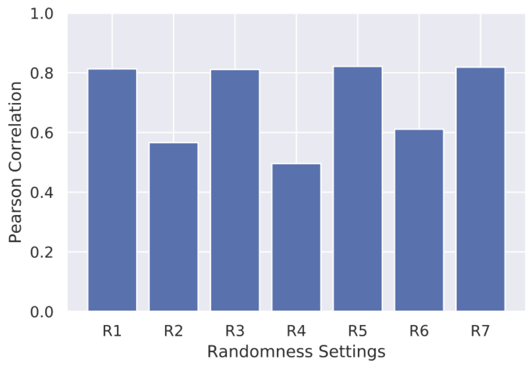

Figure 3: Pearson correlation of prediction variations between MovieLens-R and MovieLens-C.

We also compare the prediction variations obtained through regression and classification, and Figure 3 shows the Pearson correlation of prediction variations for the two tasks on various randomness settings for all the testing examples. We find that whether the variations are strongly correlated depends on the randomness settings. As shown in the figure, the prediction variations are highly correlated (with Pearson correlation more than 0.8) when we add the RandInit randomness source, otherwise the two tasks are less correlated. we think the reason is that RandInit controls model parameters and the loss function plays an important role, while underlying data properties affect Shuffle and Jackknife more.

6 PREDICTION VARIATION ESTIMATION

In this section, we study the problem of using neuron activation strength to infer prediction variation. We first discuss the prediction variation estimation task setup in Section 6.1, and then show the experiment results on using neuron activation strength to estimate prediction variation in Section 6.2.

6.1 Variation Estimation Task Setup

As shown in Figure 1, the variation estimation task collects the neuron activation information collected during the target task inference time. We use the prediction variation estimated by the ensemble model for training, and then during inference time, the variation estimation model infers the prediction variation as an auxiliary task. Now we set up the variation estimation task as follows.

Evaluation Procedure — On both MovieLens or Criteo, we first split the data into training D_t and testing D_e for the target task. On MovieLens, we split the 1M data into 60% for training and 40% for testing; on Criteo, we use the same setting as in [40], 37M data for training, 4.4M for validation and 4.4M for testing. The training data D_t is used to train the target task model m_t, and is also used for calculating the neuron output distribution mean and standard deviation for feature normalization.

Given the trained target task model m_t, we further split D_e into two sets: 50% as $D_{e1} = (x_{e1}, PV(x_{e1}))$ for training the variation estimation model; and another 50% as $D_{e2} = (x_{e2}, PV(x_{e2}))$ for testing. The ground-truth label of prediction variation for D_{e1} is collected in a separate process: We train 100 target task models independently with the same configurations to m_t using training data D_t, and then we obtain the predictions for D_{e1} from the 100 ensemble models to calculate the prediction variation for each each $x \in D_{e1}$. In this paper, we used 100 models for more accurate variation estimation. in practice, 5 to 10 model ensembles would work fine as we discussed in Section A.1.

Given a trained target task model m_t, we build a neural network model m_v to estimate prediction variations. m_v collects the activation strength information from m_t's neurons as features. m_v trains on D_{e1} and tests on D_{e2} with fully connected layers of size $[100, 50]$, batch size 256, Adam optimizer with learning rate 0.001, and 150 training epochs with early stopping. We find that m_v takes less than one epoch to converge on Criteo, but takes longer to converge on MovieLens due to its much smaller data size.

Now we explain m_v's input features and objective in detail.

Input Features — We consider two types of input features collected from neuron activation strength. We use ReLU [35] as the activation function. We believe our activation strength feature is general and can be applied to other activation functions, such as Softplus [15], ELU [8], GELUs [18], and Swish [41]. Due to space limitations, we only experiment with ReLU.

Binary — On ReLU neurons, we consider whether a neuron is activated as the input feature. This binary feature represents whether the neuron output is greater than 0.

Value — The raw value of a neuron's output directly represents the strength of an activated neuron. Therefore, we experiment with normalized activation value as the input feature. We normalize the neuron outputs according to the neuron output mean and standard deviation collected from the training data.

Objective — We estimate the prediction variation in two ways.

Regression — In this setting, the model directly estimates the prediction variation as a regression task. We use Mean Squared Error (MSE) as the loss function. However, by directly optimizing for MSE, this regression task's output range could be huge. As a result, we limit the minimum output of the model to be 0 as prediction variation is always positive, and limit the maximum output to be $mean + 3 * std$ where the mean and std are estimated on the training data's prediction variations. $mean + 3 * std$ should be able to cover 99.7% of the data in a Guassian distribution.[6]

Classification — In this setting, we divide prediction variation into multiple buckets according to the percentile, and then predict which variation bucket it belongs to. We set the bucket number to be 5, and use cross entropy as the loss function for the prediction variation classification model.

6.2 Variation Estimation Performance

In this section, we show the variation estimation performance using neuron activation strength on MovieLens and Criteo.

Regression Performance — When we run the prediction variation estimation as a regression task, we directly output a score as the estimated prediction variation.

In Table 2, we show the Mean Squared Error (MSE) and R^2 for the three target tasks on the 7 randomness control settings. From the table, on all the three target tasks and all the 7 randomness control settings, we observe strong prediction power of neuron activation strength for ensemble prediction variations. The average

[6]https://en.wikipedia.org/wiki/68-95-99.7_rule

	MovieLens-R		MovieLens-C		Criteo	
	MSE	R^2	MSE	R^2	MSE	R^2
(R1) R	0.0022	0.5416	3.6159	0.4586	0.0063	0.7617
(R2) S	0.0011	0.5636	1.5015	0.4683	0.0068	0.7863
(R3) R+S	0.0019	0.5514	3.4288	0.4569	0.0062	0.8100
(R4) J	0.0025	0.3885	2.5986	0.3386	0.0086	0.7817
(R5) R+J	0.0024	0.5219	3.7727	0.4646	0.0085	0.7868
(R6) S+J	0.0017	0.4938	2.3175	0.4125	0.0091	0.7739
(R7) R+S+J	0.0022	0.5123	4.0496	0.4368	0.0092	0.7761
Average	0.0020	0.5104	3.0407	0.4338	0.0078	0.7824

Table 2: Performance of variation estimation as regression on 7 randomness settings.

R^2 on MovieLens-R is 0.51, on MovieLens-C is 0.43, and on Criteo is 0.78. The variation estimation performance is the best on the Criteo data, while MovieLens-R is better than MovieLens-C. The reasons could be: First, Criteo has more training data than MovieLens. To train the variation estimation model, we have 2.2M (50% of 4.4M) training data on Criteo, while only 0.2M (50% of 0.4M) training data on MovieLens; Second, Criteo has a larger relative range of prediction variations, compared to MovieLens. As shown in Table 1, Criteo shows around 20% of prediction variation sway from the mean prediction, which is much higher than 4-6% on MovieLens; Finally, the Criteo task is probably the easiest task among the three: it is a binary classification task, while MovieLens-R is a regression task and MovieLens-C is a multi-class classification task.

Classification Performance — We also run the prediction variation estimation as a classification task, by predicting which variation bucket it should be in. We use both binary and value input features. Due to the space limit, we only show the results on the R3 randomness setting, as R3 uses training data shuffling and random initialization which is the most common setting in practice.

Figure 4 shows the AUC scores and Figure 5 shows the confusion matrix for the 5-bucket prediction variation classification on both MovieLens and Criteo. The numbers in Figure 5 are normalized by the actual example number in each bucket. Bucket 1 represents the lowest variation slice, while bucket 5 represents the highest.

As we can see from Figure 4, our variation estimation model is fairly good at distinguishing examples at different variation buckets, especially for the lowest and highest buckets. The average AUC score for the three tasks on bucket 1 is about 0.92 and on bucket 5 is about 0.89. Figure 5 shows that the classification errors mostly happen on adjacent buckets. For example, on Criteo, most of the mis-classifications assign bucket 1 examples to bucket 2. When we divide the prediction variation buckets on training data, we notice that the bucket thresholds are close. For example, under the randomness control setting R3, the thresholds of the 5 buckets for MovieLens-R are [0.1420, 0.1672, 0.1950, 0.2366], and the thresholds for Criteo is [0.0194, 0.0287, 0.0398, 0.0515]. As show in the figures, Criteo seems to have the best performance among the three tasks. Again the reasons could be that the Criteo task has more training data, is probably the easiest among the three tasks, and it has much larger relative prediction variation range.

Activation Feature Study — In Table 3, we show the contribution of the two input activation strength features. We try two feature

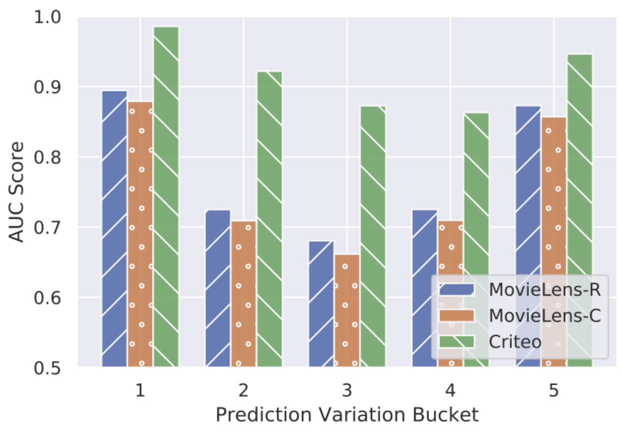

Figure 4: AUC for variation bucket prediction with the randomness setting R3.

	MovieLens-R		MovieLens-C		Criteo	
	MSE	R^2	MSE	R^2	MSE	R^2
B	0.0025	0.4196	4.1616	0.3408	0.0124	0.6210
BV	0.0019	0.5514	3.4288	0.4569	0.0062	0.8100

Table 3: Activation feature study for variation estimation as regression with the randomness setting R3.

settings: *B* refers to using the binary feature only; and *BV* refers to using both the binary and value features. As shown in the table, each type of features makes a non-negligible contribution towards prediction variation estimation. Thus, it is beneficial to have both the binary and value features for prediction variation estimation.

Reproducibility — When we evaluate the variation estimation model, the performance is calculated based on one target task model. To check whether the performance is reproducible, we train 5 new target task models. To simplify the problem, we also use the randomness setting R3, which is the most commonly used setting in practice. Table 4 shows the MSE and R^2 for each run on both MovieLens and Criteo. As shown in the table, the standard deviation of MSE and R^2 for the 5 runs is small for each of the three tasks. Thus, we conclude that activation strength is useful for estimating prediction variation and it is reproducible.

Comparison with Dropout [14] — Dropout is also a standard way to estimate model uncertainty. We estimate prediction variation using dropout as follows: We train one model by randomly dropping 20% of the neurons on the ReLU layers using the randomness settings R3. During inference time, we keep the dropout turned on to obtain the predicted results for all the testing data. We run the inference 100 times, and obtained the prediction variation for each testing example. We find that the prediction variation estimated by dropout is not very correlated with the variation estimated by the ensemble method. On MovieLens-R, Pearson correlation of prediction variations for dropout and ensemble is 0.25, RMSE is 0.0798, and R^2 is -0.5010. On Criteo, Pearson correlation of prediction variations for dropout and ensemble is 0.37, RMSE is 0.0279, and R^2 is -1.3709. As a result, we did not conduct further comparison with our activation strength based method.

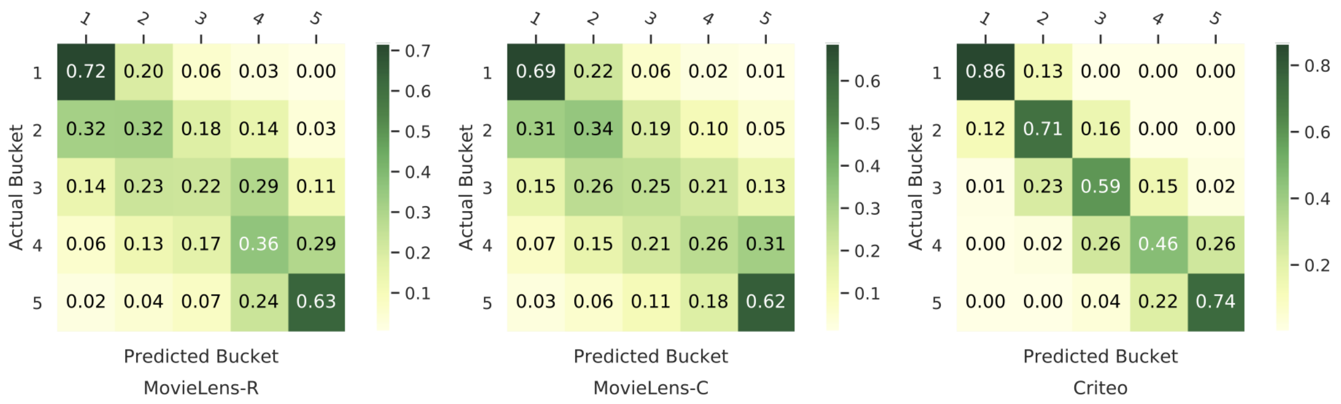

Figure 5: Confusion matrix for variation estimation as classification for the randomness setting R3.

	MovieLens-R		MovieLens-C		Criteo	
Run	MSE	R^2	MSE	R^2	MSE	R^2
1	0.0019	0.5514	3.4288	0.4569	0.0062	0.8100
2	0.0020	0.5160	3.3243	0.4734	0.0061	0.8140
3	0.0019	0.5418	3.1825	0.4959	0.0064	0.8036
4	0.0019	0.5384	3.2375	0.4872	0.0070	0.7863
5	0.0021	0.4994	3.2631	0.4831	0.0066	0.7969
Std	0.00008	0.0190	0.0842	0.0133	0.0003	0.0098

Table 4: Reproducibility test for variation estimation as regression with the randomness setting R3.

7 CONCLUSION AND FUTURE WORK

In this paper, we conduct empirical studies to understand the prediction variation estimated by ensembles under various randomness control settings. Our experiments on two public datasets (MovieLens and Criteo) demonstrate that with more variation sources, ensemble tends to produce more accurate point estimates with higher prediction variations. More importantly, we demonstrate strong predictive power of neuron activation strength to infer ensemble prediction variations, which provides an efficient way to estimate prediction variation without the need to run inference multiple times as in ensemble methods.

In the future, we are interested in exploring the proposed activation strength based methods in various applications, such as model-based reinforcement learning, and curriculum learning. In addition, we plan to improve our activation strength based approach as follows. First, we aim to simplify the training procedure of prediction variation estimation. Our current activation strength based approach requires a tedious process to collect prediction variations from ensembles. We hope to make this step easier, or remove the dependency of relying on the ensemble models. Second, we aim to improve the variation estimation accuracy by considering more neuron activation features such as neuron patterns, and to improve the prediction variation generalizability by understanding the contributions of neurons activation at different positions and layers. Finally, we hope to demonstrate the prediction power of our activation strength based approach on other neural network architectures aside from the simple MLP framework.

A APPENDIX

A.1 Model Ensemble Sizes

In this section, we are interested in finding out how many models in ensemble are need to estimate prediction variation accurately. In this section, we conduct empirical experiments on MovieLens-R, and Criteo to answer this question.

For each target task, using the same training data and model configuration, we first train 1000 models as the ensemble universe M_{gt} to obtain ground-truth prediction variations, using the R3 randomness setting. Given an example x, we obtain its prediction variation from the 1000 ensemble models as $PV_{gt}(x)$. We calculate the mean prediction variation for all the examples as $\bar{PV}_{gt}$.

Then we evaluate the prediction variation difference between an ensemble M_N of a smaller size N and M_{gt}. We use *delta ratio* to quantify the difference between the prediction variation estimated from the two ensembles M_N and M_{gt} as follows.

Definition A.1. (Delta Ratio) Let prediction variation delta $\delta_{M_N}(x)$ be the absolute difference of the estimated prediction variation between a model ensemble M_N of size N and the ground-truth model ensemble M_{gt}, as $\delta_{M_N}(x) = |PV_{M_N}(x) - PV_{gt}(x)|$. We obtain the average prediction variation delta for all the examples in a dataset D as $\delta_{M_N} = \frac{1}{|D|}\sum_{x \in D} \delta_{M_N}(x)$. We define delta ratio dr_{M_N} to be the ratio of prediction variation delta δ_{M_N} to the average prediction variation in the ground-truth ensemble models $\bar{PV}_{gt}$, as $dr_{M_N} = \delta_{M_N} / \bar{PV}_{gt}$

In Figure 6, we show the delta ratio of different ensemble sizes for the MovieLens regression task and the Criteo task. For each ensemble size N, we sample N models without replacement from the 1000 ground-truth model universe and obtain its delta ratio. We repeat this sampling process for 20 times, and obtain the mean and standard deviation of the delta ratio for the given N. In the figure, we can see the delta ratio decreases when the ensemble size increases. The delta ratio statistics is similar on both datasets, MovieLens and Criteo. When the ensemble size is 100, the delta ratio is about 7% which indicates 93% of prediction variation from the ground-truth ensemble of 1000 models is captured. As a result, in this paper, we use 100 as the default ensemble size for all experiments.

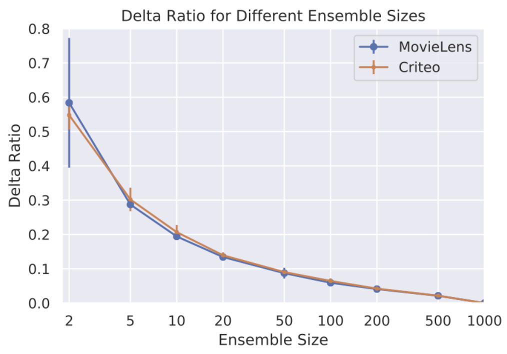

Figure 6: Delta ratio for different ensemble sizes on the MovieLens regression task and the Criteo task.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.
[2] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. Citeseer, 35.
[3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*. 5049–5059.
[4] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.
[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424* (2015).
[6] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.
[7] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*. 4754–4765.
[8] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).
[9] Jeffrey De Fauw, Joseph R Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O'Donoghue, Daniel Visentin, et al. 2018. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine* 24, 9 (2018), 1342–1350.
[10] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural safety* 31, 2 (2009), 105–112.
[11] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.
[12] Michael W Dusenberry, Dustin Tran, Edward Choi, Jonas Kemp, Jeremy Nixon, Ghassen Jerfel, Katherine Heller, and Andrew M Dai. 2020. Analyzing the role of model uncertainty for electronic health records. In *Proceedings of the ACM Conference on Health, Inference, and Learning*. 204–213.
[13] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757* (2019).
[14] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. 1050–1059.
[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.
[16] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599* (2017).
[17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 173–182.
[18] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).
[19] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.
[20] Geoffrey E Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*. 5–13.
[21] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109* (2017).
[22] Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. 2018. To trust or not to trust a classifier. In *Advances in neural information processing systems*. 5541–5552.
[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[24] Frank Hyneman Knight. 1921. *Risk, uncertainty and profit*. Vol. 31. Houghton Mifflin.
[25] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. 2018. Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation. (2018).
[26] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*. 6402–6413.
[27] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325* (2017).
[28] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. 2017. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports* 7, 1 (2017), 1–14.
[29] Jeremiah Liu, John Paisley, Marianthi-Anna Kioumourtzoglou, and Brent Coull. 2019. Accurate uncertainty estimation and decomposition in ensemble learning. In *Advances in Neural Information Processing Systems*. 8952–8963.
[30] Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. 2020. Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness. *arXiv preprint arXiv:2006.10108* (2020).
[31] Christos Louizos and Max Welling. 2017. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961* (2017).
[32] Zhiyun Lu, Eugene Ie, and Fei Sha. 2020. Uncertainty Estimation with Infinitesimal Jackknife, Its Distribution and Mean-Field Approximation. *arXiv preprint arXiv:2006.07584* (2020).
[33] David JC MacKay. 1992. A practical Bayesian framework for backpropagation networks. *Neural computation* 4, 3 (1992), 448–472.
[34] Avery McIntosh. 2016. The Jackknife estimation method. *arXiv preprint arXiv:1606.00497* (2016).
[35] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.
[36] Radford M Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.
[37] Roger A Nicoll. 2017. A brief history of long-term potentiation. *Neuron* 93, 2 (2017), 281–290.
[38] David A Nix and Andreas S Weigend. 1994. Estimating the mean and variance of the target probability distribution. *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)* 1 (1994), 55–60.
[39] David A Nix and Andreas S Weigend. 1995. Learning local error bars for nonlinear regression. *Advances in neural information processing systems* (1995), 489–496.
[40] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*. 13991–14002.
[41] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for Activation Functions. *CoRR* abs/1710.05941 (2017).
[42] Carlos Riquelme, George Tucker, and Jasper Snoek. 2018. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127* (2018).
[43] Mojdeh Saadati, Syed Shihab, and Mohammed Shaiqur Rahman. 2019. Movie Recommender Systems: Implementation and Performance Evaluation. *arXiv preprint arXiv:1909.12749* (2019).
[44] Peter Schulam and Suchi Saria. 2019. Can you trust this prediction? Auditing pointwise reliability after learning. *arXiv preprint arXiv:1901.00403* (2019).
[45] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. 2015. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*. 2171–2180.
[46] Asa Cooper Stickland and Iain Murray. 2020. Diverse ensembles improve calibration. *arXiv preprint arXiv:2007.04206* (2020).
[47] Dongqi Su, Ying Yin Ting, and Jason Ansel. 2018. Tight Prediction Intervals Using Expanded Interval Minimization. *arXiv preprint arXiv:1806.11222* (2018).
[48] Natasa Tagasovska and David Lopez-Paz. 2019. Single-model uncertainties for deep learning. In *Advances in Neural Information Processing Systems*. 6417–6428.
[49] Yeming Wen, Dustin Tran, and Jimmy Ba. 2020. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning. *Eighth International Conference on Learning Representations (ICLR 2020)* (2020).
[50] Felix A Wichmann and N Jeremy Hill. 2001. The psychometric function: II. Bootstrap-based confidence intervals and sampling. *Perception & psychophysics* 63, 8 (2001), 1314–1329.
[51] Dongruo Zhou, Lihong Li, and Quanquan Gu. 2019. Neural Contextual Bandits with UCB-based Exploration. arXiv:1911.04462 [cs.LG]