Inference Calibration of Neural Network Models

1st Xiaohu Xie
Supply Chain Optimization Techniques
Amazon
Toronto, Canada
xxie@amazon.com

2nd Reza Madankan
Supply Chain Optimization Techniques
Amazon
Seattle, USA
madankan@amazon.com

3rd Kairui Wu
Supply Chain Optimization Techniques
Amazon
Toronto, Canada
karrywu@amazon.com

***Abstract*—The development of both the theory and the practice of neural network models has been significant in the last decade. Novel solutions including Dropout and Batch Normalization (BN) have been proposed and pervasively adopted to overcome issues like over-fitting and to improve the convergence of the models. Despite of their remarkable success, in this paper we show that Dropout and BN can make the model biased and suboptimum in inference time because of the shift of their behaviour from training to inference. We propose a simple method, called Inference Calibration, to reduce this bias and improve the performance for neural network models. Our experiments show that Inference Calibration algorithm can effectively reduce prediction error and improve model's accuracy, while reducing the calibration error for both regression and classification models.**

***Index Terms*—Deep Learning, Neural Network, Dropout, Batch Normalization**

I. Introduction

Neural networks are increasingly used in various domains such as computer vision, natural language processing, etc. For deep neural networks, Dropout and Batch Normalization are widely used approaches to improve the model's robustness and generalizability by optimizing the training process. Dropout (Nitish *et al.* [1]) prevents the training of deep neural networks from co-adapt issue by only updating the weights of the hidden nodes that are randomly picked up in each batch, which reduces overfitting. Batch Normalization (Sergey *et al.* [2]) improves training stability and accelerates training process of deep neural networks significantly by reducing the internal covariate shift.

Despite the well-known benefits those approaches bring to deep neural networks, Dropout and Batch Normalization add bias to NN models due to the gap they introduce between training and inference mode. When applying Dropout, outputs of the Dropout layer are scaled in inference mode to preserve a consistent distribution with that in training mode. However, the scaling only preserve the expected value but not the variance of the distribution. The gap of variance will be translated to bias in the model's output by the non-linear activation functions of the subsequence layers. Similarly, the application of Batch Normalization introduces a gap between training and inference as well given that the scaling in training is based on statistics on each batch, but in inference mode, it is based on the statistics of the training set. The bias brought by such gaps is harmful to model accuracy and robustness. Furthermore, it has been noticed that the bias is one cause of modern neural network models suffering from poor calibration despite of their growing predictive power in terms of accuracy (Matthias *et al.* [3]).

This paper proposes an approach called Inference Calibration to reduce the bias caused by Dropout, Batch Normalization and other layers in neural networks that behavior differently during training and inference. The effect of Inference Calibration is evaluated on both regression and classification problems with various datasets. Aside from evaluating on shallow network trained from scratch, this study also assesses the effect of Inference Calibration on transfer learning (Karl *et al.* [4]), which is becoming increasingly popular in applications based on large and complex neural network given that it speeds up training process and improves the generalizability by transferring the pattern learned from one domain to another related domain without training from scratch (Lixin *et al.* [5], Brian *et al.* [6], Yin *et al.* [7], Chang *et al.* [8], Tianyi *et al.* [9], Tianyi *et al.* [10]). Our numerical simulations demonstrate that our proposed approach results in lower error and better accuracy for both regression and classification problems.

II. Related Work

The negative impact of the bias on NN model has been noticed while related researches and studies are limited. Ozgur *et al.* [11] evaluated the negative impact of Dropout on classical regression cases. Chuan *et al.* [12] analyzed the factors that impact the confidence calibration of NN classification models and found Batch Normalization caused NN models to be more miscalibrated.

There has been fewer number of works about the impact of Dropout on classical regression problems, in comparison with classification problems. In a relatively recent work, Piotrowski *et al.* [13] developed a shallow neural network for stream temperature modeling, where a combination of Dropout with early stopping was used to improve performance of the model. In another research, Ozgur *et al.* [11] has performed an in-depth study about the impact of Dropout in classical regression problems. They demonstrated the inefficiency of using Dropout method using multiple regression examples.

While many researches demonstrate the negative impact brought by Dropout and Batch Normalization, only a few provides in-depth root cause analysis and correction approaches. Many of the proposed approaches are post-processing steps focusing on improving the model's calibration. These approaches include Histogram Binning [14] and Isotonic Regression [15] proposed by Zadrozny *et al.*, and Bayesian Binning into Quantiles proposed by Naeini *et al.* [16]. The current state

of the art in calibration of neural networks is achieved by the Temperature Scaling algorithm proposed by Chuan *et al.* [12], where itself is a simple extension to Platt scaling (Platt [17]). The key idea of this technique is to introduce a single scalar parameter T into the softmax function, where its value is optimized with respect to the negative likelihood on the validation dataset.

All the above mentioned algorithms are for classification problems. For regression problems, scatterplot smoothing techniques are commonly used to de-bias a model by fitting the points of the predictions and ground truth using a straight line or a polynomial curve.

This study proposes a recalibration approach named Inference Calibration to be applied in training process to reduce the bias brought by Dropout and Batch Normalization due to train-inference gap. Unlike other methods such as Temperature Scaling method (Chuan *et al.* [12]), this approach does not rely on any post-processing on validation data, applies to neural network models for both regression and classification problems universally, and requires no post-processing steps at the inference time. In addition, it does not only improves the calibration of the final model, but also improves other error metrics such as Root Mean Square Error (RMSE) for regression, and accuracy and cross-entropy for classification.

The structure of this manuscript is as follows: In section III, we explain the underlying idea of the Inference Calibration algorithm in detail. Measures of model performance are then described in section IV. Next, we demonstrate performance of the Inference Calibration algorithm by some numerical experiments in section V. Finally, conclusion and summary are provided in section VI.

III. The Inference Calibration Algorithm

Let x_i be the i-th sample and y_i be its label. Let $\bar{y}_i = G_\theta(x_i)$ be the prediction of the neural network model G_θ with trainable parameters θ. During the training process, we aim to find θ solving the following problem:

$$\arg\min_\theta \sum_i l\left(G_\theta(x_i), y_i\right) \tag{1}$$

Gneiting *et al.* [18] show that if the loss function l represents a proper scoring rule for the problem, e.g., squared error loss for regression problems and cross-entropy for classification problems, solving (1) should theoretically result in a calibrated model G_θ.

It's common for modern NN models to contain operations, such as Dropout and BN, that demonstrate different behaviours in training and inference. To describe the shift in model's behaviour, we denote the model in the inference mode as $\hat{G}_\theta$ and the inference output as $\hat{y}_i = \hat{G}_\theta(x_i)$. Because of the behaviour shift, the model at inference time $\hat{G}_\theta$ is not guaranteed to be optimal nor calibrated even if θ solves (1).

To address this issue, we propose the Inference Calibration (IC) algorithm which get an initial solution θ by training a model to solve (1) first. Then, it runs a few additional calibration epochs to fine tune θ by solving (1) again but with G replaced by $\hat{G}$. Since the Dropout layers are disabled in $\hat{G}$, a new mechanism needs to be implemented to prevent overfitting in calibration epochs. In the IC algorithm, this is done by limiting the number of parameters being updated in the calibration epochs.

Without loss of generality, we assume

$$h_i = G^h_{\theta_h}(x_i), \tag{2}$$

$$\bar{y}_i = G^o_{\theta_o}(h_i), \tag{3}$$

where $G^h_{\theta_h}$ and $G^o_{\theta_o}$ are the hidden layers and output layer of the model G_θ respectively. We split G_θ in a way such that only $G^h_{\theta_h}$ contains Dropout or batch normalization which alter their behaviour in inference time. Due to the inference behaviour shifts, we denote the the inference time hidden layers as $\hat{h}_i = \hat{G}^h_{\theta_h}(x_i)$. Here $\hat{h}_i$ collects the biases from the inference shifting operators in the model. For typical neural network models, the output layer is a linear layer with an output activation function, i.e., $G^o_{\theta_o}(\hat{h}_i) = f(W_o\hat{h}_i + b_o)$.

We propose two strategies to avoid overfitting when fine tuning θ for $\hat{G}$. The first strategy is to freeze parameters in θ_h and update the parameters of the output layer θ_o only in the calibration epochs. This avoid overfitting as θ_o is usually a very small subset of θ. We call the variant of IC algorithm adopting this stragety the IC-O algorithm (O for Output layer).

The second strategy is to apply an affine transformation to each entity of $\hat{h}_i$ to correct the biases before it is passed to the original output layer. This gives us the following formula for inference:

$$\hat{h}_i = \hat{G}^h_{\theta_h}(x_i), \tag{4}$$

$$\hat{y}_i = G^o_{\theta_o}(D\hat{h}_i + b), \tag{5}$$

where $D = \text{diag}(d)$, with $d \in \mathbb{R}^{\dim(h)}$, is a diagonal matrix and $b \in \mathbb{R}^{\dim(h)}$ is an offset vector. In this strategy, we initialize $d = \mathbf{1}$ and $b = \mathbf{0}$ so that the affline transformation starts as an identity transformation. In the calibration epochs, we freeze both θ_h and θ_o, and learn D and b to corrects the bias in $\hat{h}$. We call the variant of the IC algorithm adopting this strategy as the IC-A algorithm (A for Affine transformation).

Algorithm 1 gives the pseudo-code of the two variants of the IC algorithm.

Consider a typical NN model output layer $G^o_{\theta_o}(\hat{h}_i) = f(W_o\hat{h}_i + b_o)$. The IC-O algorithm updates W_o and b_o in the calibration epochs. The IC-A algorithm introduces new parameters d and b and updates them only. We would like to call out that for such output layer, d and b can merge back to existing parameters W_o and b_o so that the number of parameters in the final model does not increase. To see that, let

$$W'_o = W_oD, b'_o = W_ob + b_o. \tag{6}$$

We have $G^o_\theta(D\hat{h}_i + b) = f(W'_o\hat{h}_i + b'_o)$. In this sense, IC-A also updates W_o and b_o in the calibration epochs but in a more constrained way compared to the IC-O variant. When the dimension of the output is 1, i.e., $\dim(y) = 1$, the constrains (6) become trivial and IC-A and IC-O are equivalent. In section V, we show the performance comparison of IC-A and IC-O

Algorithm 1: NN Training with Inference Calibration

Input: x_i, y_i

Output: $\hat{G}_\theta$

1 Train the neural network model with Dropout layers (and/or Batch Normalization etc.), which reduce overfitting but introduce biases to the model:

$$\theta \leftarrow \arg\min_{\theta} \sum_i l\left(G_\theta(x_i), y_i\right)$$

2 Get $\hat{G}^h_{\theta_h}(x_i)$ by setting the hidden layers $G^h_{\theta_h}(x_i)$ to inference mode.

3 [IC-O]: Run additional calibration epochs to update θ_o of the output layer $G^o_{\theta_o}$:

$$\theta_o \leftarrow \arg\min_{\theta_o} \sum_i l\left(G^o_{\theta_o}(\hat{G}^h_{\theta_h}(x_i)), y_i\right).$$

4 [IC-A]: Run additional calibration epochs to learn the linear projection required to correct the bias:

$$D, b \leftarrow \arg\min_{D,b} \sum_i l\left(G^o_{\theta_o}(D\hat{G}^h_{\theta_h}(x_i) + b), y_i\right).$$

5 Return the calibrated model [IC-O] $\hat{G}_\theta = G^o_{\theta_o}(\hat{G}^h_{\theta_h}(\cdot))$ or [IC-A] $\hat{G}_{\theta,D,b} = G^o_{\theta_o}(D\hat{G}^h_{\theta_h}(\cdot) + b)$.

for different models and provide a guideline of choosing one over the other in different scenarios.

IV. Measure of Model Performance

The IC algorithm can be used to correct the biases in NN models for both regression problems and classification problems. In this section, we describe the metrics we use to measure the impact of the IC algorithm across different types of models.

Regression problems: For regression problems, we measure the model's performance using two metrics. The first metric is RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{7}$$

where $y_i \in \mathbb{R}$ and $\hat{y}_i \in \mathbb{R}$ are the ground truth and prediction for the i^{th} sample respectively. For regression models, the smaller the RMSE, the better.

In addition to RMSE, which measures the performance of the model at the sample level, we also measure the bias of a model at the aggregate level using the average error between the predictions and ground truth:

$$e_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i). \tag{8}$$

Unlike RMSE, the calibration error e_{avg} can be negative (over-biased) or positive (under-biased). The closer is e_{avg} to 0, the more calibrated the model is.

Classification problems: For classification problems, we measure the model's performance using three metrics. The first metric is cross entropy, which is often the loss of classification models as well:

$$\text{cross-entropy} = -\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{c=1}^{M} y_{i,c}\log(\hat{y}_{i,c})\right) \tag{9}$$

where $y_{i,c} \in \{0, 1\}$ equals to 1, if class c is the correct classification of sample i, and 0 otherwise, and $\hat{y}_{i,c} \in (0, 1)$ is the corresponding predicted probability distribution for the i^{th} sample.

In addition to cross-entropy, we also consider accuracy as another measure of calibration, when it comes to classification problems:

$$\text{accuracy} = \frac{\sum_{m=1}^{M} C_{m,m}}{\sum_{i=1}^{M}\left(\sum_{j=1}^{M} C_{i,j}\right)} \tag{10}$$

where $\mathbf{C} \in \mathbb{R}^{m\times m}$ is the confusion matrix.

Following [12], [19], we also measure the top-label calibration error of the model. To calculate that, one typically divide the predictions into M bins $B_1, \ldots, B_m$ based on the top-label probability first, and then compute the Expected Calibration Error (ECE) over the bins, i.e.,

$$ECE = \sum_{m=1}^{M}\frac{|B_m|}{N}|\text{accuracy}(B_m) - \text{confidence}(B_m)| \tag{11}$$

TABLE I: Overview of Datasets for Numerical Experiments

Dataset	Number of observations	Problem Type
California housing dataset	18528	Regression
Shipment arrival prediction	13958	Regression
Poker hand dataset	850000	Classification
CIFAR-10 dataset	55000	Classification

V. Numerical Experiments

The proposed inference calibration framework is examined across multiple datasets covering both classification and regression problems. These datasets include California housing[1] and shipment arrival[2] datasets, which are used to study the impact of the proposed approach on regression problems. On the other hand, poker hand and CIFAR-10 datasets are used as a benchmark to examine performance of the proposed technique across classification problems. An overview of these datasets is shown in Table I.

A. Regression

We focus on regression problems using the following two datasets:

California housing dataset California housing dataset pertains to the houses found in a given California district and some summary stats about them based on the 1990 census data. Specifically, it includes median house value, median income, housing median age, total rooms, total bedrooms,

[1]https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

[2]https://aws.amazon.com/blogs/machine-learning/using-machine-learning-to-predict-vessel-time-of-arrival-with-amazon-sagemaker/

population, households, latitude, and longitude. The key goal in this problem is to predict the median house price, given the other features. There exists overall 18528 rows of observations, where we retained 25% of it for validation and remaining rows were used for training the network.

Shipment arrival dataset For this example, we focus on prediction of vessel time of arrival, based on the following features: origin and destination ports, distance between origin and destination, ship's draught and efficiency, and ship type. The data consists of 13958 observations, with some of the features being numerical (ship's draught and efficiency, distance between the origin and destination), and some of them being categorical (ship type, origin and destination ports). Similar to previous example, a neural network with 4 hidden layers was used to model vessel time of arrival in this problem. Architecture of the model is shown in the Table II. Similar to previous example, We have used Dropout layers across all hidden layers (except the last one) with probability of 0.5 for the Dropout. Also, 75% of the data was used for training and remaining 25% was used for validation.

The last layer of the two regression models are dense linear layers with identity activation function, i.e., $\hat{y} = W_o\hat{h}_i + b_o$. For the two regression models, IC-A and IC-O are equivalent and they both solve the following linear regression:

$$\arg\min_{W_o', b_o'} \sum_i \|W_o'\hat{h}_i + b_o' - y_i\|_2^2 \tag{12}$$

$$\text{s.t. } W_o' \in \mathbb{R}^{1\times \dim(\hat{h})}, b_o' \in \mathbb{R}. \tag{13}$$

For reference, we compare the inference calibration with the following simple linear regression:

$$\arg\min_{\alpha,\beta\in\mathbb{R}} \sum_i \|\alpha\hat{y}_i + \beta - y_i\|_2^2. \tag{14}$$

Because of the relatively small sizes of the regression datasets, we can solve (14) optimally using the least squares method over the training set.

Fig. 1 and Fig. 2 show the actual and predictions for the California housing dataset and shipment lead time dataset respectively. The left sub-figures show the case while not using inference calibration, vs. the right sub-figures where it shows the impact of using the proposed technique. From the figures, it is clear that the calibration was poor for the original NN models, and predictions made using inference calibration are more aligned with actual values, comparing to the original models.

Table III shows the RMSE and the calibration error e_{avg} for the original model, the model calibrated with Inference Calibration as well as the model calibrated with linear regression (14). As shown in the table, both inference calibration and linear regression bias correction methods significantly reduce the calibration error and RMSE. The improvements from the two methods are similar. This is expected as both of them solve linear regressions underneath with slight variance in constraints. Inference calibration tends to be marginally better in RMSE because it touches more parameters. The linear regression bias correction method is marginally better in the calibration error e_{avg} because the least squares algorithm solves for the optimal solution directly while inference calibration uses stochastic gradient descend to find a solution that converges to the optimal solution.

B. Classification

Poker Hand Dataset [20] and CIFAR-10 Dataset [21] are chosen for the experiment of classification problem. The effect of Inference Calibration is assessed across two different scenarios i) shallow network trained from scratch, and ii) deep pretrained network trained through transfer learning. For transfer learning, we experimented with both ResNet (Kaiming *et al.* [22]) and VGG (Karen *et al.* [23]) which are widely used pre-trained network structures in the domain.

In Poker Hand Dataset, 650,000 observations were used for model training and 200,000 observations were used for performance testing. A 4-layer shallow neural network (shown in Table II) with Dropout and Batch Transformation trained from scratch is adopted in the experiment.

In CIFAR-10 Dataset, 45,000 images were used for model training and 10,000 images were used for performance testing. Networks used for this experiment include a 9-layer shallow neural network trained from scratch, pretrained ResNet-18, and pretrained VGG-16 network with a linear classification header (shown in Table II).

In all experiments, Inference Calibration was implemented after normal training process in which early stopping was adopted to ensure the models were trained sufficiently and converged well before Inference Calibration. Table IV summarizes Cross Entropy Loss, Expected Calibration Error and Classification Accuracy of models on test set with no Inference Calibration, after 100 epochs of Inference Calibration, and Temperature Scaling being applied as a benchmark for comparison. Fig. 3 to Fig. 6 shows changes of performance metrics on test set over 100 epochs of Inference Calibration being implemented.

In all eight experiments being conducted, both IC-A and IC-O show good effect on reduction of Cross Entropy Loss and improvement on Classification Accuracy. For shallow neural networks trained from scratch on Poker Hand Dataset, Cross Entropy Loss reduces by 0.071 and Classification Accuracy increases by roughly 200 bps with IC-A and IC-O; For shallow neural networks trained from scratch on CIFAR-10 Dataset, Cross Entropy Loss reduces by 0.014 with IC-A and 0.018 with IC-O and Classification Accuracy increases by roughly 100 bps with both. The effect on transfer learning using ResNet-18 and VGG-16 is less significant than that on shallow neural network, in which IC-O outperforms IC-A. For ResNet-18 on CIFAR-10 Dataset, Cross Entropy Loss reduces by 0.033 and Classification Accuracy increases by roughly 100 bps with IC-O; For VGG-16 on CIFAR-10 Dataset, Cross Entropy Loss reduces by 0.018 and Classification Accuracy increases by roughly 130 bps with IC-O.

In terms of reduction on Expected Calibration Error, IC-A shows good effect on ECE reduction in all experiments. Especially for shallow neural networks trained from scratch, IC-A outperforms Temperature Scaling that is used as a

TABLE II: Structure of Models Used in Experiments

Task	Experiment	Model Type	Model Structure	# of Dropout Layer	# of BN Layer	# of Total Params	# of Updated Params in IC-A	# of Updated Params in IC-O
Regression	Housing Price	Shallow Network	4 Hidden + IC Linear	3	0	993	5	5
	Shipment	Shallow Network	4 Hidden + IC Linear	3	0	14689	5	5
Classification	Pokerhand	Shallow Network	3 Hidden + IC Linear	1	2	3674	16	160
	CIFAR-10	Shallow Network	8 Hidden + IC Linear	4	4	63514	16	160
		Pretrained ResNet	ResNet18 + IC Linear	0	20	11181642	512	5120
		Pretrained VGG	VGG16 + IC Linear	2	8	128812810	4096	40960

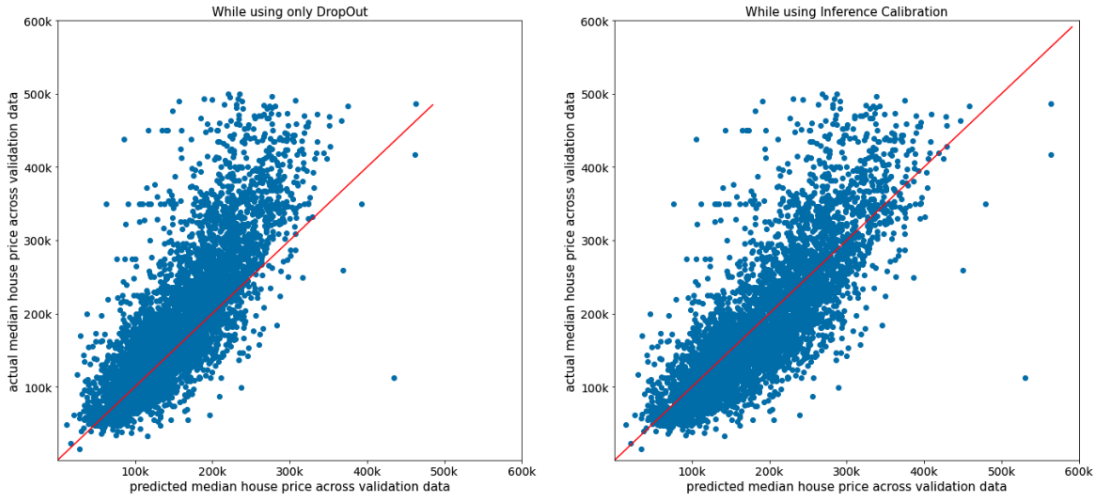


Fig. 1: California housing dataset: Prediction vs. actual values across validation dataset, left: without using inference calibration, right: with using inference calibration

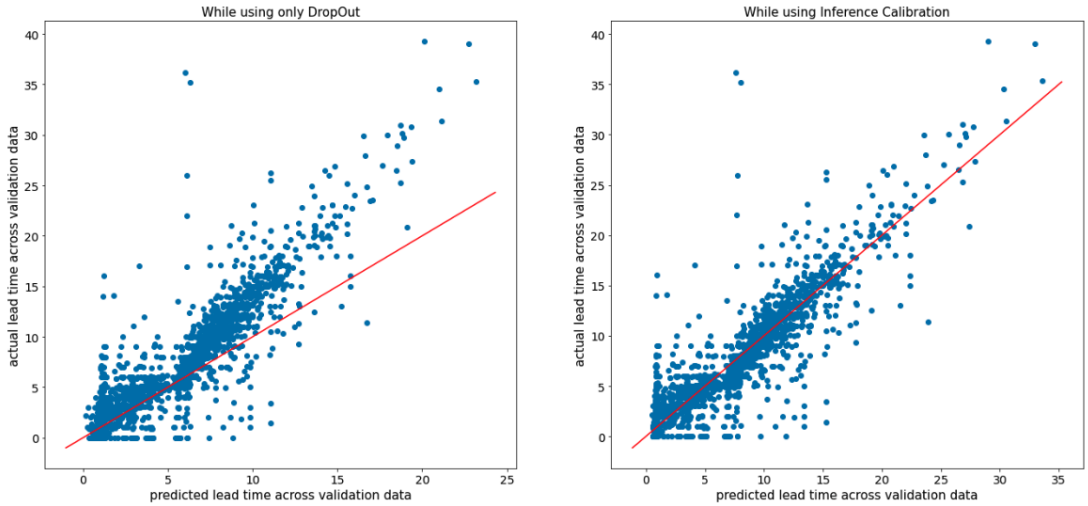
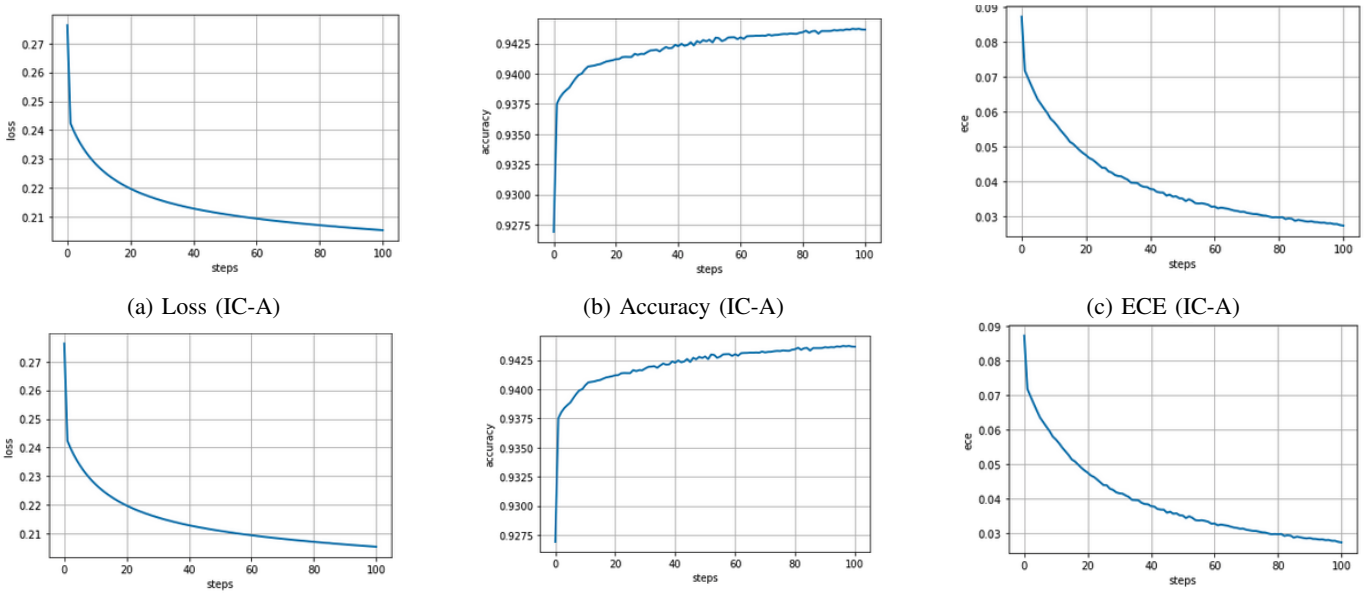


Fig. 2: Shipment lead time prediction: Prediction vs. actual values across validation dataset, left: without using inference calibration, right: with using inference calibration

TABLE III: RMSE and Calibration Error with and without Using Inference Calibration

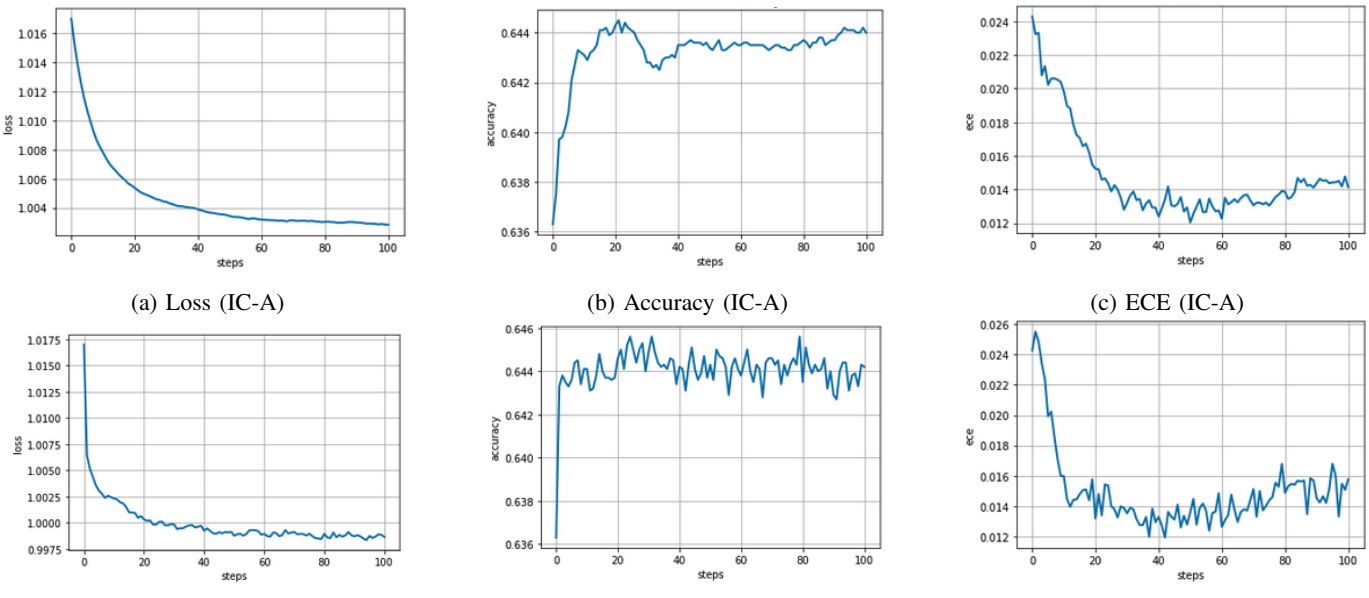
Dataset	Scenario	RMSE	e_{avg}
California housing	without Inference Calibration	68.103k	34.348K
	with Inference Calibration	**56.966k**	0.086K
	with Linear Regression bias correction	56.972k	**0.029K**
Shipment lead time	without Inference Calibration	2.669	0.968
	with Inference Calibration	**2.056**	-0.008
	with Linear Regression bias correction	2.070	**0.004**



(a) Loss (IC-A) (b) Accuracy (IC-A) (c) ECE (IC-A)

(d) Loss (IC-O) (e) Accuracy (IC-O) (f) ECE (IC-O)

Fig. 3: Effect of Inference Calibration on shallow neural network with Poker Hand Dataset



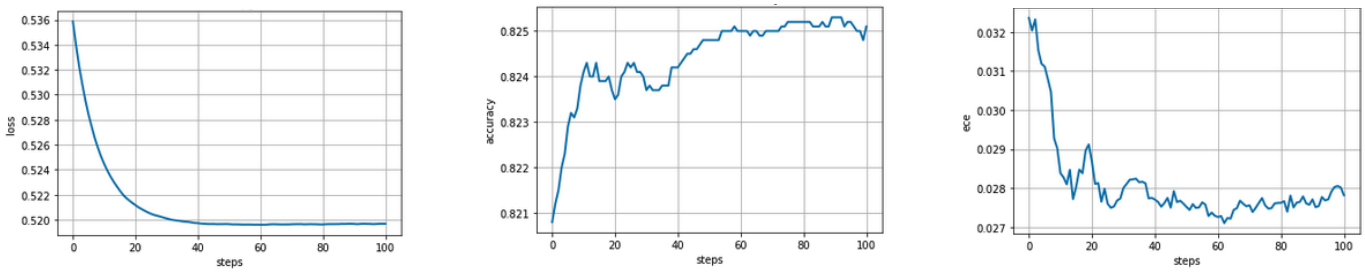
(a) Loss (IC-A) (b) Accuracy (IC-A) (c) ECE (IC-A)

(d) Loss (IC-O) (e) Accuracy (IC-O) (f) ECE (IC-O)

Fig. 4: Effect of Inference Calibration on shallow neural network with CIFAR-10 Dataset

TABLE IV: Summary of Experiment Results on Classification Problem

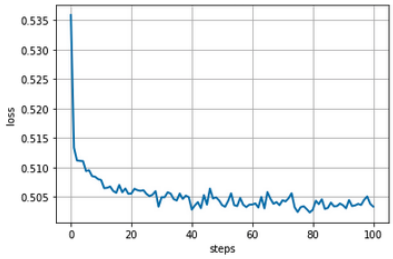
Dataset	Network	Application	Loss	Accuracy	ECE
Poker Hand	shallow network	No	0.276	92.69%	0.087
		Inference Calibration - A	**0.205**	94.36%	**0.027**
		Inference Calibration - O	**0.205**	**94.37%**	**0.027**
		Temperature Scaling	0.265	92.69%	0.073
CIFAR-10	shallow network	No	1.017	63.63%	0.024
		Inference Calibration - A	1.003	64.40%	**0.014**
		Inference Calibration - O	**0.999**	**64.42%**	0.016
		Temperature Scaling	1.029	63.63%	0.061
	ResNet-18	No	0.536	82.08%	0.032
		Inference Calibration - A	0.520	82.51%	0.028
		Inference Calibration - O	**0.503**	**83.03%**	0.033
		Temperature Scaling	0.531	82.08%	**0.021**
	VGG-16	No	0.395	86.88%	0.021
		Inference Calibration - A	0.383	87.22%	0.019
		Inference Calibration - O	**0.377**	**88.12%**	0.033
		Temperature Scaling	0.395	86.88%	**0.018**



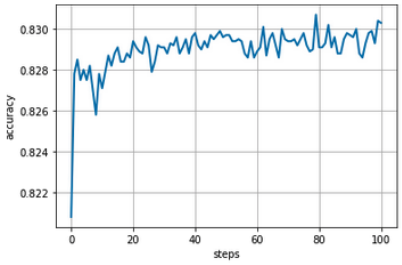
(a) Loss (IC-A)

(b) Accuracy (IC-A)

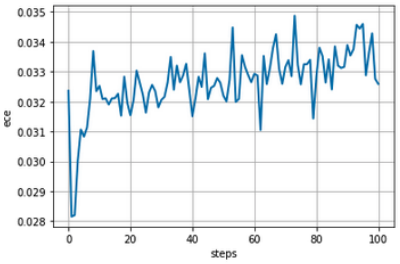
(c) ECE (IC-A)



(d) Loss (IC-O)



(e) Accuracy (IC-O)



(f) ECE (IC-O)

Fig. 5: Effect of Inference Calibration on ResNet-18 with CIFAR-10 Dataset

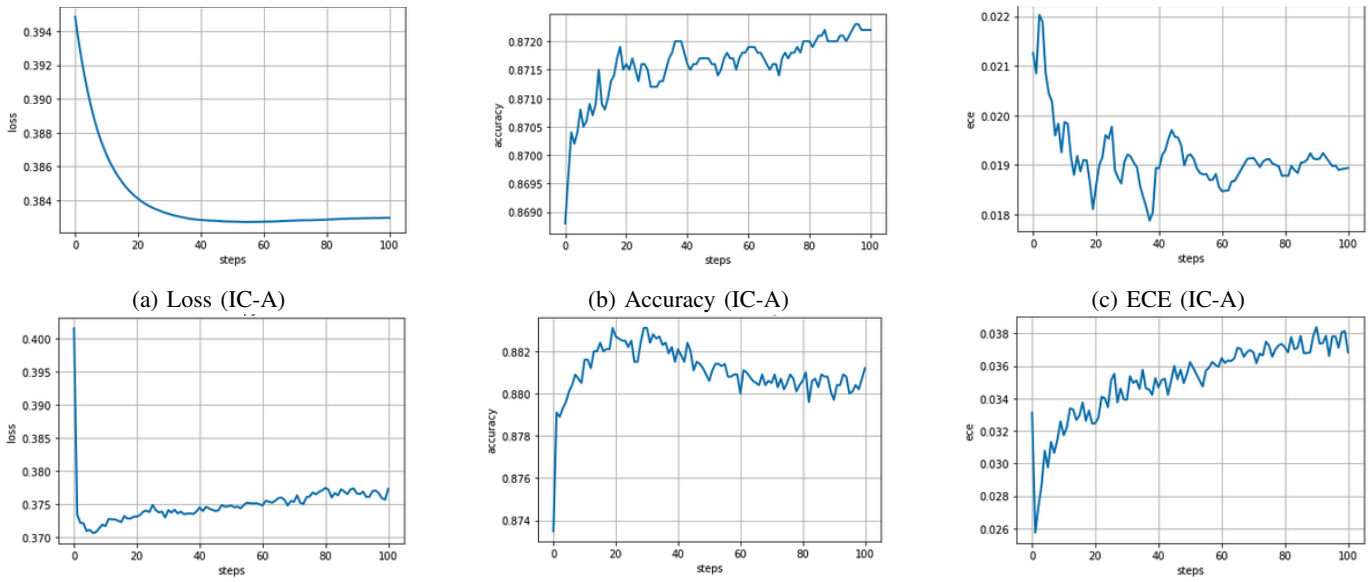
benchmark in this research and is widely used for probability calibration optimization in the industry. For transfer learning cases in the experiments on CIFAR-10 Dataset, IC-A shows comparable effect with Temperature Scaling on the reduction of ECE on VGG-16 while Temperature Scaling outperforms IC-A on ResNet-18. In comparison, while the IC-O has on-par performance to IC-A for the non-transfer learning models, it does not do a good job reducing ECE for transfer learning models. We suspect that the reduction on ECE is highly sensitive to the number of parameters being updated in Inference Calibration. This is aligned with the trend observed in Fig. 5 (f) and Fig. 6 (f) in which ECE is suddenly increased after a very limited number of IC-O iteration.

VI. Conclusion

Operations like Dropout and Batch Normalization cause neural network models to behave differently during the training and inference time. While Dropout plays an important role on preventing the model from overfitting and Batch Normalization makes the parameter be updated smoothly thus speeding up the training process, it has been noticed that they have negative impacts on the model accuracy and calibration due to the gap of implementation between training and inference. In this work, we proposed a general algorithm to correct these negative impacts, that is applicable across both regression and classification problems.

Our simulation results for regression problems demonstrate our proposed technique's ability to significantly mitigate RMSE and average error of original model's output. Our experiment results also show that the proposed technique is equivalent with linear regression bias correction, when it comes to reducing inherent biases resulted from Dropout.

For classification problems, our approach shows broad impact across different metrics such as loss, accuracy, and ECE. As shown by numerical experiments, using IC-A version of the algorithm results in improvement across all these metrics, for



(a) Loss (IC-A) (b) Accuracy (IC-A) (c) ECE (IC-A)

(d) Loss (IC-O) (e) Accuracy (IC-O) (f) ECE (IC-O)

Fig. 6: Effect of Inference Calibration on VGG-16 with CIFAR-10 Dataset

both shallow networks and pre-trained networks such as ResNet-18 and VGG-16. Compared to IC-A, IC-O gives better loss and accuracy although it is more prone to overfit in terms of ECE, probably due to more parameters being updated in the calibration.

In summary, the experiments show that the Inference Calibration algorithm effectively improves model's performance for regression models, classification models, and sophisticated pre-trained models on various datasets.

REFERENCES

[1] S. Nitish, H. Geoffrey, K. Alex, S. Ilya, and S. Ruslan, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[2] I. Sergey and S. Christian, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, pp. 448–456, 2015.

[3] M. Matthias, D. Josip, R. Rob, H. Frances Ann, Z. Xiaohua, H. Neil, T. Dustin, and L. Mario, "Revisiting the calibration of modern neural networks," *ArXiv*, vol. abs/2106.07998, 2021.

[4] W. Karl, K. Taghi M., and W. DingDing, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 9, 2016.

[5] D. Lixin, X. Dong, and T. Ivor W., "Learning with augmented features for heterogeneous domain adaptation," in *the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK*, pp. 1134—-1148, IEEE, 2012.

[6] K. Brian, S. Kate, and D. Trevor, "What you saw is not what you get: domain adaptation using asymmetric kernel transforms," in *Conference on Computer Vision and Pattern Recognition*, pp. 1785–1792, IEEE, 2011.

[7] Z. Yin, C. Yuqiang, L. Zhongqi, P. Sinno Jialin, X. Gui-Rong, Y. Yong, and Y. Qiang, "Heterogeneous transfer learning for image classification," in *The Twenty-Fifth AAAI Conference on Artificial Intelligence*, p. 1304–1309, 2011.

[8] W. Chang and M. Sridhar, "Heterogeneous domain adaptation using manifold alignment," in *The Twenty-Second International Joint Conference on Artificial Intelligence*, p. 541–546, 2011.

[9] Z. Tianyi, P. Jialin, T. Ivor W., and Y. Yan, "Hybrid heterogeneous transfer learning through deep learning," in *The Twenty-Eighth AAAI Conference on Artificial Intelligence*, p. 2213–2220, 2014.

[10] Z. Tianyi, T. Ivor W., P. Jialin, and T. Mingkui, "Heterogeneous domain adaptation for multiple classes," in *The Seventeenth International Conference on Artificial Intelligence and Statistics*, p. 1095–1103, PMLR, 2014.

[11] A. Özgür and F. Nar, "Effect of dropout layer on classical regression problems," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2020.

[12] G. Chuan, P. Geoff, S. Yu, and Q. Kilian, "On calibration of modern neural networks," *ICML'17: Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 1321–1330, 2017.

[13] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Impact of deep learning-based dropout on shallow neural networks applied to stream temperature modelling," *Earth-Science Reviews*, vol. 201, p. 103076, 2020.

[14] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), p. 609–616, Morgan Kaufmann Publishers Inc., 2001.

[15] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, (New York, NY, USA), p. 694–699, Association for Computing Machinery, 2002.

[16] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, p. 2901–2907, AAAI Press, 2015.

[17] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[18] T. Gneiting, F. Balabdaoui, and A. E. Raftery, "Probabilistic forecasts, calibration and sharpness," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 2, pp. 243–268, 2007.

[19] A. Kumar, P. Liang, and T. Ma, "Verified uncertainty calibration," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[20] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[21] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep. 0, University of Toronto, Toronto, Ontario, 2009.

[22] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[23] S. Karen and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.