Post-hoc Models for Performance Estimation of Machine Learning Inference

Xuechen Zhang Samet Oymak Jiasi Chen

Abstract—Estimating how well a machine learning model performs during inference is critical in a variety of scenarios (for example, to quantify uncertainty, or to choose from a library of available models). However, the standard accuracy estimate of softmax confidence is not versatile and cannot reliably predict different performance metrics (e.g., F1-score, recall) or the performance in different application scenarios or input domains. In this work, we systematically generalize performance estimation to a diverse set of metrics and scenarios and discuss generalized notions of uncertainty calibration. We propose the use of post-hoc models to accomplish this goal and investigate design parameters, including the model type, feature engineering, and performance metric, to achieve the best estimation quality. Emphasis is given to object detection problems and, unlike prior work, our approach enables the estimation of per-image metrics such as recall and F1-score. Through extensive experiments with computer vision models and datasets in three use cases – mobile edge offloading, model selection, and dataset shift – we find that proposed post-hoc models consistently outperform the standard calibrated confidence baselines. To the best of our knowledge, this is the first work to develop a unified framework to address different performance estimation problems for machine learning inference.

Index Terms—performance estimation, post-hoc models, uncertainty quantification, calibration

1 Introduction

Machine learning inference pipelines typically do not have any way of knowing how well they are doing during runtime, beyond the softmax probability score of the model. However, an estimate of the current inference performance for different applications can be very useful for a variety of purposes. For instance, in natural language processing, we might want to estimate the quality of a neural translation in terms of BLEU score. In object detection, we may want to estimate the F1-score and mean Average Precision (mAP), which provide critical summaries of the output quality but are not accessible during inference. In edge computing [1], [2], resource allocation decisions are made based on the estimated inference performance on the test set. Critically, all such decisions assume that the performance on the test domain is known, or rely on either similar distributional characteristics for the test and training domains. In this work, we seek to address this key stumbling block by asking

> **Q:** Can we estimate the performance of black-box models across different metrics, domains, & applications?

The key hypothesis is that it is possible to make accurate, per-example predictions of how well a DNN will perform, and utilize these predictions to improve practical use cases. To make these predictions quickly, we focus our design space on lightweight post-hoc models that operate on the outputs of the black-box DNN model that is performing the main inference. While there has been work on specific instances of performance estimation, such as predicting resource consumption or image segmentation quality [3], [4], to the best of our knowledge, this is the first work to provide a general framework for performance prediction of DNNs in a post-hoc fashion.

Related work has used calibrated confidence as an estimate of inference performance [5], [6], [7], [8], [9], [10], [11], [12], but our approach is more general and can accommodate a richer set of input features and output performance metrics. For instance, our proposed framework can estimate the performance gap across different models, in addition to the performance of a single model. Importantly, our evaluations show that our approach handily outperforms such confidence-based approaches. Additionally, to assess estimation quality, prior works are mostly restricted to variations of Expected Calibration Error (ECE), which as we discuss in Section 4.1, may not be suitable for certain applications. Works specific to object detection [12], [13], [14], [15] focus on estimating the uncertainty of the location or scale of a given detected object, but cannot evaluate the image as a whole, such as how many objects were missed (false negatives) which is needed to predict per-image metrics F1 score or recall. Perhaps more importantly, these works provide point solutions and do not address the growing need for general-purpose methodologies that can be effortlessly adapted to new problems of interest.

When designing our general framework to work for different use cases, a variety of challenges arise. How to define a general framework that can adapt to different use cases? What is the right choice of input features and design for the post-hoc model? Can the post-hoc model work well even if there are few samples? These questions become particularly challenging for applications beyond standard image classification. For instance, in object detection, models have very high-dimensional outputs (capturing the logits and locations of multiple objects), and the output dimension is variable (depending on the number of detected objects). To overcome these challenges, we systematically explore

- *X. Zhang and S. Oymak are with the Department of Electrical and Computer Engineering, University of California, Riverside.*
J. Chen is with the Department of Computer Science and Engineering, University of California, Riverside.
E-mails: xzhan394@ucr.edu, oymak@ece.ucr.edu, jiasi@cs.ucr.edu.

different post-hoc model designs and input feature choices, to develop a post-hoc model that works well even with few number of training samples.

Overall, this work addresses both high-level and application-specific challenges through the following contributions.

- **General Framework for Post-Hoc Model Design:** We formulate the general problem of post-hoc model design, which is flexible enough to accomodate a variety of performance metrics and input features based on the desired machine learning inference pipeline. This also leads to a natural notion of *calibration error* for a general class of performance metrics. We describe how our framework applies to three practical use cases, simply by modifying certain definitions in the framework; these use cases include (a) choosing between multiple machine learning inference models, (b) deciding whether to offload machine learning inference from a mobile device to an edge server, and (c) calibrating models after dataset shift.
- **Applications, Experiments, & Insights:** We perform extensive numerical experiments of object detection and image classification to show the efficacy of our approach, compared to the baseline of using calibrated confidence as an estimate of inference performance. We show that metrics such as F1 score, precision, and recall can be accurately predicted by our post-hoc model and outperforms a calibrated confidence baseline. For instance, in COCO dataset while confidence-based approach achieves 2.67% Expected Calibration Error (ECE) we achieve 1.65% ECE for F1 score prediction. We show that our post-hoc model can accurately predict the inference performance of the three use cases described above, on different datasets (COCO, VOC) and models (SSD, MobileNets, YOLO, etc.). A key finding is that performing intelligent feature selection and reducing the dimensionality of the black-box model outputs (which are the inputs to the post-hoc model) can greatly reduce sample complexity and enhance the performance estimates. Finally, besides commonly used ECE, we propose *Spearman's rank correlation* as an alternative metric to assess calibration performance and discuss its potential benefits over ECE.

The remainder of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe our general framework, followed by numerical results in Section 4. Finally, we conclude in Section 5.

2 Related Work

Confidence calibration: Confidence is one possible metric of inference performance, and there are a wealth of confidence calibration approaches in the literature, such as Platt/Temperature scaling [5], Histogram binning [7], Bayesian Binning into Quantiles (BBQ) [9], Isotonic regression [10], and Platt scaling extensions [8]. Several works [6], [11], [16], [17], [18], [19], [20], [21] consider either algorithmic improvements or application specific challenges associated to uncertainty quantification. [22] studies model uncertainty under dataset shift and provides empirical comparison of different calibration techniques. This work studies performance estimation metrics beyond confidence that are specific to the application, such as F1-score for object detection, which may be more interpretable by practitioners (*e.g.,* “what is the predicted F1 score of this test image?” is more interpretable than “what is the predicted confidence of this test image?”).

Confidence of object detection: Several works have examined confidence calibration for object detection specifically, which is also the application domain that this work focuses on. [12] presents a framework to measure and calibrate biased (or miscalibrated) confidence estimates of the model output. This approach results in calibrated confidence estimates of the object location and box scale. [13] additionally considers the impact of post-processing methods, such as non-maximum suppression, on confidence calibration. However, these works focus on performance estimates per object, whereas this work studies more general per image performance estimation metrics. These works are only able to calibrate confidence for objects that are detected in the image, and miss on those objects that were false negatives. In contrast, our approach can calibrate metrics that incoporate false negatives (such as F1 score or recall).

Other performance metrics: Prediction of other performance metrics (*e.g.,* segmentation quality, intersection over union) have also been studied in the literature. [14] proposes meta-regression to predict the intersection over union (IoU), and also classifies true and false positives. [15] predicts when the per-frame mAP drops below a critical threshold. [3], [23] predict segmentation quality. [24] evaluated prediction uncertainty after dataset shift. Our work provides a more general framework that can encompass these disparate performance metrics.

The area of image quality assessment (IQA) [25] seeks to predict the perceptual quality of images, with recent work using DNNs to perform the prediction [26]. IQA has some similarity to this work in that IQA also predicts various (perceptual) metrics, and the original source data is also images; however, the main difference is that we seek to predict one step further down the computational pipeline – not the quality of an image itself, but the quality of a prediction generated from that image.

3 Performance Estimation Framework

3.1 Post-hoc predictions of scores

We first describe the problem setup and our general framework for designing our post-hoc model. Let $f : \mathcal{X} \rightarrow \mathbb{R}^k$ be the machine learning model of interest for which we want to estimate the performance; for example, a deep neural network that maps input space into a k dimensional output. Let $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$ be the distribution of our dataset. Given an input $\boldsymbol{x} \in \mathcal{X}$, the model outputs a prediction $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$. For image recognition, $\hat{\boldsymbol{y}}$ is a probability distribution over the classes. However, in more practical scenarios, $\hat{\boldsymbol{y}}$ can contain more complex features. For instance, for object detection, $\hat{\boldsymbol{y}}$ contains the likelihoods of multiple objects, as well as their locations and sizes in the input image. During inference time, it is desirable to know the performance of the classification. This performance can be assessed through a performance metric $m(\boldsymbol{y}, \hat{\boldsymbol{y}})$. For example, m could be defined as the F1

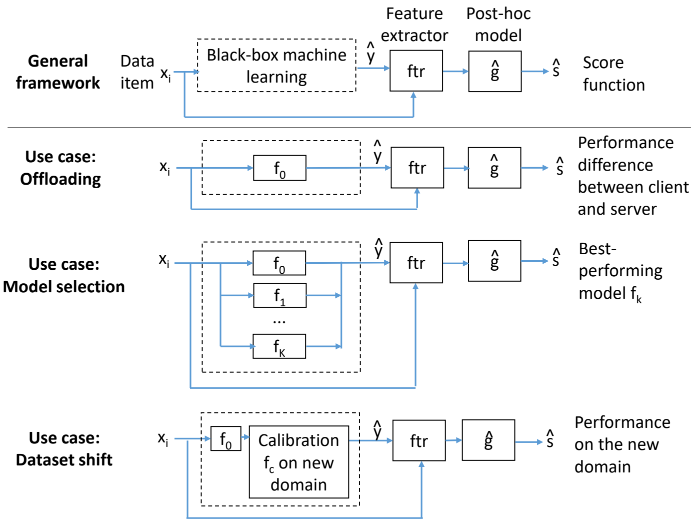

General framework
Data item
x_i
Black-box machine learning
$\hat{y}$
Feature extractor
ftr
Post-hoc model
$\hat{g}$
$\hat{s}$
Score function
Use case: Offloading
f_0
Performance difference between client and server
Use case: Model selection
f_1
...
f_K
Best-performing model f_k
Use case: Dataset shift
Calibration f_c on new domain
Performance on the new domain

Fig. 1: Overview of our performance estimation framework and its application to three use cases. The input data x_i is fed into a black-box machine learning model. The output prediction $\hat{y}$ is fed into a feature extractor (ftr), and from there into the post-hoc model $\hat{g}$. The final output is the score function $\hat{s}$. The black-box machine learning model, features, and performance estimate are defined based on the individual use cases (see Table 1). Jiasi to update figure

score, recall, mean Average Precision (mAP), BLEU score etc. We also define a score function $s^m(\cdot)$ which is a processed version of the raw performance metric m via

$$s^m(\cdot) = s \circ m(\cdot).$$

In simple use cases, we set the processing function as $s \leftarrow$ *identity*, and in more complex use cases, s^m is allowed to be a sophisticated function of m.

Our goal is to build a post-hoc model g that estimates the performance $\hat{\boldsymbol{s}}$ of the model f during inference. Let $\ell(\boldsymbol{s}, \hat{\boldsymbol{s}})$ be a loss function that takes the true performance $\boldsymbol{s}$ and the estimated performance $\hat{\boldsymbol{s}}$ as inputs. Let $\mathcal{S}$ be a training dataset for post-hoc modeling. Typically $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is a secondary training dataset of size n independent of the one used to build f. When training multiple models – as in the example of dataset shift use case (Section 3.4) – we will split $\mathcal{S}$ into disjoint sets (*e.g.*, train-validation). We use $\mathcal{S}$ to fit the model g via

$$\hat{g} = \arg\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \ell(s^m(\boldsymbol{y}_i, f(\boldsymbol{x}_i)), g \circ f(\boldsymbol{x}_i)). \tag{3.1}$$

An overview of our approach is shown in the top row of Fig. 1, with details on each of the use cases given in Table 1.

To refine this formulation, we consider the fact that the output f of the neural network might be very high-dimensional, resulting in a less interpretable g as well as being prone to overfitting during the training process. Additionally, the output f may be of variable length (for example, based on the number of objects detected by the black-box model). Instead of using the full output $f(\boldsymbol{x})$, we may only need a subset of the output to use as inputs to g. Thus, it is reasonable to utilize only a subset of elements from $f(\boldsymbol{x})$ as the inputs to g. On the other hand, some of the original input data $\boldsymbol{x}$ might also be useful to train $\hat{g}$. So overall, given a set of features $\mathrm{ftr}(f, \boldsymbol{x})$, we define the following generalization of (3.1):

Use case	Input to post-hoc	Score function
Model selection (Sec. 3.2)	Image features, output of all candidate models $\{f_k\}_{k=1}^K$'s features	Index of the best model (3.3)
Offloading (Section 3.3)	Image features, output of client model f_0's features	Performance metric improvement over f_0 (3.4)
Dataset shift (Section 3.4)	Image features, output of model after adaptation f_c's features	Performance on the new domain (3.5)

TABLE 1: Our general franework encompasses different use cases by simply changing the input features and output score function definitions.

$$\hat{g} = \arg\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \ell(s^m(\boldsymbol{y}_i, f(\boldsymbol{x}_i)), g \circ \mathrm{ftr}(\boldsymbol{x}_i, f)). \tag{3.2}$$

This is the form of $\hat{g}$ that we use throughput this paper. Later on in Section 4, we explore which features are most useful in learning $\hat{g}$. Defining the score functions and performance metrics in this general way allows our framework to cover a variety of use cases. We next discuss a simple example of how confidence calibration is a special case of our framework, before showing how our framework applies to three more complex use cases: model selection, device-server offloading, and dataset shift. The use cases are summarized in Fig. 1.

Generalized model calibration. A standard performance metric is the binary classification error $m(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \mathbf{1}_{\boldsymbol{y} \neq \arg\max_j \hat{\boldsymbol{y}}_j}$, where $\hat{\boldsymbol{y}}_j$ is the j^{th} element of $\hat{\boldsymbol{y}}$. In this scenario, $\hat{g}$ aims to predict the accuracy of f. To accomplish this, (3.1) can be optimized with a Bayes-consistent loss function such as cross-entropy as discussed in [8] to output an estimate of the correct probability $\mathbb{P}(\mathbf{1}_{\boldsymbol{y} = \arg\max_j \hat{\boldsymbol{y}}_j} \mid \boldsymbol{x}) = \hat{g} \circ f(\boldsymbol{x})$.

More generally, suppose the metric m takes values in $[0, 1]$ and we output an estimate $\hat{m} = \hat{g} \circ f$ by solving (3.1). We can then introduce calibration errors for m similar to confidence. For instance, the continuous Expected Calibration Error takes the form $\mathrm{ECE}(\hat{m}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[|\,\mathbb{E}[m(\boldsymbol{x}, \boldsymbol{y}) | \hat{m}(\boldsymbol{x}) = \alpha] - \alpha|]$, which is the average mismatch between the predicted and actual performance. Section 4 describes how solving problem (3.1) leads to refined calibration outputs for metrics specific to object detection.

3.2 Use Case: Model Selection

We next show how our framework can incorporate multiple models for the model selection use case. Besides $f \triangleq f_0$, suppose we have multiple candidate models $F = (f_k)_{k=0}^K$. Understanding the best model for an input $\boldsymbol{x}$ can be useful in a variety of scenarios, for example if the inference runs in the cloud with a library of models available [2], [27], and we need to choose the best model to run. Specifically, we define the index of the best model as

$$s^m_{\text{model}}(\boldsymbol{y}, \boldsymbol{x}, F) = \arg \max_{0\le k\le K} m(\boldsymbol{y}, f_k(\boldsymbol{x})).$$

In this use case, we set the score function to be $s(\{m_i\}_{i=0}^K) = \arg\max_{0\le k\le K} m_i$ with $m_i = m(\boldsymbol{y}, f_k(\boldsymbol{x}))$. We solve the index prediction problem by setting $s^m \leftarrow s^m_{\text{model}}$ in (3.2)

$$\hat{g} = \arg\min_{g\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^{n} \ell(s^m_{\text{model}}(\boldsymbol{y}_i, \boldsymbol{x}_i, F), g \circ \text{ftr}(\boldsymbol{x}_i, F)). \quad (3.3)$$

The predicted $\hat{s}$ obtained from $\hat{g}$ is the index of the (predicted) best model.

3.3 Use Case: Device-Server Offloading

Another use case is for a resource-constrained end device (*e.g.*, a smartphone) that wants to choose between a smaller local DNN model, or the library of larger models available in the cloud [28]. Here, let f_0 be the lightweight inference-time efficient model that is run on the client device. f_1 is a more accurate model that is more computationally intensive and hence run on in the cloud. If f_0 on the mobile device gives an inaccurate result, ideally we would like to offload the data to the cloud for inference by the stronger models. Thus the question is: Can we predict when f_0 fails, and if the data should be offloaded; and if so, which server model f_k should be used? Note that the difference between this offloading use case and the previous model selection use case (Section 3.2) is that we only have the result of f_0 as input to $\hat{g}$, whereas for model selection $\hat{g}$ has knowledge of all the models $f_0, f_1, \ldots, f_K$.

A naive approach is to send images with low confidence from the mobile device to the server [29]. A good calibration method can improve this naive approach through accurate confidence estimation; however, confidence alone as the offloading criterion is insufficient, because in some instances offloading even a low-confidence data sample to the server doesn't help (shown later through our experiments). Therefore, we define the offloading score s^m_{offload} as the *difference* between the performance of the mobile device's model, and the best server model

$$s^m_{\text{offload}}(\boldsymbol{y}, \boldsymbol{x}, F) = \max_{0\le k\le K} m(\boldsymbol{y}, f_k(\boldsymbol{x})) - m(\boldsymbol{y}, f_0(\boldsymbol{x})) \quad (3.4)$$

We then use s^m_{offload} in (3.2) to train $\hat{g}$.

Threshold policy: After solving for $\hat{g}$ using (3.2) and (3.4), to utilize the performance estimate, we propose a simple thresholding policy as a proof of concept. This empirical threshold is used to decide whether to offload new samples. Let $0 \le \rho \le 1$ be the offloading fraction, *i.e.*, the fraction of data items sent for processing on the cloud by one of the models $(f_k)_{k=1}^K$. $\rho = 0$ implies that only f_0 is used for inference on the mobile device, and $\rho = 1$ implies that all data is offloaded to the cloud (which is not a feasible solution due to high latency/communication costs). Then the threshold is defined as:

$$s^\rho_{\text{threshold}} = \lfloor \rho \rfloor\text{'th largest element of } (s^m_{\text{offload}}(\boldsymbol{y}_i, \boldsymbol{x}_i, F))_{i=1}^n.$$

In other words, we sort the data samples i by their performance gap s^m_{offload}, and given a desired offloading fraction ρ, we can compute the corresponding performance gap $s^\rho_{\text{threshold}}$. A new input data item $\boldsymbol{x}$ is offloaded if its predicted performance gap $\hat{s}^m_{\text{offload}}(\boldsymbol{y}, \boldsymbol{x}, F) \ge \max\{s^\rho_{\text{threshold}}, 0\}$.

3.4 Use Case: Dataset Shift

In our final example, we consider a dataset shift use case, where the model f has been trained on one domain with data distribution $\mathcal{D}_{\text{source}} \in \mathcal{X} \times \mathcal{Y}$, and is used in a related domain with distribution $\mathcal{D}_{\text{target}} \in \mathcal{X} \times \mathcal{Y}_{\text{target}}$. The motivation is that while existing techniques [30] can improve the accuracy on the new domain $\mathcal{D}_{\text{target}}$, they may result in mis-calibrated confidence, hence necessitating post-hoc adjustment.

We consider two post-hoc models f_c and $\hat{g}$, which are applied sequentially to the outputs of the black box model. First, model f_c minimizes the test error in the new domain by using the prediction error as a performance metric, *i.e.*, by solving (3.1) with $s^m(\boldsymbol{y}, f(\boldsymbol{x})) = \boldsymbol{y}$. In many instances, this can be accomplished through logit adjustment to account for changes in class priors [30], although more sophisticated strategies might be needed depending on the amount of dataset distribution shift. f_c is not the focus of this work, as adapting to dataset shift is a well-studied problem [24]; rather, we focus on performance estimation after dataset shift through model $\hat{g}$.

Following this adaptation to the new domain, we ask the question: Can we accurately predict the inference performance on the new domain using a post-hoc model $\hat{g}$? To do this, we define the score function for use in (3.2):

$$s^m_{\text{domain}}(\boldsymbol{y}, f(\boldsymbol{x})) = m(\boldsymbol{y}, f_c \circ f(\boldsymbol{x})). \quad (3.5)$$

This is perhaps the most straightfoward score function compared to the other use cases, as the score function is simply equal to the performance metric m, albeit on the test set in the new domain $\mathcal{D}_{\text{target}}$.

4 Numerical Experiments

In this section, we evaluate our post-hoc model framework through the three use cases (model selection, mobile offloading, and dataset shift) described in Section 3, with different performance metrics. We first describe our setup (Section 4.1), our experiments with a single model including the dataset shift use case (Section 4.2), and finally our experiments with multiple models (Section 4.3), including the offloading and model selection use cases. In summary, our results show that both gradient boosting and neural network-based post-hoc models uniformly outperform the baseline confidence calibration. Among these, gradient boosting trained with handcrafted input features attains the best performance in most scenarios, as opposed to full-dimension input features, suggesting that more information is not always better.

4.1 Experiment Setup

Datasets and pre-trained models: We evaluate our framework using two types of machine learning tasks: image classification and object detection.

Image classification: We use CIFAR-10 [35], which consists of 60,000 32x32 color images in 10 classes, with 6000 images per class. We split the dataset into training, two validation, and test sets, with 4500/100/300/1000 images per class,

Object detection model	Dataset	ECE
SSD MobileNets [31]	COCO	0.01648
SSD ResNet-50 [31]	COCO	0.01657
SSD Inception [31]	COCO	0.01748
Fast R-CNN ResNet-101 [32]	COCO	0.01976
Tiny YOLO [33]	VOC	0.01847
YOLO v2 [34]	VOC	0.01790

TABLE 2: Models used in our evaluation. Local prediction: The ECE of Post-hoc-XGB for F1-score prediction is low across models and datasets, demonstrating the generality of our approach across different black-box models.

Image features	**Color histogram entropy**
	Image size
	Number of corners
Model features	**Class score**, representing the importance of a class towards m; computed as the weights from a linear regression between the # of objects per class and m.
	Location score, representing the importance of a location in an image towards m; computed as the weight from a linear regression between 25 grid locations and m.
	Min confidence across objects in an image
	Max confidence across objects in an image
	Mean confidence across objects
	# of bounding boxes in an image
	Min bounding box size across objects
	Mean bounding box size across objects

TABLE 3: Handcrafted features used in our post-hoc model.

respectively. The target dataset $\mathcal{D}_{\text{target}}$ for the dataset shift use case (V1, V2) is a modified version of the two validation sets. Essentially, to model the distribution shift on the new domain, we randomly select 3 out of the original 10 classes and sample these classes with frequencies 3:3:1), then V1 is used to train the dataset shift calibrator f_c, and V2 is used to train the post-hoc model. The black-box machine learning model is a ResNet model [36].

Object detection: We use the VOC and COCO datasets. VOC [37] contains 11,540 images with objects from 20 target classes. We randomly pick 1000/500 images for validation and test, respectively. COCO [38] contains 91 classes, with 2.5 million labeled instances in 328k images in the dataset. We use 4000/1000 images for validation and test, respectively. We evaluate 9 different pre-trained models on these datasets, ranging from compressed models designed for mobile devices to more powerful models, as summarized in Table 2.

Post-hoc model: For the post-hoc model, we experimented with two model designs: a 3-layer fully connected neural network and gradient boosting using XGBoost [39]. We label the former as ***Post-hoc-NN*** throughout our experiments, and the latter as ***Post-hoc-XGB***. The neural network has 2 hidden layers. The number of neurons is two thirds of the input size for each layer (*e.g.,* for dataset shift, the input size is 20, so the number of neurons per layer is 13 and 9 respectively). The activation function is a ReLu layer. The learning rate is set to 0.03. The gradient boosting method uses trees with maximum depth 5, subsampling ratio 0.7, learning rate 0.1 and numer of epoch 300. We varied these hyperparameters and chose the values above based on their overall performance.

Per-image features: The inputs to the post-hoc model $\hat{\boldsymbol{g}}$ include model features (outputs of the black-box machine learning model) and image features (features pertaining to the original input image), as shown in Table 3. We created handcrafted features to summarize these inputs. The model features require summarization because the black-box model outputs might be very high-dimensional and their dimension is not fixed. The image features similarly require summarization because the high image resolution results in high-dimensional features. The intuition behind per-image features, considering the multiple models use case and number of bounding boxes feature as an example, is that knowing that there are many bounding boxes suggests a more complex image, suggesting that the post-hoc model should predict that a more powerful machine learning model is needed. In Section 4.2.1, we systematically investigate which of the per-image features correlate best with the performance metrics.

We note that, different from previous work that uses per-object features [12], [13], we use per-image features (*i.e.,* an aggregation of the features of all the objects in an image). This allow us to overcome the influence of undetected ground truth. In other words, using per-object features alone results in undetected objects, whose information can't be incorporated into the performance estimate (since they are false negatives and never detected). Per-image features, as we use, contain general information of the whole image, so it contains information about those undetected objects and can help us estimate per-image performance metrics such as recall and F1-score.

Metrics: The overall evaluation metrics include:

- **Expected calibration error (ECE)** [8], [9]: The ECE measures the calibration accuracy, by sorting the predictions into J bins ($J = 10$ in our experiments), and counting how many samples were put into the correct bins, weighted by the empirical probability of that bin: $\text{ECE} = \sum_{j=1}^{J} \frac{|B_j|}{n} \left|\text{true}\left(B_j\right) - \text{predict}\left(B_j\right)\right|$, where n is the number of samples, $\text{true}(B_j)$ is the number of samples that fall in bin B_j, and $\text{predict}(B_j)$ is the number of samples that were predicted to fall into bin B_j. ECE are used in generally throughout Section 4.2, as it is one of the most popular metrics to evaluate calibration accuracy.
- **Coefficient of determination (R^2):** The R^2 value is a measure of correlation, defined as $1 - \frac{\sum_i (y_i - \hat{y})^2}{\sum_i (y_i - \hat{y}_i)^2}$, where y_i is true value, $\bar{y}$ is the mean value of y, and $\hat{y}_i$ is the predicted value. This metric is used in Figure 2 to show the correlation between features and performance metrics.
- **Spearman's rank correlation coefficient:** The Spearman correlation coefficient is defined as the Pearson correlation coefficient between two rankings. In our case, we compute the Spearman correlation coefficient between the ground truth ranking and the ranking produced by the post-hoc model (e.g., convert the predicted F1 scores to a ranking). This is used in Section 4.2.1 and Section 4.2.2 for the basic evaluation and the dataset shift use case.

Different evaluation metrics are appropriate for different scenarios. Although the ECE is widely used to evaluate calibration, it has several drawbacks, leading us to consider the other additional metrics listed above. First, when the prediction output is continuous, the ECE computation requires bins, which are artifically introduced just for evaluation purposes. Second, ECE is sensitive to arbitrary bin settings, (*e.g.*, the more number of bins we use, the larger the error). Finally, we sometimes care more about whether one sample performs better than another, rather than the exact value of the performance metric. In the offloading use case, for example, we want to know which sample has better performance on the server compared to its peers, rather than exactly how much better it will perform. Rank correlation gives us the accuracy of these rankings, without being influenced by arbitrary settings, such as the number of bins for the ECE calculation.

The metrics we use to evaluate object detection and image classification include:

- **Intersection over Union (IoU)** [40]: IoU measures how well a detected object's location matches with the ground truth (for those objects correctly classified), and is defined as: IoU $= \frac{|A \cap B|}{|A \cup B|}$ where A and B are areas of the predicted and ground truth bounding boxes, respectively.
- **F1-score, recall, and precision:** F1-score is a measure of accuracy of the classification task, defined in terms of recall ($\frac{\text{true positives}}{\text{true positives + false negatives}}$) and precision ($\frac{\text{true positives}}{\text{true positives + false positives}}$) as $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. We compute the F1-score per image.
- **Accuracy:** The classification accuracy is defined as $\frac{1}{n} \sum_i 1_{\hat{y}_i = y_i}$.

Baselines: The baseline methods that we compare our post-hoc model against include:

- **Confidence:** For image classification, we use the regular confidence values output by the machine learning model. For object detection, the model may output a both a location confidence and class confidence/probability for each object in an image. For such models (*e.g.*, YOLO), we compute the combined confidence by multiplying the class probability (first term) and the location confidence (second and third terms) as: $P\,(\text{class} \mid \text{object}) * P(\text{object}) * IOU$. For models without location confidence, such as SSD, we just use the class confidence.
- **Calibrated confidence:** Confidence calibration methods include temperature scaling and vector scaling [8]. Briefly, temperature scaling scales all the logits by a scalar parameter T, while vector scaling is a multiclass extension that applies a linear transformation to the logits. To extend this to object detection, which also cares about an object's location, when training the calibration model we label a detected object as having 0 confidence if there is no object actually present.

4.2 Performance Prediction of a Single Model

4.2.1 Base Case: Local Prediction

We first study in-depth whether our post-hoc model works well to predict performance of object detection locally on a device. Success in this base scenario is necessary before applying our framework to more complex use cases in the following subsections. We examine the impact of different setups, including feature selection, post-hoc model performance, generality to different black-box models and datasets, and sample complexity.

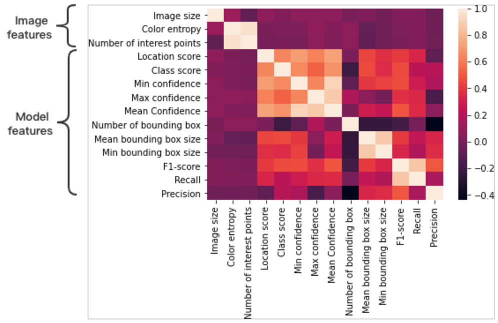


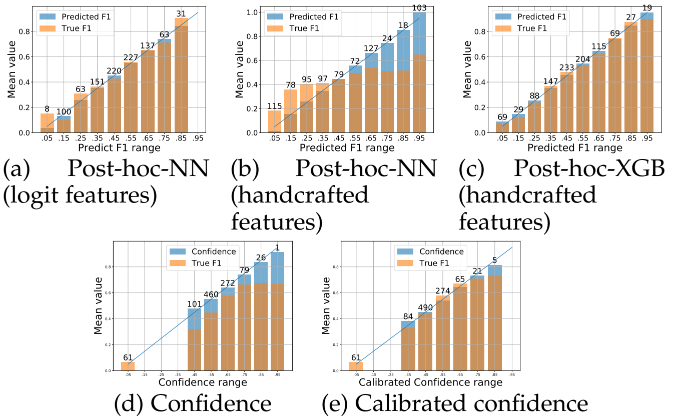
Fig. 2: Correlation between the features and the performance metrics (F1-score, recall, precision). The black-box model features are generally more highly correlated with the performance metrics than image features.

Feature selection: First, we investigate which handcrafted features to use as inputs to the post-hoc model. We find that the features relating to the black-box model, rather than image features, are more highly correlated with the performance estimates, and thus more useful to the post-hoc model. This is shown in Figure 2, which illustrates the correlation between the handcrafted features and the performance metrics (F1-score, recall, precision). The feature selection depends on the size of dataset, cases and object detection/image classification models. For instance, the class score and location score are highly correlated with the performance in normal scenarios with large enough validation set to train the post-hoc model, but can hurt if there is not enough training data (discussed later in the sample complexity experiments).

Post-hoc model performance: Using these handcrafted features, we next evaluate the performance of the post-hoc model. We trained a post-hoc model to predict three different performance metrics – precision, recall, and F1-score – of the object detector SSD MobileNets. To evaluate the performance of post-hoc model, we compute ECE to show the the difference in expectation between the predicted value and the true object detection model's performance metrics. Taking F1-score as an example, the calibrated confidence has an ECE of around 0.08, while our post-hoc model (specifically, Post-hoc-XGB) can decrease ECE to less than 0.02 (latter shown in Table 2).

To further examine this performance gain, in Figure 3, we plot the reliability diagrams of our approach and the baselines. These reliability diagrams show the mean value of the metric in question, as a function of the predicted metric. Figures 3(a), (b), and (c) show the different post-hoc model variants as a predictor of F1, while (d) and (e) show confidence as a predictor of F1. XGBoost with handcrafted features as input (Figure 3(a)) performs the best, as it most closely tracks the diagonal.

Figure 4 shows the ability of Post-hoc-XGB to predict



(a) Post-hoc-NN (logit features) (b) Post-hoc-NN (handcrafted features) (c) Post-hoc-XGB (handcrafted features)

(d) Confidence (e) Calibrated confidence

Fig. 3: Local prediction: Reliability diagrams, generated by binning the test examples by F1-score predictions from our post-hoc models (a,b,c), confidence (d) and calibrated confidence (e), and plotting the average predicted value (blue) and average true value (orange). With perfect prediction, the true value should align exactly with the predicted value along the diagonal.

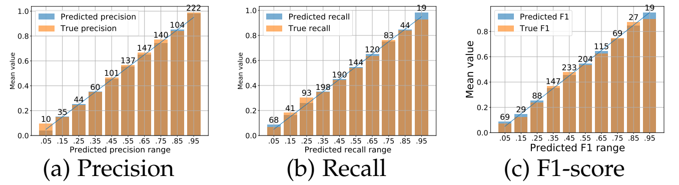
other performance metrics (precision and recall); the results indicate that our model can successfully predict these other performance metrics.

Generality across models and datasets: The results of our experiments with five additional object detection models and one additional dataset are shown in Table 2, leading to similar conclusions on the good performance of Post-hoc-XGB. Although different models have different structures and principles, and are trained for different datasets, our post-hoc model can estimate their performance accurately.

In summary, our results show that XGBoost with handcrafted features can accurately predict F1-score, precision, and recall, and outperforms the baselines across 6 different black-box models and 2 datasets.

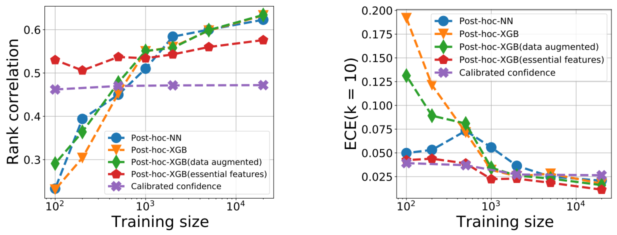
Sample complexity: We also experiment with how much data is needed to train the post-hoc model and the impact of a small training dataset. In Figure 5, we show the ECE and the rank correlation for the post-hoc model trained with different dataset sizes (*i.e.,* the validation set size). To counteract the effect of small training set sizes, we introduce two new variants of the post-hoc model: “Post-hoc-XGB (essential features)” and “Post-hoc-XGB(data augmented)”. The former excludes class and location score from the handcrafted features, as those scores are harder to infer from when there are few samples; the latter adds data augmentation (e.g., cropping, rotating) when training the post-hoc model.

The results show that while data augmentation is beneficial for small sample size its impact is rather limited (comparing the green curves to the orange curves). This makes sense as the post-hoc model needs to analyze what is contained each image (*e.g.,* number of objects, object classes) in order to make good predictions, and simply rotating or cropping images doesn’t give the post-hoc model more training examples with more diverse contents (*e.g.,* more objects or diverse object classes). Overall, as long as we can select the handcrafted features correctly (i.e., use Post-hoc-XGB (essential features) when the training set is small, and the regular Post-hoc-XGB when the training set is larger), Post-hoc-XGB can achieve good performance, better than or on par with other baselines across all training set sizes. The fact that “Post-hoc-XGB (essential features)” performs quite well even when there is limited training data demonstrates the algorithmic benefits of feature selection and reducing the dimensionality of the post-hoc model’s input.



(a) Precision (b) Recall (c) F1-score

Fig. 4: Local prediction: Reliability diagram of Post-hoc-XGB for different performance metrics, generated by binning the test examples by predicted (a) precision, (b) recall, and (c) F-1 score.



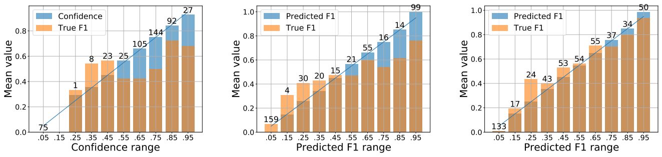
(a) Rank correlation (b) ECE

Fig. 5: Local prediction: We investigate the sample complexity for predicting F1-score. Post-hoc training is done for varying sizes of validation dataset. As sample size increases, more complex model tends to perform better both for ECE and rank correlation.

4.2.2 Use Case: Dataset Shift

Setup: We emulate dataset shift by generating new datasets with different class distributions. First, for image classification, we use the CIFAR-10 dataset, which has 10 classes, and a ResNet model. To model the dataset distribution shift on the new domain, we randomly select 3 out of the original 10 classes and sample these classes with frequencies 3:3:1. Let (V1,V2) be the datasets of the new domain. Second, for object detection, a ResNet model is trained on the COCO dataset and the VOC dataset is used as the new domain. V1 and V2 are from the VOC dataset. Recall that we must train two models, f_c for dataset shift and $\hat{g}$ for performance estimation. To obtain f_c, we use vector scaling trained on V1. $\hat{g}$ is trained on V2 using either the handcrafted features for object detection, or for image classification, a slightly modified set of features (logits, the label predicted by the black-box model, and compressed image using PCA)

Object detection results: Here we focus on the accuracy of the predicted class for each detected object in the image. We analyze the performance through reliability diagrams as before. Without a post-hoc model, the confidence of the original model plus dataset shift f_c is mis-calibrated (Figure 6a). Figures 6(b) and (c) show the reliability of our post-hoc model. Post-hoc-XGB (Figure 6c) achieves the best reliability (closest to diagonal) and thus can be helpful to predict the performance after dataset shift.



(a) Confidence of f_c (b) Post-hoc-NN (c) Post-hoc-XGB

Fig. 6: Dataset shift: Reliability diagrams for (a) the confidence of the dataset shift model f_c, and (b)(c) the F1-score of our post-hoc models.

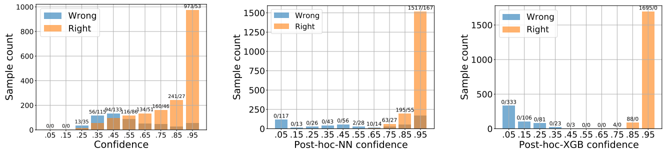
	ECE	Rank correlation
Confidence	0.17330	0.281
Post-hoc-NN	0.11344	0.560
Post-hoc-XGB	0.04299	0.584

TABLE 4: Dataset shift: Comparison of ECE, MAE, and R^2 for the confidence baseline and post-hoc models.

Table 4 highlights the performance improvement of our post-hoc model compared to the baseline domain calibration method for different metrics, with Post-hoc-XGB having the best (lowest) ECE, and slightly better rank correlation than Post-hoc-NN. These results are similar to that of the object detection base case discussed previously (Section 4.2.1), and provides further evidence that the simpler XGBoost-based post-hoc model outperforms a more complex post-hoc model based on neural networks.

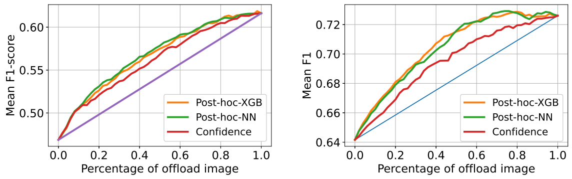
Image classification results: The reliability diagram results for image classification are consistent with that of object detection, showing that the post-hoc model can accurately predict the performance metrics after dataset shift, and are therefore omitted for brevity. Instead, we examine another facet of the problem: whether the post-hoc model can use an estimated performance metric (i.e., confidence) to separate mis-classified samples from the correct ones. Ideally, we would expect that a low predicted confidence would correspond to an incorrectly classified sample by the black-box model, and a high predicted confidence corresponds to correctly classified samples. In other words, is there a threshold for which samples with predicted confidence below the threshold are classified wrongly, while samples above the threshold are classified correctly?

Figure 7 shows the number of rightly/wrongly classified samples for different confidence bins. Ideally, we would expect all the wrongly classified samples (blue) to be to the left of the figure (low predicted confidence), while all of the correctly classified samples (orange) to be on the right of the figure (high predicted confidence). We see that post-hoc-XGB predicts very well (Figure 7c), achieving good separation between the wrongly/correctly classified samples, with the wrongly classified samples corresponding to predicted confidence of less than 0.7, and the correctly classified samples corresponding to predicted confidence larger than 0.5. In contrast, the vanilla dataset shift model f_c (Figure 7a) has high-confidence samples that are misclassified, and vice versa.



(a) Confidence of f_c (b) Post-hoc-NN (c) Post-hoc-XGB

Fig. 7: Dataset shift: Distribution of rightly/wrongly classified samples, generated by binning the test samples by confidence, as achieved by the (a) dataset shift model f_c or our post-hoc prediction models (b)(c). Ideally, we should see high confidence samples classified rightly, and vice versa.



(a) SSD MobileNet (client) & SSD ResNet-50 (server)

(b) SSD Inception (client) & Fast R-CNN ResNet-101 (server)

Fig. 8: Offloading: Comparison of post-hoc models with baselines to decide which samples to offload, for two client-server model pairs. The x-axis is the fraction of images offloaded (as determined by the predicted F1-score difference for the post-hoc models, or confidence for the confidence baseline).

In summary, our post-hoc model works well to predict performance after dataset shift, for both image classification and object detection, decreasing ECE by 0.13 for example, compared to the confidence baseline in object detection.

4.3 Performance Prediction of Multiple Models

In this section, our goal is to evaluate how well post-hoc models can estimate the performance of a combination of black-box machine learning models.

4.3.1 Use Case: Device-Server Offloading

In this set of experients, we study whether the post-hoc model can correctly determine the F1-score gap (3.4) between the mobile device's model and the server's model. This is challenging because we do not know server model's output in advance to make the prediction, but intuitively, for images that are more complex or with many small objects, the gap will be larger and the image should be offloaded to a server with a more powerful black-box model. We set $K = 1$, *i.e.*, there is one machine learning model stored on a client, and one machine learning model available on a cloud server. The client and server machine learning models we used are listed in Table 2; we tried many combinations with similar results, so for brevity, only the results from SSD MobileNet paired with SSD ResNet-50, as well as SSD Inception paired with Fast R-CNN ResNet-101 are discussed.

Figure 8 plots the mean F1 score of the various methods (ground truth, post-hoc model, and confidence baseline) vs. the fraction of test images offloaded. The test images are sorted by the predicted performance difference between the two models (or the confidence, for the confidence baseline). A simple thresholding policy is used to determine which images should be offloaded according to their predicted

Model	Mean F1-score
SSD MobileNets	0.46872
SSD ResNet-50	0.61620
SSD Inception	0.64959
Fast R-CNN ResNet-101	0.64613
Combined (Post-hoc-NN)	0.65399
Combined (Post-hoc-XGB)	0.66841
Optimal	0.70983

TABLE 5: Model selection: Mean F1-score of the individual black-box models, and the model chosen by the post-hoc model.

performance difference; a lower threshold leads to more images being offloaded, and a higher threshold leads to more images being offloaded, as discussed in Section 3.3.

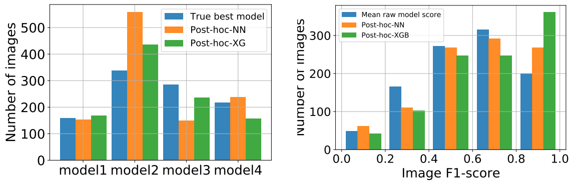
The results show that as we sweep across thresholds, the offloading fraction increases, more images will be offloaded to the server, and hence the mean F1-score across all images improves. Our post-hoc model (orange/green/red) has a monotonically increasing trend and can increase the mean F1-score across all test images by precisely picking which images to offload. The baseline confidence method performs worse as it has lower mean F1-score across offloading fractions; for example, when offloading 20% of the images (corrresponding to $S^{\rho}_{\text{threshold}} = 0.32$), the improvement of our post-hoc model over the confidence baseline is 0.01 in terms of mean F1 score. When the offloading fraction is 40% (corresponding to $S^{\rho}_{\text{threshold}} = 0.19$), the improvement is 0.013 (for scale, note that the total average true performance difference is only 0.147).

In summary, our post-hoc model accurately predicts the performance difference, resulting in an offloading of certain images to a more powerful black-box model, hence improving the average F1-score across all test images. The average F1 improvement over the confidence baseline is up to 0.013 for SSD Inception + Fast R-CNN ResNet-101.

4.3.2 Use Case: Model Selection

How well can the post-poc model predict the performance of several models, and hence choose the right model for a given image? We first adjusted the classification thresholds of the models to get four models with diverse performance across the different test images (otherwise, one black-box model might always consistently outperform the others). The post-hoc model is trained as discussed in Section 4.3.2. Table 5 shows the mean F1-score of each model individually (rows 1-4), as well as from our post-hoc model (rows 5-6) that tries to pick the best model. The results show that the post-hoc model can choose the appropriate black-box model and achieve higher F1-score, compared to using a individual model alone.

To delve deeper into these results, we perform a more detailed analysis of the models chosen by each approach. Figure 9a shows the number of test images assigned to each model, according to the optimal policy (labeled as "true best model"), as well as chosen by the post-hoc model. Note that although SSD Inception generally has the highest mean F1 score, it's not the optimal choice for all samples. Ideally, any post-hoc model should match with the true best model (blue bars). We can see that the distribution of Post-hoc-XGB (orange bars) most closely matches the optimal policy, suggesting that it's able to accurately predict the F1-scores of the images.



(a) Histogram of test samples corresponding to each model

(b) Histogram of mean F1-score achieved by the different methods

Fig. 9: Model selection: Distribution of test results. (a) We count the number of images matched to each model by the post-hoc model or the ground truth; more closely matching the true best model (blue) is better. (b) We bin the test samples by F1-score and plot the number of images in each bin; more images with higher F1-score is better.

Figure 9b shows the distribution of F1-score across all the test images, as achieved by the post-hoc models, compared to the average F1-score across all models achieved by each image (labeled "mean raw model score" – essentially, a measure of how easy/difficult the test images are). Ideally, there will be more samples with high F1-score, and vice versa. In other words, we hope to see the low F1-score bins (left side) contain fewer images, and the high F1-score bins (right side) with more images.

The results show that the number of images with low F1-score (lower than 0.2) according to Post-hoc-XGB is small, while many images achieve an F1-score between 0.8-1. *In summary, the post-hoc model is able to select the most suitable black-box model for each image and achieve higher mean F1-score, an improvement of 0.018 over the best-performing solo black-box model.*

5 Conclusions

In this paper, we proposed a framework to predict inference performance for a diverse set of metrics and scenarios. We developed a gradient boosting-based post-hoc model that is easy to train and flexible to use. Our model predicts the per-image performance for a variety of metrics, going beyond baseline confidence methods that only predict the class probabilities. By incorporating handcrafted features (*e.g.*, the number of bounding boxes) into our post-hoc model, we were able to outperform a confidence baseline. Future work includes extending the framework to additional performance metrics and use cases, such as hyperparameter optimization for machine learning model architecture search.

Acknowledgments

This work is supported in part by NSF CAREER awards CCF-2046816 and CNS-1942700, by Army Research Office under MURI Grant W911NF-21-1-0312, and by a Facebook Faculty Research Award.

References

[1] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1421–1429.

[2] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 236–252.

[3] R. Robinson, O. Oktay, W. Bai, V. V. Valindria, M. M. Sanghvi, N. Aung, J. M. Paiva, F. Zemrak, K. Fung, E. Lukaschuk *et al.*, "Real-time prediction of segmentation quality," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 578–585.

[4] Z. Lu, S. Rallapalli, K. Chan, and T. La Porta, "Modeling the resource requirements of convolutional neural networks on mobile devices," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1663–1671.

[5] 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.

[6] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[7] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *Icml*, vol. 1. Citeseer, 2001, pp. 609–616.

[8] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1321–1330.

[9] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[10] 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*, 2002, pp. 694–699.

[11] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," *arXiv preprint arXiv:1901.09960*, 2019.

[12] F. Kuppers, J. Kronenberger, A. Shantia, and A. Haselhoff, "Multivariate confidence calibration for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[13] F. Schwaiger, M. Henne, F. Küppers, F. S. Roza, K. Roscher, and A. Haselhoff, "From black-box to white-box: Examining confidence calibration under different conditions," *arXiv preprint arXiv:2101.02971*, 2021.

[14] M. Schubert, K. Kahl, and M. Rotmann, "Metadetect: Uncertainty quantification and prediction quality estimates for object detection," *arXiv preprint arXiv:2010.01695*, 2020.

[15] Q. M. Rahman, N. Sünderhauf, and F. Dayoub, "Performance monitoring of object detection during deployment," *arXiv preprint arXiv:2009.08650*, 2020.

[16] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

[17] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O'Donoghue, D. Visentin *et al.*, "Clinically applicable deep learning for diagnosis and referral in retinal disease," *Nature medicine*, vol. 24, no. 9, p. 1342, 2018.

[18] S. Thulasidasan, G. Chennupati, J. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," *arXiv preprint arXiv:1905.11001*, 2019.

[19] A. Kumar, P. Liang, and T. Ma, "Verified uncertainty calibration," *arXiv preprint arXiv:1909.10155*, 2019.

[20] Y. Zhao, J. Chen, and S. Oymak, "On the role of dataset quality and heterogeneity in model confidence," *arXiv preprint arXiv:2002.09831*, 2020.

[21] A. Kumar, S. Sarawagi, and U. Jain, "Trainable calibration measures for neural networks from kernel mean embeddings," in *International Conference on Machine Learning*, 2018, pp. 2810–2819.

[22] J. Snoek, Y. Ovadia, E. Fertig, B. Lakshminarayanan, S. Nowozin, D. Sculley, J. Dillon, J. Ren, and Z. Nado, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 969–13 980.

[23] M. Rottmann, P. Colling, T. P. Hack, R. Chan, F. Hüger, P. Schlicht, and H. Gottschalk, "Prediction error meta classification in semantic segmentation: Detection via aggregated dispersion measures of softmax probabilities," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–9.

[24] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *arXiv preprint arXiv:1906.02530*, 2019.

[25] Z. Wang and A. C. Bovik, "Modern image quality assessment," *Synthesis Lectures on Image, Video, and Multimedia Processing*, vol. 2, no. 1, pp. 1–156, 2006.

[26] X. Yang, F. Li, and H. Liu, "A survey of dnn methods for blind image quality assessment," *IEEE Access*, vol. 7, pp. 123 788–123 806, 2019.

[27] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 377–392.

[28] J. Chen and X. Ran, "Deep learning with edge computing: A review." *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[29] S. Wang, S. Yang, and C. Zhao, "Surveiledge: Real-time video query based on collaborative cloud-edge deep learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2519–2528.

[30] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar, "Long-tail learning via logit adjustment," *arXiv preprint arXiv:2007.07314*, 2020.

[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[32] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[33] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2503–2510.

[34] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[35] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[37] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[38] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[39] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.

[40] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union," June 2019.