Design and Evaluation of Object Classifiers for Probabilistic Decision-Making in Autonomous Systems

Hamad Ullah[1], Weisi Fan[1], and Tichakorn Wongpiromsarn[1]

[1]Department of Computer Science, Iowa State University

***Abstract*—Object classification is a key element that enables effective decision-making in many autonomous systems. A more sophisticated system may also utilize the probability distribution over the classes instead of basing its decision only on the most likely class. This paper introduces new performance metrics: the absolute class error (ACE), expectation of absolute class error (EACE) and variance of absolute class error (VACE) for evaluating the accuracy of such probabilities. We test this metric using different neural network architectures and datasets. Furthermore, we present a new task-based neural network for object classification and compare its performance with a typical probabilistic classification model to show the improvement with threshold-based probabilistic decision-making.**

I. Introduction

Perception and control are two main components of autonomous systems, as shown in figure 1. The perception component extracts the relevant features from the environment and performs the classification task. The control component takes this perceived environment as input and makes the corresponding decision for the system to react to the real-world environment.

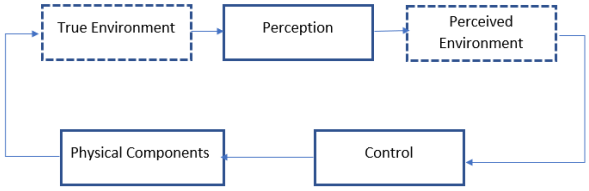


Fig. 1. A typical architecture of autonomous systems.

Object classifiers are typically evaluated based on classification accuracy, i.e., the ratio of correct classifications [1]. Recently, deep learning models, especially convolutional neural networks (CNN), have been widely deployed in the perception component of many autonomous systems [2, 3]. These models take samples that are drawn from the whole population to perform the statistical inference in the training process. For example, a model for handwritten digits recognition can be constructed from the MNIST dataset [4], which contains samples drawn from the handwritten digits data. A model is, therefore, only an estimator obtained from the samples. As a result, it is impracticable for a deep learning model to make a correct prediction on all of the new data [5]. In particular, most existing models cannot achieve 100% accuracy in real-world classification tasks [6, 4, 7].

Specifically, there are two cases in which a model would make incorrect classifications [5]. The first case is where the input data follows the i.i.d. assumption [8], but the trained model cannot capture the correct conditional distribution since we only have limited data (samples). To fix this problem, we can collect more data to train the model. The second case is where the input data does not follow the i.i.d. assumption (non-i.i.d) or comes from out-of-distribution (OOD). The OOD issue happens when the new observed data comes from a different distribution or a shifted distribution compared to the data used to train the model [5]. In this case, it is very likely that the trained model would fail to make statistical inference on such OOD data as the OOD data is from a different population from the data used to train the model. On the other hand, the non-i.i.d issue happens when the data is collected from different distributions [9]. If the model is trained under the i.i.d assumption with non-i.i.d data, it would make inference based on the wrong assumption, potentially leading to a poor performance.

Incorrect object classifications potentially lead to unsafe behaviors of autonomous systems. Let us take an autonomous car as an example. Suppose the car detects an object next to a crosswalk. The control component will stop the car if and only if the perception component identifies that the most likely class of the object is a pedestrian. In this case, the car will make a correct decision only if the object classification is perfect, which is not realistic in most systems.

With imperfect classifications, the controller may utilize the probability distribution over the object classes to make better decisions. For example, suppose the object near the crosswalk is, in fact, a pedestrian and the perception component reports that the object belongs to the non-pedestrian class and the pedestrian class with probability 0.51 and 0.49, respectively. In this case, the classification result based on the most likely class will be incorrect. However, the controller may take the high probability of the object being

This work was partially supported by NSF Grant 2141153.

a pedestrian into account and decide to stop the car. As a result, the probability distribution over the classes can play an essential role in decision making. Thus, it is important to estimate the accuracy of such probability estimates as a model may achieve a high classification accuracy based on the most likely class even if the probability estimates are not accurate. In fact, in Section V-A, we give an example of two networks that achieve a similar accuracy but with a significant difference in the uncertainty of each class.

Uncertainty quantification and reduction is an important topic in the machine learning area. Epistemic uncertainty and aleatoric uncertainty are two major types of uncertainty that can be modeled [10]. Epistemic uncertainty accounts for the uncertainty in the model. Recently, many methods, including Bayesian and non-Bayesian approaches, focus on quantifying and reducing the epistemic uncertainty. Dropout-based variational inference [11] and stochastic gradient MCMC [12] are two popular Bayesian approaches for epistemic uncertainty. On the other hand, the non-Bayesian methods consist of training multiple probabilistic neural networks with bootstrap [12] or ensembling [13]. As epistemic uncertainty can be reduced through the collection of additional data, data augmentation [14] is another way to reduce epistemic uncertainty. Aleatoric uncertainty comes from the noise within the observations and can be captured by MAP inference [10].

In this paper, we quantify the uncertainty of the probability estimates to measure their accuracy. For the rest of the paper, we will use the term "uncertainty" to refer to the uncertainty of these probability estimates. Such uncertainty quantifies the confidence interval of the probabilities.

With this quantified uncertainty, the controller can make more rational decisions based on the quantified confidence values. Recent work focuses on two directions relating to the uncertainty of the model: quantifying the uncertainty to make the model selection and calibrating the model to obtain lower uncertainty [5]. Section II will discuss these two directions in more details.

The main contributions of this paper are two-fold. Firstly, we present a new evaluation metric to measure the accuracy of the probability distribution over the classes. Secondly, we propose a task-based classifier for the case where the controller makes a decision based on whether the probability that an object belongs to a certain class is above a given threshold. The experimental results show the improved system-level performance compared to ordinary probabilistic classifiers.

II. Related Work

Consider the data $\mathcal{D} = \{(x_j, y_j)\}$, where $j \in \{1, 2, ..., n\}$, n as the size of the data. Here, $x_j \in \mathbb{R}^d$ is a given d-dimensional input and $y_j \in \mathbb{K} = \{1, 2, ..., K\}$ is its true label. The output of a probabilistic model $p(y|x_j, \theta)$ is the predicted probability of the specific class $y \in \mathbb{K}$ and θ is the model's parameters. Several metrics are widely used to quantify the uncertainty for evaluating probabilistic models.

1) Brier Score (BS) [15] is defined as

$$BS = \frac{1}{n}\sum_{j=1}^{n}\sum_{y\in\mathbb{K}}(p(y|x_j,\theta) - 1\{y = y_j\})^2.$$

This score measures the the difference between the predicted probability and the true probability. It is also a proper scoring rule [16] since it would be minimized under fully correct prediction results,

2) Expected Calibration Error (ECE) [17] uses the binning methods to compute the gap between the predicted probability and the true probability and is defined as

$$ECE = \frac{1}{n}\sum_{s=1}^{S}|\sum_{j\in B_s} 1\{y_j = \tilde{y}_j\} - \sum_{j\in B_s} p(\tilde{y}_j|x_j,\theta)|,$$

where $B_s = \{j \in (1, 2, ..., n) : p(y_j|x_j, \theta) \in (\rho_s, \rho_{s+1}]\}$ and $\tilde{y}_j$ is the predicted results of n-th test data. The purpose of ECE is to average the difference between the predicted probability and the true probability by different buckets. Thus, it computes the gap of the data in different confident groups. By assigning different weights based on the number of data points in the group, the weighted results would be reported by ECE.

3) Negative Log-Likelihood (NLL) [18] is defined as

$$NLL = -\sum_{j=1}^{n}\log(p(y_j|x_j,\theta)).$$

This metric is a direct measurement of the output probability, since it is a monotonically increasing function with respect to the output probability.

All the above metrics can give us the quantified uncertainty of the model and can be used as a performance metric [5]. However, they have some limitations. First, the Brier score is not sensitive to the predicted probability related to frequent events. Additionally, ECE is not the proper scoring rule, i.e., it is not monotonically increasing when the predicted probability is close to the true label. A model with good prediction results could show up with a higher ECE score. For the Negative Log-Likelihood, it can over-emphasize tail probability [5, 19].

Another major concern with the above metrics is that they only measure the overall uncertainty of the model. For autonomous systems, different object classes may not be equally important. For example, some classes such as pedestrians or vehicles cannot afford misclassification due to safety consideration. Thus, it requires the model to have low uncertainty on some of the classes under different tasks when making the prediction. Only providing the overall uncertainty of the model is not enough to help choose the most suitable model.

Classification deep learning models usually use a softmax layer before the output layer. Strictly speaking, the output of the softmax layer cannot be treated as the probability of

each class [20, 21, 22]. Additionally, many have observed that this approach may do a poor job of predicting probability [22]. Nevertheless, many training techniques, including Bayesian Binning into Quantiles (BBQ) [17], Temp Scaling [23], Dropout [11], Ensemble [24], explain the output of the softmax layer as the probability distribution over the classes and calibrate the model to reduce its uncertainty. Moreover, BS, ECE, and NLL are commonly used to show the real improvement of calibration. In this paper, we also treat the output of the softmax layer as the probability distribution over the classes. To test the real performance of the calibration methods, we design a new metric to capture more information about the model uncertainty. As opposed to BS, ECE and NLL, our metric can measure the uncertainty on the classification results of each class.

By treating the output of the softmax function as the probabilities of the object belonging to the associated classes, the controller may treat an object as belonging to the class with the maximum probability from resultant softmax vector. Another widely-used decision-making framework that utilizes the probability estimates is based on the probability thresholds. In this case, a threshold t_c is introduced for each object class c. The controller then treats an object as belonging to class c if its probability is at least t_c. Note that in this threshold-based probabilistic decision-making approach, an object may belong to multiple classes. Thus, the classification results could be ambiguous and the controller may treat the object as belonging to the class that would lead to the most conservative action. In this paper, we present a new task-based network structure to be utilized by such a controller. Instead of explicitly computing the probability, our network directly determines whether the probability is above or below a given threshold. More details will be discussed in Section IV-B.

III. Problem Setup

The classification task is to predict the label (i.e., the class) for any given input data. Specifically, let $\mathbb{K}$ be the set of all the object classes. A deep learning model first takes a subset of the data $\mathcal{D}_{train} \subset \mathcal{D}$ to train the model. The trained model can be treated as a function $f : \mathbb{R}^d \to \mathbb{K}$. Given any input data $x \in \mathbb{R}^d$, it will predict the label $\tilde{y} \in \mathbb{K}$. The rest of the data $\mathcal{D}_{test} = \mathcal{D} \setminus \mathcal{D}_{train}$ will be used to compare the predicted label $\tilde{y} = f(x)$, where $(x, y) \in \mathcal{D}_{test}$, with the real label y to get the training accuracy for each class.

Most deep learning models use the softmax layer before the output layer. The softmax function outputs a value $p_c \in [0, 1]$ for each class $c \in \mathbb{K}$. The label c with the largest p_c among all the classes will be outputted as the classification result, i.e.,

$$\tilde{y} = \max_{c \in \mathbb{K}} p_c$$

The output value p_c is often treated as the probability of class c. Our first problem is to quantify the uncertainty of p_c to evaluate the performance of the model. Our second problem is to design neural networks to improve the performance of the system for the case where the controller makes a decision based on whether the probability that an object belongs to a certain class is above a given threshold.

IV. Methods

This section discusses the proposed evaluation metric and task-based neural network. Section IV-A introduces the definition of our evaluation metric. Section IV-B demonstrates the proposed task-based neural network.

A. Evaluation of Model Uncertainty

Let $\#c$ denote the number of samples from class c, where $c \in \mathbb{K}$, for any dataset $\mathcal{D}$. The softmax layer of a deep learning model returns a probability vector $P_{x_j} = (p_{j1}, ..., p_{jK}) \in [0, 1]^K$ for any $x_j \in \mathbb{R}^d$. Based on the output probability P_{X_j}, the expected number of samples belonging to each class c can be defined as:

$$E(\#c) = \sum_{j=1}^{n} p_{jc}.$$

The basic idea is that the model with lower uncertainty should have $E(\#c)$ close to $\#c$ [8]. Moreover, this implies that if the model is uncertain about its output, $E(\#c)$ would be far from $\#c$. A suitable model should have $E(\#c)$ closer to the actual $\#c$. So the error of the model can be defined as the difference between the expected number and the actual number, i.e., $error = |E(\#c) - \#c|$. This $error$ can be used as a metric to evaluate the deep learning model. However, this single metric cannot provide enough information to evaluate the model. From the statistical perspective, as we have a large number of test data, we can generate multiple sampled subsets $\tilde{\mathcal{D}}_{test} \subset \mathcal{D}_{test}$ of the test data and compute the corresponding $error$ for each subset. The uncertainty of the model can then be obtained from these errors. For example, we can compute the mean and variance of the errors associated with these sampled subsets.

In summary, we propose the following steps to evaluate the model.

(a) Sample multiple subsets of the test data by a given ratio $p \in (0, 1)$ to get sub-test data $\{\mathcal{D}^1_{test}, \mathcal{D}^2_{test}, ..., \mathcal{D}^m_{test}\}$ from the original test data $\mathcal{D}_{test}$. For any $i \in \{1, 2, ..., m\}$, $\mathcal{D}^i_{test} \subset \mathcal{D}_{test}$, $|\mathcal{D}^i_{test}| = p|\mathcal{D}_{test}|$, and the data of $\mathcal{D}^i_{test}$ is randomly drawn from $\mathcal{D}_{test}$.
(b) For each $\mathcal{D}^i_{test}$, compute the prediction error of each class c based on $error^i_c = |E(\#c) - \#c|$.
(c) Compute the mean and variance of the error of each class

$$E_c(error^i_c) = \sum_{i=1}^{m} error^i_c/m,$$

$$Var_c(error^i_c) = \sum_{i=1}^{m} (error^i_c - E_c(error^i_c))^2/m.$$

Based on the computed information $error_c^i$, $E_c(error_c^i)$ and $Var_c(error_c^i)$, which we denote as the absolute class error (ACE), expectation of absolute class error (EACE) and variance of absolute class error (VACE), respectively, we can plot all these statistic evaluations to compare the performance of each model.

B. Task-based Networks

The threshold-based probabilistic decision-making approach described in Section II requires comparing a probability estimate against the corresponding threshold [13]. As a result, the performance of the system depends largely on the setting of the thresholds. The closer the predicted probability is to the threshold, the lower the uncertainty is required for the system to make a correct decision. For example, if the threshold is 0.6, the probability estimate of 0.5999 and 0.6001 could lead to a different decision. Expecting such accurate values of probability estimates may not be realistically possible. On the other hand, if the probability estimate is 0.1, the system can tolerate much higher uncertainty.

Our task-based network is based on the observation that many decisions in autonomous systems are often discrete in nature, e.g., stopping, keeping constant speed, or accelerating for longitudinal control. In fact, the task-based technique is already deployed in some autonomous systems [25]. Thus, it should not require explicit computation of the probabilities. Instead, the actual problem it needs to solve is whether the probability is above a given threshold. As a result, we propose a tasked-based neural network that directly outputs such decisions, instead of having to explicitly compute the probability distribution over the classes. Specifically, a task-based model is defined as a function $M : \mathbb{R}^d \rightarrow \{0, 1\}$, i.e., it is a binary model that only outputs 'Yes' (i.e., 1) or 'No' (i.e., 0) for the task.

For example, consider an autonomous driving problem, where the car needs to (a) stop if the probability that the object near the crosswalk is a pedestrian is at least p_{ped}, (b) keep constant speed if the probability that the object may move is at least p_{mov} and the the probability it is a pedestrian is less than p_{ped}, and (c) accelerate otherwise. In this case, we construct 2 task-based neural networks M_a and M_b that identify 2 distinct classes, namely a pedestrian class and a moving-object class. Specifically, model M_a is trained to determine whether the probability that the object is a pedestrian is at least p_{ped}, while model M_b is trained to determine whether the probability that the object may move is at least p_{mov}. Note that both networks only output 'Yes' or 'No', without actually computing the probabilities. We first feed an input X_i to M_a. If $M_a(x_j) = 1$, the car would stop. Otherwise, x_j will be fed into M_b. If $M_b(x_j) = 1$, the car would keep the current speed. Otherwise, the car would accelerate.

V. Experimental Results

This section provides experimental results both for the evaluation metric (Section V-A and V-B) and for the task-based classifier (Section V-C). All the code used in these experiments is publicly available [26].

A. Evaluation Metric on the MNIST Dataset

In this experiment, we train the CNN [4] and Resnet [7] models on the MNIST dataset [4]. Table I shows the BS, ECE and accuracy results.

TABLE I
Results on MNIST dataset

Model	BS	ECE	Accuracy
CNN	0.0284	0.0142	0.9811
Resnet	0.0198	0.0104	0.9873

From the results of Table I, we can see that the CNN and the Resnet models have very similar accuracy. However, from the BS and ECE metrics, the Resnet model has noticeably better performance than the CNN model on the model uncertainty. Using the evaluation metrics proposed in Section IV-A, figure 2(a) shows the errors associated with randomly sampled subsets of the test data, while figure 2(b) shows the variance of the errors. Both BS and ECE can only tell us the overall uncertainty of the model performance. However, autonomous system needs to pay more attention to some specific classes such as pedestrians. From the results of figure 2, it is now clear to see which model performs better on each class. For example, the mean and variance of the errors associated with class 5 of CNN are both lower than those of Resnet. Even though the two networks show the same accuracy on the predicted results of class 5 (CNN 882/892, Resnet 883/892), we can see that CNN actually has lower uncertainty than Resnet on the predicted results of class 5 based on the mean and variance of the errors.

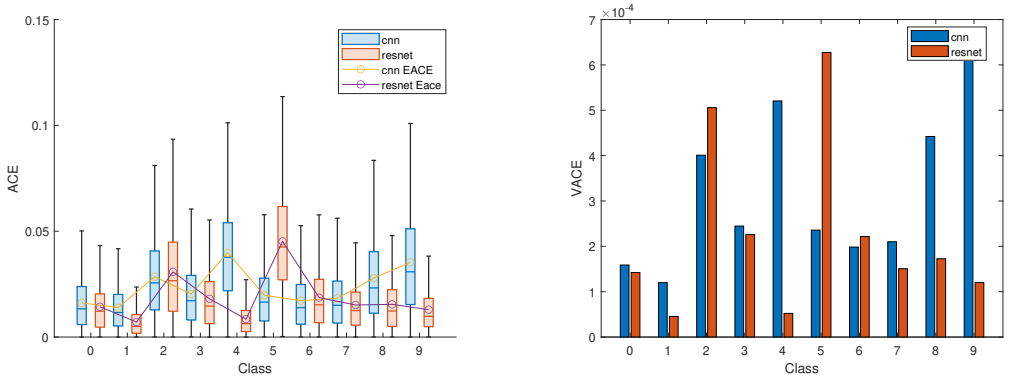
B. Evaluation Metric on the CIFAR10 Dataset

Section V-A provides an example of the case where two models have a similar performance based on accuracy. However, their uncertainties are significantly different. To further verify the performance of our metric, we perform a similar experiment on the CIFAR10 dataset [27]. The BS, ECE and accuracy results are summarized in table II.

TABLE II
Results on MNIST dataset

Model	BS	ECE	Acc
CNN	0.3733	0.0477	0.7703
Resnet	0.2049	0.0463	0.8646

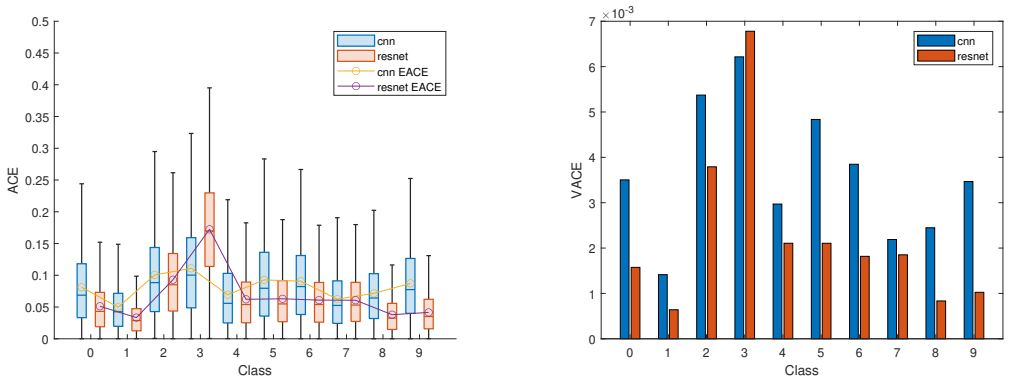
Table II shows that Resnet performs better than CNN since it achieves higher accuracy. BS is also consistent with the accuracy. This indicates that the output of Resnet has lower uncertainty than that of CNN. Using the proposed evaluation



(a) Statistics of error on each class

(b) Variance of error on each class

Fig. 2. Experimental results on MNIST dataset



(a) Statistics of error on each class

(b) Variance of error on each class

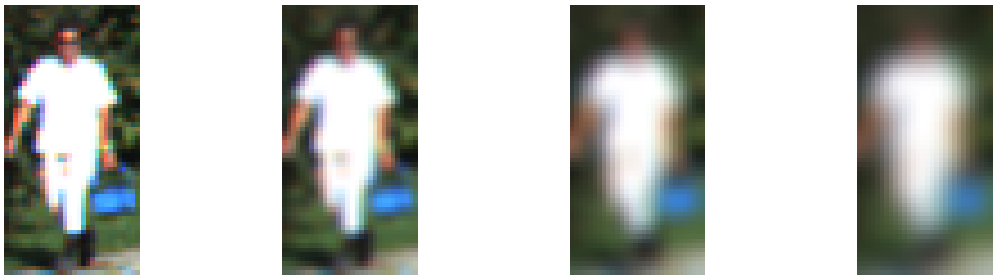
Fig. 3. Experimental results on CIFAR10 dataset

metrics on CNN and Resnet, figure 3(a) shows the error and mean of random sampling sub-datasets, figure 3(b) shows the variance of error among all sub-datasets. It can be seen that the Resnet classification results of most categories are better than CNN from figure 3(a). Figure 3(b) shows the same pattern. However, CNN has lower uncertainty than Resnet on the classification results of class 3. Even though the correct classified result of class 3 is 649/1000 for CNN and 835/1000 for Resnet from prediction, CNN has lower uncertainty about this specific class from the mean and variance. Based on the experiment results from this section, we can see even a model has lower accuracy than the other model, it is still possible the model are more confident about the output probability than the other model on some specific classes. Based on this analysis, our evaluation metric can be utilized to select a neural network model with more attention to specific classes.

C. Task-Based VS Probabilistic Classification Networks

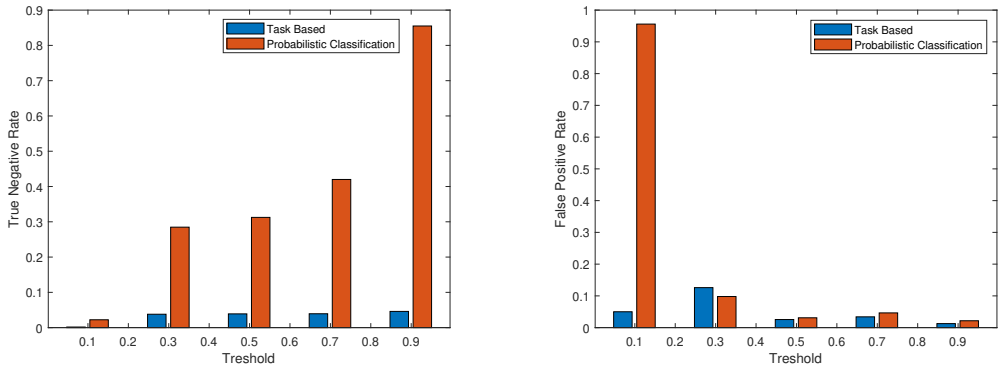
In this section, we use the KITTI dataset [28] to compare the performance of the proposed task-based network and a typical probabilistic classification network that explicitly computes the probability distribution over the object classes. The data used to train the network is the cropped images from the KITTI dataset and contains two classes, pedestrian and non-pedestrian. The label of each image corresponds to the probability that the image belongs to the pedestrian class. For the original images that are cropped from the KITTI dataset, we set the label as 1 if it belongs to the pedestrian class and 0 otherwise. In practice, the data may be labeled by different annotators, who may not always agree on the label, especially when the object is far away or the image is blurry. The probability of each class can then be calculated from the proportion of annotators who label the image with each class. To imitate this process, we use different sizes of Gaussian blur kernel to filter the image. Then, we assign the probability accordingly. Figure 4 shows an example of the different level blurred images with the probability. After having the data with its corresponding probability, we can train the network on the data.

Both the task-based network and probabilistic classification networks use the CNN structure and Adam loss function [29]. Before training the task-based network, the label of the images are assigned based on a specific threshold t. For each



(a) Original $p = 1$ (b) 3×3 kernel $p = 0.9$ (c) 7×7 kernel $p = 0.7$ (d) 11×11 kernel $p = 0.5$

Fig. 4. Images with different levels of blur



(a) True negative rates (b) False positive rates

Fig. 5. Experimental results of task-based and probabilistic classification networks on the stop decision.

image x_j, if $p_j > t$ then $y_j = 1$, otherwise $y_j = 0$. After the labels are set up, the task-based network will use 85% of the data to train the network and reserve 15%, randomly selected for the collected data, for evaluation. On the other hand, the probabilistic classification network feeds the training image x_j with its probability p_j into the network to train the model. To avoid the data augmentation, we trained both network with blurred images as well as original images.

Figure 5 show the true negative rate and false positive rate of both networks on the stop decision. From the results of figure 5, we can see that both the true-negative and false-positive rates of the task-based network do not vary much with changing thresholds. On the other hand, the results of the probabilistic classification network highly depend on the threshold. For example, with threshold $t = 0.9$, the true negative rate is very high. This observation implies that the predicted probability is not reliable. Moreover, the true negative rates are very high under all thresholds. This implies that even though we feed the network with probability and train it, the output probability can still be unreliable. This can be particularly problematic in safety-critical applications such as autonomous vehicles.

D. Summary

The results of sections V-A and V-B show that our proposed evaluation metric can capture the uncertainty of the probability estimate of each class. Such detailed uncertainty information can be utilized by the controller to improve its decision, especially for autonomous systems that are required to pay special attention to specific classes, e.g., pedestrian, cyclist, etc., to reduce the safety risks. We also test our evaluation metric on small-size test sub-dataset. The computed ACE, EACE and VACE can vary with different test sub-datasets, but it still can distinguish the uncertainty of each class of given model. In section V-C, the experimental results show that the proposed task-based neural network achieves better performance compared to an ordinary neural network.

VI. Conclusion

We propose a new evaluation metric to measure the uncertainty of probabilistic neural networks. As opposed to existing metrics, it provides the uncertainty information for each class on the classification task, which is important for applications in the autonomous system domain. Furthermore, we present a task-based neural network for threshold-based probabilistic decision-making and show the improved performance compared to a typical probabilistic classification network.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] L.-C. Wang, "An autonomous system view to apply machine learning," in *2018 IEEE International Test Conference (ITC)*. IEEE, 2018, pp. 1–10.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] 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.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[7] 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.

[8] G. Casella and R. L. Berger, *Statistical inference*. Cengage Learning, 2021.

[9] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[10] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.

[11] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[12] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 681–688.

[13] S. Jha, V. Raman, D. Sadigh, and S. A. Seshia, "Safe autonomy under perception uncertainty using chance-constrained temporal logic," *Journal of Automated Reasoning*, vol. 60, no. 1, pp. 43–62, 2018.

[14] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[15] G. W. Brier *et al.*, "Verification of forecasts expressed in terms of probability," *Monthly weather review*, vol. 78, no. 1, pp. 1–3, 1950.

[16] M. H. DeGroot and S. E. Fienberg, "The comparison and evaluation of forecasters," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 32, no. 1-2, pp. 12–22, 1983.

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

[18] I. J. Good, "Rational decisions," in *Breakthroughs in statistics*. Springer, 1992, pp. 365–377.

[19] J. Quinonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, and B. Schölkopf, "Evaluating predictive uncertainty challenge," in *Machine Learning Challenges Workshop*. Springer, 2005, pp. 1–27.

[20] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[21] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[22] K. Nguyen and B. O Connor, "Posterior calibration and exploratory analysis for natural language processing models," *arXiv preprint arXiv:1508.05154*, 2015.

[23] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[24] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *arXiv preprint arXiv:1612.01474*, 2016.

[25] A. Badithela, T. Wongpiromsarn, and R. M. Murray, "Leveraging classification metrics for quantitative system-level analysis with temporal logic specifications," *arXiv preprint arXiv:2105.07343*, 2021.

[26] [Online]. Available: https://github.com/MachineLearningProb/machine-learning

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

[28] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.