Reliable Edge Intelligence in Unreliable Environment

Minah Lee, Xueyuan She, Biswadeep Chakraborty, Saurabh Dash, Burhan Mudassar, and Saibal Mukhopadhyay
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
{minah.lee, xshe, biswadeep, saurabhdash, burhan.mudassar, smukhopadhyay6}@gatech.edu

***Abstract*—A key challenge for deployment of artificial intelligence (AI) in real-time safety-critical systems at the edge is to ensure reliable performance even in unreliable environments. This paper will present a broad perspective on how to design AI platforms to achieve this unique goal. First, we will present examples of AI architecture and algorithm that can assist in improving robustness against input perturbations. Next, we will discuss examples of how to make AI platforms robust against hardware induced noise and variation. Finally, we will discuss the concept of using lightweight networks as reliability estimators to generate early warning of potential task failures.**

***Index Terms*—Autonomous System, Edge intelligence, Machine Learning, Reliability**

I. Introduction

Artificial Intelligence (AI) platforms [1], in particular, deep neural networks (DNNs), are being increasingly deployed in real-time, safety-critical, and autonomous system at the edge such as self-driving cars, unmanned aerial vehicles, drones, robots, to name a few [2], [3]. Unlike AI at the cloud environments, the AI in real-time edge devices need to perform reliably under changing environmental conditions and robust against different types of noise, while meeting stringent energy and time constraints. Hence, a critical challenge in adopting AI in safety-critical edge applications is to ensure its reliability even in the unreliable environment.

The unreliability can originate from various sources of variations and uncertainties as illustrated in Fig. 1. During data acquisition, noise can be injected due to the inherent device variations of the sensor [4], [5] and non-ideal environment such as the adverse weather conditions [6], [7] leading to errors in DNN inference [7], [8] (Fig. 2). The energy-efficient implementation of DNNs has emerged as a key challenge for real-time perception. Processing-in-memory (PIM) has emerged as an attractive technique to achieve energy- and resource- efficiency; but at the expense of inherent hardware variations leading to errors in parameters and computations [9], [10]. DNN models also include inherent uncertainty due to a lack of full understanding of the process that generated the training data and inherent complexity/noise in the observation (model/data uncertainty) [11], [12]. These variations and uncertainties cause unreliable task behavior that, at the worst case, can lead to catastrophic system failures. Therefore, generating reliable intelligence under unreliable environment is a major challenge for adoption of AI in real-time environment.

In this paper, we present a broad perspective on how to enhance reliability of AI platforms under various sources of

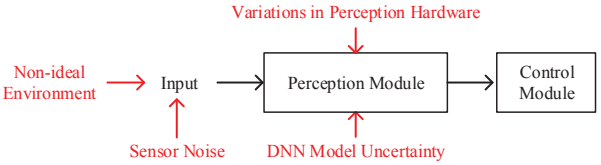

Fig. 1: Sources of variations or uncertainties in unreliable sensor-NN processing pipeline

(a) (b)

Fig. 2: Object detection failures under (a) noise and (b) rain

variation and uncertainty. Many efforts seek to design DNN robust to input perturbation by using various filters/preprocessors and augmenting with noise or perturbed data during training [7], [8]. However, preprocessing and training based approaches require exact knowledge of perturbation statistics, which is difficult to obtain in edge environment. As an alternative, we introduce a hybrid learning method that couples supervised learning using back propagation and unsupervised learning using spike-time-dependent-plasticity (STDP) to design DNNs that are naturally robust to input perturbations [13]. Several algorithmic approaches have been developed to improve robustness to hardware variation [9], [10]. As an example of such techniques, we next discuss a design approach for robust PIM where a set of important parameters is selected and protected from hardware variations [14].

Although, preceding methods help improve the reliability, they cannot eliminate all sources of failures. We argue that it is essential to be able to predict unreliable task outputs in advance to avoid a catastrophic system failure in safety-critical systems. To achieve this goal, we introduce the concept of warning generation that aims to estimate reliability of a task. Model uncertainty is a measure of task reliability due to input perturbations and out-of-distribution inputs [11]. However, quantifying uncertainty is challenging and requires unacceptable latency in real-time operations such as the case of Monte-Carlo sampling of inference outputs [12]. We present a lightweight framework to predict task reliability.

978-3-9819263-5-4/DATE21/©2021 EDAA

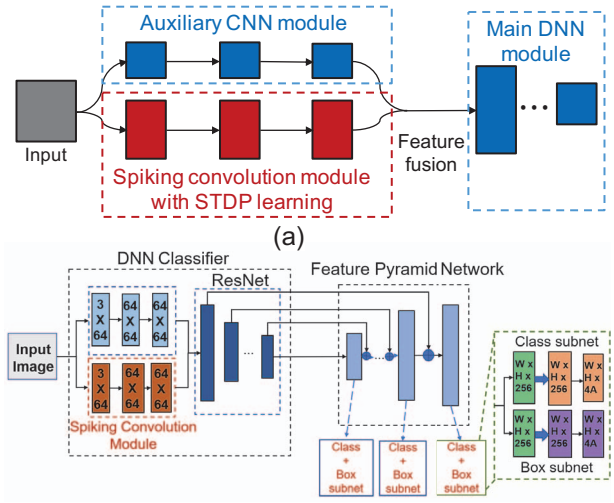

Fig. 3: SAFE-DNN: (a) a classifier and (b) an object detector

Fig. 4: Visual results (a) classification and (b) object detection.

II. Hybrid Learning for Noise-Tolerant DNN

In stochastic gradient descent (SGD) based DNN training process, gradient of the loss with respect to a network parameter is affected by all input pixels in the entire input image. This facilitates global learning and optimizes accuracy of classification, but also makes it difficult to strongly impose local constraints during training. Hence, the network *does not learn to ignore* local perturbations during low-level feature extraction as it is trained to consider global impact of each pixel for accurate classifications. This means that during inference, perturbation from pixel level noise can propagate through the network, and degrade the classification accuracy.

To improve noise resiliency of a DNN, we develop a hybrid network architecture of DNN and spiking neural network (SNN), referred to as Spike Assisted Feature Extraction based Deep Neural Network (SAFE-DNN) [13]. SNN is a type of network using biologically plausible neuron and synapse models that can exploit temporal relationship between spiking events [15]. In SNN, two neurons connected by one synapse are referred to as pre-synaptic neuron and post-synaptic neuron. Weight of the synapse determines strength of connection and learning can be achieved through STDP [16]. With STDP, SNN is able to extract the causality between spikes of two connected neurons from their temporal relationship without back propagation of global loss. As a result, weight update of each synapse in SNN depends only on input signal within the same (local) receptive field [13]. STDP also helps network learn the expected spatial correlation between pixels in a local region. During inference, if the input image contains noise, intensity of individual pixel can be contaminated but within a close spatial proximity the correlation is better preserved. As the SNN has learned to respond to local correlation, rather than individual pixels, the neuron's activity experiences less interference from local input perturbation. In other words, the SNN "learns to ignore" local perturbations and hence, the extracted features are robust to noise.

A. SAFE-DNN Image Classification

A SAFE-DNN classifier contains spiking layers placed contiguously to form the spiking convolution module, along with conventional CNN layers (Fig. 3(a)). The spiking convolution module is placed at the front to extract local and low-level features. We place several SGD- trained convolutional layers of smaller size in parallel with the spiking convolution module to also consider global features in early layers. This is called the auxiliary CNN module. The output feature map of the two parallel modules is concatenated along the depth to be used as input tensor to the remaining DNN layers, referred to as the main DNN module. The main DNN module are designed based on existing deep learning models. Training of SAFE-DNN takes two stages. First, SNN is simulated in isolation to learn all images in the training set without supervision. The learned SNN parameters are migrated to the spiking convolution module of SAFE-DNN. The entire SAFE-DNN is then trained using SGD while weights in the spiking convolution module are frozen.

We test SAFE-DNN on CIFAR-10 dataset [17] with three noise structures: Gaussian, Poisson and salt-and-pepper (S&P). We consider a SAFE-DNN variant (SAFE-MobileNetV2) trained with only clean images, a MobileNetV2 trained with clean images, and a MobileNetV2 trained with noisy images. We observe that SAFE-MobileNetV2 is more noise robust than the baseline trained with clean images and the one trained with noisy images (Table I (top)). We also test the networks with frames from rainy traffic footage. As shown in Fig. 4(a), SAFE-MobileNetV2 is able to correctly predict object class with higher confidence than MobileNetV2. Further, we use a black-box fast gradient sign method (FGSM) [18] to generate adversarial perturbations in test images for CIFAR-10 and ImageNet (traffic subset) datset. Accuracy results with different perturbation intensity ϵ are shown in Table I (bottom). It can be observed that perturbation created with black-box FGSM

TABLE I: Accuracy results for noisy CIFAR-10 (top) and adversarial perturbation (bottom)

Model	Clean	Gaussian	Wald	Poisson	S&P
MobileNetV2 [19]	91.30	66.25	48.41	37.70	32.37
n.t. MobileNetV2	90.54	86.36	71.80	74.51	63.84
SAFE-MobileNetV2	91.33	90.68	88.50	82.58	80.83

	CIFAR-10		ImageNet (Top-1/Top-3)		
Model	$\epsilon = 3$	$\epsilon = 16$	$\epsilon = 3$	$\epsilon = 8$	$\epsilon = 16$
MobileNetV2	83.50	47.93	52.3/81.8	40.5/71.8	23.4/59.7
SAFE-MobileNetV2	87.14	59.18	59.0/83.0	52.1/81.2	39.0/73.8

TABLE II: Table showing the performance of the different object detectors with input noise during testing

No Noise	{mAP,mAR}	AP{S,M,L}	AR{S,M,L}
RetinaNet	0.388,0.501	0.207,0.420,0.498	0.301,0.566,0.624
SAFE-RetinaNet	0.386,0.502	0.209,0.417,0.491	0.305,0.561,0.626

SNR = 15dB	{mAP,mAR}	AP{S,M,L}	AR{S,M,L}
RetinaNet	0.277,0.416	0.104,0.237,0.349	0.204,0.401,0.426
SAFE-RetinaNet	0.281,0.419	0.106,0.240,0.351	0.207,0.402,0.429

transfers to target network trained with the same method (SGD), while for SAFE-DNN implementations that contain features learned by STDP, the perturbation is less effective.

B. SAFE-DNN Object Detection

We test the performance of SAFE-DNN in object detection tasks. Here, Retinanet [20] with ResNet-101 [21] backbone is used as the baseline network. The SAFE-DNN object detector uses the SAFE-ResNet101 as the backbone instead of the ResNet101 as shown in Fig. 3(b). MS-COCO [22] is used for the evaluations. Besides the non-noisy condition, we further test the two object detectors with input Gaussian noise at different noise levels (SNR) and the observations are shown in Table II. We compare the two models with respect to the mean AP/AR and individual AP/AR for small, medium, and large objects. We see that for the case with no noise, RetinaNet and SAFE-RetinaNet perform very similarly. However, as we include Gaussian Noise in the testing images, we observe the SAFE-Retinanet outperforms the standard Retinanet. Fig. 4(b) demonstrates the object detection output for a use-case noisy input for both the RetinaNet and the SAFE-RetinaNet. We see that though both the object detectors detect the large objects in the frame fairly similarly, the standard RetinaNet misses the traffic signal that is successfully detected by the SAFE-RetiaNet. This highlights another important feature of the SAFE-Retinanet object detector that is to improve the AP for small object detection.

III. Dealing with Hardware Noise

Processing-in-Memory has shown great promise for high efficiency DNN inference [23], [24]. However, a major challenge in PIM based designs has been degradation in classification accuracy due to process variations in the memory bit-cells

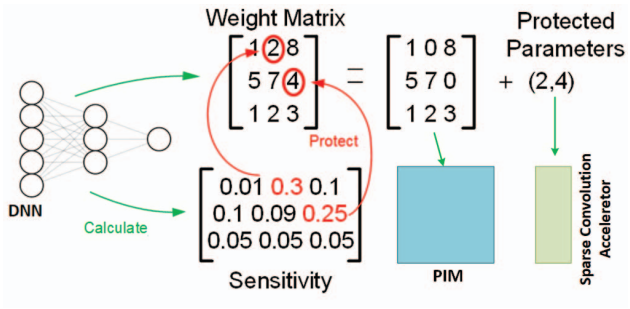

Fig. 5: Overview of proposed approach [14]

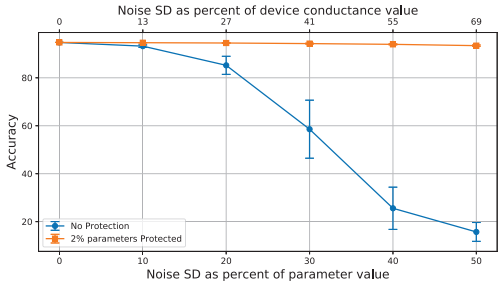

Fig. 6: Classification Accuracy for DenseNet121 for unprotected and protected designs under device variation.

[9]. Recent works have tried to approach this challenge from a purely hardware perspective, for example by using a write-verify scheme to obtain tight conductance distributions [10] in the bit-cells. An orthogonal approach has been to introduce noise during DNN training that mimics the device variation to make the model more robust during inference [9].

In contrast, we seek to leverage both algorithmic and hardware components to achieve robust inference at high efficiency. As modern DNN models are heavily over parameterized, we observe that "protecting" a small fraction of the critical parameters in a network can significantly improve robustness against process variation. Using this observation, the non-critical parameters are mapped to the PIM to perform (error-prone yet high-efficiency) mixed-signal Vector-Matrix Multiplication while (accurate yet low-efficiency) digital multiply and accumulate (MAC) operations are performed on the critical parameters as illustrated in Fig. 5.

To identify these critical parameters, we propose a Hessian based sensitivity metric ($\boldsymbol{s}$) given by $\boldsymbol{s} = (\sum_{i=1}^{n} |\lambda_i| \boldsymbol{q}_i^2) \odot \boldsymbol{w}^2$. λ_i's are the eigenvalues and $\boldsymbol{q}_i$'s are the corresponding eigenvectors of the Hessian. ($\lambda_1 \geq \lambda_2 ... \geq \lambda_n \ggg \lambda_{n+1} ... \lambda_N$) where, $n \ll N$. $\boldsymbol{w}$ are the parameters and $\odot$ denotes the Hadamard Product. The Hessian matrix contains information about the local geometry of the loss landscape and thus plays a crucial role in identifying the critical parameters, however it is intractable to compute and store for modern state-of-the-art DNNs. The computation of our sensitivity metric can be done without access to the entire Hessian and can be done in one-shot for all the network parameters as illustrated

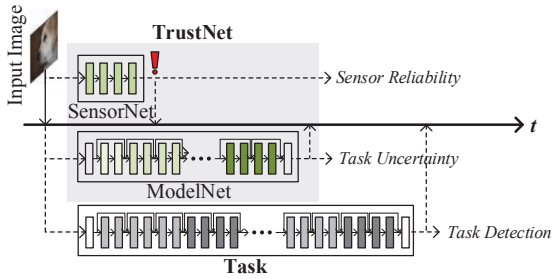

Fig. 7: Overall framework of TrustNet

in the equation above. Fig. 6 shows the performance of a protected network design compared to an unprotected one. Upon using the sensitivity metric (s) to identify and protect just top 2% DenseNet121 parameters, we observe that the mean classification accuracy drops $< 2\%$ even under high device variation, whereas the classification of the unprotected design degrades to random guessing.

IV. Warning Generation

In this section we present the concept of TrustNet, a module that can be added to a DNN to generate (early) warning of potential failures (Fig. 7). TrustNet seeks to estimate uncertainty associated with detections from a DNN and generate warning when uncertainty is high. The uncertainty of a task can be estimated using Monte Carlo (MC) sampling based techniques such as dropout [11], [12], [25]. However, MC sampling requires running a task network multiple times to estimate uncertainty which is prohibitive in real-time environment. TrustNet uses a set of neural networks to quickly estimate the uncertainty and predict potential failures of the task DNN.

An illustrative implementation of the Trustnet contains two DNN based uncertainty estimators, SensorNet and ModelNet. The SensorNet is designed to predict uncertainty due to input perturbations, for example, noise, rain, snow, to name a few, within a fraction of time required by the task DNN, and generates, early warning if input perturbation increases uncertainty beyond a threshold (Fig. 7). However, SensorNet cannot predict increased model uncertainty, for example, due to out-of-distribution input. We add a second neural network, ModelNet, to predict model uncertainty and generate warning related to associated failures, for example, due to novel (out-of-distribution) inputs (Fig. 7). The ModelNet has similar complexity as the task DNN and generates a reliability related warning around the same time when the task DNN is completed. In summary, a reliability warning from SensorNet indicates perturbed input, while the warning from ModelNet indicates novel inputs (along with including noisy inputs). Therefore, augmenting TrustNet with a task DNN increases overall reliability of the autonomous platform.

We implement a Bayesian model from a DNN based task network by using MC based Concrete dropout [25] and adopt it as a teacher model. Teacher is only used during the training, and SensorNet and ModelNet can predict the uncertainty

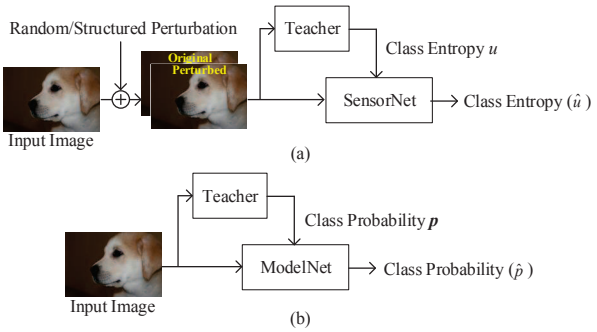

Fig. 8: Training frameworks: (a) SensorNet and (b) ModelNet

behavior without running multiple iterations of task network during the inference.

A. SensorNet

The SensorNet is designed based on WarningNet [26], a lightweight DNN platform (Fig. 8(a)). However, WarningNet predicts task accuracy based on the confidence decrease of ground truth class by input perturbation; but predictive uncertainty is a more well-known metric to quantify the task reliability. Therefore, we design SensorNet following the architecture of WarningNet (three convolutional layers followed by one linear layer) and train with task uncertainty rather than confidence.

Task uncertainty u, which is used to teach SensorNet, is obtained from the estimation of the teacher model $f_{w(m)}$ on input image $\mathbf{x}$ with M MC iterations and defined as follow:

$$u = -\sum_{c} \left\{ \frac{1}{M} \sum_{m} f^{c}_{w(m)}(\mathbf{x}) \right\} \ln \left\{ \frac{1}{M} \sum_{m} f^{c}_{w(m)}(\mathbf{x}) \right\} \quad (1)$$

where $f^{c}_{w(m)}$ denotes the c-th class output of $f_{w(m)}$. We use L_2 loss to train the SensorNet as follow where $\hat{u}$ denotes the estimated uncertainty from SensorNet:

$$\mathcal{L} = ||\hat{u} - u||_2^2 \quad (2)$$

B. ModelNet

ModelNet is trained to predict the expected class probability of a task based on the spatial characteristics of image. Similar to SensorNet, we adopt a Bayesian model using MC dropout as a teacher to teach the expected class probability (Fig. 8(b)). Uncertainty estimation can be obtained by calculating the class entropy of confidence estimations. Compared to the prior works that distill a Bayesian model to improve accuracy and calibrate uncertainty of a student network [27], [28], ModelNet is only trained to estimate the uncertainty of a given *task DNN*, rather than the uncertainty of itself, and the estimated uncertainty should be calibrated to the accuracy of the *task DNN*. The training objective of ModelNet is to only predict the expected class probability of teacher, even when teacher's prediction is wrong compared to the ground truth.

Let $f^{c}_{w(m)}(\mathbf{x})$, $h^{c}(\mathbf{x})$ denote the c-th class output from teacher model $f_{w(m)}$ and ModelNet h, respectively on input

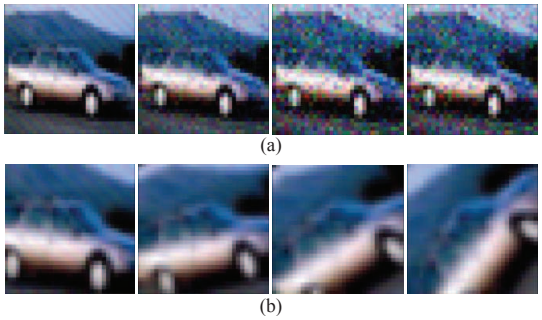

Fig. 9: Sample images of (a) Gaussian noise (σ=0, 0.05, 0.10, 0.15), and (c) image rotation (0°, 20°, 40°, 60°).

image $\mathbf{x}$. We use cross entropy $\mathcal{L}_{CE}$ and mean square error $\mathcal{L}_{MSE}$ to train the ModelNet as follows:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{MSE}}, \tag{3}$$

$$\mathcal{L}_{\text{CE}} = -\sum_{c} \left\{ \frac{1}{M} \sum_{m} f^{c}_{w(m)}(\mathbf{x}) \right\} \ln h^{c}(\mathbf{x}), \tag{4}$$

$$\mathcal{L}_{\text{MSE}} = \sum_{c} || \left\{ \frac{1}{M} \sum_{m} f^{c}_{w(m)}(\mathbf{x}) \right\} - h^{c}(\mathbf{x}) ||_2^2 \tag{5}$$

ModelNet does not need to learn the teacher's behavior under various input perturbations. This allows to detect the out-of-distribution samples including the unexpected distribution, which cannot be trained for SensorNet.

C. Verification of Uncertainty Estimation

We evaluate uncertainty estimation of SensorNet and ModelNet on *Gaussian Noise* and *image rotation*, as an example of pixel-level and structured perturbations, respectively (Fig. 9). We consider Resnet-18 [21] based image classifier as a target task network, and train a MC dropout based Bayesian model considering CIFAR10 dataset [17]. This Bayesian model is also a teacher of SensorNet and ModelNet to learn the task uncertainty. We train SensorNet with Gaussian noise only to observe the effect of known/unknown input perturbation. ModelNet does not required input perturbation during training.

As shown in Fig. 10(a), both SensorNet and ModelNet can estimate the uncertainty increase under Gaussian noise. Note, although SensorNet is trained on pixel-level noise, ModelNet has not seen such perturbations during training. The estimated uncertainty is linearly correlated to the uncertainty obtained from the teacher under low level of Gaussian noise, but teacher model becomes overconfident on highly noisy images which decreases the predictive uncertainty as illustrated in Fig. 10(c). Such uncertainty decrease is not desirable as the task accuracy continues to drop at the high noisy level. On the other hand, as SensorNet is only trained with the uncertainty of low noise, it does not learn the teacher's overconfidence on high level of noise. As input perturbations are not involved during the training of ModelNet, it also does not learn the teacher's overconfidence on high level of noise and predicts high uncertainty on highly noisy images. Besides, ModelNet can correctly estimate the uncertainty increase on

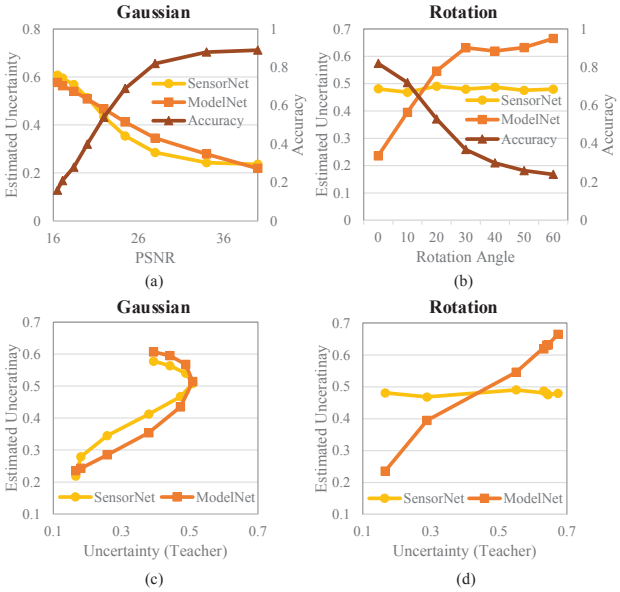

Fig. 10: Verification of TrustNet: (a-b) Comparison of accuracy of task network and estimated uncertainty from (a) SensorNet and (b) ModelNet; and (c-d) correlation of the uncertainty estimated by the DNN task and the uncertainty estimated from (c) SensorNet and (d) ModelNet.

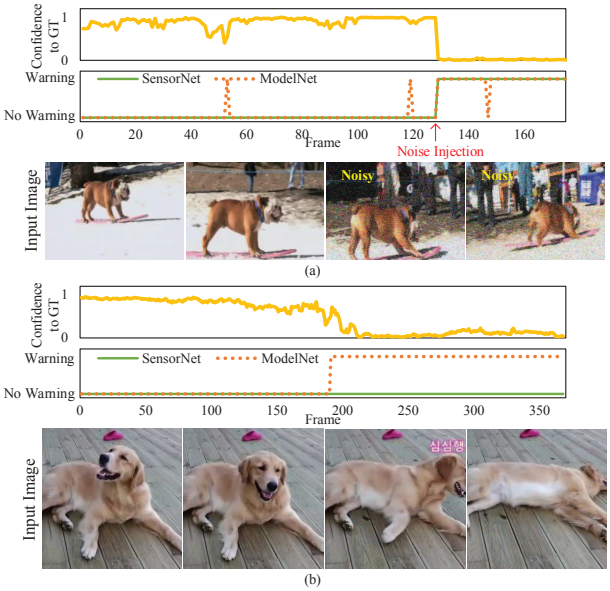

Fig. 11: Confidence of ground truth class and estimated uncertainty from SensorNet and ModelNet on example videos.

rotated images and its estimation shows linear correlation to teacher's estimation(Fig. 10(b),(d)). In contrast, SensorNet predicts similar uncertainty for image rotation with different angles, because SensorNet can only predict the uncertainty for known types of perturbation. As rotated images are not included during training, it cannot correctly predict.

TABLE III: Computational complexity analysis (Input image: 32×32; NVIDIA GeForce GTX 1080Ti)

	SensorNet	ModelNet	Task	MC Task (5 iters)
# params (MB)	0.0055	11.17	11.17	11.17
Ops (GFLOP)	0.0017	0.557	0.557	2.784
Run Time (ms)	1.30	4.83	4.83	26.36

D. Example of Real-Time Warning Generation

Based on the estimated uncertainty from SensorNet ($\hat{u}_{sensor}$) and ModelNet ($\hat{u}_{trust}$), the task is estimated to be reliable if the estimated uncertainty is lower than a threshold ($\hat{u} < u_{Th}$). Fig. 11 elaborates the performance of SensorNet and ModelNet on real world videos. Fig. 11(a) is a sample video where the Gaussian noise is added after 130-th frame. Due to the noise injection, the confidence of ground truth class *dog* of the image classifier is dramatically decreased and fails the task. Both SensorNet and ModelNet can estimate the unreliable task after noise injection and raise warning by predicting the uncertainty increase due to input noise. Fig. 11(b) is a video of a dog slowly lying down. As laid down dogs are not included in training dataset, the confidence of ground truth class *dog* is slowly decreased and task fails. SensorNet cannot predict the task failure as such out-of-distribution data is unknown, but ModelNet can detect it from the uncertainty increase. Therefore, ModelNet can predict the uncertainty increase on the out-of-distribution.

E. Complexity of Networks

Table III shows that SensorNet has $2031\times$ less number of parameters, $328\times$ less number of operations, and $3.7\times$ shorter run time compared to the task network, so early warning of sensor unreliability can be generated. ModelNet and the task network have same number of parameters and run-time. However, as the task DNN needs to run in a MC loop (say 5 iterations), ModelNet can generate uncertainty estimate in $5.5\times$ shorter run time.

V. Conclusions

We discuss a broad perspective of designs methods to improve reliability of AI models used in real-time systems in various unreliable conditions. We first present a hybrid learning methods that couples supervised and unsupervised learning to design a naturally robust DNN to input noise. It allows improved task accuracy for both image classification and object detection under natural/adversarial noise. We next discuss an example of designing a robust processing-in-memory architecture to improve tolerance to hardware induced noise. Finally, we introduce the concept of a reliability estimator, TrustNet, that predicts the unreliable task based on the uncertainty estimation for safety-critical applications. As AI models are being increasingly used in safety-critical real-time systems at the IoT edge, the need for reliable intelligence even under unreliable conditions will continue to grow. The concepts presented in this paper will motivate future works to address this critical challenge by innovations in algorithm and hardware designs to enhance a system's reliability.

Acknowledgment

This material is based on work supported in parts by the Defense Advanced Research Projects Agency (#HR0011-17-2-0045) and National Science Foundation (#1740197). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or NSF.

References

[1] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, May 2015.
[2] S. Thrun *et al.*, *Probabilistic robotics*. MIT Press, 2005.
[3] H. Sjafrie, *Introduction to Self-Driving Vehicle Technology*. CRC Press, 2019.
[4] R. D. Gow *et al.*, "A Comprehensive Tool for Modeling CMOS Image-Sensor-Noise Performance," *IEEE T-ED*, vol. 54, no. 6, pp. 1321–1329, June 2007.
[5] M. Lee *et al.*, "Effect of Process Variations in Digital Pixel Circuits on the Accuracy of DNN based Smart Sensor," in *AICAS*, March 2020.
[6] W. Wei *et al.*, "Should We Encode Rain Streaks in Video as Deterministic or Stochastic?" in *ICCV*, 2017.
[7] M. Lee *et al.*, "A Spatiotemporal Pre-processing Network for Activity Recognition under Rain," in *BMVC*, 2019.
[8] T. Na *et al.*, "Mixture of Pre-processing Experts Model for Noise Robust Deep Learning on Resource Constrained Platforms," in *IJCNN*, 2019.
[9] Y. Long *et al.*, "Design of reliable dnn accelerator with un-reliable reram," *DATE*, pp. 1769–1774, 2019.
[10] C. Ho *et al.*, "Integrated HfO_2-RRAM to achieve highly reliable, greener, faster, cost-effective, and scaled devices," in *IEDM*. IEEE, Dec. 2017.
[11] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *NIPS*, vol. 30. Curran Associates, Inc., 2017, pp. 5574–5584.
[12] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016.
[13] X. She *et al.*, "Safe-dnn: A deep neural network with spike assisted feature extraction for noise robust inference," in *IJCNN*. IEEE, 2020, pp. 1–8.
[14] S. Dash and S. Mukhopadhyay, "Hessian-driven unequal protection of dnn parameters for robust inference," in *ICCAD*, 2020.
[15] R. Moreno-Bote and J. Drugowitsch, "Causal inference and explaining away in a spiking network," *Scientific reports*, vol. 5, p. 17531, 2015.
[16] T. V. P. Bliss and A. R. Gardner-Medwin, "Long-lasting potentiation of synaptic transmission in the dentate area of the unanaesthetized rabbit following stimulation of the perforant path," *The Journal of Physiology*, 1973.
[17] A. Krizhevsky *et al.*, "Cifar-10 (canadian institute for advanced research)," *Technical report*, 2009.
[18] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[19] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.
[20] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2980–2988.
[21] K. He *et al.*, "Deep Residual Learning for Image Recognition," in *CVPR*, June 2016.
[22] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014, pp. 740–755.
[23] Y. Long *et al.*, "A ferroelectric fet based power-efficient architecture for data-intensive computing," in *ICCAD*, 2018.
[24] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA*, 2016, pp. 27–39.
[25] Y. Gal *et al.*, "Concrete Dropout," in *NIPS*, 2017.
[26] M. Lee *et al.*, "Warningnet: A deep learning platform for early warning of task failures under input perturbation for reliable autonomous platforms," in *DAC*, 2020, pp. 1–6.
[27] C. Gurau *et al.*, "Dropout distillation for efficiently estimating model confidence," 2018.
[28] A. Malinin *et al.*, "Ensemble distribution distillation," in *ICLR*, 2020.