PERSPECTIVE
https://doi.org/10.1038/s41928-020-00476-7

nature electronics

Hardware design and the competency awareness of a neural network

Yukun Ding[1], Weiwen Jiang[1], Qiuwen Lou[1], Jinglan Liu[1], Jinjun Xiong[2], Xiaobo Sharon Hu[1], Xiaowei Xu[3]✉ and Yiyu Shi[1]✉

The ability to estimate the uncertainty of predictions made by a neural network is essential when applying neural networks to tasks such as medical diagnosis and autonomous vehicles. The approach is of particular relevance when deploying the networks on devices with limited hardware resources, but existing competency-aware neural networks largely ignore any resource constraints. Here we examine the relationship between hardware platforms and the competency awareness of a neural network. We highlight the impact of two key areas of hardware development — increasing memory size of accelerator architectures and device-to-device variation in the emerging devices typically used in in-memory computing — on uncertainty estimation quality. We also consider the challenges that developments in uncertainty estimation methods impose on hardware designs. Finally, we explore the innovations required in terms of hardware, software, and hardware–software co-design in order to build future competency-aware neural networks.

Deep neural networks (DNNs) have achieved state-of-the-art performance in areas such as computer vision, machine translation, and speech recognition, and have been successfully integrated into various commercial products[1–5]. Recent advances have also highlighted their potential for solving challenging tasks including scene representation and rendering[6], navigation[7], and visual question answering[8]. These developments have been accompanied by a surge in the creation of DNN-specific hardware accelerators with a wide range of size, power and capacity[9–15]. However, a key problem in adopting DNNs in real-world mission-critical applications (and possibly many artificial intelligence systems based on other approaches) is the lack of competency awareness. Even if cutting-edge models trained with a large amount of data are used on platforms with virtually unlimited hardware resources, the complicated nature of practical problems, and the long tail of data distribution on which the models are potentially not trained or evaluated, can lead to the failure of DNNs — and this failure often happens silently[16].

This is in sharp contrast to the competency awareness of humans. When a person observes and makes predictions their actual decision is based on a most likely prediction and also an associated estimation of competency or confidence. Doctors will, for example, conduct further investigations whenever they are in doubt about a diagnosis, even when their best guess is a simple flu, and drivers will slow down when they cannot confidently recognize a traffic sign. Being overconfident or overcautious can though lead to mistakes or inferior efficiency.

In competency-aware neural networks (illustrated in Fig. 1a with an application in traffic sign detection), the competency assessment provides extra information that informs the decision-making model when the prediction is not reliable. Appropriate strategies, such as asking for human intervention or using a conservative action, can be then used to ensure safety. In order for people to trust DNNs, it is important to equip DNNs with good self-awareness of their task competency. Without such competency awareness, the completion of a target task will still need to be inspected by a human expert in order for it to be reliable in critical tasks, even if the prediction is reasonably accurate.

Though such ideas predate the prevalence of DNNs[17,18], considerable effort has recently been focused on providing an accurately quantified score representing the confidence of a neural network prediction through uncertainty estimation[19–24]. The confidence score is usually a scalar normalized to [0,1] where wrongly predicted samples are expected to be assigned with low confidence scores and correctly predicted ones are expected to be assigned with high confidence scores[25]. (Confidence is the additive inverse of uncertainty with respect to 1, so they are used interchangeably in the literature.)

The confidence score (either in its fine-grained form or coarse-grained form) is increasingly used to enable the competency-awareness of DNNs, and such DNNs offer capabilities that are crucial for their application in critical tasks (Fig. 1b), including medical diagnosis and autonomous vehicles[26–28]. Competency awareness will be an increasingly important aspect of DNNs in the next decade, especially for those deployed in commercial products where hardware costs matter and where legal and responsibility issues may arise[29–31].

Uncertainty estimation of neural networks appears to offer a route to competency-aware neural networks, if the uncertainty estimation is accurate enough. However, with the ever-increasing size of neural network models, and the pressure it places on hardware accelerators, little is known about whether hardware designs will affect the uncertainty estimation quality, and vice versa. The pursuit of competency awareness introduces a new objective in DNN-based systems design (Fig. 1c). In this Perspective, we examine recent solutions for competency-aware neural networks, and show that hardware advances do affect the uncertainty estimation quality and this needs to be taken into consideration by neural architects. We discuss the challenges involved in building competency-aware neural networks in resource-constrained hardware platforms, and explore promising approaches to address them.

[1]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. [2]IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. [3]Guangdong Cardiovascular Institute, Guangdong Provincial People's Hospital, Guangdong Academy of Medical Sciences, Guangzhou, China. ✉e-mail: xiaoweixu.ai@gmail.com; yshi4@nd.edu

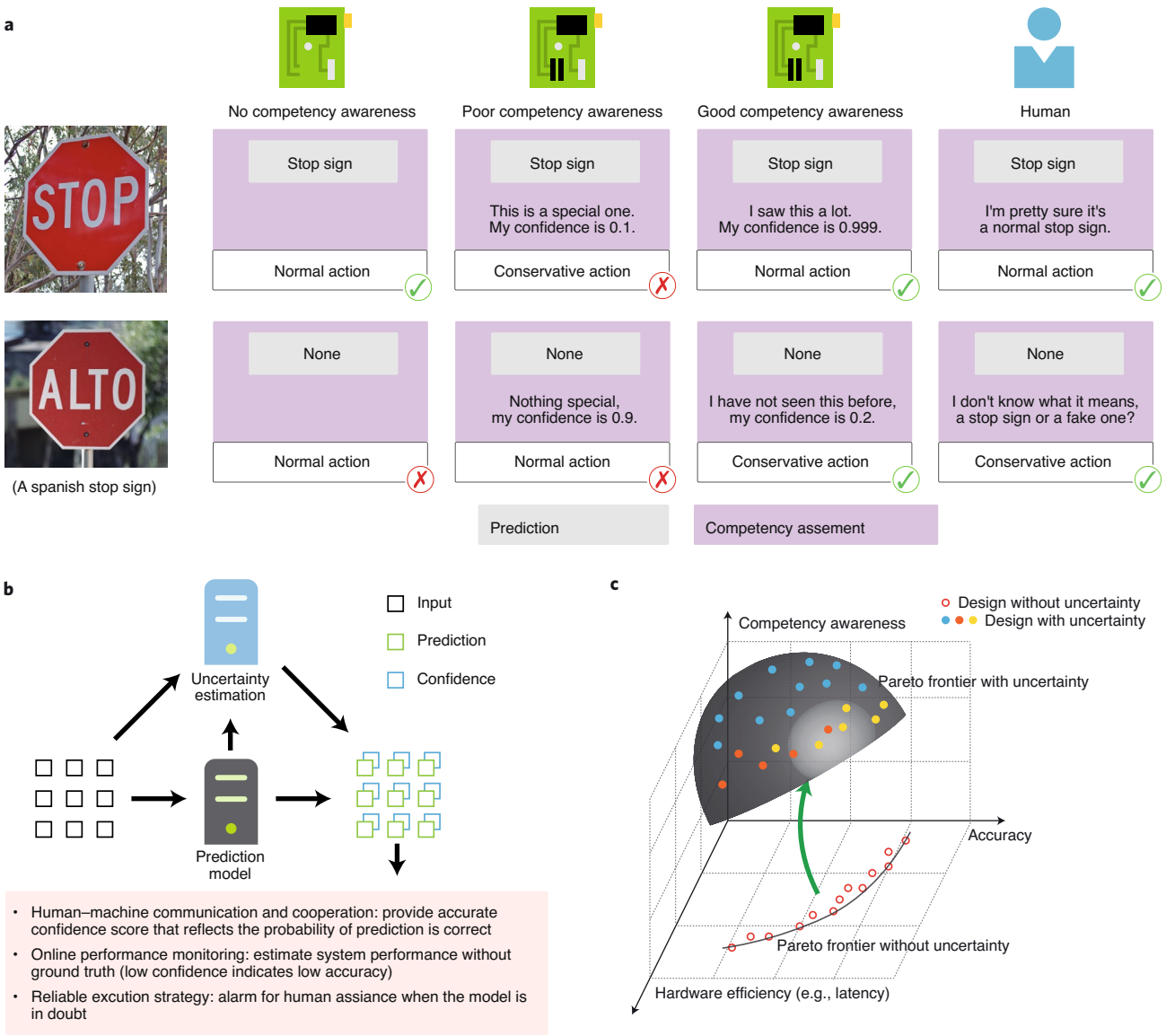

Fig. 1 | Competency awareness of neural networks. a, Illustrative example of neural network competency awareness in traffic sign detection. Competency awareness is critical in order for a machine to make human-like decisions. **b**, Benefits of models with competency awareness. **c**, New objective to be considered in system design. Competency awareness adds a new dimension and changes the pareto frontier we are looking for. Different colours represent different uncertainty estimation methods used. Data points are for illustration only.

Uncertainty estimation for competency awareness

Competency-aware neural networks are dominated by uncertainty estimation that gives a confidence score r normalized to [0, 1] as a direct and interpretable indication of the neural network's competency on given input x. Ideally, neural networks with competency awareness should know perfectly whether they can make the correct prediction on a given input, which can be defined as r=0 for wrong prediction or incompetency and r=1 for correct prediction or competency. In practice, the confidence score r is used in different use cases, which are briefly described below.

Selective prediction. In selective prediction, with a confidence score r_i for each input x_i, the model abstains from making prediction on x_i if r_i is smaller than a given threshold. In this way, competency awareness is used to avoid making decisions with low confidence. The input on which the model is uncertain is forwarded to corresponding procedures that typically lead to the involvement of human expertise or other models with higher capacity, or simply give up on the specific cases. By selectively making predictions on a subset of inputs, a given prediction model can achieve a much higher accuracy to meet the requirements[32]. The ability of identifying wrong predictions is usually measured by the area under the precision-recall curve (AUPR) and the overall selective prediction performance is measured by the area under the risk coverage curve (AURC)[25].

Confidence calibration. The aim of a confidence calibration is to give a confidence score $r \in [0, 1]$ equal to the probability of the prediction being correct and thus directly interpretable. Besides being used in the communications with human, the well-calibrated bias-free confidence score is also necessary to build a standard interface across various automatic decision-making modules[21,33,34]. The quality of the confidence score is usually measured by the expected calibration error (ECE[21]).

Other use cases. Sometimes the interest in uncertainty estimation also lies in modelling the uncertainty according to its sources. Aleatoric uncertainty captures noise inherent in the observations, while epistemic uncertainty measures the uncertainty in the model parameters[35,36]. The ability of modelling different sources of uncertainty enables more fine-grained control (for example, leveraging the aleatoric uncertainty to make the model more robust to noisy data). There is also a line of research on out-of-distribution (OOD) detection[37] that detects when a sample fed in is not drawn from the training distribution[38]. This helps the model to identify the situation on which it is not trained, such as when a cat is on the hood of the car. It is also important for the defence of malicious attacks because it is easier to manipulate the model with something it has never seen before. Distinguishing correct classification and incorrect classification and distinguishing in- and out-of-distribution samples can usually be done with the same methods[23,39] or very similar methods[20,38]. We anticipate that the reason is that, given enough model capacity, the samples in the test set are misclassified because they are not well covered by the training set, or are far from the high-density area of the training data distribution, which is similar to the out-of-distribution case.

Among all the approaches for uncertainty estimation with different theoretical justification and implementation overhead, the maximum softmax probability is the most popular way of uncertainty estimation[21,39]. It directly uses the maximum of the softmax probability produced by the softmax layer that assigns decimal probabilities to each class in a classification network. Although the networks are not trained explicitly for uncertainty estimation, maximum softmax probability can be an effective confidence score because commonly used loss functions are strictly proper scoring rules for uncertainty estimation[23]. The maximum softmax probability is shown to be a strong baseline for selective prediction but is poorly calibrated for confidence calibration. However, the calibration issue can be largely fixed by temperature scaling[21]. Temperature scaling uses a scalar parameter to scale the input to the softmax function and does not affect the model accuracy. It is a state-of-the-art confidence calibration technique that performs similarly to other alternatives[39]. (We use the maximum softmax probability with temperature scaling in our analysis below.)

Potential impact of hardware developments

There is a continued interest in using larger, more powerful and more resource-demanding networks such as BigGAN[40] and BERT[41]. As well as making neural networks more parameter efficient or floating-point operations (FLOPs) efficient[42], considerable effort has been focused on developing more powerful and more energy-efficient hardware platforms to accommodate bigger DNN models efficiently. In particular, the computation overhead mostly lies in the movement of data between function units and different memory hierarchies[43], and there are two general approaches to try to address this issue.

First, there are customized application-specific hardware accelerators that come with larger memory and higher computation density, such as Google TPU v3 (ref. [9]) and Intel Movidius (ref. [10]). Second, there are emerging hardware architectures, such as near-data processing[44] and computing-in-memory (CiM) accelerators[45,46], which use novel devices to reduce the data movement between functional units and the memory array. However, a major concern with using such emerging devices is their non-ideal behaviour, and these devices typically exhibit larger variations than conventional metal–oxide–semiconductor field-effect transistors (MOSFETs). Device-to-device variation is, in particular, dominant when using the CiM architecture for inference. Larger device-to-device variation leads to a larger overlap between two neighbouring current levels, limiting the number of bits a device can represent.

Figure 2 summarizes how the memory capacity of DNN accelerator architectures and the memory window of emerging memory devices (typically used in CiM) has progressed over the last few years. In Fig. 2d, we use a measure of memory window (I_{on}/I_{off} ratio) to capture the device-to-device variation.

It is important to understand how such advances in hardware might affect the uncertainty estimation quality of neural networks, and we thus consider the potential impact of two types of computation paradigm: traditional von Neumann architectures (which separate computation and memory) and CiM architectures. In particular, we explore the two most prominent trends: increasing memory size due to continuous technology scaling and advanced architecture design, and the quantization and device-to-device variations prominent in emerging device-based CiM accelerators.

Impact of memory size. With increasing on-chip memory size, the off-chip memory access that significantly affects the power consumption and latency can be reduced. This margin can then be used to accommodate bigger and more powerful neural networks.

To study how a change of neural network size may affect the uncertainty estimation quality, we varied the size of popular network structures DenseNet[47] and WideResNet[48] by changing their depth and width. Figure 3a shows the trend of uncertainty estimation quality based on the maximum softmax[39] with respect to the memory footprint of the model parameters. It highlights that the ability of identifying wrong prediction measured by AUPR decreases as the model size increases. This may appear counter intuitive, but the reason is that a bigger model usually comes with higher accuracy. The easy-to-identify wrong predictions become correct predictions, and then the remaining wrong predictions are more difficult to identify. Considering the overall selective prediction performance measured by AURC (Fig. 3b), we see that the performance (with AURC, the lower the better) improves with a bigger model, which suggests that the improved accuracy has a stronger effect on the selective prediction performance.

This leads to our first key observation: with increasing model memory footprint, it is increasingly difficult to identify wrong predictions, but the overall selective prediction performance still improves due to increased accuracy.

Confidence calibration has a different trend with respect to memory footprint (Fig. 3b,c). In the standard setting without calibration measure, both DenseNet and WideResNet on both datasets show a clear increase-then-decrease trend. The trend can be affected by many factors. We anticipate that the two most important factors are increasing over-confidence and improved accuracy. With increasing model size, the well-known over-confidence issue leads to higher calibration error per incorrect prediction. Meanwhile, the model gets higher accuracy and the number of incorrect predictions decreases. After temperature scaling is applied, the calibration error is effectively reduced confirming that current techniques are able to do a decent job on providing calibrated confidence scores. Interestingly, the resulting error is not proportional to the original error and does not show a consistent strong trend with memory footprint, though both DenseNet and WideResNet show a decrease-and-then-increase trend on Cifar100.

This leads to our second key observation: the commonly used temperature scaling can erase overlarge calibration errors and change the impact of memory footprint on calibration quality.

Impact of in-memory computing based on emerging devices. The emerging devices, such as memristors, that are typically used in in-memory computing require weights to be quantized to match their finite state representations, and induce device-to-device variations that affect the weight values. To study their impact on the uncertainty estimation quality, we first train models in floating-point and then simulate the inference stage assuming an emerging devices-based crossbar architecture. The maximum device current is a representative current for emerging devices such as ferroelectric

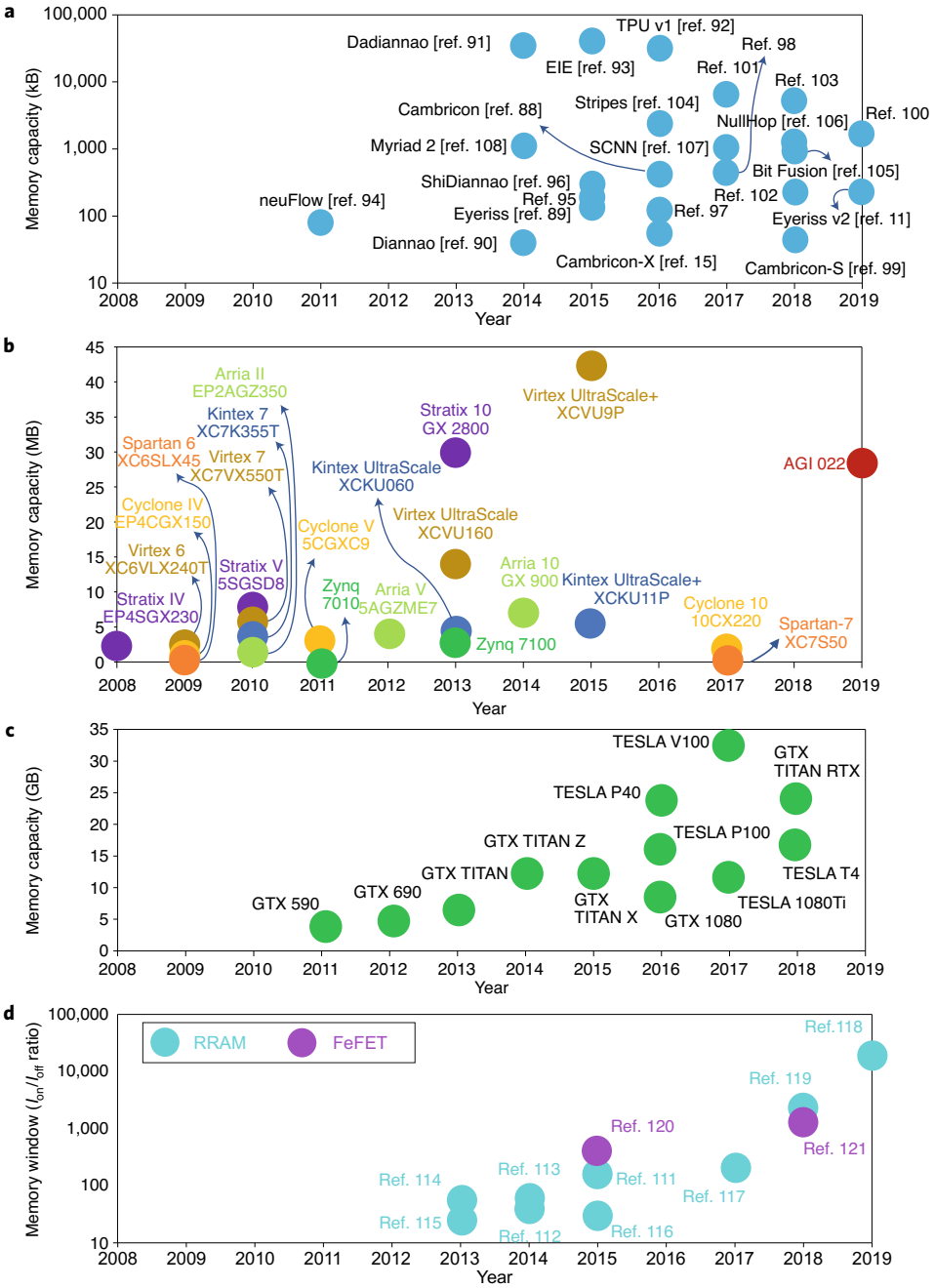

Fig. 2 | Trends in memory capacity and memory window. a–c, Progress in memory capacity of DNN accelerator architectures using application-specific integrated circuit (ASIC) memory (**a**), field-programmable gate array (FPGA) memory (**b**), and graphics processing unit (GPU) memory (**c**). Memory capacity is based on on-chip static random-access memory (SRAM) for ASIC and FPGA, and dynamic random-access memory (DRAM) for GPU. **d**, Progress in the memory window of emerging memory devices (resistive random-access memory (RRAM) and ferroelectric field-effect transistors (FeFET)). The memory window is based on the ratio of high resistance and low resistance of the device, which is a measure of the distinctness between these two extreme current levels. The larger memory window means more levels could fit in two extreme levels, and less device-to-device variation. As such, a larger memory window is desired. Data are from refs. [88–108] (**a**), ref. [109] (**b**), ref. [110] (**c**), refs. [111–121] (**d**).

field-effect transistors (FeFETs) and memristors (for example, resistive random-access memory (RRAM)). The device-to-device variation typically follows a Gaussian distribution[49–51]. We use the RRAM device model in ref. [51] as a reference while modelling the variation. In principle, the Gaussian type of device-to-device variation exists in most emerging devices (such as FeFETs, RRAMs, and spintronic devices). Therefore, our exploration is general and could be extended to other emerging devices.

The device-to-device variation induces variation in the read current for the devices, ranging from 0 nA to 10,000 nA, and we use this current variation in our simulations. One of the representative read current variations was reported as 800 nA in ref. [51]. However, device variation can typically be modulated by a write-and-verify mechanism. Therefore, device variation can be controlled to some extent by the number of write pulses allowed. Different current levels can also be achieved by applying different write pulse schemes.

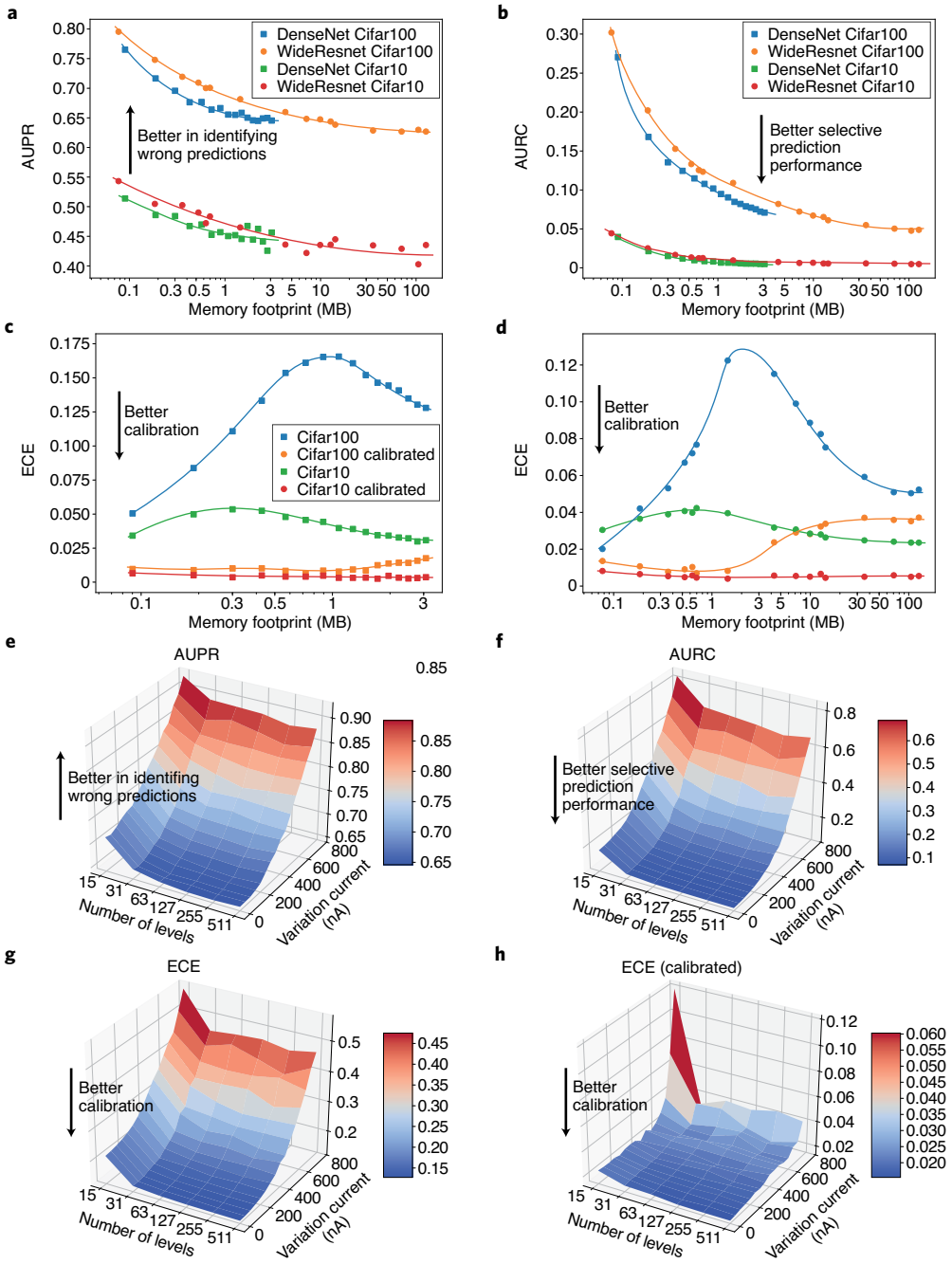

Fig. 3 | Uncertainty-related performance with respect to memory footprint, quantization and device-to-device variation. a, Area under precision-recall (AUPR) with respect to memory footprint. **b**, Area under risk-coverage (AURC) with respect to memory footprint. **c**, Expected calibration error (ECE), and ECE after calibration, with respect to memory footprint of DenseNet. **d**, ECE, and ECE calibration, with respect to memory footprint of WideResNet. Curves are hand drawn to highlight the trend. **e**–**h**, AUPR (**e**), AURC (**f**), ECE (**g**), and ECE after calibration (**h**) with respect to quantization and device-to-device variation. Only DenseNet Cifar100 results are shown here for clarity; the results for other settings, as well as full details of how this analysis was performed, can be found in the Supplementary Information.

As such, we deviate from these reported measurements and sweep the number of current levels and device read current variation in order to evaluate the impact of different device-to-device variation.

Typically, one crossbar array can only be used to compute a subset of convolution operations. After the crossbar computation, analogue to digital converters (ADCs) are used to convert the analogue signal to a digital signal for accumulation. Recent work has shown that these ADCs can be eliminated to further reduce the energy[52], and ADCs are only required after the entire convolution operation. We apply this architecture in our study. In the architecture simulation, we apply a 10-bit ADC to ensure the number of bits for activation is larger than the number of bits for weights.

Figure 3e-h depicts the quality of uncertainty estimation of the same model with different levels of quantization and current variation. Both quantization and device-to-device variation lead to increasing AUPR, but the selective prediction performance

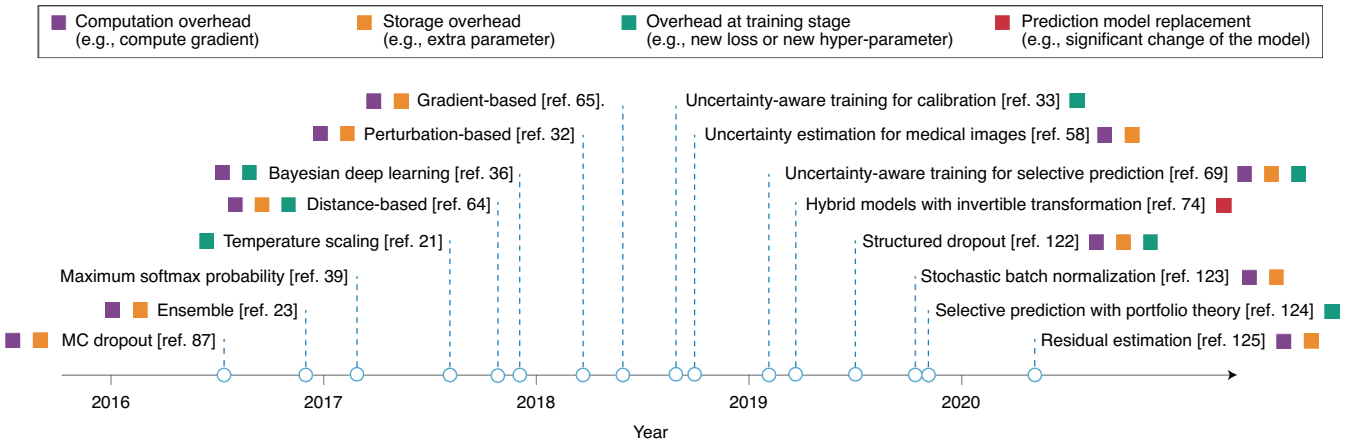

Fig. 4 | Timeline of developments in uncertainty estimation methods. Notable recent neural network uncertainty estimation methods are highlighted. Most of these incur overheads and these are indicated by the coloured squares. Data are from refs. [21,23,32,33,36,39,58,64,65,69,74,87,122–125].

becomes worse together with the accuracy. For the confidence calibration, the error grows significantly when the number of quantization levels is too small, or when the device-to-device variation is too big. The model performance is in general improved with more levels and smaller variation but there is a saturation point where the performance stops to improve further. For the calibrated model, the calibration error is reduced greatly while the impact from quantization and variation also decreases.

This leads to our third key observation: with increased number of quantization levels and reduced device-to-device variation, overall selective prediction performance improves due to the higher accuracy, but it is more difficult to identify wrong predictions as measured by AUPR. In contrast to the accuracy, when temperature scaling is applied, the confidence calibration performance is not sensitive to the quantization and device-to-device variation in a larger range of settings.

Note that these observations are not intended to be conclusive, but rather examples to show that uncertainty estimation has unique characteristics that need to be considered along with the hardware advancement.

Challenges of uncertainty estimation on hardware designs

Though the maximum softmax probability is a well-established approach for uncertainty estimation with competitive performance and negligible cost, more sophisticated methods are sought. Figure 4 highlights recent developments of uncertainty estimation methods. Popular new approaches, such as Bayesian analysis[53–57], Monte Carlo dropout[26,58] and ensemble[23,59], can achieve better estimation quality or generalization than the maximum softmax probability, but require much greater hardware resources.

A remedy for high hardware resource demand is to use relatively lightweight estimation methods. One of the most successful attempts is learned uncertainty estimation where the model, as a whole or in part, is explicitly optimized to provide accurate confidence score in addition to the original prediction task. We categorize these methods into two types based on whether extra computation is required.

The first type, which does not need extra computation in the inference stage, typically uses the maximum softmax probability and improves upon it through customized training with carefully designed uncertainty-aware optimization objectives[38,60,61]. Platt scaling and its variants can also be applied for confidence calibration with negligible overhead[21,62]. For the second type, which requires extra computation, a straightforward idea is to add a module that is specifically optimized for uncertainty estimation objectives. Such an uncertainty estimation model typically uses embedding properties of the original prediction networks so that clustering-based methods can be applied. Meanwhile, it minimizes the overhead by sharing computation with the prediction model[63,64]. Note that, in either method, training the prediction model with an uncertainty objective may lead to compromised prediction accuracy[22].

Hardware challenges. Most uncertainty estimation methods come with considerable computation or storage overhead. One of the reasons behind this is that current development in competency-aware neural networks is mostly driven by performance merit. Significant differences can also be found among the uncertainty estimation methods, especially the workload characteristic. For example, the uncertainty estimation process may require repetition of the DNN-type computation with a different set of weights[23], a *k*-nearest neighbours[64] or a backward pass to compute the gradient even in inference stage[65].

Since competency-aware neural network design is a relatively new direction, there is no consensus on which approach among all possible solutions mentioned above is the best for each scenario. As a result, it is challenging to adopt these methods for efficient implementation on hardware. On most dedicated neural network accelerators, the workload that cannot be accelerated effectively could easily become the performance bottleneck. The fact that the characteristics of some of these uncertainty estimation workloads are distinct from normal neural network workload raises the question that how practical these uncertainty estimation methods are given the current hardware platform and how we should accommodate these workloads in the future. Even if the characteristics of uncertainty estimation workload appear to be the same with the original neural network workload (for example, Monte Carlo dropout, which does multiple forward passes on the same network with dropout applied), the roofline model changes and the workload becomes parallelizable. As such, building competency-aware neural networks leads to changes in the workload on hardware and imposes new challenges that cannot be solved in the software level alone.

Multi-objective optimization. Despite all the advances in uncertainty estimation, a more fundamental problem is that uncertainty estimation is an optimization objective different from the prediction objective theoretically. As a result, given a fixed resource budget, there exists a trade-off between the prediction performance and the uncertainty estimation quality[22]. In order to not compromise the prediction performance, we expect an increasing resource demand from competency-aware neural networks especially when targeting a higher level of competency awareness.

Future directions

To address all of the challenges involved in building competency-aware neural networks, innovations in terms of hardware, software and hardware–software co-design are required.

Hardware. As well as increasing a neural network's workload, competency awareness can change the characteristics of an existing workload and boost efficiency. With the mechanism of early exits[66], for relatively simple input, if a model is confident to make a prediction in early layers, the inference can be terminated and the computation on later layers is thus skipped. Considering the fact that most inputs to neural networks are relatively simple[67], there is a high upper bound for potential computation saving. With the uncertainty estimation quality as the key factor, such an early exit method boosts the throughput with neglectable performance loss[68]. While the overall computation requirement drops significantly, the storage requirement for model parameters increases marginally due to the uncertainty estimation overhead.

The trend of decoupled prediction and uncertainty estimation objectives[33,63,64] can be formulated as a new problem: what is the best hardware resource partition for them? With limited hardware resources, a carefully designed trade-off among various factors including accuracy, competency awareness, cost, and energy efficiency is critical. In this regard, there are two key problems to be considered by hardware designers.

First, what is the difference between the workload of prediction and uncertainty estimation? The variety of different approaches for neural network uncertainty estimation makes the problem more difficult, as we see that some methods have the same type of workload with the prediction (for example, ensemble or dropout) while some others have significant differences (for example, distance-based). We found that ensemble and dropout are most computation extensive but can be effectively accelerated with parallelized process units where possible. When gradient is needed, the memory requirement would be increased greatly for saving all the intermediate results. Other more unstructured computation may better fit into a general processing unit instead of a dedicated accelerator.

Second, how do hardware platforms affect the performance of prediction and uncertainty estimation? For example, while the resource–performance relations are benchmarked frequently for prediction model[68], the correlation between uncertainty estimation and hardware resources remains unclear. As discussed above, competency-awareness of neural networks can have some interesting and maybe unexpected behaviour due to the memory constraint, quantization and device-to-device variation, which requires hardware–software co-design. In addition, and as shown in Fig. 3, the saturation point of the number of quantization levels and device-to-device variation for uncertainty estimation has not been reached by the state-of-the-art[49,51]. Therefore, we expect that, as the fabrication processes of emerging devices mature, the variation will reduce, and a higher level of uncertainty estimation quality can be achieved. When a new type of device is invented, it usually suffers from large device-to-device variations before the fabrication process becomes mature. As such they should first be used in applications where the key metric is less variation sensitive, such as calibrated ECE.

Software. The training methods of both prediction and uncertainty estimation have new considerations to gain maximum performance from a certain resource budget. The new training methodologies for both prediction model and uncertainty estimation model can be treated as uncertainty-aware training[33,36] which may replace the original training methodology for competency-aware neural networks.

Partially motivated by the compatibility with conventional development tools and training methods, most existing uncertainty estimation approaches only add a new uncertainty objective but leave the prediction objective untouched[33,38,63] during training. However, when we take uncertainty estimation into account, even the prediction objective needs new consideration. For example, training with a conventional loss function assigns equal weights to all training instances and try to minimize the average loss. When we assume the model has the capability to learn the desired function, such training methodology should work very well.

However, if there are some difficult cases in the input that the model just does not have enough capacity or data to learn, forcing the model to minimize the average loss not only cannot solve the difficult cases (at least in terms of generalization) but also harms the learning on the rest of cases. Instead of forcing the model to solve all cases, it is more reasonable to let it give up the difficult cases and focus on the majority easy or normal cases especially in a selective prediction scenario. As one of the major benefits comes with uncertainty estimation, such a problem needs to be solved for better prediction model training. This promising direction is partially explored in some recent works where customized loss functions are used[69]. In addition, as the uncertainty estimation can also be affected by hardware variations, the quantization/variation-aware training that has been used for the conventional prediction model[70] should also be used for training the uncertainty estimation model.

Another shared property of most current solutions for uncertainty estimation is that they make little change to the model architecture. From ResNet[71] and DenseNet[47] to weight-agnostic neural networks[72] and HoloGAN[73], the advances in neural network architectures suggest that the inductive biases from neural network architectures can be critical for model performance. Some initial attempts use a branch-out structure on the original prediction model[63]. However, some fundamental change of the model architecture, or even model type, may be needed to achieve a higher level of competency-awareness. There are successful attempts using invertible neural networks as a theoretically sound approach to do out-of-distribution detection[74]. Other more practical approaches include better ways to represent and memorize experience about correct predictions and wrong predictions so that the model can avoid making same mistake twice. This ability is important because it is common that operators catch the mistakes of a deployed model, but do not have an easy way to prevent the same mistakes happening again.

Hardware–software co-design. While the challenges can be partially addressed from the software or hardware perspective individually, we believe there is substantial space for hardware–software co-design of competency-aware neural networks based on three reasons.

First, although the network architecture matters, the low-level structure or hyper-parameters, such as the number of layers, number of filters, input dimension, numerical precision, usually have design flexibilities. As shown in Fig. 3, the change in model hyper-parameters leads to a smooth curve of uncertainty estimation quality. Meanwhile, certain levels of uncertainty estimation quality can be achieved with various combinations of these parameters.

Second, it has been shown that a small change in hardware design or network architecture can lead to significant efficiency change in a machine learning production workload[75,76]. As a result, instead of conducting model design and hardware design separately, an end-to-end exploration in both hardware space and software space promises better trade-off between uncertainty estimation quality and the resource requirement.

Third, for neural networks without competency awareness, there has been a transition from basic network design[77], optimized network design[71], neural architecture search (NAS)[78], hardware-aware NAS[79], to hardware–software co-exploration[80]. Such a trend validates the practical benefits of hardware–software co-design and establishes a potential path for the transition of competency-aware neural networks.

Even with great potential, sophisticated hardware–software co-design for competency-aware neural networks is not straightforward. As of today, even the hardware–software co-design of 'vanilla' neural networks takes significant expertise and labour due to the high-dimensional search space and the lack of accurate modelling especially for the performance of neural networks. Compared with existing hardware–software co-design techniques for neural networks, co-design for competency-aware neural networks means an additional uncertainty-related objective and enlarged design space associated with uncertainty estimation.

It is expected that competency awareness will be integrated into these existing hardware–software co-exploration frameworks in multiple ways to work as uncertainty-aware hardware–software co-exploration. First, the uncertainty estimation model can be searched separately and then combined with the prediction model. Second, the whole model can be searched end-to-end in a multi-objective optimization setting. In either way, proper heuristics based on the understanding of uncertainty estimation can be adopted to facilitate the search process. Yet there are still considerable challenges. First, NAS that only aims at the conventional accuracy metric already suffers from long search time and high computation cost, and the additional competency-awareness objective would make it even worse. Second, many uncertainty estimation methods break the structure of stacked regular layers in the vanilla neural network and result in unstructured computation and complex scheduling problems[25,63,74]. Third, the various choice of uncertainty estimation methods and their non-differentiable nature make it hard to fit them into the state-of-the-art differentiable NAS[81].

Building competency-aware neural networks may lead to a whole spectrum of changes to current neural networks including the training algorithm, model architecture, and eventually the workload on hardware. The uncertainty estimation methods that stand the test of time will be integrated into the design pipeline of the competency-aware neural-network-based system starting from the hardware design level. We anticipate that, to better handle uncertainty estimation, future hardware needs to accommodate heavier and more diverse workloads found in current uncertainty estimation methods, potentially including more irregular computation patterns[72], more complex non-linear activation operations other than ReLU[61] and irregular memory access[64]. Hardware–software co-design will need to be extended to optimize both conventional performance and competency awareness.

Conclusions

It is increasingly unrealistic to train models sufficiently in all possible scenarios to deploy deep neural networks in critical practical tasks; building competency-aware neural networks is an intuitive and effective solution. At the same time, competency awareness is not the only property needed to build trusted DNN-based systems. Other important components include explainability of prediction and uncertainty estimation, robustness with or without defence against adversarial attacks[82–84] and privacy preservation[85,86].

As the primary tool to enable competency awareness of neural networks, uncertainty estimation is successful at preventing silent mistakes, providing calibrated confidence scores, and even improving sample efficiency for reinforcement learning[87]. Building competency-aware neural networks may though lead to substantial changes to both the amount and the characteristics of the workload, and these challenges call for innovations in terms of hardware, software, and, in particular, hardware–software co-design.

Data availability

The data that support the findings of this study are available from the corresponding author upon request.

Code availability

The code that support the findings of this study are available from the corresponding author upon request.

Received: 17 August 2019; Accepted: 18 August 2020;
Published online: 18 September 2020

References

1. Zhu, J.-Y., Park, T., Isola, P. & Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE Int. Conference on Computer Vision (ICCV)* 2223–2232 (IEEE, 2017).
2. Oord, A. V. D. et al. Wavenet: A generative model for raw audio. Preprint at https://arxiv.org/abs/1609.03499 (2016).
3. Wu, Y. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. Preprint at https://arxiv.org/abs/1609.08144 (2016).
4. Johnson, J., Alahi, A. & Li, F.-F. Perceptual losses for real-time style transfer and super-resolution. In *Proc. European Conference on Computer Vision* 694–711 (Springer, 2016).
5. He, Y. et al. Streaming end-to-end speech recognition for mobile devices. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 6381–6385 (IEEE, 2019).
6. Ali Eslami, S. M. et al. Neural scene representation and rendering. *Science* **360**, 1204–1210 (2018).
7. Anderson, P. et al. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) 3674–3683 (IEEE, 2018).
8. Yi, K. et al. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)* 1031–1042 (MIT Press, 2018).
9. *Cloud TPU* (Google, 2020); https://cloud.google.com/tpu
10. *Intel Movidius Myriad X Vision Processing Unit Technical Specifications* (Intel, 2020); https://www.intel.com/content/www/us/en/products/processors/movidius-vpu/movidius-myriad-x.html.
11. Chen, Y.-H., Yang, T.-J., Emer, J. & Sze, V. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE J. Em. Sel. Top. Circuits Syst.* **9**, 292–308 (2019).
12. Guo, K. et al. Angel-eye: A complete design flow for mapping CNN onto embedded fpga. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **37**, 35–47 (2017).
13. Valavi, H., Ramadge, P. J., Nestler, E. & Verma, N. A 64-tile 2.4-mb in-memory-computing CNN accelerator employing charge-domain compute. *IEEE J. Solid-State Circuits* **54**, 1789–1799 (2019).
14. Xu et al. Scaling for edge inference of deep neural networks. *Nat. Electron.* **1**, 216–222 (2018).
15. Zhang, S. et al. Cambricon-x: An accelerator for sparse neural networks. In *49th Annual IEEE/ACM Int. Symposium on Microarchitecture* https://doi.org/10.1109/MICRO.2016.7783723 (IEEE, 2016).
16. Holzinger, A., Biemann, C., Pattichis, C. S. & Kell, D. B. What do we need to build explainable ai systems for the medical domain? Preprint at https://arxiv.org/abs/1712.09923 (2017).
17. Vovk, V., Gammerman, A. & Shafer, G. *Algorithmic Learning in a Random World* (Springer, 2005).
18. Papadopoulos, H., Vovk, V. & Gammermam, A. Conformal prediction with neural networks. In *19th IEEE Int. Conference on Tools with Artificial Intelligence (ICTAI 2007)* **2**, 388–395 (IEEE, 2007).
19. DeVries, T. & Taylor, G.W. Leveraging uncertainty estimates for predicting segmentation quality. Preprint at https://arxiv.org/abs/1807.00502 (2018).
20. DeVries, T. & Taylor, G.W. Learning confidence for out-of-distribution detection in neural networks. Preprint at https://arxiv.org/abs/1802.04865 (2018).
21. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K.Q. On calibration of modern neural networks. In *Proc. 34th International Conference on Machine Learning* **70**, 1321–1330 (ACM, 2017).
22. Malinin, A. & Gales, M. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)* 7047–7058 (MIT Press, 2018).
23. Lakshminarayanan, B., Pritzel, A. & Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)* 6402–6413 (MIT Press, 2017).
24. Geifman, Y., Uziel, G. & El-Yaniv, R. Bias-reduced uncertainty estimation for deep neural classifiers. In *Proc. 7th Int. Conference on Learning Representations (ICLR)* (ICLR, 2019).
25. Ding, Y., Liu, J., Xiong, J. & Shi, Y. Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off. In *CVPR workshop on Fair, Data Efficient and Trusted Computer Vision* 4–5 (IEEE, 2020).
26. Roy, A. G., Conjeti, S., Navab, N. & Wachinger, C. Inherent brain segmentation quality control from fully convnet Monte Carlo sampling. In *Int. Conference on Medical Image Computing and Computer-Assisted Intervention* 664–672 (Springer, 2018).

27. Su, H., Yin, Z., Huh, S., Kanade, T. & Zhu, J. Interactive cell segmentation based on active and semi-supervised learning. *IEEE Trans. Med. Imaging* **35**, 762–777 (2015).
28. McAllister, R. et al. Concrete problems for autonomous vehicle safety: advantages of Bayesian deep learning. In *Int. Joint Conferences on Artificial Intelligence*, 4745–4753 (IJCAI, 2017).
29. Gasser, U. & Almeida, V. A. A layered model for ai governance. *IEEE Internet Comput.* **21**, 58–62 (2017).
30. O'sullivan, S. et al. Legal, regulatory, and ethical frameworks for development of standards in artificial intelligence (AI) and autonomous robotic surgery. *Int. J. Med. Robot. Comput. Assist. Surg.* **15**, e1968 (2019).
31. Shih, P.-J. Ethical guidelines for artificial intelligence (AI) development and the new "trust" between humans and machines. *Int. J. Autom. Smart Technol.* **9**, 41–43 (2019).
32. Liang, S., Li, Y. & Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proc. 6th Int. Conference on Learning Representations (ICLR)* (ICLR, 2018).
33. Kumar, A., Sarawagi, S. & Jain, U. Trainable calibration measures for neural networks from kernel mean embeddings. In *Int. Conference on Machine Learning* 2810–2819 (MLR, 2018).
34. Naeini, M. P., Cooper, G. & Hauskrecht, M. Obtaining well calibrated probabilities using Bayesian binning. In *29th AAAI Conference on Artificial Intelligence* 2901–2907 (AAAI, 2015).
35. Kiureghian, A. D. & Ditlevsen, O. Aleatory or epistemic? Does it matter? *Struct, Saf.* **31**, 105–112 (2009).
36. Kendall, A. & Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30 (NIPS 2017)* 5574–5584 (MIT Press, 2017).
37. Shalev, G., Adi, Y. & Keshet, J. Out-of-distribution detection using multiple semantic label representations. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)* 7375–7385 (MIT Press, 2018).
38. Lee, K., Lee, K., Lee, K. and Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *Proc. 6th Int. Conference on Learning Representations (ICLR)* (ICLR, 2018).
39. Hendrycks, D. & Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proc. 5th Int. Conference on Learning Representations (ICLR)* (ICLR, 2017).
40. Brock, A., Donahue, J. & Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *Proc. 7th International Conference on Learning Representations (ICLR)* (ICLR, 2019).
41. Devlin, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* 4171–4186 (MIT Press, 2019).
42. Sandler, M. et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520 (IEEE, 2018).
43. Chen, Y.-H., Emer, J. Sze, V. Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Proc. 43rd International Symposium on Computer Architecture*, 367–379 (IEEE, 2016).
44. Gao, M., Ayers, G. & Kozyrakis, C. Practical near-data processing for in-memory analytics frameworks. In *2015 International Conference on Parallel Architecture and Compilation (PACT)* 113–124 (IEEE, 2015).
45. Xue, C.-X. et al. 24.1 a 1Mb multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for CNN based AI edge processors. In *2019 IEEE International Solid-State Circuits Conference (ISSCC)* 388–390 (IEEE, 2019).
46. Jiang, W., Xie, B. & Liu, C. et al. Integrating memristors and CMOS for better AI. *Nat. Electron.* **2**, 376–377 (2019).
47. Huang, G., Liu, Z., Maaten, L.V.D. & Weinberger, K.Q. Densely connected convolutional networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 4700–4708 (IEEE, 2017).
48. Zagoruyko, S. & Komodakis, N. Wide residual networks. *In Proc. British Machine Vision Conference (BMVC)* 87.1–87.12 (BMVA, 2016).
49. Ni, K. et al. Fundamental understanding and control of device-to-device variation in deeply scaled ferroelectric FETs. In *Proc. 2019 Symposium on VLSI Technology* 40–41 (IEEE, 2019).
50. Jerry, M. et al. Ferroelectric FET analog synapse for acceleration of deep neural network training. In *Proc. 2017 IEEE International Electron Devices Meeting (IEDM)* 6.2.1–6.2.4 (IEEE, 2017).
51. Zhao, M. et al. Investigation of statistical retention of filamentary analog RRAM for neuromophic computing. In *Proc. 2017 IEEE International Electron Devices Meeting (IEDM)* 39.4.1–39.4.4 (IEEE, 2017).
52. Chou, T., Tang, W., Botimer, J. & Zhang, Z. CASCADE: Connecting RRAMs to Extend Analog Dataflow In An End-To-End In-Memory Processing Paradigm. In *Proc. 52nd Annual IEEE/ACM International Symposium on Microarchitecture* 114–125 (IEEE, 2019).
53. MacKay, D. J. A practical Bayesian framework for backpropagation networks. *Neural Comput.* **4**, 448–472 (1992).
54. Neal, R. M. *Bayesian Learning for Neural Networks* 118 (Springer, 2012).
55. Blundell, C., Cornebise, J., Kavukcuoglu, K. & Wierstra, D. Weight uncertainty in neural network. In *Proc. International Conference on Machine Learning* 1613–1622 (ACM, 2015).
56. Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems 24 (NIPS 2011)* 2348–2356 (MIT Press, 2011).
57. Louizos C. & Welling, M. Multiplicative normalizing flows for variational Bayesian neural networks. In *Proc. 34th International Conference on Machine Learning* https://doi.org/10.5555/3305890.3305910 (ACM, 2017).
58. Nair, T., Precup, D., Arnold, D. L. & Arbel, T. Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. In *Proc. Int. Conference on Medical Image Computing and Computer-Assisted Intervention* 655–663 (Springer, 2018).
59. Ovadia, Y. et al. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *Advances in Neural Information Processing Systems 32 (NIPS 2019)* 13991–14002 (MIT Press, 2019).
60. Dhamija, A. R., Günther, M. & Boult, T. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)* 9175–9186 (MIT Press, 2018).
61. Hein, M., Andriushchenko, M. & Bitterwolf, J. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)* 41–50 (IEEE, 2018).
62. Alexandari, A., Kundaje, A. & Shrikumar, A. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. Preprint at https://arxiv.org/abs/1901.06852 (2019).
63. Chen, T., Navrátil, J., Iyengar, V. & Shanmugam, K. Confidence scoring using whitebox meta-models with linear classifier probes. In *Proc. 22nd International Conference on Artificial Intelligence and Statistics* 1467–1475 (PMLR, 2019).
64. Mandelbaum, A. & Weinshall, D. Distance-based confidence score for neural network classifiers. Preprint at https://arxiv.org/abs/1709.09844 (2017).
65. Oberdiek, P., Rottmann, M. & Gottschalk, H. Classification uncertainty of deep neural networks based on gradient information. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition* 113–125 (Springer, 2018).
66. Teerapittayanon, S., McDanel, B. & Kung, H.-T. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd Int. Conference on Pattern Recognition (ICPR)* 2464–2469 (Springer, 2016).
67. Wang, X. et al. Idk cascades: fast deep learning by learning not to overthink. In *Proc. Conference on Uncertainty in Artificial Intelligence* 580–590 (UAIA, 2018).
68. Sze, V., Chen, Y.-H., Yang, T.-J. & Emer, J. S. Efficient processing of deep neural networks: a tutorial and survey. *Proc. IEEE* **105**, 2295–2329 (2017).
69. Geifman, Y. & El-Yaniv, R. Selectivenet: a deep neural network with an integrated reject option. In *Proc. Int. Conference on Machine Learning* 2151–2159 (MLR, 2019).
70. Song, C., Liu, B., Wen, W., Li, H. & Chen, Y. A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system. In *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)* https://doi.org/10.1109/NVMSA.2017.8064465 (IEEE, 2017).
71. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE conference on Computer Vision and Pattern Recognition* 770–778 (IEEE, 2016).
72. Gaier, A. & Ha, D. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems 32 (NIPS 2019)* 5364–5378 (MIT Press, 2019).
73. Nguyen-Phuoc, T., Li, C., Theis, L. Richardt, C. & Yang, Y.-L. Hologan: Unsupervised learning of 3D representations from natural images. In *Proc. IEEE Int. Conference on Computer Vision* 7588–7597 (IEEE, 2019).
74. Nalisnick, E. et al. Hybrid models with deep and invertible features. In *Proc. Int. Conference on Machine Learning* 4723–4732 (MLR, 2019).
75. Wu, C.-J. et al. Machine learning at Facebook: understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* 331–344 (IEEE, 2019).
76. Gupta, U. et al. The architectural implications of Facebook's DNN-based personalized recommendation. In *2020 IEEE Int. Symposium on High Performance Computer Architecture (HPCA)* 488–501 (IEEE, 2020).
77. Krizhevsky, A., Sutskever, I. & Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)* 1097–1105 (MIT Press, 2012).
78. Zoph, B. & Le, Q.V. Neural architecture search with reinforcement learning. In *Proc. 5th Int. Conference on Learning Representations (ICLR)* (ICLR, 2017).
79. Tan, M. et al. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 2820–2828 (IEEE, 2019).
80. Yang, L. et al. Co-exploring neural architecture and network-on-chip design for real-time artificial intelligence. In *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)* 85–90 (2020).

81. Liu, H., Simonyan, K. & Yang, Y. DARTS: Differentiable architecture search. In *Proc. 7th Int. Conference on Learning Representations (ICLR)* (ICLR, 2019).
82. Hendrycks, D. & Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. 7th Int. Conference on Learning Representations (ICLR)* (ICLR, 2019).
83. Huang, X., Kwiatkowska, M., Wang, S. & Wu, M. Safety verification of deep neural networks. In *Proc. Int. Conference on Computer Aided Verification* 3–29 (Springer, 2017).
84. Papernot, N. et al. Practical black-box attacks against machine learning. In *Proc. 2017 ACM Asia Conference on Computer and Communications Security* 506–519 (ACM, 2017).
85. Abadi, M. et al. Deep learning with differential privacy. In *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security* 308–318 (ACM, 2016).
86. Papernot, N. et al. Practical black-box attacks against machine learning. In *Proc. 2017 ACM Asia Conference on Computer and Communications security* 506–519 (ACM, 2017).
87. Gal, Y. & Ghahramani, Z. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In *Proc Int. Conference on Machine Learning* 1050–1059 (MLR, 2016).
88. Liu, S. et al. Cambricon: An instruction set architecture for neural networks. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* 393–405 (IEEE, 2016).
89. Chen, Y.-H., Krishna, T., Emer, J. S. & Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-State Circ.* **52**, 127–138 (2016).
90. Chen, T. et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Comput. Archit. News* **42**, 269–284 (ACM, 2014).
91. Chen, Y. et al. Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM Int. Symposium on Microarchitecture* 609–622 (IEEE, 2014).
92. Jouppi, N. P. et al. In-datacenter performance analysis of a tensor processing unit. In *Proc. 44th Annual Int. Symposium on Computer Architecture* https://doi.org/10.1145/3140659.3080246 (ACM, 2017).
93. Han, S. et al. EIE: efficient inference engine on compressed deep neural network. *ACM SIGARCH Comput. Archit. News* **44**, 243–254 (2016).
94. Farabet, C. et al. Neuflow: A runtime reconfigurable dataflow processor for vision. In *CVPR2011 Workshops* 109–116 (IEEE, 2011).
95. Yoo, H.-J. et al. A 1.93 tops/w scalable deep learning/inference processor with tetra-parallel MIMD architecture for big data applications. In *IEEE Int. Solid-State Circuits Conference* 80–81 (IEEE, 2015).
96. Du, Z. et al. ShiDianNao: Shifting vision processing closer to the sensor. In *Proc. 42nd Annual International Symposium on Computer Architecture* 92–104 (IEEE, 2015).
97. Moons, B. & Verhelst, M. A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets. In *Proc. 2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)* https://doi.org/10.1109/VLSIC.2016.7573525 (IEEE, 2016).
98. Whatmough, P. N. et al. 14.3 A 28nm SoC with a 1.2 GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications. In *Proc. 2017 IEEE Int. Solid-State Circuits Conference (ISSCC)* 242–243 (IEEE, 2017).
99. Zhou, X. et al. Cambricon-s: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach. In *Proc. 2018 51st Annual IEEE/ACM Int. Symposium on Microarchitecture (MICRO)* 15–28 (IEEE, 2018).
100. Song, J. et al. 7.1 An 11.5 TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC. In *2019 IEEE Int. Solid-State Circuits Conference-(ISSCC)* 130–132 (IEEE, 2019).
101. Desoli, G. et al. 14.1 a 2.9 TOPS/W deep convolutional neural network SOC in FD-SOI 28nm for intelligent embedded systems. In *2017 IEEE Int. Solid-State Circuits Conference (ISSCC)* 238–239 (IEEE, 2017).
102. Lee, J. et al. UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In *2018 IEEE Int. Solid-State Circuits Conference-(ISSCC)* 218-220 (IEEE, 2018).
103. Park, E., Kim, D. & Yoo, S. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *2018 ACM/IEEE 45th Annual Int. Symposium on Computer Architecture (ISCA)* 688–698 (IEEE, 2018).
104. Judd, P., Albericio, J., Hetherington, T., Aamodt, T. M. & Moshovos, A. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM Int. Symposium on Microarchitecture (MICRO)* https://doi.org/10.1109/MICRO.2016.7783722. (IEEE, 2016).
105. Sharma, H. et al. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *2018 ACM/IEEE 45th Annual Int. Symposium on Computer Architecture (ISCA)* 764–775 (IEEE, 2018).
106. Aimar, A. et al. Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps. *IEEE Trans. Neural Netw. Learn. Syst.* **30**, 644–656 (2018).
107. Parashar, A. et al. Scnn: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH Comput. Archit. News* **45**, 27–40 (2017).
108. Moloney, D. et al. Myriad 2: Eye of the computational vision storm. In *2014 IEEE Hot Chips 26 Symposium (HCS)* https://doi.org/10.1109/HOTCHIPS.2014.7478823 (IEEE, 2014).
109. *Intel Agilex FPGAs and SOCs* (Intel, 2020); https://www.intel.com/content/www/us/en/products/programmable/fpga/agilex.html
110. *List of Graphics Processing Units* (Wikipedia, 2020); https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units
111. Kumbhare, P. et al. A selectorless RRAM with record memory window and nonlinearity based on trap filled limit mechanism. In *2015 15th Non-Volatile Memory Technology Symposium (NVMTS)* https://doi.org/10.1109/NVMTS.2015.7457491 (IEEE, 2015).
112. Larcher, L. et al. A compact model of program window in HfO x RRAM devices for conductive filament characteristics analysis. *IEEE Trans. Electron Dev.* **61**, 2668–2673 (2014).
113. Lee, S. et al. Engineering oxygen vacancy of tunnel barrier and switching layer for both selectivity and reliability of selector-less ReRAM. *IEEE Electron Dev. Lett.* **35**, 1022–1024 (2014).
114. Lee, S. et al. Selector-less ReRAM with an excellent non-linearity and reliability by the band-gap engineered multi-layer titanium oxide and triangular shaped AC pulse. In *2013 IEEE Int. Electron Devices Meeting* 10.6.1-10.6.4 (IEEE, 2013).
115. Woo, J. et al. Selector-less RRAM with non-linearity of device for cross-point array applications. *Microelectron. Eng.* **109**, 360–363 (2013).
116. Lee, S. et al. Effect of AC pulse overshoot on nonlinearity and reliability of selectorless resistive random access memory in AC pulse operation. *Solid-State Electron.* **104**, 70–74 (2015).
117. Dongale, T. D. et al. Effect of write voltage and frequency on the reliability aspects of memristor-based RRAM. *Int. Nano Lett.* **7**, 209–216 (2017).
118. Gismatulin, A., Volodin, V., Gritsenko, V. & Chin, A. All nonmetal resistive random access memory. *Sci. Rep.* **9**, 6144 (2019).
119. Grossi, A. et al. Experimental investigation of 4-kb RRAM arrays programming conditions suitable for TCAM. *IEEE Trans. VLSI Syst.* **26**, 2599–2607 (2018).
120. Mulaosmanovic, H. et al. Evidence of single domain switching in hafnium oxide based FeFETs: Enabler for multi-level FeFET memory cells. In *Proc. 2015 IEEE Int. Electron Devices Meeting (IEDM)* 26.8.1–26.8.3 (IEEE, 2015).
121. Ni, K., Li, X, Smith, J. A., Jerry, M. & Datta, S. Write disturb in ferroelectric FETs and its implication for 1T-FeFET AND memory arrays. *IEEE Electron Device Lett.* **39**, 1656–1659 (2018)
122. Zhang, Z., Dalca, A. V. & Sabuncu, M. R. Confidence calibration for convolutional neural networks using structured dropout. Preprint at https://arxiv.org/abs/1906.09551 (2019).
123. Atanov, A., Ashukha, A., Molchanov, D., Neklyudov, K. & Vetrov, D. Uncertainty estimation via stochastic batch normalization. In *Proc. Int. Symposium on Neural Networks* 261–269 (Springer, 2019).
124. Liu, Z. et al. Deep gamblers: learning to abstain with portfolio theory. In *Advances in Neural Information Processing Systems 32 (NIPS 2019)* 10622–10632 (MIT Press, 2019).
125. Qiu, X., Meyerson, E. & Miikkulainen, R. Quantifying point-prediction uncertainty in neural networks via residual estimation with an I/O kernel. In *Proc. 8th Int. Conference on Learning Representations (ICLR)* (ICLR, 2020).

Author contributions

Y.D. contributed to all aspects of the project. X.X., and W.J. contributed to data collection and discussion. J.L., Q.L., J.X., and X.H. contributed to discussion and writing. Y.S. contributed to project planning, development, discussion, and writing.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41928-020-00476-7.

Correspondence should be addressed to X.X. or Y.S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.