Assessing Reliability of Deep Learning Through Robustness Evaluation and Operational Testing

Xingyu Zhao[1,3], Wei Huang[1], Alec Banks[2], Victoria Cox[2], David Flynn[3], Sven Schewe[1], and Xiaowei Huang[1](✉)

[1] University of Liverpool, Liverpool, L69 3BX, U.K.
{xingyu.zhao,w.huang23,xiaowei.huang,sven.schewe}@liverpool.ac.uk
[2] Defence Science and Technology Laboratory, Salisbury, SP4 0JQ, U.K.
{abanks,vcox}@dstl.gov.uk
[3] Heriot-Watt University, Edinburgh, EH14 4AS, U.K.
d.flynn@hw.ac.uk

Abstract. Utilisation of Deep Learning (DL) is advancing into increasingly more sophisticated applications. While it shows great potential to provide transformational capabilities, DL also raises new challenges regarding its dependability in critical functions. In this paper, we present a bespoke reliability assessment model for DL, based on evidence from both, robustness evaluation and operational testing. We first partition the input space into small cells and then "assemble" their robustness according to the operational profile (OP), where both, the estimates on the cell robustness and the OP, are continuously updated. We develop a prototype tool of our model that is publicly available and demonstrate it with case studies. Compared to accuracy testing and average cell robustness, our method is more comprehensive by considering both, the robustness and the OP, while being conservative, flexible, and evolvable.

Keywords: Safe AI · Bayesian inference · uncertain operational profiles

1 Introduction

Deep Learning (DL) is being explored to provide transformational capabilities to industrial sectors including automotive, healthcare, aviation and finance. The reality that DL is not actually as dependable as envisioned now becomes a major impediment. E.g., key industrial foresight reviews identified that the biggest obstacle to gaining benefits of DL-powered robots is the assurance and regulation on their safety and reliability [15]. Thus, there is an urgent need to develop methods to enable the dependable use of DL in critical applications and, more importantly, to *assess* and *demonstrate* the dependability in certification/regulation.

For traditional systems, safety and reliability analysis is guided by well-established standards, and supported by mature development processes and verification and validation (V&V) techniques/tools. The situation is different for systems that utilise DL: they require new and advanced analysis reflective of the complex requirements in their safe and reliable function. Such analysis also

needs to be tailored to fully evaluate the inherent character of DL [7,14], despite the progress made recently in this direction [10]. In [31] we propose a safety case framework tailored for DL systems, in which we describe a methodology for combining robustness verification and operational testing for reliability claims. In this paper, we implement such methodology as a reliability assessment model (RAM) with theoretical details and optimise and evaluate it against case studies.

The proposed RAM is inspired by partition-based testing and operational testing with uncertain operational profiles (OP), corresponding to two major stages: (a) prior to operational testing, the input space is partitioned into three types of cells. The robustness (to the ground truth label) of cells is first evaluated and then "assembled" according to their estimated OPs to form prior knowledge on the system reliability; (b) during operational testing we update the OP of cells as well as the system reliability given newly collected operational data. Subject to certain conditions being met, a light-weight robustness verifier will be invoked for certain cells, whose results will be incorporated into the RAM.

Detailed steps of each stage involve conservative treatment that is in the best interest for safety—e.g. setting the robustness of a cell to 0 if its ground truth label is uncertain. Each step corresponds to a formally stated problem that is not restricted to any certain solution. Our new model therefore provides a flexible and evolvable *framework*: it is flexible in the sense that it can be tailored for DL applications with different safety-integrity and/or scalability requirements, and evolvable by integrating new techniques (e.g. on verifying DL robustness).

The key contributions of this work are:

a) A first bespoke RAM for DL systems, which is based on evidence of robustness evaluation and operational testing.

b) A prototype tool of our new RAM with illustrative case studies, publicly available at `https://github.com/havelhuang/ReAsDL`.

c) The role of related concepts of reliability, accuracy, generalisability and robustness are discussed and summarised.

Organisation of the paper. We first present preliminaries on software reliability, DL accuracy, generalisability and robustness. Section 3 then describes the conceptual model in detail, and Section 4 evaluates the model against case studies. We close with a discussion of related work in Section 5 and a discussion of the assumption of our model and conclusions in Section 6.

2 Preliminaries

Software Reliability and its Assessment. Prior research into software reliability has advocated formalisation in probabilistic terms [18]. The main argument of the objectors is that software fails *systematically* due to design faults, i.e. an input/circumstance causes the software to fail once, will *deterministically* cause it to fail again upon seeing the same input/circumstance. Littewood and Strigini [20] state that the word "systematic" refers to the mechanism whereby a software fault reveals itself as a failure when given an input, while the successive inputs

(potentially uncovering some faults) are random. Thus our real interest, the failure process, is *stochastic*, due to the uncertainties of: (a) the random sampling of inputs; and (b) if a future sampled input (potentially unseen/untested before) would cause a failure (otherwise the fault can be fixed before the execution).

During this debate on RAM of software, it should be highlighted that characteristics of DL (e.g. learning from limited samples from an ultra-high dimensional input space, its black-box nature, interaction with its environment, self-learning, and deployment in complex/dynamic environments, etc) increase non-determinism in terms of the two aforementioned sources of uncertainties. Consequently we believe reliability claims on DL, compared to traditional software, are even more naturally (if not inevitably) to be stated in probabilistic terms.

Reliability, as a probabilistic claim, needs to be formally defined by a measure. Typically, we have *continuous systems* that are being continuously operated in the active control of some process and *on-demand systems* that are only called upon to act on receipt of discrete demands. Usually we study the failure rate (number of failures in one time unit) of the former (e.g. flight control software) and the *probability of failure per demand* (*pfd*) of the latter (e.g. the emergency shutdown system of a nuclear plant). Without loss of generality, we focus on *pfd*[4], which aligns with most DNN classifiers for perception, where demands are, e.g. facial images uploaded by users to a cloud service for facial recognition, or camera images fed into a traffic sign recognition component on self-driving cars, while failures are miss-classifications[5].

If we denote the unknown *pfd* as a variable λ, then

$$\lambda := \sum_{x \in \mathcal{D}} I_{\{x \text{ causes a failure}\}}(x) Op(x) \tag{1}$$

where x is a demand in the input domain $\mathcal{D}$, and $I_{\mathtt{S}}$ is an indicator function—it is equal to 1 when $\mathtt{S}$ is true and 0 otherwise. The $Op(x)$ captures the uncertainty of "which one would be the next input"—it returns the probability that x is the next random input, i.e. the *Operational Profile* (OP) [21], a notion used in software engineering to quantify how the software will be operated. We also do not know $I(x)$ for all x in $\mathcal{D}$, having observed only a very small region of it (especially for DL with a high dimensional $\mathcal{D}$), which results in uncertainty of "whether or not a given input will cause a failure".

In general, software testing methods can be classified as two categories based on their main goals—*debug* testing and *operational* testing, aiming at *improving* and *assessing/demonstrating* reliability, respectively [8]. The former, e.g. coverage-guided and partition-based testing, utilises test cases to excite as many failures as possible (in order to fix them prior to deployment). However, the test cases are normally not representative of the software's day-to-day operation. Confidence in the delivered reliability can only be gained by testing that represents the typical usage [9]—the operational testing. Indeed, tests representative

[4] We expect that our model can be adapted system reliability stated in failure rates.

[5] Specific definitions on "demands" and "failures" in *pfd* may vary case by case. E.g., depending on the sensor's working frequency, a demand could be a burst of images on the object, while a failure is a miss-classification after some "voting" mechanism.

of operation may seldom encounter failures when the system is ultra-reliable, thus prior knowledge needs to be incorporated in the RAM [33]. The proposed model is the latter and aims at assessing the reliability of a "frozen" DL model. Thus once we encountered a failure, we record it without fixing (which would result in a new version of the software, making existing failure data obsolete).

For on-demand systems, the Bernoulli process is a common choice for the mathematical abstraction of the failure process [2], implying a likelihood function

$$L(\lambda; k\&n) = Pr(k\&n \mid \lambda) = \lambda^k (1-\lambda)^{n-k} \ , \tag{2}$$

where $k\&n$ denotes the evidence of k failures in n tests sampled from the OP.

Upon establishing the likelihood function, RAMs on estimating λ vary case by case—from the basic Maximum Likelihood Estimation (MLE) to Bayesian estimators tailored for certain scenarios when, e.g., seeing no failure [5], inferring ultra-high reliability [33], with certain forms of prior knowledge (e.g., perfectioness [24]), using imprecise probabilities [30], with uncertain OPs [6,22], etc.

DNN and its Accuracy, Generalisability and Robustness. Let (X, Y) be the dataset (including both the training data (X^{train}, Y^{train}) and the testing data (X^{test}, Y^{test})), where X is a vector of inputs and Y is a vector of output labels such that $|X| = |Y|$. As before, $\mathcal{D}$ denotes the whole input domain and $\mathtt{Y}$ the set of labels. A deep neural network (DNN) $\mathcal{N} : \mathcal{D} \to \mathsf{Dist}(\mathtt{Y})$ can be seen as a function mapping from $\mathcal{D}$ to probabilistic distributions over $\mathtt{Y}$. That is, $\mathcal{N}(x)$ is a probabilistic distribution that assigns, for each possible label $y \in \mathtt{Y}$, a probability value $(\mathcal{N}(x))_y$. We let $f_{\mathcal{N}} : \mathcal{D} \to \mathtt{Y}$ be a function such that for any $x \in \mathcal{D}$, $f_{\mathcal{N}}(x) = \arg\max_{y \in \mathtt{Y}}\{(\mathcal{N}(x))_y\}$, i.e. $f_{\mathcal{N}}(x)$ returns the classification label. Also, we assume that there is a *ground truth* function $f : \mathcal{D} \to \mathtt{Y}$, so that $f(x)$ returns the ground truth label of x. By DNN accuracy, we usually mean

$$\mathit{Accuracy}(X^{test}) := 1 - \frac{1}{|X^{test}|} \sum_{x \in X^{test}} I_{\{f_{\mathcal{N}}(x) \neq f(x)\}}(x) \ . \tag{3}$$

We note two observations on Equation (3):

a) Accuracy is associated with a set of testing data, which normally is the test set (X^{test}, Y^{test}). If the associated set of data changes—e.g. when we use only a subset of (X^{test}, Y^{test}), the accuracy changes accordingly. Reliability (stated in *pfd*, cf. Eq. (1)), on the other hand, is associated with the whole input domain $\mathcal{D}$.

b) All inputs in the test dataset are treated equally when calculating the accuracy. In contrast, inputs normally have different contributions to reliability according to the OP. Only in the case when the test data is sampled from the OP, accuracy becomes an estimate of reliability.

The related but different notion or generalisability requires that a DNN works well on all possible inputs in $\mathcal{D}$, although it is only trained and tested on the limited existing dataset (X, Y). It is usually defined in terms of the *generalisation error* $G_{\mathcal{N}}$ using some loss function $\mathit{Loss}(\cdot)$:

$$G_{\mathcal{N}} := \sum_{x \in \mathcal{D}} \mathit{Loss}(x) \times \mathit{Op}(x) \ . \tag{4}$$

Although there is no one-size-fits-all loss function to different training algorithms in DL, common $\mathit{Loss}(\cdot)$ include Cross Entropy Loss, Hinge Loss, etc. We note:

a) When we use a 0-1 loss (i.e., we assign the value 0 to $Loss(\cdot)$ for a correct classification and 1 otherwise), $G_{\mathcal{N}}$ becomes the reliability stated in *pfd* [31].

b) Like *pfd*, $G_{\mathcal{N}}$ can never be known for certain, thus $\frac{1}{|X^{test}|}\sum_{x\in X^{test}} Loss(x)$ is normally used as an estimate assuming the test set represents the OP. If $Loss(\cdot)$ was the 0-1 loss, the estimated $G_{\mathcal{N}}$ is the inaccuracy upon the test set.

Robustness requires that the decision of the DNN is invariant against small perturbations on inputs. That is, all inputs in a region $\eta \subset \mathcal{D}$ have the same prediction label, where usually the region η is a small norm ball (in some L_p-norm distance[6]) of radius ϵ around an input x. Inside η, if an input x' is classified differently to x by the DNN, then x' is an *adversarial example*. Robustness can be defined either as a binary metric (if there exists any adversarial example in η) or as a probabilistic metric (how likely the event of seeing an adversarial example in η is). The former aligns with formal verification, e.g. [11], while the latter is normally used in statistical approaches, e.g. [26]. The former 'verification approach' is the qualitative version of the latter 'stochastic approach'[7].

Similar to [26], we adopt the more general probabilistic metric of robustness:

$$R_{\mathcal{N}}(\eta, y) := \sum_{x\in\eta} I_{\{f_{\mathcal{N}}(x)=y\}}(x) \times Op(x \mid x \in \eta) \ , \tag{5}$$

where $y \in \mathtt{Y}$ is a given label, and $Op(x \mid x \in \eta)$ is the conditional OP of region η (precisely the "input model" defined in [26] and also used in [27]). We note:

a) Normally the radius ϵ of the region η is quite small, such that $\epsilon \ll r$, where r derives from the r-separation property [28]—the distance between any two data-points with different labels is at least $2r$. Thus, inputs in the region η may have the same ground truth label. When y is such a ground truth label, robustness becomes *astuteness* (i.e. robustness to the ground truth) [28].

b) Astuteness is the *conditional* reliability in that region η.

3 A Reliability Assessment Model for Deep Learning

Fig. 1 presents the overall workflow of our RAM. There are two main stages: prior to operational testing (lhs) and during operational testing (rhs). We first partition the input space into small cells, within which astuteness is evaluated. Then we "assemble" those cell-wise astuteness estimates according to the OP of cells to get a system reliability estimate. As the OP at the fine grain level of small cells is unknown to us, we use (inspired by [22]) Bayesian inference on the OP during operational testing. We also update the astuteness of cells during the operational testing with compromised solutions for online efficiency. Steps in the workflow may have multiple technical solutions, while, for brevity, we formally state the problems tackled by those steps with one or more illustrative/representative solutions. In that sense, our RAM is a flexible and evolving *framework* that can be tailored for DL in various contexts (e.g. of different safety-integrity and scalability levels) with emerging techniques.

[6] Without further clarification, distance mentioned in this paper is defined in L_∞.

[7] Thus, we use the general term robustness "evaluation" rather than "verification".

3.1 Prior to Operational Testing

The first step at this stage is to partition the whole input domain $\mathcal{D}$ into "cells":

Problem Statement 1 (cell partition) *Partition $\mathcal{D}$ into m cells $c_1, \ldots, c_m$, such that it is almost true that the following properties hold: (i) all data-points within a cell share a single ground truth label; and (ii) the conditional OP of inputs within a cell can be reasonably approximated.*

The problem of finding the right "size" of the cells, small enough for properties (i) and (ii), but large enough to be useful for prediction, is addressed using the the r-separation property [28].

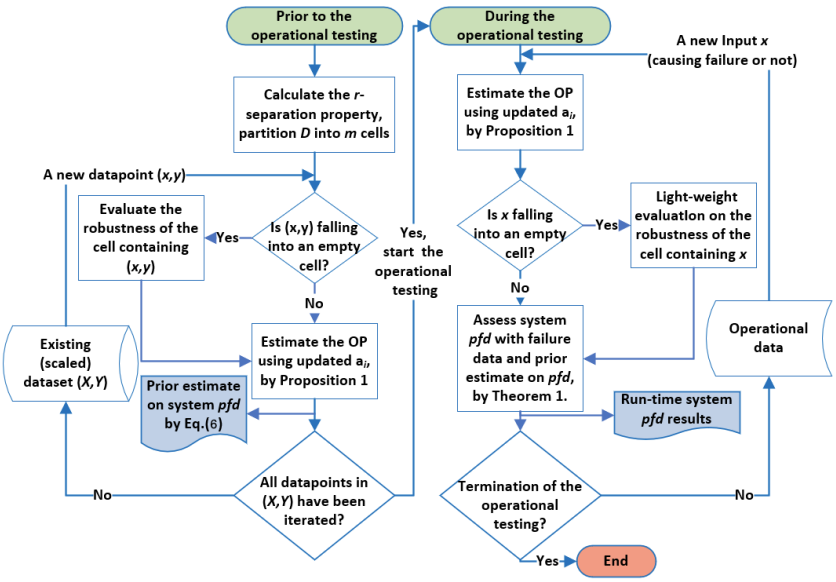

Fig. 1: Workflow of the proposed RAM.

The 'r' (for radius) in r-separation is at most half the distance between any two examples with different labels; typical, the radius ϵ of a region (e.g. a norm ball) in which we study the robustness is much smaller. For instance, an empirical observation based on four real-world image datasets studied in [28] implies r is 3 to 7 times the ϵ commonly used in robustness experiments. This provides a rule of thumb:for a given DL application, we first calculate r based on its existing dataset (X, Y), and then select the radius ϵ of our cells, say a third of r. This assures that the claim "each cell c_i has a single ground truth label" (denoted as y_i) is *almost true*: while there are likely some cells that crossing the boundaries of different ground truth labels, thanks to the r-separation property, data-points in these cells would be "ambiguous" images (without ground truth—different people may assign different labels) that are unlikely to occur in practice. Our RAM estimates the probability of such extreme case (in later steps, although it

may require many more test cases), and we conservatively set the astuteness of such "cross-boundary" cells to 0.

The second desirable property in Problem Statement 1 is regarding the conditional OP, i.e. $Op(x \mid x \in c_i)$. In our case, we conjecture the cell radius ϵ is so small that inputs from the same cell can be assumed to be uniformly distributed. Intuitively this says, those very close/similar inputs are only subject to natural noise factors (like lighting conditions) that can be reasonably modelled uniformly. While other distributions (e.g. supported by physical sensor models) for $Op(x \mid x \in c_i)$ can also be assumed in our flexible framework, we choose a uniform distribution for illustration (which is not uncommon, cf. [26,22]).

Although our way of partitioning may create a very large number of cells (e.g., the MNIST dataset has $3^{28\times 28}$ cells in our case studies), the majority of them would be "empty" in the sense that no data-point was observed (in (X, Y) and operational data) would be in those cells. However, we cannot know if a given cell will *eventually* be empty—a new input still has some chance of falling into an empty cell, which relates to the OP estimation problem stated later. Later, we treat all empty cells (based on up-to-date data) as one big cell c_{emp} with estimates on its *pooled* OP Op_{emp} and its contribution to unreliability λ_{emp}.

In summary, our way of partitioning creates three types of cells—normal, empty and cross-boundary cells. For each cell c_i, let us temporarily assume we know perfectly the values of $\lambda_i := 1 - R_{\mathcal{N}}(c_i, y_i)$ and $Op_i := \sum_{x\in c_i} Op(x)$ which are the conditional *pfd* and pooled OP of cell c_i. Then the system *pfd* is:

$$\begin{aligned}\lambda = \sum_{i=1..m} \lambda_i Op_i &= \sum_{\text{normal } c_i} \lambda_i Op_i + \sum_{\text{empty } c_i} \lambda_i Op_i + \sum_{\text{cro. bou. } c_i} \lambda_i Op_i \\ &\leq \sum_{\text{normal } c_i} \lambda_i Op_i + \lambda_{emp} Op_{emp} + \sum_{\text{cro. bou. } c_i} 1 \times Op_i \end{aligned} \tag{6}$$

where the first equality is an established relation (in, e.g., [6,22]) and the last inequality is by our conservative treatment on empty and cross-boundary cells.

In reality, we never have precise knowledge of the λ_i and Op_i due to *epistemic* uncertainties captured by distributions, say $h(\lambda_1, \ldots, \lambda_m)$ and $g(Op_1, \ldots, Op_m)$, whose marginal expectations on λ_i and Op_i are $\widehat{\lambda}_i$ and $\widehat{Op}_i$ (for which we present estimators later). Then, we can simply "plug in" them into Eq. (6), due to

$$\mathbb{E}(\lambda) = \mathbb{E}_g(\mathbb{E}_h(\lambda | Op_1, ..., Op_m)) = \mathbb{E}_g\left(\sum_{i=1..m} \widehat{\lambda}_i Op_i\right) = \sum_{i=1..m} \widehat{\lambda}_i \widehat{Op}_i \tag{7}$$

Problem Statement 2 (cell astuteness evaluation) *Given a cell c_i, evaluate the robustness to its ground truth label (astuteness).*

Essentially the aim is to get $\widehat{\lambda}_i$. First, we need determine the ground truth label y_i of the cell c_i. For this, we iterate all data-points in (X, Y) (assuming all $y \in Y$ are human labelled, thus correct). When the data-point (x, y) falls into a cell: (i) that has not been labelled before, then assign the label y as its ground truth; (ii) that has already been labelled as y, then keep y as its ground truth; or (iii) that has been labelled, but not as y, we label it as a cross-boundary cell.

For a normal cell c_i, given its ground truth label y_i, there exist many methods (cf. [10]) that can estimate λ_i, by reducing it to a robustness evaluation problem. We use the tool developed by [26] as an example[8]. Since we observe no labelled data-point in empty cells, we do not know their ground truth labels *automatically* (it is impractical to label empty cells manually). We can therefore either use some theoretical bound on the generalisation error (e.g. [12]) as an approximation of λ_{emp} or conservatively set $\widehat{\lambda}_{emp} = 1$.

Problem Statement 3 (OP estimation) *Given n samples, seeing a_i data-points in non-empty cell c_i, estimate the OP of cells—Op_i and Op_{emp}. This equals to a statistical inference problem from multinomial data with unknown categories [25]: A sample of n observations yielding the counts $\boldsymbol{a} = (a_1, \ldots, a_s)$ over s categories from set C with unknown chances $\boldsymbol{Op} = (Op_i, \ldots, Op_s)$, where the set C and its cardinality S are unknown a priori. The inference is on $\boldsymbol{Op}$.*

Compared with the basic multinomial-inference problem, there is notably *prior ignorance* about the set of categories. Whenever we observe a new data-point (either during the iteration of (X, Y) or operational testing) falling into an empty cell, the cardinality of set $C = \{\{c_i\}, c_{emp}\}$ grows by 1 (since $|\{c_i\}|$, the size of the subset of non-empty cells, grows). We choose the Imprecise Dirichlet Model (IDM) [25,3] as an example Bayesian estimator for this statistical problem.

Proposition 1 (IDM [3]). *To the problem stated in Problem Statement 3, IDM gives the lower and upper bounds on the posterior expectation of Op_i:*

$$\underline{\mathbb{E}}(Op_i \mid \boldsymbol{a}) = \frac{a_i}{n+v}, \quad \overline{\mathbb{E}}(Op_i \mid \boldsymbol{a}) = \frac{a_i + v}{n+v} \tag{8}$$

(note $a_i = 0$ for Op_{emp}) where v represents the degree of imprecision.

The hyper-parameter v determines how fast the lower and upper bounds converge one towards the other as n increases. The choice of v represents different principles of modelling prior ignorance, e.g., $v \to 0$ Haldane's improper prior, $v = 1$ Perks, $v = S/2$ Jeffreys and $v = k$ Bayes-Laplace prior, as discussed in [3,25]. To demonstrate our framework in this paper, we choose $v = 2$ as suggested by [3] for the case where S is large and the inference is on posterior mean.

The conservatively estimation for system *pfd*, can be formulated as a linear programming of Op_i's that maximise Eq. (6), subject to the interval constraints in Eq. (8) (of each cell) and the unit simplex constraint $\sum Op_i = 1$. The solution is simply to assign $\widehat{Op}_i = \frac{a_i}{v+n}$ for all but the maximal $\widehat{\lambda}_i$, normally $\widehat{\lambda}_{emp} = 1$ ($\gg \widehat{\lambda}_i$ of the non-empty cells), which leads to $\widehat{Op}_{emp} = \frac{v}{v+n}$.

3.2 During Operational Testing

Prior to the operational testing, by iterating all data-points in the (X, Y), we had a prior expectation on the system *pfd*, denoted as $\widehat{\lambda}$, which is essentially a

[8] Other tools like formal methods that yield a robust region can also be used. In this case $\widehat{\lambda}_i$ can be conservatively set set to the proportion of robust region covered in c_i.

function of the estimates $\widehat{\lambda}_i$'s and $\widehat{Op}_i$'s of non-empty cells and $\widehat{\lambda}_{emp}$ and $\widehat{Op}_{emp}$. During the operational testing, we keep updating those estimates to calculate $\widehat{\lambda}$, upon seeing operational inputs. The way of updating the OP during the operational testing is the same as before (solutions to Problem Statement 3), while how to update the astuteness of cells is different as follows:

a) When the operational input x falls into an empty cell, we need to evaluate the cell's astuteness from scratch. If the operational testing is conducted in simulators/confined-testing-field, we can have a perfect oracle telling the ground truth label of x (ditto its cell's ground truth based on which we do the astuteness evaluation). Otherwise, we cannot directly know the ground truth label of x without human intervention—we can only observe whether or not it is causing a *higher-level system failure*. When seeing a failure, the ground truth label of x remains unknown, thus we conservatively set its cell's unastuteness $\lambda_i = 1$. On the other hand, seeing no failure allows us to use the predicted label of x as the ground truth[9] of its cell, then do the astuteness evaluation. We note that the astuteness evaluator has to be *light-weight* due to efficiency constraints at runtime (e.g., by tuning some efficiency-related hyper-parameters of [26]).

b) When the operational input x falls into a non-empty cell c_i, observing x causing a failure or not will also change our beliefs (i.e. epistemic uncertainty) on the cell's astuteness. However, update such beliefs in a "cell-wise" manner is not only inefficient but also mathematically intractable due to the difficulty of modelling dependencies among cells (i.e. updating on the astuteness estimation of one cell may impact the beliefs about other cells). That said, we directly update our beliefs on the *pfd* (denoted by the variable λ) at the system level:

Problem Statement 4 (system *pfd* update) *During the operational testing, update beliefs about the system pfd when seeing k failures in n tests, i.e.* $\mathbb{E}(\lambda \mid k\&n)$*, knowing the expected pfd* $\mathbb{E}(\lambda) = \widehat{\lambda}$ *as the prior knowledge.*

Theorem 1 (conservative Beta-Binomial inference). *Assuming the system pfd has a Beta prior distribution and a Binomial likelihood of seeing k failures in n tests, let* $\mathcal{U}$ *be the set of all possible Beta priors satisfying* $\mathbb{E}(\lambda) = \widehat{\lambda}$*, then the posterior expected pfd has a supremum* $\sup_{\mathcal{U}} \mathbb{E}(\lambda \mid k\&n) = \max(\widehat{\lambda}, \frac{k}{n})$.

Proof. First, instead of assuming $\lambda \sim \text{Beta}(\alpha, \beta)$ where α and β are the two common shape parameters, we *reparametrise* it as $\lambda \sim \text{Beta}(n^{(0)}, z^{(0)})$ by setting $n^{(0)} = \alpha + \beta, z^{(0)} = \alpha/(\alpha + \beta)$. The $n^{(0)}$ and $z^{(0)}$ are called *canonical* parameters [4] that intuitively represent the prior strength and the prior expectation, respectively. Thus, in our case, there is an infinity set $\mathcal{U}$ of Beta priors $\text{Beta}(n^{(0)}, \widehat{\lambda})$ where $n^{(0)} \in (0, +\infty)$ is the variable in our optimisation problem.

For each item in $\mathcal{U}$ (i.e. given an $n^{(0)}$), thanks to the conjugacy with the Binomial likelihood in Eq. (2), the posterior is also a Beta (denoted as $\text{Beta}(n^{(1)}, z^{(1)})$) with posterior parameters [4]: $n^{(1)} = n^{(0)} + n$ and $z^{(1)} = \frac{n^{(0)}}{n^{(0)}+n}\widehat{\lambda} + \frac{n}{n^{(0)}+n}\frac{k}{n}$, from which it is easy to see the posterior expectation $z^{(1)}$ is a weighted sum of the

[9] Systems with fault-tolerance design may hide occasional DL misclassifications that needs extra care, e.g., retrospective checking on if a miss-classification was tolerated.

prior estimate $\widehat{\lambda}$ and the MLE of samples k/n, and the weights are proportional to the prior strength $n^{(0)}$ and the sample size n.

Now we treat $z^{(1)}(n^{(0)})$ as our objective function ($n^{(0)}$ is the only variable), then its derivative is $\frac{\mathrm{d}z^{(1)}(n^{(0)})}{\mathrm{d}n^{(0)}} = \frac{\widehat{\lambda}n-k}{(n^{(0)}+n)^2}$ by which we know: (i) When $\widehat{\lambda} > k/n$, $z^{(1)}(n^{(0)})$ is an increasing function. To maximise it, we take $\lim_{n^{(0)}\to+\infty} z^{(1)}(n^{(0)}) = \widehat{\lambda}$. (ii) When $\widehat{\lambda} < k/n$, $z^{(1)}(n^{(0)})$ is a decreasing function. To maximise it, we take $\lim_{n^{(0)}\to 0} z^{(1)}(n^{(0)}) = k/n$. Finally, since $n^{(0)}$ is taking the limiting value in both cases, the corresponding result is a supremum. □

Intuitively, Theorem 1 says, in the quite flexible and common Beta-Binomial parametric models, the *worst-case* posterior system *pfd* is either the prior estimate $\widehat{\lambda}$ or the MLE result k/n, whichever is more conservative. Such conservatism is in the best interest of safety-critical systems that normally see non or very rare failures in operation, where MLE alone tends to be unwittingly optimistic. For instance, MLE always indicates a perfect software when $k = 0$, which is misleadingly optimistic, while Theorem 1 gives a more conservative result based on solid evidence from robustness evaluation. On the other hand, we may reduce any unnecessary conservatism in Theorem 1 if prior knowledge allowed us to. That is, the prior strength $n^{(0)}$ on our prior estimate can be narrowed down to a smaller interval than $(0, +\infty)$ when there is sound evidence that supports upper or lower bounds.

Finally, although $\widehat{\lambda}$ is called a "prior" estimate, it is only a prior in the Beta-Binomial model. Thus it is not surprising that $\widehat{\lambda}$ itself is also continuously updated, due to the updates of the OP and newly incorporated cell robustness.

4 Case Studies

As shown in Table 1, we conduct case studies to demonstrate our RAM with 3 DNNs trained on the common MNIST, CIFAR-10 and also a real-word traffic sign recognition dataset GTSRB [23]. In line with the r-separation property, we first calculate the minimum label separation ($2r$) based on all training data. Then the cell diameter (2ϵ) is determined such that $\epsilon < r$, where the cell diameter refers to the value range of a single pixel as an element of the image array (after scaled to 0-1). E.g., an MNIST image has 28×28 pixels. If we split the scaled pixel value into 3 segments, the cell diameter is approximately 0.333, and there are $3^{28\times28}$ mutually exclusive cells. Without loss of generality, we use a subset of the training data to form the prior estimates on *pfd*, compared with the Average Cell Unastuteness (ACU) and MLE results (accuracy upon the operational data).

For each DNN, we synthesise two operational test sets, OP1 and OP2, to simulate two operational scenarios. OP1 consists of data randomly sampled from both the training and testing sets with perturbation: it represents an ideal case where the existing dataset statistically depicts the future operational use. OP2 represents the case where the future OP and the existing dataset used for training are mismatched, which is common and not necessarily bad. Training data is normally collected in a *balanced* way, since the DNN is expected to perform well

in all categories of input, especially when the OP is unknown at the time of training and/or expected to change in future. The delivered reliability would be higher if we see more inputs from the category that the DNN performs well on (e.g. images of a specific ground truth label that are inherently easier to classify). In our case studies, we mimic the opposite situation in which the unreliable input categories are associated with higher OP, as shown in the last column of Fig. 2. For more operational scenarios, cf. our project website.

Table 1: Minimum label separation, cell size & prior estimates by used samples.

Dataset	Mini. Label Separation	Cell Diameter	No. of Used Samples	Prior Estimates		
				pfd	ACU	MLE
MNIST	0.737	0.333	5000	0.04103	0.04064	0.00380
CIFAR-10	0.212	0.100	5000	0.16968	0.16935	0.01580
GTSRB	0.059	0.050	4300	0.02144	0.02099	0.01813

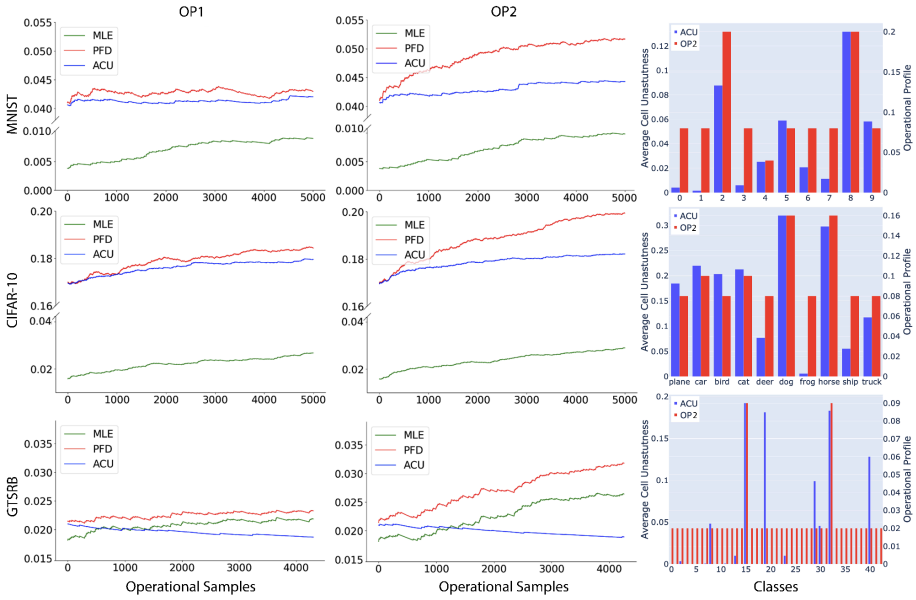

Fig. 2: Case studies on three DNNs with two types of operational test sets.

We observe in Fig. 2: (i) For DNNs trained on MNIST and CIFAR-10, ACU estimates are higher than MLE, indicating the lack of robustness. MLE is misleadingly optimistic in this case, while ACU does not take account of the OP of cells. Our *pfd* results address both problems. (ii) The DNN trained on GTSRB is indeed very robust due to the augmented samples (representing extreme cases, e.g. overexposed conditions) in the training data. Our conservative RAM still provides good estimates without unnecessary conservatism. (iii) As expected, all *pfd* estimates in OP2 are worse than those in OP1, since we deliberately scaled the OP such that unrobust classes of inputs have higher probability to be seen in operation. In contrast, MLE and ACU are much less sensitive to the OPs.

5 Related Work

In recent years, there has been extensive efforts in evaluating/verifying DL robustness, detecting adversarial examples and estimating DL generalisation errors [11,12,26,27,28]. A comprehensive review of those techniques may sourced from recent survey papers [10,29]. To the best of our knowledge, the only papers on testing DNNs within an operational context are [16,17]. In [16], novel stratified sampling methods are used to improve the operational testing efficiency. Operational calibration, a Bayesian approach that gradually corrects the output confidence function of a DNN against its operational data, is proposed in [17].

At the abstract level of systems utilising DL, although there are RAMs based on operational data, knowledge from low-level DL components is ignored, e.g., [13]. In [33] we improved [13] by providing a Bayesian mechanism to combine such knowledge, but did not show how to obtain that knowledge. In that sense, this paper is an important extension of [33] by providing a more comprehensive and end-to-end RAM. A notion of quantified assurance is proposed by [1], where a bespoke measure is needed for some specific quantifiable system-level property, thus the underlying stochastic model varies case by case. While, our model is more like DL-component oriented and generic to all systems where the common reliability measure *pfd* (and failure rate, as planned in future work) is applicable.

6 Discussion, Conclusion & Future Work

We present a bespoke RAM for DL combining evidence from robustness evaluation and operational testing. By partitioning the input space into 3 types of cells, dynamic reliability estimates are obtained by "assembling" the astuteness of cells according to the continuously updated OP.

Several key assumptions and observations are made on our cells:

a) Determination of the ground truth label of a cell is non-trivial. Thanks to the r-separation property [28], we may confidently assume a *small enough* cell has a single ground truth label, and know what that label is when the cell contains human-labelled data. We do not exclude the possibility of cells containing data with different labels (called cross-boundary cells which are indeed very rare thanks to the r-separation). Our RAM can estimate such small probability given sufficient operational data, and conservatively set those cells' astuteness to 0.

b) Given a dataset, the cell size is restricted by the minimum label separation which should be periodically recalculated when massive new data is collected and labelled. Upon an obvious decrease of the r-separation distance, redo the partition to calibrate our model is needed (as part of dynamic assurance cases).

c) We note the limiting cases of partitioning cells—each data-point is a cell and the whole input space is a cell. The former becomes accuracy testing (against the operational test set) ignoring robustness evidence, while the later is a one big cross-boundary cell whose reliability is 0 by our conservative treatment.

d) There are two side effects due to the small size of cells. First, most of the cells are empty and we cannot know their ground truth labels. Second, for

non-empty cells, it is likely that only one data-point is observed per cell at earlier stages of the operation, thus the OP estimates are not accurate. For the former, we treat all empty cells as one, conservatively set the astuteness to 0, and then estimate the OP by reducing it to a known statistical problem [3]. For the latter, indeed we need a "burn-in" stage (not unusual for Bayesian inference) to collect enough data to get accurate estimates. We envisage ML data-augmentation techniques may speed it up, for which we plan to do in future.

e) For illustration, we assume the conditional OP within each cell is uniform. Although we conjecture this is the common case due to the small size of cells, the real situation indeed may vary case by case that requires justification in safety cases. Our model is flexible in terms of assuming different distributions.

We confine our model on general DL classifiers, where reliability is measured in *pfd*. Given a specific application, the definitions of "demands" and "failures" in the *pfd* may change. E.g., for most robotics missions, a demand feed into the perception component is a burst of images on an object (depends on the sensor's frequency) and a failure is a miss-classification after some "voting" mechanism on a burst of labels. Our model is still applicable in this case but with some extra cares, e.g., the robustness evaluation has to be done in a statistical way such that the voting mechanism is part of the black-box component under study.

Our model can approximate non-constant OPs during the operation, but it might be slow to react to sudden changes. We plan to pair our OP estimator with change-point detectors, similar to [32]. We plan to implement a concrete safety case as outlined in [31] in which our new RAM is the main probabilistic argument. Finally, our model has the potential to assess quantified security of DL, if open questions (e.g., as discussed in [19]) are addressed in future work.

References

1. Asaadi, E., Denney, E., Pai, G.: Quantifying assurance in learning-enabled systems. In: SafeComp'20. LNCS, vol. 12234, pp. 270–286. Springer, Cham (2020)
2. Atwood, C., et al: Handbook of parameter estimation for probabilistic risk assessment. Report NUREG/CR-6823, U.S. Nuclear Regulatory Commission (2003)
3. Bernard, J.M.: An introduction to the imprecise Dirichlet model for multinomial data. International Journal of Approximate Reasoning **39**(2), 123 – 150 (2005)
4. Bernardo, J.M., Smith, A.F.M.: Bayesian theory. Wiley (1994)
5. Bishop, P., Bloomfield, R., Littlewood, B., Povyakalo, A., Wright, D.: Toward a formalism for conservative claims about the dependability of software-based systems. IEEE Transactions on Software Engineering **37**(5), 708–717 (2011)
6. Bishop, P., Povyakalo, A.: Deriving a frequentist conservative confidence bound for probability of failure per demand for systems with different operational and test profiles. Reliability Engineering & System Safety **158**, 246–253 (2017)
7. Bloomfield, R., et al: Disruptive innovations and disruptive assurance: Assuring machine learning and autonomy. Computer **52**(9), 82–89 (2019)
8. Frankl, P.G., Hamlet, R.G., Littlewood, B., Strigini, L.: Evaluating testing methods by delivered reliability [software]. IEEE Tran. on Softw. Eng. **24**(8), 586–601 (1998)
9. Hamlet, D., Taylor, R.: Partition testing does not inspire confidence (program testing). IEEE Transactions on Software Engineering **16**(12), 1402–1411 (1990)

10. Huang, X., et al: A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review **37**, 100270 (2020)
11. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: CAV'17. LNCS, vol. 10426, pp. 3–29. Springer, Cham (2017)
12. Jin, G., et al: How does weight correlation affect generalisation ability of deep neural networks? In: NeurIPS'20. Vancouver, Canada (2020)
13. Kalra, N., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transportation Research Part A: Policy and Practice **94**, 182 – 193 (2016)
14. Koopman, P., Kane, A., Black, J.: Credible autonomy safety argumentation. In: 27th Safety-Critical Sys. Symp. Safety-Critical Systems Club, Bristol, UK (2019)
15. Lane, D., Bisset, D., Buckingham, R., Pegman, G., Prescott, T.: New foresight review on robotics and autonomous systems. Tech. Rep. No. 2016.1, LRF (2016)
16. Li, Z., Ma, X., Xu, C., Cao, C., Xu, J., Lü, J.: Boosting operational DNN testing efficiency through conditioning. In: ESEC/FSE'19. pp. 499–509. ACM (2019)
17. Li, Z., Ma, X., Xu, C., Xu, J., Cao, C., Lü, J.: Operational calibration: Debugging confidence errors for DNNs in the field. In: ESEC/FSE20. pp. 901–913. ACM (2020)
18. Littlewood, B., Strigini, L.: 'Validation of ultra-high dependability...' - 20 years on. Safety Systems, Newsletter of the Safety-Critical Systems Club **20**(3) (2011)
19. Littlewood, B., et al: Towards operational measures of computer security. Journal of Computer Security **2**(2-3), 211–229 (1993)
20. Littlewood, B., Strigini, L.: Software reliability and dependability: A roadmap. In: Proc. of the Conf. on The Future of Softw. Eng. pp. 175–188. ICSE'00 (2000)
21. Musa, J.D.: Operational profiles in software-reliability engineering. IEEE Software **10**(2), 14–32 (1993)
22. Pietrantuono, R., Popov, P., Russo, S.: Reliability assessment of service-based software under operational profile uncertainty. Rel. Eng. & Sy. Saf. **204**, 107193 (2020)
23. Stallkamp, J., et al: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Networks **32**, 323 – 332 (2012)
24. Strigini, L., Povyakalo, A.: Software fault-freeness and reliability predictions. In: SafeComp'13. LNCS, vol. 8153, pp. 106–117. Springer, Berlin, Heidelberg (2013)
25. Walley, P.: Inferences from multinomial data: Learning about a bag of marbles. Journal of the Royal Statistical Society. Series B **58**(1), 3–57 (1996)
26. Webb, S., Rainforth, T., Teh, Y.W., Kumar, M.P.: A statistical approach to assessing neural network robustness. In: ICLR'19. New Orleans, LA, USA (2019)
27. Weng, L., et al: PROVEN: Verifying robustness of neural networks with a probabilistic approach. In: ICML'19. vol. 97, pp. 6727–6736. PMLR (2019)
28. Yang, Y.Y., Rashtchian, C., Zhang, H., Salakhutdinov, R., Chaudhuri, K.: A Closer Look at Accuracy vs. Robustness. In: NeurIPS'20. Vancouver, Canada (2020)
29. Zhang, J.M., Harman, M., Ma, L., Liu, Y.: Machine learning testing: Survey, landscapes and horizons. IEEE Tran. on Software Engineering (2020), early access
30. Zhao, X., et al: Probabilistic model checking of robots deployed in extreme environments. In: AAAI'19. vol. 33, pp. 8076–8084 (2019)
31. Zhao, X., et al: A safety framework for critical systems utilising deep neural networks. In: SafeComp'20. LNCS, vol. 12234, pp. 244–259. Springer (2020)
32. Zhao, X., Calinescu, R., Gerasimou, S., Robu, V., Flynn, D.: Interval change-point detection for runtime probabilistic model checking. In: ASE'20. pp. 163–174 (2020)
33. Zhao, X., Salako, K., Strigini, L., Robu, V., Flynn, D.: Assessing safety-critical systems from operational testing: A study on autonomous vehicles. Information and Software Technology **128**, 106393 (2020)