Provisioning Robust and Interpretable AI/ML-based Service Bundles

Alun Preece, Dan Harborne
Crime and Security Research Institute
Cardiff University
Cardiff, UK
{preecead,harborned}@cardiff.ac.uk

Ramya Raghavendra
Thomas J. Watson Research Center
IBM US
Yorktown Heights, USA
rraghav@us.ibm.com

Richard Tomsett, Dave Braines
Emerging Technology
IBM UK
Hursley, UK
{rtomsett,dave_braines}@uk.ibm.com

***Abstract*—Coalition operations environments are characterised by the need to share intelligence, surveillance and reconnaissance services. Increasingly, such services are based on artificial intelligence (AI) and machine learning (ML) technologies. Two key issues in the exploitation of AI/ML services are robustness and interpretability. Employing a diverse portfolio of services can make a system robust to 'unknown unknowns'. Interpretability — the need for services to offer explanation facilities to engender user trust — can be addressed by a variety of methods to generate either transparent or post hoc explanations according to users' requirements. This paper shows how a service-provisioning framework for coalition operations can be extended to address specific requirements for robustness and interpretability, allowing automatic selection of service bundles for intelligence, surveillance and reconnaissance tasks. The approach is demonstrated in a case study on traffic monitoring featuring a diverse set of AI/ML services based on deep neural networks and heuristic reasoning approaches.**

***Index Terms*—intelligence, surveillance and reconnaissance; robustness; interpretability; reasoning; machine learning**

I. Introduction

A *coalition* is an alliance of partners with a common goal. In a military context, partners will typically need to share assets for intelligence, surveillance and reconnaissance tasks, e.g., sensing and information processing services [1]. Operational efficiency can be improved via agile approaches to sharing of services near the edge of the network [2], [3].

Recent years have seen a resurgence in the effectiveness of artificial intelligence (AI) and machine learning (ML) approaches for tasks including processing of imagery and text data, particularly approaches based on deep neural networks [4] and deep question-answering techniques [5]. Such methods are highly applicable to intelligence, surveillance and reconnaissance tasks, where there is commonly a need to process multiple data modalities and distinguish features at multiple semantic scales [6]. *Robustness* is a key issue in deploying AI/ML-based services, with 'unknown unknowns' being a matter of particular concern: the ability of a system to appropriately handle inputs on which it was not trained or which fall outside of its knowledge model [7]. A well-established method for addressing this issue is the use of *portfolio methods* where a diversity of approaches/models is employed. To some extent, the coalition context benefits this approach: multiple partners are more likely than a single partner to provide such a diversity of services.

A second key issue in deploying AI/ML-based services for intelligence, surveillance and reconnaissance tasks is service *interpretability*: the ability to offer an appropriate explanation for an output [8]. Unlike 'classical' symbolic AI approaches where it was possible for a system to explicate its chain of reasoning, ML services based on deep neural networks lack easily-communicable internal representations. This makes the generation of useful explanations or justifications for an output difficult, especially in terms appropriate for an end-user (e.g., an intelligence analyst). While such systems can be trained to provide effective classifications with intelligible explanations, there is a performance trade-off in doing so [9].

The focus of this paper is to examine the problem of agile service provisioning in a coalition context, with a particular focus on meeting requirements for robustness through diversity, and interpretability of various kinds appropriate for end-users. In part this builds on previous work in dynamic assignment of intelligence, surveillance and reconnaissance assets to mission tasks though, to the best of our knowledge, no previous work has considered both interpretability and robustness conditions for AI/ML-based services in this context. It should be borne in mind that the coalition context imposes a variety of constraints, including ones concerned with security, policy, and resources (network and physical), which will impact on robustness, accuracy and interpretability. These coalition constraints are not the primary focus of this paper but their impact is acknowledged.

The paper is organised as follows: Section II outlines previous work in dynamic asset provisioning for coalition intelligence, surveillance and reconnaissance tasks; Section III introduces a case study in the domain of traffic monitoring featuring multiple services based on ML and heuristic reasoning approaches; Section IV extends previous asset provisioning with features to handle interpretability and robustness requirements; Section V hows how the extended provisioning model

This research was sponsored by the U.S. Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the UK Ministry of Defence or the UK Government. The U.S. and UK Governments are authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

is applied in the case study; finally, Section VI reflects on the approach and identifies future work.

II. Background and Related Work

Dynamic service provisioning for coalition operations is a well-studied problem [1]. A key approach is to model the capabilities of available intelligence, surveillance and reconnaissance assets, e.g., sensors, and use these models — commonly in the form of *ontologies* [10] — to determine which assets are appropriate for given mission tasks. This process can be automated and optimised [11]. Commonly, a single asset is insufficient to meet the needs of a task, so assets are grouped into *bundles* during the process of asset-task matching [12]. Figure 1 shows the asset-task matching model used in [11]. Here, sensors are the only kind of asset considered. Bundles are computed by an algorithm that optimises the assignment of sensors to bundles, and bundles to tasks, in order to maximise the aggregate priority of the set of satisfied tasks, such that the utility demand of each satisfied task is met or exceeded by the joint utility of the bundle assigned to that task.

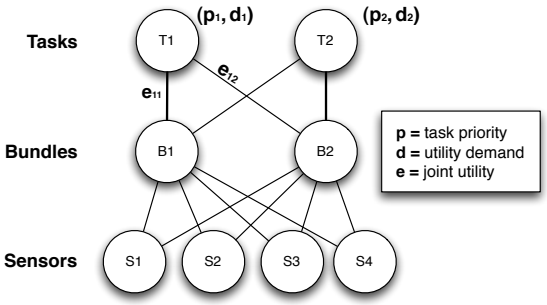

Fig. 1. Task-Bundle-Sensor model from [11]

The purpose of the asset ontology is to define the search space for the optimisation problem by classifying assets in terms of the capabilities they provide, and tasks in terms of the capabilities they require. An asset is a candidate for inclusion in a bundle to satisfy a task if it provides a capability required by the task. The model in Figure 1 can be generalised from sensors to all intelligence, surveillance and reconnaissance assets — including services — and will be referred to as the *task-bundle-asset* (TBA) model in subsequent discussion.

The TBA model is agnostic to the kind of asset employed, including AI or ML-based services. As discussed in the introduction, two key issues with these kinds of service asset are robustness and interpretability. The most significant aspect of the former is robustness to 'unknown unknowns' [7]: phenomena outside the AI/ML system's models. These occur because models are always incomplete; there will be aspects of the world that fall outside any given model. One of the main approaches to addressing this robustness problem is to employ a portfolio of diverse models so that weaknesses in one model are compensated for by strengths of another. The diversity property is important and can be implemented via training ML-based models on different sets of data and/or features, or by employing a combination of AI/ML techniques, for example reasoning and learning-based services.

On the one hand, the portfolio approach is facilitated by the coalition context with asset sharing at the network edge [3]: the coalition collectively will tend to have a greater diversity of assets than each individual partner. However, while the bundle approach used for asset provisioning is compatible in principle with having multiple ways to accomplish a task, previous work has not considered the requirement for diversity explicitly. Moreover, optimal asset allocation algorithms would disfavour bundles that appear to have over-provision of 'redundant' assets that 'do the same thing'. Therefore, previous work needs to be extended with (a) explicit requirements for asset diversity and (b) algorithms that favour diverse bundles where required.

Turning now to the requirement for interpretability, it has been acknowledged that there are varying definitions of this, partly arising due to there being multiple intents behind making AI/ML systems interpretable [8], [13]. There is as yet no formal ontology of interpretability types, though it has been argued [8] that these can generally be categorised as either *transparency* or *post hoc explanation*. The former provides direct evidence from internal working of how the AI/ML system arrived at an output. In a 'classical' AI system this would include a trace of the symbolic reasoning steps executed by a theorem prover or rule-based system. In a deep neural network, a transparent interpretation needs to be based on the firings of artificial neurons and it is far less clear how to make this process usefully intelligible to humans [14]. The dominant post hoc interpretability approach for deep neural networks is to use *saliency mapping* methods that visualise features of the input that have the most significant positive/negative effect on an output decision; e.g., regions of an image that were most significant in determining a classification for that image, visualised as a 'heatmap' [15], [16]. Another common post hoc interpretability techniques include explanations by example (e.g., using a case-based reasoning approach to select an appropriately-similar example from training set [17]) and natural language explanations (e.g., automatic generation of image captions [18]).

To ground the subsequent discussion of how to provision service bundles that meet both robustness (diversity) and interpretability (transparency and post hoc) requirements, the next section introduces a sensor-based system for traffic monitoring featuring services based on ML and reasoning approaches.

III. Motivating Example

The system described in this section was created as a testbed for situational understanding technologies in a coalition context [19]. The main requirements for the system were: (i) to address a plausible situational understanding problem (traffic monitoring) for which datasets of multiple modalities were readily available; (ii) to feature both AI-based (heuristic reasoning) and ML-based services; and (iii) to exemplify coalition constraints (service ownership, information flow). Traffic monitoring, with a specific focus on congestion detection,

was chosen for (i) because congestion can be viewed as a higher-level situational element with relationships to lower-level elements such as vehicles (e.g., cars, buses, bicycles) and road system objects (e.g., traffic lights, pedestrian crossings). Moreover, data is readily available: in the UK, Transport for London (TfL) provides an application programming interface (API) to obtain still imagery and video from their traffic camera network around London[1], while Open Street Maps (OSM) provides information about the road network, e.g., speed limit[2].

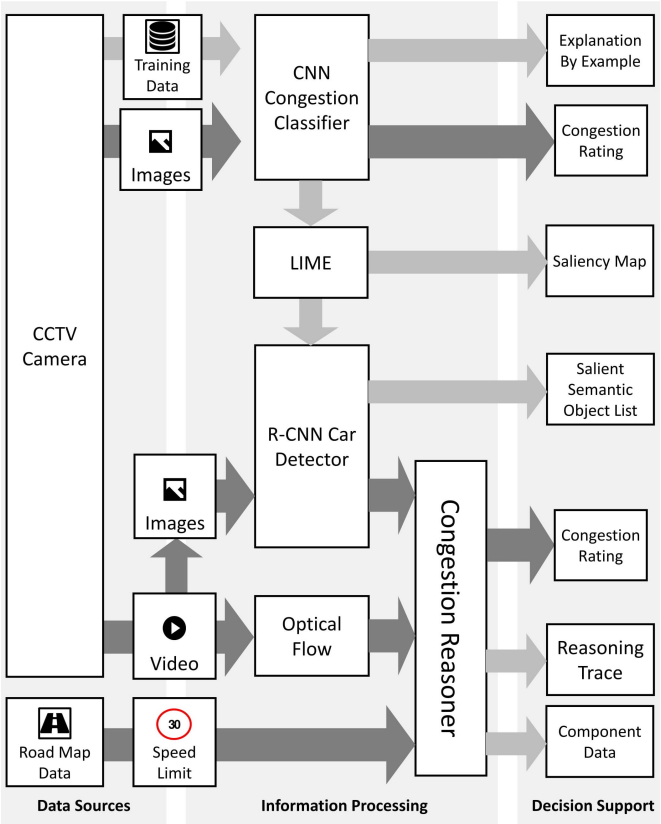

Fig. 2. Architecture for traffic monitoring case study from [19]

The example traffic monitoring system illustrated in Figure 2 was engineered to feature multiple heterogeneous methods to detect congestion. There are two main 'paths' through the system, discussed in more detail below. The first path, shown in the top part of the figure, feeds still traffic camera images to a service incorporating a convolutional neural network (CNN) trained to classify images as either *congested* or *uncongested*. The second path, shown in the bottom part of the figure, feeds video imagery to a reasoner that (after pre-processing the video to detect 'blobs in motion', and verifying that the 'blobs' are vehicles) uses heuristic-based rules to conclude whether traffic is flowing (therefore *uncongested*) or *congested*. Each path and set of services is capable of generating distinct kinds of explanation.

[1]http://www.trafficdelays.co.uk/london-traffic-cameras/
[2]http://www.openstreetmap.org/

As previously noted, coalition constraints are not the focus of this paper but the traffic monitoring problem exemplifies a multi-service environment where, plausibly, different services may be operated and controlled by different agencies (e.g., local authorities, police and other emergency services) that must share information and work in cooperation to keep the transport network running smoothly and safely.

The most important services in the architecture shown in Figure 2 are described below.[3]

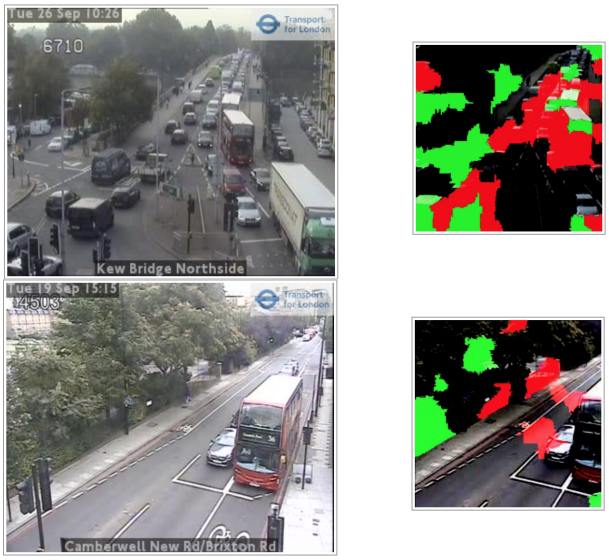

Fig. 3. Examples of *congested* (top) and *uncongested* (bottom) images with corresponding saliency maps; the lower image is misclassified as congested and the positive (red) regions of the map are falsely-positive in this case

Congestion classifier: This service comprises two CNNs. The first uses the GoogLeNet Inception network [21], pre-trained on ImageNet data, for feature extraction. The feature vector output from this network is input to a five-layer fully-connected network trained from a dataset consisting of 4,117 images labelled as *congested*, *uncongested* or *unknown* by human annotators. Example images are given on the left of Figure 3. The output from this second CNN (congestion rating) is the class conditional probability for the *congested* class. Recorded precision and recall are 0.98 and 0.96 respectively.

Explanation by example (EBE) service: This service implements the post hoc explanation-by-example technique by retrieving similar previous cases from the training set (note the training set as input to the classifier service that 'feeds' the EBE service). When an explanation is required for a runtime output, the service retrieves training examples with similar classifications (congestion ratings) and presents these alongside the runtime case. These examples indicate what the congestion classifier considers to be similar instances of the output class. (Similar in terms of class conditional probability, not in terms of visual features.)

[3]The description here is intended to be reasonably self-contained, but fuller details are provided in [19] and [20]. System code and data is available at: https://osf.io/wm3t9

LIME service: This service is based on the LIME (Local Interpretable Model-agnostic Explanations) [16] package which generates a saliency map (post hoc explanation) in the form of highlighted regions of the original input that were important in the input model's assessment of the likelihood of an output class (i.e., strong evidence for or against the class). Here, the LIME service works in conjunction with the congestion classifier to generate saliency maps for the outputs (congestion ratings) of that classifier. Examples are shown on the right of Figure 3: red regions show evidence towards *congested*, green regions show evidence towards *uncongested*. Both images are classified as *congested* but the lower one is misclassified; the saliency map confirms that the red regions here do not in fact provide evidence for *congested*.

Car detector: This is a 'utility' object detection service implemented as a retrained instance of the VGG-16 regional-convolutional neural network (R-CNN) model [22], able to identify cars in still image frames. This service is used by both the congestion reasoner and the SSO service (below).

Congestion reasoner: This service uses rules to infer the level of congestion based on traffic flow rates. Video clips from the TfL API are passed to an optical flow algorithm which produces data on 'blobs in motion'. The car detector service is then used to verify that the 'blobs' are cars. Heuristics are used to compare the pixel velocities of the car objects with the likely speed limit of the location (from the OSM API), giving a 'traffic flow ratio', and another heuristic determines at what point the flow ratio indicates low and high levels of traffic flow, allowing the level of congestion to be inferred. For this we use the rule: *if flow_ratio* $\leq$ *0.4 then congestion*, where the constant 0.4 was derived from observation of a range of London inner-city locations. The rule-based approach makes it easy to use different flow ratio thresholds for different regions of the city and, because the decision is rule-based, the service can provide an explicit trace of its reasoning as transparent explanation. Recorded precision and recall are 0.79 and 0.87 respectively.

Salient semantic object (SSO) service: A salient semantic object is an object that is detected within the highlighted positive region of a saliency map for a target class and which has a close semantic relationship to the target classIn the example, the target class is *congestion* (i.e., the positive class label for the saliency maps generated by the LIME service for the congestion classifier) and the semantically-related object is *car*. An example is shown in Figure 4. Such semantic relationships can be provided by a custom ontology or from an open knowledge base such as ConceptNet (http://conceptnet.io).

The services are integrated using the Node-RED programming environment[4] which allows them to be connected dynamically to support agile service provisioning; however, the original version of the system was composed statically [19]. The next section proposes extensions of the TBA service bundle model and associated task-bundle matching algorithm to allow dynamic selection of services that meet interpretation and robustness criteria.

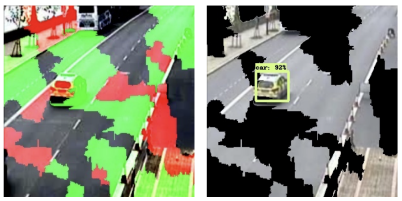

Fig. 4. A LIME saliency map (left) with an SSO (car) identified in the salient (red) region via the car detector service (right)

IV. Extended TBA Model

Details of the original matching procedure associated with the TBA model appear in [12]. Here, the TBA model is extended as follows:

- A *task* originally had a type (e.g., detect) and a target (e.g., congestion), as well as an area of interest and temporal interval; here, we extend task to also require an accuracy (e.g., expressed as precision/recall constraints) and specified kinds of interpretability and robustness (see Section V).
- An *asset* was originally defined in terms of its type and a set of capabilities that it provides, and any constraints on its deployment with other assets (e.g., dependencies, compatibilities, and coalition constraints); here, we extend the capability types to include kinds of interpretability (Section V); the asset type facilitates choosing bundles that address robustness via diversity (selecting assets that satisfy the same capability but being of different types) discussed in Section V.
- A *bundle* for a given task is a set of assets that meets all task requirements, satisfies constraints on asset deployment, and is minimal; here, the additional task requirements in terms of accuracy, interpretability and robustness must also be met in bundle provisioning.

The main procedure of the matching algorithm is given as a logic program below:[5]

$$
\begin{aligned}
&\mathit{matches}(T, B') \leftarrow \\
&\quad \mathit{requiresCapabilities}(T, C) \wedge \\
&\quad \mathit{serviceBundle}(B) \wedge \\
&\quad \mathit{providesCapabilities}(B, C) \wedge \\
&\quad \mathit{isMinimal}(B, C) \wedge \\
&\quad \mathit{satisfiesDependencies}(B, B') \wedge \\
&\quad \mathit{isConfigurable}(B').
\end{aligned}
$$

T is a task object with associated requirements. B and B' are bundles (sets of assets). Given a task instance T, the *matches* procedure produces a matching bundle B'; via backtracking, the procedure produces all matching bundles. The predicate *requiresCapabilities* identifies the set of capabilities C associated with T. The predicate *serviceBundle* generates a valid bundle B (taking account only of constraints between

[4]http://nodered.org

[5]A full Prolog implementation is provided at https://osf.io/wm3t9

assets, e.g., deployment compatibilities, not the requirements of T).[6] The predicate *providesCapabilities* returns true if the generated bundle B provides all capabilities of T. The predicate *isMinimal* returns true if B contains no assets that are redundant in providing the set of required capabilities C. The predicate *satisfiesDependencies* generates an extended version of B, B' that satisfies any dependencies of its assets on supporting assets. Finally, *isConfigurable* returns true if B' satisfies all constraints between its constituent assets.

The next section presents a worked example, applying the above model and procedure to the system described in Section III.

V. Worked Example

To apply the TBA model and matching procedure to the example traffic monitoring system, it is first necessary to define the requirements of a task. These include capabilities that a bundle must provide — including task type, accuracy and interpretability — and robustness. The task type is no different from previous work: the example focuses on the *detect* task with target *congestion* (a secondary task provided by the example system is *detect car*). We can add requirements for specific threshold levels of accuracy (e.g., precision and recall) but these do not significantly add to the previous work. More significantly, interpretability capabilities need to be defined. Based on the discussion in Section II these are:

- *transparent explanation* (a super-type)
- *post hoc explanation* (a super-type)
- *reasoning trace* (a sub-type of *transparent explanation*)
- *saliency map* (a sub-type of *post hoc explanation*)
- *explanation by example* (a sub-type of *post hoc explanation*)
- *SSO explanation* (a sub-type of *post hoc explanation*)

Relationships between assets and capabilities are defined by the *providesCapability* predicate used internally by the *requiresCapabilities* predicate in the matching procedure. For the example system these are:

providesCapability('congestion classifier', 'detect congestion').
providesCapability('EBE service', 'explanation by example').
providesCapability('LIME service', 'saliency map').
providesCapability('car detector', 'detect car').
providesCapability('congestion reasoner', 'detect congestion').
providesCapability('congestion reasoner', 'reasoning trace').
providesCapability('SSO service', 'SSO explanation').

Sub-type/super-type relationships between capabilities are handled in the matching procedure by means of an *entailsCapability* predicate:

entailsCapability('saliency map', 'post hoc explanation').
entailsCapability('reasoning trace', 'transparent explanation').
entailsCapability('explanation by example',
'post hoc explanation').
entailsCapability('SSO explanation', 'post hoc explanation').

These definitions allow the matching procedure to infer:

$$providesCapability(A, C) \leftarrow providesCapability(A, C') \wedge entailsCapability(C', C).$$

Thus, for example, *providesCapability('LIME service', 'post hoc explanation')* and *providesCapability('SSO service', 'post hoc explanation')* are inferred facts.

The *satisfiesDependencies* predicate in the matching procedure uses the following dependencies for the example system:

dependsOn('EBE service', 'congestion classifier').
dependsOn('LIME service', 'congestion classifier').
dependsOn('SSO service', 'LIME service').
dependsOn('SSO service', 'car detector').
dependsOn('congestion reasoner', 'car detector').

With these declarations, the matching procedure is now able to provide bundles for a variety of task requirements, as illustrated by the examples in Table I. Note that the requirements do not include accuracy for the *detect congestion* task since requiring particular thresholds for precision and/or recall would simply maintain or reduce the available congestion classifier or congestion reasoner service options.

The requirement for robustness in terms of service diversity needs to be handled above the level of bundle generation since it requires multiple bundles to be assigned to a task such that diversity constraints are satisfied by the combination of bundles. Service diversity for a task T with a given type and target (e.g., *detect congestion*) requires that there be at least two bundles such that the member services that provide the capability for the type and target of T are based on different models. For this, it is necessary to provide a categorisation of the asset type for AI/ML-based services, for example:

subClassOf('congestion classifier', 'CNN-based service').
subClassOf('congestion reasoner', 'rule-based service').

The original implementation of the matching system described in [11] computed bundles for all task type/target permutations and showed how, for a reasonably-comprehensive intelligence, surveillance and reconnaissance domain, a complete set of task-bundle combinations could be stored in a lookup table on a standard mobile device. Therefore, the problem of selecting multiple bundles for the same task type/target to satisfy a diversity requirement is a relatively straightforward extension.

VI. Discussion and Conclusion

The main contribution of this paper has been to consider the problem of service bundle provision with additional requirements for interpretability and robustness pertinent to AI/ML-based services. For this, it was necessary to also propose an initial typology of interpretability requirements based on [8].The approach was implemented by extending a previous model and procedure for task-asset provisioning with the additional requirements for interpretability and robustness. A pre-existing system for traffic monitoring was taken as a case study to illustrate the approach.

There is significantly more work to be done to extend the typology of interpretability types and, indeed, development

[6]Note that this formulation of the matching procedure is not intended to be efficient, merely descriptive.

Requirements	Bundle
{ *'detect congestion', 'saliency map'* }	{ *'congestion classifier', 'LIME service'* }
{ *'detect congestion', 'explanation by example'* }	{ *'congestion classifier', 'EBE service'* }
{ *'detect congestion', 'sso explanation'* }	{ *'congestion classifier', 'SSO service', 'LIME service', 'car detector'* }
{ *'detect congestion', 'transparent explanation'* }	{ *'congestion reasoner', 'car detector'* }

TABLE I
BUNDLES GENERATED BY THE *matches* PROCEDURE FOR SAMPLE TASK REQUIREMENTS

of standard terminology in this area is a key requirement going forward. However, our immediate focus is to examine the relationship between interpretability types and user roles [23]. To support experimentation in this area we are creating an open source testbed of interpretability services and ML classifiers trained on multiple datasets. The testbed will allow us to explore trade-offs in ML service performance and interpretability types, and also enable benchmarking of the computational resource utilisation of interpretability services. Ultimately, it might be feasible to optimise the selection of ML and interpretability services in terms of utility functions taking account of service performance and computational resource utilisation.

Extension of the above approach to handling robustness criteria is also desirable. The service type could be extended to indicate the scope, modality, and provenance of training data, for example, or model parameters. While task requirements could be made arbitrarily specific in this way, it is debatable how fine-grained this needs to be since such requirements — like precision/recall — will tend to reduce the set of available services and therefore restrict the amount of information available to a task. Arguably it is better to collect more information where available, and employ a 'reasoning layer' to handle uncertainty where multiple services disagree in cases of diversity; for some initial ideas and discussion in this direction, see [24].

The approach to asset description using an ontology involves effort in knowledge representation. It was argued in previous work that the amount effort required is low because the range of physical asset types is relatively static; new kinds of sensing system are rarely introduced and persist for many mission runs. However, software assets, being virtual entities, may be introduced far more dynamically, especially for ML-based services. Therefore, future work will address using ML to learn asset descriptions (i.e., TBA model instances).

REFERENCES

[1] T. Pham, G. Cirincione, D. Verma, and G. Pearson, "Intelligence, surveillance, and reconnaisance fusion for coalition operations," in *Proc 11th International Conference on Information Fusion*, 2008.

[2] D. S. Alberts and R. E. Hayes, *Power to the Edge: Command and Control in the Information Age*. CCRP, 2003.

[3] D. S. Alberts, R. K. Huber, and J. Moffat, *NATO NEC C2 Maturity Model*. CCRP, 2010.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[5] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building watson: An overview of the deepqa project," *AI Magazine*, vol. 31, no. 1, pp. 59–79, 2010.

[6] A. Preece, F. Cerutti, D. Braines, S. Chakraborty, and M. Srivastava, "Cognitive computing for coalition situational understanding," in *First International Workshop on Distributed Analytics InfraStructure and Algorithms for Multi-Organization Federations*, 2017.

[7] T. G. Dietterich, "Steps toward robust artificial intelligence," *AI Magazine*, vol. 38, no. 3, pp. 3–24, 2017.

[8] Z. C. Lipton, "The mythos of model interpretability," in *2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*, 2017, pp. 96–100.

[9] A. S. Ross, M. C. Hughes, and F. Doshi-Velez, "Right for the right reasons: Training differentiable models by constraining their explanations," *arXiv preprint arXiv:1703.03717*, 2017.

[10] W3C, "Semantic sensor network ontology," World Wide Web Consortium, Oct. 2017. [Online]. Available: https://www.w3.org/TR/vocab-ssn/

[11] D. Pizzocaro, A. Preece, F. Chen, T. L. Porta, and A. Bar-Noy, "A distributed architecture for heterogeneous multi sensor-task allocation," in *Proc 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'11)*, 2011.

[12] A. Preece, T. Norman, G. de Mel, D. Pizzocaro, M. Sensoy, and T. Pham, "Agilely assigning sensing assets to mission tasks in a coalition context," *IEEE Intelligent Systems*, vol. Jan/Feb, pp. 57–63, 2013.

[13] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[14] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018, 10.23915/distill.00010.

[15] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2016.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. ACM, 2016, pp. 1135–1144.

[17] R. Caruana, H. Kangarloo, J. Dionisio, U. Sinha, and D. Johnson, "Case-based explanation of non-case-based learning methods," in *Proceedings of the AMIA Symposium*, 1999, pp. 212–215.

[18] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating visual explanations," in *European Conference on Computer Vision (ECCV 2016)*. Springer, 2016, pp. 3–19.

[19] D. Harborne, C. Willis, R. Tomsett, and A. Preece, "Integrating learning and reasoning services for explainable information fusion," in *First International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI 2018)*, 2018.

[20] C. Willis, D. Harborne, R. Tomsett, and M. Alzantot, "A deep convolutional network for traffic congestion classification," in *Proc NATO IST-158/RSM-010 Specialists' Meeting on Content Based Real-Time Analytics of Multi-Media Streams*. NATO, 2017.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[23] R. Tomsett, D. Braines, D. Harborne, A. Preece, and S. Chakraborty, "Interpretable to whom? A role-based model for analyzing interpretable machine learning systems," in *2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*, 2018.

[24] F. Cerutti, M. Alzantot, T. Xing, D. Harborne, J. Z. Bakdash, D. Braines, S. Chakraborty, L. Kaplan, A. Kimmig, A. Preece, R. Raghavendra, M. Şensoy, and M. Srivastava, "Learning and reasoning in complex coalition information environments: a critical analysis," in *21st IEEE International Conference on Information Fusion*, 2018.