Towards a Neural-Symbolic Generative Policy Model

Daniel Cunnington*, Mark Law†, Alessandra Russo†, Elisa Bertino‡ and Seraphin Calo§
*IBM Research, Winchester, UK. Email: dancunnington@uk.ibm.com
†Imperial College, London, UK. Email: {mark.law09,a.russo}@imperial.ac.uk
‡Purdue University, West Lafayette, IN, USA. Email: bertino@purdue.edu
§IBM Research, Yorktown Heights, NY, USA. Email: scalo@us.ibm.com

***Abstract*—To facilitate information sharing between systems and devices in a distributed environment, unstructured data from various sensors must be analysed accordingly. Recent work has developed the notion of a context-dependant generative policy framework capable of learning generative policy models from strings and text-based data in a tabular format. However, it is vital that unstructured contextual information can be analysed alongside tabular data, potentially at the edge of the network to enable generative policy models to be applied to more complex tasks. This paper performs a deep-dive into the field of neural-symbolic machine learning with a view towards enabling future neural-symbolic generative policy models that are capable of analysing both structured and unstructured data, whilst providing full transparency and enabling edge of network reasoning capability. Firstly, an existing technique called *DeepProbLog* is investigated and applied to a policy *inferencing* task based on unstructured data and secondly, a recent inductive logic programming technique currently used for learning generative policy models is evaluated with respect to a policy *learning* task also based on unstructured data. Finally, the results of both tasks are discussed to provide a platform for future research into enabling neural-symbolic generative policy models.**

***Index Terms*—Distributed Analytics, Information Science, Generative Policy Models, Neural-Symbolic Learning**

I. Introduction

Generative policy models [1] have been proposed as a means of autonomously managing the interactions and behaviours between systems and devices within distributed environments across varying contexts. Recent work has developed a formal framework for learning context-dependant generative policy models [2] using Answer Set Grammars (ASGs) [3] and Inductive Learning of Answer Set Programs (ILASP) [4] with applications in autonomous vehicles [5], logistical re-supply [6], access control [7] and information sharing [8].

Developing a generative policy model involves learning the policies to be obeyed in a given scenario across a variety of different contexts. This can be achieved through continuous observation of policy examples over a period of time, so that policies can be learned and refined according to contextual changes. Policy examples can be used across systems and devices in a distributed environment to collaboratively learn a generative policy model. Once a suitable generative policy model has been learned, it can be used to generate new policies within any of the observed contexts. The generative policy model can be represented in a number of different forms such as a list of logical rules or a machine learning model.

In all of the applications explored to-date, generative policy models have been learned from string and tabular based training examples. However, to extend and utilise generative policy models for more complex tasks it is crucial to also analyse and learn from unstructured data to gain a better understanding of the context in which devices operate. A hybrid neural-symbolic learning approach for learning generative policy models could match the required explainable properties of symbolic learning with the powerful performance of neural models for analysing unstructured data.

This paper performs a deep-dive into the field of neural-symbolic machine learning with three main contributions. Firstly, an initial experiment is performed to evaluate the suitability of an existing technique called *DeepProbLog* [9] for combining neural learning of contextual features from unstructured data together with symbolic-based inference of policy decisions. DeepProbLog's performance is evaluated with respect to an end-end neural baseline for a logistical resupply policy *inferencing* task.

Secondly, ILASP is applied to a more complex policy *learning* task that requires the learning of both symbolic rules and unstructured, contextual features using advanced neural architectures such as object detection models. Policy rules are learned symbolically using the predicted output from the neural components, additional structured information and a ground truth policy annotation, where confidence scores resulting from neural softmax layers are used to weight examples in the ILASP learning process. Prior to the symbolic rule learning, the neural components learn to detect and classify contextual features within unstructured data. The symbolic and neural components are trained independently in a modular, *hybrid* fashion, with no back-propagation between components. The performance of the hybrid neural-ILASP approach is compared to other hybrid learning approaches and an end-to-end neural baseline. The hybrid neural-ILASP approach outperforms other learning approaches whilst retaining explainability over the learned logical rules.

Finally, we summarise the results of both tasks and suggest a platform for a wider body of research into the field of neural-symbolic learning of generative policy models, which aims at developing an *end-to-end* learning system that is capable of integrating both neural and symbolic learning components to learn symbolic policy rules whilst simultaneously training neural networks to classify contextual features from unstructured data.

The paper is structured as follows. In section II we summarise some of the key state-of-the-art work in the field of neural-symbolic learning, in section III we define our two policy learning tasks, in section IV we outline our experimental results for each task and in section V we discuss our results and provide a direction for future work. Finally, we conclude in section VI.

II. Related Work

The field of neural-symbolic learning has made significant progress in the last decade. Among the initial contributions are the work of Garcez et al. [10] and Hammer and Hitzler [11], summarised, together with related approaches, in Besold et al. [12]. Among these, *CILP* [13] uses a neural network to approximate logical rules such that inductive learning can be used to refine logic-based background knowledge with examples. The emphasis is on using neural computation to improve symbolic knowledge.

More recently, Hu et al., [14] distill logical rules into the weights of a neural network to assist the neural network in learning challenging aspects of a classification task from unstructured text, such as contrastive sentences in sentiment analysis. The emphasis here is to use symbolic knowledge to improve a neural learning task. Manhaeve et al. [9] introduce a probabilistic logic programming language, called *DeepProbLog*, that integrates neural networks with probabilistic logic programming modelling and reasoning. This extends *ProbLog* [15] to support the learning of neural objects from unstructured data such as images by guiding the neural classification process through probabilistic logic-based inference. This integrated neural-symbolic model, with given fixed probabilistic logic-based rules is trained end-to-end from examples of labelled consequences that the logic program is modelled to infer. DeepProbLog does not perform learning of symbolic rules and has not been applied to more advanced neural architectures such as object detection models that require learning to classify an object's type as well as identifying it's location within an image.

In this paper, we evaluate the suitability of DeepProbLog for combining the training and classification of two neural models together with a probabilistic logic-based program and, to the best of our knowledge, perform a first experimental step combining ILASP with neural networks to learn both symbolic rules and contextual features from unstructured data, in a modular, hybrid fashion.

III. Illustrative Scenario

Let us outline an illustrative scenario to motivate the need for future neural-symbolic generative policy models and define the policy inferencing and learning tasks used in this paper.

A. Logistical Resupply Scenario

Consider a logistical resupply mission [16] where a resupply convoy must reach a particular destination, navigating through areas of adversarial activity. Let us assume the resupply convoy does not have communication to high performance back-end computing facilities although does have access to local sensing infrastructure such as camera networks and weather sensors. The resupply convoy must autonomously decide various actions to take at various points within the mission, in order to mitigate the risk of adversarial compromise and reach the destination successfully.

B. Policy Inferencing Task

For the *policy inferencing* task, we assume the resupply convoy has access to a local CCTV camera network that contains a neural model capable of classifying the presence of an enemy vehicle from the unstructured CCTV image (either *present*, *not_present*, or *no_data_available*—which occurs if the data provider chooses to redact a particular camera image). The resupply convoy also has access to a team of ground troops that may be *available* or *unavailable* at a given time. We also assume that the resupply convoy can take the following actions: *a)* ***deploy_ground_troops****. Deploy the ground troops to handle enemy activity. b)* ***re_route_convoy****. Modify the convoy's route to avoid enemy activity.* Or, *c)* ***proceed_as_normal****. Proceed with the currently selected route*. The resupply convoy in this task behaves according to the following policy rules:

```
IF enemy_vehicle == 'present' AND
   ground_troops == 'available' THEN
   deploy_ground_troops.

IF (enemy_vehicle == 'present' AND
   ground_troops == 'unavailable') OR
   enemy_vehicle == 'no_data_available' THEN
   re_route_convoy.

IF enemy_vehicle == 'not_present' THEN
   proceed_as_normal.
```

Listing 1: Ground truth rules for the policy inferencing task

The goal of this task is to *infer* the correct policy action based on the classification of images within neural components using DeepProbLog [9]—where neural components are trained with respect to the resulting policy action as oppose to a raw image annotation. This method enables a human operator to rapidly construct integrated neural-symbolic learning and reasoning components by specifying probabilistic logical rules that can be used to train multiple deep neural network components in one training session.

C. Policy Learning Task

As an extension to the policy inferencing task, we also explore a *policy learning* task, where symbolic rules can be learned by observation of policy examples, removing the need for a human operator to specify the rules up-front.

We assume the resupply convoy has access to a local CCTV camera network and information regarding ground troop availability as in the policy inferencing task as well as some additional information which includes an estimate of the enemy's capability (*low*, *medium* or *dangerous*) and current visibility conditions (an integer between 0 and 10 inclusive). We also assume that in the policy learning task, an additional action is available to the resupply convoy, which is ***deploy_UAV***. This action deploys an Unmanned Aerial Vehicle (UAV) to further investigate the status of enemy vehicles, provided there is sufficient visibility (e.g. no smoke present or heavy fog).

The resupply convoy in this task behaves according to the following policy rules:

```
IF enemy_vehicle == 'present' AND
    enemy_capability == 'low' AND
    ground_troops == 'available' THEN
    deploy_ground_troops.

IF enemy_vehicle == 'present' AND
    ground_troops == 'unavailable' THEN
    re_route_convoy.

IF enemy_vehicle == 'no_data_available' AND
    visibility > 5 THEN deploy_UAV.

IF enemy_vehicle == 'not_present' THEN
    proceed_as_normal.

ELSE re_route_convoy.
```

Listing 2: Ground truth rules for the policy learning task

The goal of this task is therefore to *learn* the policy rules as well as training neural components to detect and classify contextual information. This task adopts a modular, hybrid approach, where in order to learn the symbolic rules, a forward pass is performed over pre-trained neural components which then feeds the downstream ILASP symbolic learner [4] to learn the symbolic rules based on the predictions of the neural models, additional structured information and a ground-truth label.

IV. Experimental Approach and Results

In this section we outline our experimental approach and present our results for each task.

A. Policy Inferencing Task

1) Approach: For this task there are two neural components integrated with a symbolic component. Firstly a neural model is constructed to analyse CCTV data and determine the status of an enemy vehicle and secondly, a neural model is constructed to represent a decision regarding ground troop availability. The neural components are integrated with fixed, hard-coded policy rules in the form of a DeepProbLog program.

To determine enemy vehicle status, we utilise a Convoloutional Neural Network (CNN) pre-trained on the ResNet18 architecture[1] and adopt a transfer learning approach, freezing the weights of all layers except the final fully connected layer which remains trainable. To simulate a CCTV network within a logistical resupply mission, we use the Transport for London (TfL) CCTV dataset[2] and consider red buses to represent enemy vehicles, an example of which is shown in Figure 1. The enemy vehicle CNN outputs 3 classes (*present*, *not_present*, or *no_data_available*) to align with the scenario as discussed in section III.

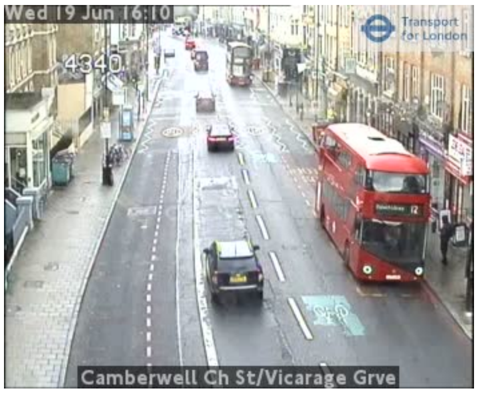


Fig. 1: Sample CCTV image indicating the presence of an enemy vehicle, represented as a red London bus from the TfL dataset[2].

To determine the status of ground troop availability we utilise the MNIST[3] dataset to build a binary CNN classifier, assuming for the logistical resupply scenario that the MNIST digit 0 represents *available* and the MNIST digit 1 represents *unavailable*. For this model, we adopt the same CNN architecture that is used in the DeepProbLog MNIST experiments [9].

Using the DeepProbLog data generation method [9], we generate $10,000$ random training examples and 500 random test examples, where an example consists of a CCTV image and an MNIST image annotated with a resulting policy action of the form $mission\_step(cctv\_image, mnist\_image, policy\_action)$. The logical rules outlined in Listing 1 are formalised in DeepProbLog and we present the full DeepProbLog program for the policy inferencing task in Appendix A. The goal is to train the neural CNN components whilst inferring the `Result` of the `mission_step/3` predicate as listed in Appendix A. We implement this task in PyTorch, extending the DeepProbLog framework[4].

2) Results: Figure 2 details the results of this task, comparing the accuracy of the DeepProbLog approach with a baseline neural approach over $30,000$ training iterations. We use the same dataset for both approaches and take the average accuracy at each number of training iterations over 5 repeats to account for random weight initialisation in the neural components. Error bars indicating the standard deviation at each number of iterations are shown in Figure 2. For the baseline, we adopt a similar approach to the DeepProbLog experiments [9] and concatenate the CCTV images with MNIST images to train a CNN to classify the resulting *mission_step* policy action from the concatenation of the two images.

With the addition of the logical rules, the DeepProbLog approach significantly outperforms the baseline after $30,000$

[1]https://www.kaggle.com/pytorch/resnet18
[2]https://data.london.gov.uk/dataset/tfl-live-traffic-cameras
[3]http://yann.lecun.com/exdb/mnist/
[4]https://bitbucket.org/problog/deepproblog/src

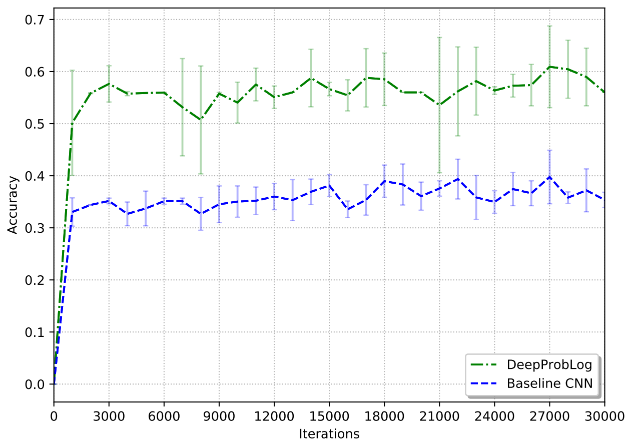


Fig. 2: Policy inferencing task experimental results

iterations. Nevertheless, the accuracy of this policy inferencing task is far below acceptable levels for real-world application. This is due to the challenging nature of identifying red buses in TfLs CCTV cameras. We substantiated this by running a second experiment in which the CCTV CNN was replaced with a 3-digit MNIST CNN. After only $1,000$ iterations the trained model achieved near 100% accuracy.

Analysing the results in Figure 2, it is apparent that the DeepProbLog approach exhibits more variance in it's performance when compared to the baseline CNN, especially at $21,000$ iterations. We hypothesise that this is due to the different random weight initialisations in the neural components in comparison to the baseline. In the DeepProbLog approach both the CCTV CNN *and* the MNIST CNN are subject to random weight initialisation, whereas in the baseline approach there is only one random weight initialisation.

Also, we note that a key limitation of the DeepProbLog approach and the policy inferencing task is that logical rules require manual specification from humans and therefore are unsuitable for direct integration into generative policy model architectures. To address this limitation, in the following task we take a first step towards *learning* the policy rules as well as training the neural components in a modular, hybrid fashion, with a view to obtaining similar performance benefits as exhibited in the DeepProbLog approach used in this task.

B. Policy Learning Task

1) Approach: For this task there are also two neural components and a symbolic component, except the neural and symbolic components are combined in a modular, hybrid fashion as oppose to being closely integrated. Following the policy inferencing task, we analyse CCTV camera images using a neural component although simplify ground troop availability to a Boolean *available/unavailable* decision. We assume the decision could be input by a human, or obtained from a database, as oppose to requiring calculation using a neural model. The purpose of this is to represent structured information that may be available to the resupply convoy alongside unstructured information present in the environment. We also supplement the structured information by adding a visibility feature assumed to be given by an appropriate weather sensor. The structured information can be given directly to the symbolic component with no requirement for any neural calculations. For this task, we analyse a new feature, enemy capability, using a neural component and learn the policy rules symbolically.

To account for uncertainty in the neural components, confidence scores (i.e. as output by a neural softmax layer) are also input to the symbolic component to accompany neural classifications. This enables the symbolic component to apply a weight to the neural classifications when learning resulting policy rules.

To address the low performance with the CNN used for enemy vehicle detection in the policy inferencing task and to explore the use of more advanced neural models, a "single shot" object detector [17] is trained using the TensorFlow object detection framework [18], using a transfer learning approach with neural weights initialised from a pre-trained Inception V2 model [19] on the Common Objects in Context dataset[5]. An example image from the TfL CCTV dataset[2] after passing through the enemy vehicle object detector is shown in Figure 3. We note that in the policy inferencing task it was not possible to integrate object detection based neural models using the current DeepProbLog framework[6].

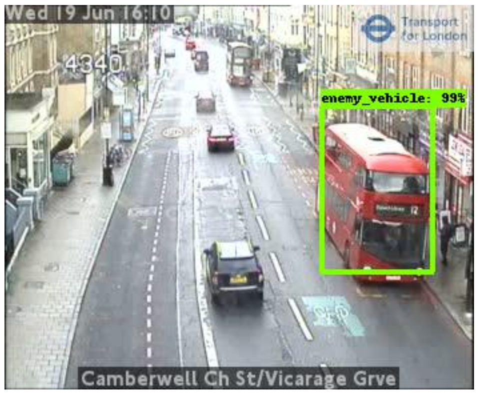


Fig. 3: Sample CCTV image from the TfL dataset[2] showing a red London bus annotated as an enemy vehicle, as output from the object detector used in the policy learning task.

In order to account for the potential *no_data_available* scenario with a TfL CCTV camera image, an example of which is shown in Figure 4, a statistical colour histogram classification approach is applied to the image before the object detection step. If the most dominant colour is determined to be RGB (128,128,128) then the resulting classification is

[5]http://cocodataset.org/

[6]This is due to the requirement to calculate the loss function with respect to the classification *and* the localisation of an object within a particular image. In the DeepProbLog approach, the loss function is calculated with respect to the resulting policy action and therefore it was not possible to include object localisation information. Future work should explore extending the DeepProbLog framework to accommodate more advanced neural models.

no_data_available. Otherwise the image is passed to the object detector to locate and classify the potential presence of red buses (enemy vehicles).

Fig. 4: Sample CCTV image from the TfL dataset[2] indicating no data is available. This may occur if TfL decide to redact a particular camera image.

To simulate the estimation of enemy capability, a neural network is trained on the MNIST dataset where the MNIST digit 0 represents *low*, the MNIST digit 1 represents *medium* and the MNIST digit 2 represents *dangerous* enemy capability respectively. We utilise the same neural network architecture used in the policy inferencing task for classifying MNIST digits in this task. For the symbolic learning component, we use the ILASP system version 3 [4] that is currently the preferred tool for learning symbolic, rule-based generative policy models [2] using inductive logic programming.

In order to structure our policy learning task, we adopt a careful dataset preparation approach as follows. Firstly, to prepare the enemy vehicle data from the TfL CCTV camera dataset[2], we annotate 429 images with bounding boxes indicating the presence of an enemy vehicle (represented by a red bus). 880 images are manually selected where no enemy vehicles are present and 133 images are manually selected where no data is available to create a dataset of 1442 CCTV camera images. From this dataset, we set aside 30% of each class at random for training and testing the symbolic component and utilise the remaining 70% of each class for training and testing the enemy vehicle detector. This ensures the distribution of images between classes is reflected in the respective datasets following the split for the object detector and the symbolic component. For both the object detector split and the symbolic component split, we further split each class at random into separate training and test sets containing 80% and 20% of the data respectively. This therefore leaves 240 *present*, 492 *not_present* and 74 *no_data_available* training examples as well as 60 *present*, 124 *not_present* and 19 *no_data_available* test examples for training and testing the enemy vehicle object detector. Note, the training examples for *not_present* and *no_data_available* are discarded as they are not required for training the object detector. We do not move these examples to the test set to avoid further skewing the class distribution. Once the enemy vehicle detector is trained and tested, the symbolic learning component receives 103 *present*, 212 *not_present* and 32 *no_data_available* training examples as well as 26 *present*, 52 *not_present* and 8 *no_data_available* test examples representing the enemy vehicle feature, originating from the initial 30% split of the entire dataset.

To prepare the enemy capability data, we first filter the original MNIST training and test datasets[2] and remove all images except those that represent digits 0, 1 and 2 to simulate the *low*, *medium* and *dangerous* classes of enemy capability respectively. We group the filtered MNIST datasets into digit classes and extract a total of 347 random training images and 86 random test images, maintaining an even distribution between digit classes. This data preparation step ensures the number of training and test examples for enemy capability matches the number of training and test examples obtained from the enemy vehicle dataset, such that the datasets can be merged for training and testing the symbolic learning component. The remaining images within the filtered MNIST datasets are used for training and testing the MNIST enemy capability neural network with 18723 training and 3058 test examples respectively. For each feature containing structured information (i.e. ground troop availability and visibility), training and test sets of length 347 and 86 are generated by enumerating each possible feature value until the desired dataset length has been reached.

Finally, training and test policy datasets are constructed by merging all the training features together and all the test features together, such that each row in the policy dataset contains a CCTV image representing an enemy vehicle, an MNIST image representing enemy capability, ground troop availability information and a visibility score. For each row in the policy dataset, a forward pass is performed over each of the neural components with their respective images, to obtain a prediction and a confidence score for the presence of enemy vehicles or classification of enemy capability in each image. The CCTV and MNIST images are then replaced with their respective neural predictions and confidence scores in the policy dataset. Furthermore, for the purposes of training the symbolic learning component, each row in the policy dataset is labelled with a ground truth policy action by evaluating the policy rules outlined in Listing 2 with respect to a *ground truth label* for the enemy vehicle and enemy capability images (instead of the *predicted* label from the neural components). This ensures each row is labelled with the correct policy action, regardless of the predicted class for the enemy vehicle and enemy capability image the forward pass over each neural component returns. Including the confidence score of the predicted class from each neural component enables the symbolic learner to weight certain examples, taking into account uncertainty and incorrect predictions when learning policy rules. To calculate the weight for the symbolic learner, the confidence scores of each neural component are multiplied together. The symbolic learner can then be evaluated with respect to the labelled ground truth policy action.

The goal of this task is therefore to train the neural components and also train the symbolic ILASP component to learn policy rules depending on the prediction and confidence of the neural components as well as structured information input directly to the symbolic ILASP component.

2) Results: Figure 5a details the average test set accuracy for each neural component over 5 repeats for $3,000$ training iterations. For the MNIST CNN, models are saved every 100 iterations and evaluated on the MNIST CNN test set. For the enemy vehicle object detector, neural models are saved every 40 iterations for the first 120 iterations, every 200 iterations from 300 to 1100 iterations and every 500 iterations thereafter. Each model is evaluated on the object detector test set. The standard deviation between repeats is indicated with error bars at each number of iterations. Following the policy inferencing task, 5 repeats are performed to account for random neural weight initialisation. The enemy capability MNIST CNN demonstrates strong performance as expected and the enemy vehicle object detector appears to outperform the enemy vehicle CNN used in the policy inferencing task, achieving over 80% accuracy compared to 58% accuracy after $3,000$ training iterations[7]. This is attributed to the addition of an object detection neural network model to locate enemy vehicles alongside the colour dominance classification approach to detect *no_data_available* images.

Figure 5b details the results of the policy learning task, comparing the performance of ILASP to a Random Forest (RF), a Fully Connected Neural Network (FCN) and a baseline end-to-end CNN for learning the policy rules outlined in Listing 2. The accuracy results shown in Figure 5b represent the performance over the policy test set, where ILASP, the RF and the FCN predict the resulting policy action based on predictions from each saved neural model alongside structured ground troop availability and visibility information. The policy action predictions are compared to the ground truth policy action to generate an accuracy score. For the baseline CNN, policy actions are learned in an end-to-end fashion, where for each row in the policy dataset, a CCTV image, an MNIST image and one-hot encoded representations of the structured information are concatenated to form one input to the network. The baseline is trained with respect to the ground truth policy action label. This follows the baseline approach used in the DeepProbLog policy inferencing task.

For the RF, we use 100 trees within the ensemble, for the FCN we use 3 hidden layers with 64 neurons in each layer and use the Adam optimiser to optimise the sparse categorical cross-entropy loss. L2 regularisation is used with penalty 0.1 and the network is trained with training examples in batches of 12. For the baseline approach, we use the same CNN architecture used to learn enemy vehicle classifications in the policy inferencing task, with pre-trained weights on the ResNet18 architecture available in PyTorch[1]. The baseline CNN also uses the Adam optimiser to optimise the sparse categorical cross-entropy loss and is trained with training examples in batches of 12.

Given the stable performance of the MNIST CNN, we use the same MNIST model to predict enemy capability paired with a different neural model at each number of iterations for the enemy vehicle object detector. As shown in Figure 5b, ILASP outperforms the other learning algorithms for each pair of neural models at each number of iterations. Also, ILASP appears to account for poor neural performance in the object detection model in earlier training iterations, (e.g. at 1600 iterations), which we hypothesise is due to the weighting of examples used during the ILASP learning process based on the confidence of the neural object detector.

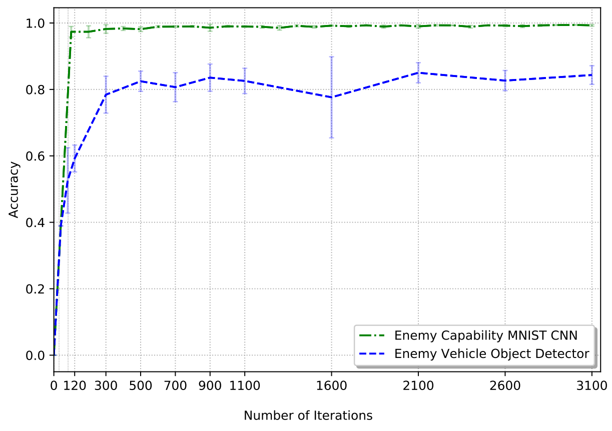
Results for ILASP are absent for iterations below 300 due to an increased amount of noise resulting from poor predictive performance in the neural components in early training iterations. In such cases, there are a large number of potential hypotheses (i.e. combination of potential policy rules) that cover completely different sets of training examples which are of similar size. As ILASP searches for an optimal solution by maximising rule coverage over the training examples, ILASP takes a long time to converge to the optimal hypothesis. One solution to this problem would be to reduce the search space of the task, although this is not performed in order to provide a fair comparison to other learning algorithms.

The ILASP results displayed in Figure 5b do not take into account the confidence of the neural predictions when evaluating test set performance, as the confidence scores are used as a weight within the ILASP learning process as oppose to a feature. To account for this and in an attempt to increase the ILASP performance, an additional experiment was performed to utilise ProbLog [15] to reason over the neural predictions using the neural confidence scores with respect to the learned ILASP hypothesis at each number of iterations.

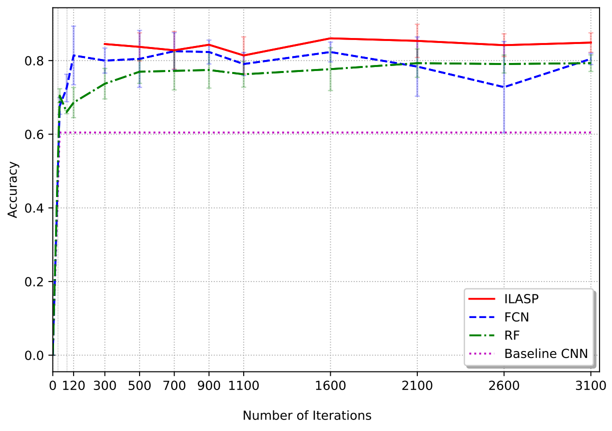
Figure 6 details the results of using the ProbLog reasoning approach for the policy learning task, with two sets of results presented. Firstly, the raw ILASP approach without ProbLog (as presented in Figure 5b) is presented alongside the test set performance obtained when swapping the raw ILASP hypothesis for the ground truth hypothesis. This evaluates whether the policy rules learned by ILASP represent the ground truth rules in Listing 2 for classifying the correct policy action on the policy test set. Secondly, the ProbLog results are presented, where the confidence scores output from the neural components are used to form a ProbLog fact with a given probability. The ILASP hypothesis in the form of policy rules is implemented in ProbLog, alongside the ground truth hypothesis to evaluate the performance of the learned ILASP hypothesis.

In Figure 6 it is clear that ILASP demonstrates very strong performance and almost perfectly matches the ground truth hypothesis with and without ProbLog reasoning. However, using ProbLog to reason over the neural confidence scores appears to decrease the overall performance of predicting a policy action on the policy test set. We attribute this to two possible causes. Firstly, the probability for each outcome of the

[7]In the policy inferencing task, policy decisions are inferred from the enemy vehicle CNN predictions and the ground troop availability MNIST CNN which achieved near 100% accuracy, so we assume the accuracy results in Figure 2 closely match the performance of the enemy vehicle CNN sub-component.



(a) Test set accuracy of each neural component



(b) Overall test set accuracy on the policy dataset

Fig. 5: Policy learning task results

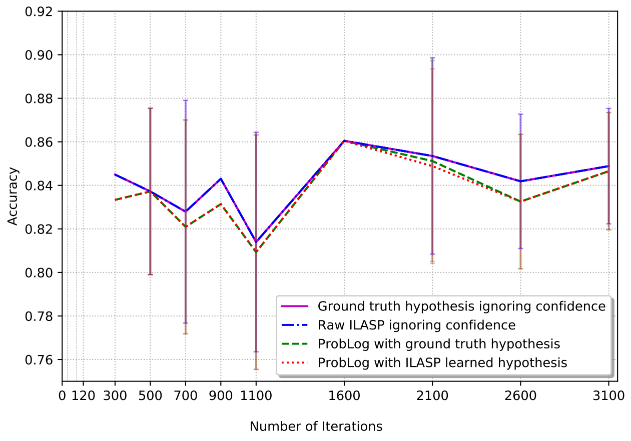


Fig. 6: ProbLog reasoning over learned ILASP hypotheses taking into account neural prediction confidence scores.

enemy vehicle object detector does not necessarily sum to 1. The confidence score returned for an object detection indicates the confidence of the object detected in a given bounding box belonging to a particular class. If the object detector classifies a bounding box as containing an enemy vehicle with probability 0.6, it doesn't necessarily mean that there isn't an enemy vehicle present in the entire image with probability 0.4. Also, recent work has suggested the confidence scores output from neural networks may not be fully accurate. Nguyen et al. demonstrate obtaining high confidence predictions from neural networks trained to classify images containing lions, when images representing white noise are passed to the network [20]. Future work should address this limitation in deep neural networks by investigating more reliable measures of confidence for neural network predictions.

V. Discussion and Future Work

Given the experimentation conducted and the results of both the policy inferencing and policy learning tasks, let us now outline some directions for future work.

Firstly, in the policy inferencing task as detailed in Figure 2, it is apparent that the DeepProbLog approach exhibits more variance between training runs than the baseline CNN. As discussed, we hypothesise that this is due to random weight initialisation within each neural sub-component. This highlights a potential disadvantage for the DeepProbLog approach, especially when considering the logistical resupply scenario presented in Section III. Learning consistency in this task is important, as it may not be possible in terms of time to conduct many repeated training runs to select a high performing model. Also, given the vast number of potential contextual features, the number of different neural components could be large. Future work should investigate increasing the number of neural components to observe the effect on performance consistency. Also, as evidenced in Figure 5a in the policy learning task, a hybrid approach enables each component to be easily evaluated independently. Future work could investigate such an approach within the DeepProbLog framework, to increase interpretability and help diagnose poor performing DeepProbLog models.

In the policy learning task, ILASP, a recent symbolic learning technique based on inductive logic programming demonstrates promising performance when compared to other learning algorithms whilst retaining full explainability. This enables probabilistic reasoning over learned rules as demonstrated in Figure 6.

In order to realise the vision of a neural-symbolic generative policy model based on ILASP, future work should firstly ensure ILASP can approximate an optimal solution when finding the optimal solution is infeasible. Experimental evidence should be obtained to ensure the absence of ILASP results for early training iterations in Figure 5b can be addressed. Also, future work should be performed to create a more stable approach to estimating the confidence of neural predictions, such as investigating state-of-the-art Bayesian approaches for uncertainty estimation in deep neural networks [21]. The

ProbLog reasoning approach should then be re-evaluated with respect to more stable confidence scores for neural predictions.

Finally, to enable future neural-symbolic generative policy models to be rapidly constructed and trained across a complex set of unstructured features, an *end-to-end* learning approach should be explored and developed which back-propagates outcomes of learned policy rules through the neural components to help improve the contextual feature classification and consequently the learned rules. This is similar to the DeepProbLog approach except logical rules are learned as opposed to being specified by humans. In the results of the *hybrid* policy learning task shown in Figure 5b, no knowledge transfer occurs between ILASP learning tasks. Future work should explore passing symbolic knowledge between iterations with a view to improving overall learning performance.

VI. Conclusion

This paper has performed a deep-dive into a state-of-the-art neural-symbolic learning technique called DeepProbLog, as well as investigated a symbolic learning technique based on inductive logic programming, called ILASP for the tasks of inferring and learning policy rules. The results of both tasks have been evaluated with respect to baseline neural approaches and in both cases, the neural-symbolic integration leads to stronger performance as well as more explainable results. This suggests a platform for future research into enabling neural-symbolic generative policy models, with a view to developing an end-to-end neural-symbolic policy learner, capable of learning policy rules whilst training neural sub-components to classify contextual features from unstructured data.

Acknowledgment

This research was sponsored by the U.S. Army Research Laboratory and the U.K. 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 U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] D. Verma, S. Calo, S. Chakraborty, E. Bertino, C. Williams, J. Tucker, and B. Rivera, "Generative policy model for autonomic management," in *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug 2017, pp. 1–6.

[2] S. Calo, I. Manotas, G. de Mel, D. Cunnington, M. Law, D. Verma, A. Russo, and E. Bertino, "AGENP: An ASGrammar-based GENerative policy framework," in *Policy-Based Autonomic Data Governance*. Springer, Sep. 2019, pp. 3–20. [Online]. Available: https://doi.org/10.1007/978-3-030-17277-0_1

[3] M. Law, A. Russo, B. Elisa, B. Krysia, and L. Jorge, "Representing and learning grammars in answer set programming." in *AAAI*, 2019.

[4] M. Law, A. Russo, and K. Broda, "The ILASP system for learning answer set programs," https://www.doc.ic.ac.uk/~ml1909/ILASP, 2015.

[5] D. Cunnington, I. Manotas, M. Law, G. de Mel, S. Calo, E. Bertino, and A. Russo, "A generative policy model for connected and autonomous vehicles," *2019 IEEE Intelligent Transportation Systems Conference*, Oct. 2019, due to appear.

[6] G. White, J. Ingham, M. Law, and A. Russo, "Using an asg based generative policy to model human rules," *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, Jun. 2019.

[7] S. Calo, D. Verma, S. Chakraborty, E. Bertino, E. Lupu, and G. Cirincione, "Self-generation of access control policies," in *Proceedings of the 23Nd ACM on Symposium on Access Control Models and Technologies*, ser. SACMAT '18. New York, NY, USA: ACM, 2018, pp. 39–47. [Online]. Available: http://doi.acm.org/10.1145/3205977.3205995

[8] D. Cunnington, G. White, M. Law, and G. de Mel, "A demonstration of generative policy models in coalition environments," in *Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection*. Springer International Publishing, 2019, pp. 242–245.

[9] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt, "Deepproblog: Neural probabilistic logic programming," in *Advances in Neural Information Processing Systems*, 2018, pp. 3749–3759.

[10] A. S. Garcez, L. C. Lamb, and D. M. Gabbay, *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.

[11] B. Hammer and P. Hitzler, *Perspectives of neural-symbolic integration*. Springer, 2007, vol. 77.

[12] T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, D. Lowd, P. M. V. Lima *et al.*, "Neural-symbolic learning and reasoning: A survey and interpretation," *arXiv preprint arXiv:1711.03902*, 2017.

[13] A. S. Avila Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Applied Intelligence*, vol. 11, no. 1, pp. 59–77, Jul 1999. [Online]. Available: https://doi.org/10.1023/A:1008328630915

[14] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing, "Harnessing deep neural networks with logic rules," *arXiv preprint arXiv:1603.06318*, 2016.

[15] L. De Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic prolog and its application in link discovery." in *IJCAI*, vol. 7. Hyderabad, 2007, pp. 2462–2467.

[16] G. White, S. Pierson, B. Rivera, M. Touma, P. Sullivan, and D. Braines, "DAIS-ITA scenario," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, Apr 2019, p. 110061F. [Online]. Available: https://doi.org/10.1117/12.2520150

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[18] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[20] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *CoRR*, vol. abs/1412.1897, 2014. [Online]. Available: http://arxiv.org/abs/1412.1897

[21] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

APPENDIX A
DEEPPROBLOG LOGISTICAL RESUPPLY PROGRAM FOR THE POLICY INFERENCING TASK

```
nn(red_bus_net,[X],Y, [present,not_present,unsure])
    :: neural_red_bus(X,Y).
nn(ground_troops_net,[X],Y, [available,unavailable])
    :: neural_ground_troops(X,Y).

outcome(present,available,deploy_ground_troops).
outcome(present,unavailable,re_route_convoy).
outcome(not_present,_,proceed_as_normal).
outcome(unsure,_,re_route_convoy).

mission_step(CCTV,GT_Info,Result) :-
        red_bus(CCTV,RB_Status),
        ground_troops(GT_Info,GT_Status),
        outcome(RB_Status, GT_Status, Result).

red_bus(CCTV,present) :-
    neural_red_bus(CCTV,present).
red_bus(CCTV,not_present) :-
    neural_red_bus(CCTV,not_present).
red_bus(CCTV,unsure) :- neural_red_bus(CCTV,unsure).
ground_troops(GT_Info,available) :-
    neural_ground_troops(GT_Info,available).
ground_troops(GT_Info,unavailable) :-
    neural_ground_troops(GT_Info,unavailable).
```