



ARL-TR-8173 • SEP 2017



Developing Scene Understanding Neural Software for Realistic Autonomous Outdoor Missions

by Arnold D Tunick and Ronald E Meyers

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.



Developing Scene Understanding Neural Software for Realistic Autonomous Outdoor Missions

by Arnold D Tunick and Ronald E Meyers
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

*Form Approved
OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) September 2017		2. REPORT TYPE Technical Report		3. DATES COVERED (From - To) May 2016–September 2017	
4. TITLE AND SUBTITLE Developing Scene Understanding Neural Software for Realistic Autonomous Outdoor Missions				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Arnold D Tunick and Ronald E Meyers				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CII-A Adelphi, MD 20783-1138				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-8173	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT We present a deep learning neural network model software implementation for improving scene understanding of realistic autonomous outdoor missions in complex and changing environments. Scene understanding for realistic outdoor missions has been considered an unsolved problem due to the uncertainty of inferring the mutual context of detected objects and the changing weather, terrain, and environmental surroundings. We report proof-of-principle progress in autonomously searching for and recognizing key activities or scenarios by identifying both salient objects and relevant environmental settings depicted in outdoor scenes. Importantly, we demonstrate autonomous detection of targeted scenarios using neural network models separately trained on both objects and places image databases. In addition, using instructive analysis of 5 representative real-world mission scenarios, we show that adding dynamic environmental data and physics-based modeling could minimize unpredictably by constraining neural predictions to physically realizable solutions.					
15. SUBJECT TERMS computer vision, software, visual scene exploration, convolutional neural network, physics-based modeling, dynamic environment					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 58	19a. NAME OF RESPONSIBLE PERSON Arnold D Tunick
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) (301) 394-1233

Contents

List of Figures	v
List of Tables	vi
Acknowledgments	vii
1. Introduction	1
2. Neural Network Models, Training, and Concepts	3
3. Neural Network Model Software Implementation: Training, Validation, and Analysis	4
4. Places-Centric Neural Network Model	9
5. Autonomous Detection of Key Scenarios	11
5.1 Computer Processing and Timing	15
6. Comprehensive Approach: Adding Dynamic Environmental Data and Physics Based Modeling	15
6.1 Dynamic Environmental Data Retrieval	15
6.2 Physics-Based Modeling	18
6.3 A Comprehensive Approach to Improve Scene Understanding Software	19
6.3.1 Case 1: Ground Search and Rescue	19
6.3.2 Case 2: Aerial Reconnaissance	20
6.3.3 Case 3: Nuclear, Biological, and Chemical (NBC) Hazard Detection	21
6.3.4 Case 4: Cave and Tunnel Reconnaissance	21
6.3.5 Case 5: Sniper Detection	22
7. Conclusions	23
8. References	24

Appendix A. Installed Software and Dependencies	31
Appendix B. Representative Convolutional Neural Network (CNN) Deep Learning Libraries and Open-Source CNN Model Codes	41
Appendix C. Integrated Scene Understanding Approach	45
List of Symbols, Abbreviations, and Acronyms	47
Distribution List	48

List of Figures

Fig. 1	Neural network model training and validation results: top-1 training accuracy (red line) and top-1 validation accuracy (blue diamonds).....	5
Fig. 2	Neural network model initial results showing the top-5 most likely class labels and corresponding top-5 confidence levels	6
Fig. 3	Neural network model results for images of a lighthouse in clear sky and unfavorable environmental conditions: a) clear sky–daytime, b) clear sky–snow covered field, c) daytime–fog and haze, d) lightning, e) storm clouds, f) daytime–foggy, g) clear sky–sunset, h) clear sky–twilight, and i) nighttime–silhouette.....	7
Fig. 4	Neural network model results showing images of Army tanks in clear sky and unfavorable environmental conditions: a) clear sky–sandy soil, b) clear sky–dirt field, c) cloudy sky–daytime, d) sand/dust storm, e) tall/dense vegetation, f) rain/muddy soil, g) rain/city street, h) snow–mountains, and i) snow–forest	8
Fig. 5	Places-CNN scene recognition demo results for selected test images from Figs. 2, 3 and 4: a) Fig. 2a, b) Fig. 2c, c) Fig. 2d, d) Fig. 3d, e) Fig. 3e, f) Fig. 3f, g) Fig. 4f, h) Fig. 4h, i) Fig. 4d.....	10
Fig. 6	Representative neural network model results for the autonomous detection of targeted scenarios. Top-5 image classification labels and top-5 probabilities are annotated for the object-centric trained CNN (above) and location/places-centric trained CNN (below). A scenario is “detected” when the top-1 classification correctly identifies both the object and place of interest.	12
Fig. 7	Visual demonstration that slight differences in the image input due to brightness, blurring, and/or cropping can result in differences in the top-5 image classification: a) image shown in Fig. 4 and b) image shown in Fig. 6.....	13
Fig. 8	Neural network model results for the autonomous detection of the “baseball game” scenario. Top-5 image classification labels and top-5 probabilities are annotated for the object-centric trained CNN (above) and location/places-centric trained CNN (below). A scenario is “detected” when the top-1 classification correctly identifies both the object and place of interest.	14
Fig. 9	Neural network model results for images of persons playing sports other than baseball on a grass field. Top-5 image classification labels and top-5 probabilities are annotated for the object-centric trained CNN (above) and location/places-centric trained CNN (below). In these examples, there were no neural detections of the “baseball game” scenario. Significantly, this sports image series presented no false positives.	14

List of Tables

Table 1	Comparison of neural network model results	11
Table 2	Neural network model results for the autonomous detection of scenarios.....	12
Table 3	Neural network model results for the autonomous detection of the “baseball game” scenario	13
Table 4	Time- and space-varying elements of scene understanding	16
Table 5	Available and accessible dynamic environmental data	17
Table 6	Scene understanding for realistic autonomous outdoor missions (case 1)	20
Table 7	Scene understanding for realistic autonomous outdoor missions (case 2)	20
Table 8	Scene understanding for realistic autonomous outdoor missions (case 3)	21
Table 9	Scene understanding for realistic autonomous outdoor missions (case 4)	22
Table 10	Scene understanding for realistic autonomous outdoor missions (case 5)	22
Table B-1	Convolutional neural network deep learning libraries: open source frameworks	43
Table B-2	Convolutional neural network open source codes	44
Table C-1	Scene understanding for realistic autonomous outdoor missions (summary)	46

Acknowledgments

We thank G Warnell, B Byrne, C Karan, and S Gutstein for helpful discussions. This research was supported by the US Army Research Laboratory.

INTENTIONALLY LEFT BLANK.

1. Introduction

Neurons in the brain enable us to understand scenes by assessing the spatial, temporal, and feature relations of objects in the scene to the environment and ourselves. Clarifying how humans gain scene understanding is an active area of theoretical and experimental research (Biederman 1987; Bar 2004; Battaglia et al. 2013; Sofer et al. 2015; Malcolm et al. 2016) that even extends to experiments to relate magnetic resonance imaging and other sensor measurements of brain neural activity to images of scenes viewed by subjects (Aminoff and Tarr 2015; Aminoff et al. 2015; Suave et al. 2017). Despite the complexity of mimicking human neural activity and because of the potential for enormous gains, there has been intense effort to use computer neural networks to augment human neural intelligence to improve our scene understanding (Krizhevsky et al. 2012; Zhou et al. 2014; Karpathy and Fei-Fei 2015; Lim et al. 2017; Qiao et al. 2017). In the age of robots and drones, achieving rapid and robust understanding of scenes has become a critical goal for autonomous robotic systems performing tasks to support realistic outdoor missions in complex and changing environments (ARL 2014; Judson 2016; Sustersic et al. 2016). Exemplars of complex outdoor missions are military operations, in particular, those where there is a need to reduce the analytical burden to process time-sensitive information on an information-saturated and rapidly changing battlefield (Howard and Cambria 2013). Clearly, the development of an approach for automated scene understanding that can handle complex outdoor missions is needed. Here we present and analyze a deep learning neural network model software implementation for improving scene understanding of realistic autonomous outdoor missions in complex and changing environments.

Scene understanding for realistic outdoor missions is a difficult challenge and remains an unsolved problem (RCTA 2012; Piekiewicz et al. 2016; Tai and Liu 2017) due to the uncertainty of inferring the mutual context of detected objects and the changing weather, terrain, and environmental surroundings. While gaining scene understanding is challenging even in static surroundings, dynamic applications such as terrain and obstacle traversal in changing environments greatly complicates the missions. By developing a prototype scene understanding software tool, we report proof-of-principle progress in autonomously searching for and recognizing key activities or scenarios by identifying both salient objects and relevant environmental settings depicted in outdoor scenes. Importantly, we demonstrate the autonomous detection of targeted scenarios using neural network models separately trained on both objects and places image databases. Analysis using this scene understanding tool can be helpful to autonomously sift through large image data sets for such applications as intelligence, surveillance, and

reconnaissance to identify potential threats to personnel or equipment. Previously, Xiang et al. (2002) and Gori et al. (2016) reported on experiments conducted using indoor video images to autonomously search for and identify human activities that happen in sequence or concurrently. However, their methods did not address neural network models or outdoor environments as ours do.

Neural networks have been applied mainly to static problems. However, their application to realistic scenarios requires attention to changing weather and terrain features encountered along paths that introduce uncertainty in neural predictions. As an example, an automated vision system may readily detect changes in the ground surface along its path as a new or different object in the field of view. Understandably, it would be a challenge for neural networks alone to be able to differentiate such features as shallow or deep water; thick, thin, or melting ice; freezing rain; blowing dust and sand; or snow, mud, or quicksand if they had not been encountered in the training data sets. By using an instructive analysis of 5 representative real-world mission scenarios, we investigate adding physics-based modeling and dynamic environmental data, such as terrain, morphology, weather, visibility, and illumination, to neural network training to minimize unpredictability by constraining neural predictions to physically realizable solutions (Battaglia et al. 2013; Ullman et al. 2014; Fragkiadaki et al. 2015; Wu et al. 2015; Lake et al. 2016; Zhang et al. 2016). Indeed, the integration of neural network software, multiple training data sets, environmental data, and physical modeling would provide a much more comprehensive approach to improve scene understanding software for robotic systems operating in realistic outdoor environments.

In this report, we present our software implementation of a deep learning convolutional neural network (CNN) model that was based on the earlier works of Krizhevsky et al. (2012) and Ding et al. (2015). In Section 2, key concepts related to neural network models and training are discussed. The training, validation, testing, and analysis of our object-trained CNN software implementation are discussed in Section 3. For comparison, test image analyses using a places-trained CNN model (Zhou et al. 2014), which is similarly based on the work of Krizhevsky et al. (2012), but is trained instead on a places/locations image database, are presented in Section 4. In Section 5, we use our object-trained CNN model software implementation together with the places-trained CNN model to demonstrate the autonomous detection of specific scenarios by jointly identifying salient objects and the environmental settings depicted in the outdoor scenes. In Section 6, we outline a more comprehensive approach to improve the next generation of scene understanding software and we summarize our overall conclusions in Section 7.

2. Neural Network Models, Training, and Concepts

In this section, neural network models, training, and concepts are reviewed. Much research has focused on the concept of saliency as if the importance of an object to us or our missions depends solely on how much it visually stands out in a scene. However, as we shall see, to be useful for outdoor missions, this is not sufficient since the environment and mission objectives must be considered. The concept of saliency estimation has been helpful to computationally identify elements in a scene that immediately capture the visual attention of an observer (Itti et al. 1998; Xu et al. 2010; Perazzi et al. 2012). Several recent papers have discussed concepts associated with visual saliency to enhance automated navigation and scene exploration (Roberts et al. 2012; Yeomans et al. 2015; Warnell et al. 2016). However, the most active or salient object(s) in a scene, by this definition, may not represent the most important or meaningful feature(s) of the scene. Also, environmental factors such as changing illumination, precipitation, and vegetation can modify saliency and context of an outdoor scene, obscure features, and significantly degrade object recognition (Wohler and Anlauf 2001; Narasimhan and Nayar 2002; Pepperell et al. 2014; Sunderhauf et al. 2015; Neubert and Protzel 2016; Valada et al. 2016).

When applied to neural network object or place recognition, an autonomous robotic system may predict the correct class label for the principal object shown in a test image (Krizhevsky et al. 2012; Zhou et al. 2014; Karpathy and Fei-Fei 2015), yet overlook key environmental features that can provide important information related to the outdoor mission. In Section 3, we show that an object-trained neural network model correctly classifies an image of a tank (in the near-field view) with high confidence. Nevertheless, in the far field of the image, a tan background is indicative of a sand storm. While the neural network training produced a single label for object recognition, it was not sufficient to identify a potentially vital piece of information that otherwise may have been recognized by a human observer.

Information related to changing environmental dynamics can enhance the scene understanding analysis and can be essential to support the goals and intent of the outdoor mission (Howard and Cambria 2013). For example, the consequence of low visibility can impede many types of navigation, reconnaissance, and target acquisition, and blowing dust or sand can decrease the effectiveness of embedded equipment and personnel. Alternately, the incomplete results of the neural network image classifier in the previous example may be related to the problem of domain transfer (You et al. 2015; Zhang et al. 2016), where knowledge learned by neural networks trained on a data set containing images of a particular domain cannot be

transferred easily to produce an outcome that is outside the training data set. Next, we show how neural network models trained on multiple and diverse data sets can improve scene understanding software for applications in dynamic environments.

There are several alternate neural network models that incorporate scene parsing, (identification of multiple objects in natural scenes [Socher et al. 2011; Karpathy and Fei-Fei 2015; Wigness 2015, 2016; Zhou et al. 2016]). Primarily, these model codes generate automated semantic labeling for image classification of the scene, to include identification of the sky, roads, grass, trees, and buildings, as an example. However, for realistic mission scenarios, salient and meaningful details about these environmental features are not yet provided (e.g., storm cloud covered skies, muddy or icy roads, or thick or impenetrable grass field). Similarly, there are neural network models that have been trained on pictures of indoor and outdoor places and natural environments (Zhou et al. 2014), as opposed to images of objects. In Zhou et al. (2014), image classification predictions for outdoor places have even been augmented with a list of scene understanding attributes (Xiao et al. 2010; Patterson and Hays 2012) that have included several environmental descriptors, such as trees, foliage, dirt soil, cold, snow, ice, sunny, smoke, and clouds, to name a few. Zhou et al. (2014) also reported on a hybrid CNN model that was trained on both scene categories and object categories. However, their hybrid model and training has not been used to jointly identify both salient objects and relevant environmental settings to detect targeted scenarios, as we do in this report.

3. Neural Network Model Software Implementation: Training, Validation, and Analysis

In this section, we present our installation, training, and analysis of the Theano-AlexNet CNN model (Krizhevsky et al. 2012; Ding et al. 2015) implemented on a Windows 10 notebook computer using a single graphics processing unit (GPU). To the best of our knowledge, an implementation of the open-source Python-based AlexNet CNN on a Windows notebook computer has not been previously reported. A description of the installed software and dependencies is given in Appendix A and also can be found in Tunick (2016a). A list of representative CNN deep learning libraries and open-source CNN model codes are provided in Appendix B.

The training and validation of our neural network model software is presented as follows. The CNN was trained to detect 1,000 different object categories (i.e., object classes) using the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) data set (Russakovsky et al. 2014). All of the 1.2 million training images (i.e., 5004 mini-batches of 256 images) were processed during each of 65 computer cycles. On average, training on 20 mini-batches (or iterations) took

approximately 172 s. As a result, the full 65 cycles took approximately 32 days to complete. For comparison, Ding et al. (2015) reported training times of about 40–49 s per 20 iterations for 1 GPU (e.g., approximately 9 days to complete 65 cycles, and 24–29 s per 20 iterations for 2 GPUs). Our proof-of-concept implementation of this CNN software achieved 56.6% validation accuracy for the top-1 class labels (Fig. 1) and 79.7% accuracy for the top-5 class labels (calculated but not shown), even though the training time was long (Bahrapour et al. 2016). These results are in close agreement with those reported by Krizhevsky et al. (2012) and Ding et al. (2015) (i.e., a top-5 accuracy of 81.8% and 80.1%, respectively). For the evaluation of neural network models, the top-5 accuracy rate is defined as the fraction of test images for which the correct class label is among the 5 most likely class labels determined by the model (Krizhevsky et al. 2012).

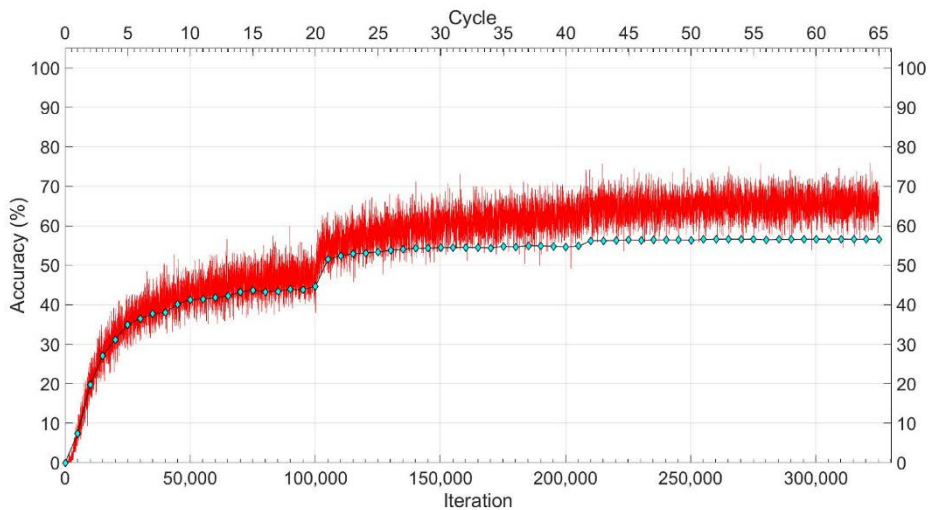


Fig. 1 Neural network model training and validation results: top-1 training accuracy (red line) and top-1 validation accuracy (blue diamonds)

Our CNN software implementation results and analysis using test images that depict representative dynamic environmental features are presented later on. We see that addressing dynamic environments in scene understanding software can be useful for robotic systems operating in realistic outdoor environments. As an example, adverse weather may darken images making it difficult for conventional neural networks to interpret the outdoor scene. However, proper characterization of weather and environmental features can enhance outdoor missions.

For the initial example shown in Fig. 2, we had the model output the top-5 most likely class labels and corresponding confidence levels (i.e., top-5 probabilities) for 4 test images taken from the ILSVRC2012 data set. To do this, we modified the model code and incorporated an inference calculation (Ma 2016; Tunick 2016b) to extract the desired results (i.e., the $p_{y_given_x}$ output from the CNN softmax layer

[see Appendix A, Section A-3.10]). We found that the neural network model predicted the correct class labels for the principal object(s) shown in the test images, generally with high confidence.

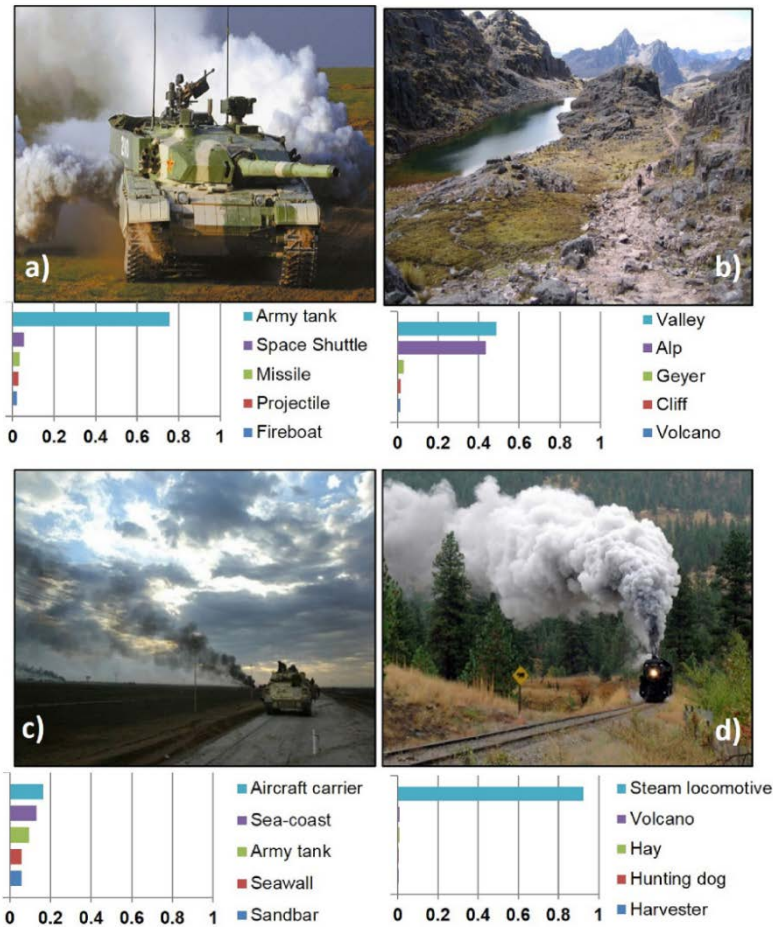


Fig. 2 Neural network model initial results showing the top-5 most likely class labels and corresponding top-5 confidence levels

However, a person viewing these images would likely see several additional features, such as clouds, haze, smoke plumes/exhaust, sandy soil, rocky terrain, mountains, river water, trees, and forests. Identifying these key environmental features is essential to the scene understanding result, especially when the images are viewed in the context of mission impact. Obscured visibility due to smoke plumes, low illumination due to cloud cover, or difficult navigation due to rocky or forested terrain are all vital pieces of information that can provide enhanced operational awareness (Howard and Cambria 2013). As an example, the tank exhaust and trailing smoke cloud in Fig. 2a are indicative of objects moving in an active scene. Similarly, Figs. 2c and 2d depict an Army tank on a roadway and a locomotive on tracks, respectively, each with a trailing smoke plume. However, the Army tank in Fig. 2c was identified with a much lower probability (i.e., 9.4%) than

the locomotive in Fig. 2d (i.e., 92.4%), even though both objects appear set back from the near-field view. Low illumination of the scene and low visibility of the object in Fig. 2c likely affected the CNN model result. Many similar effects were found after testing the CNN model with additional sets of images depicting dynamic environment features (Figs. 3 and 4).

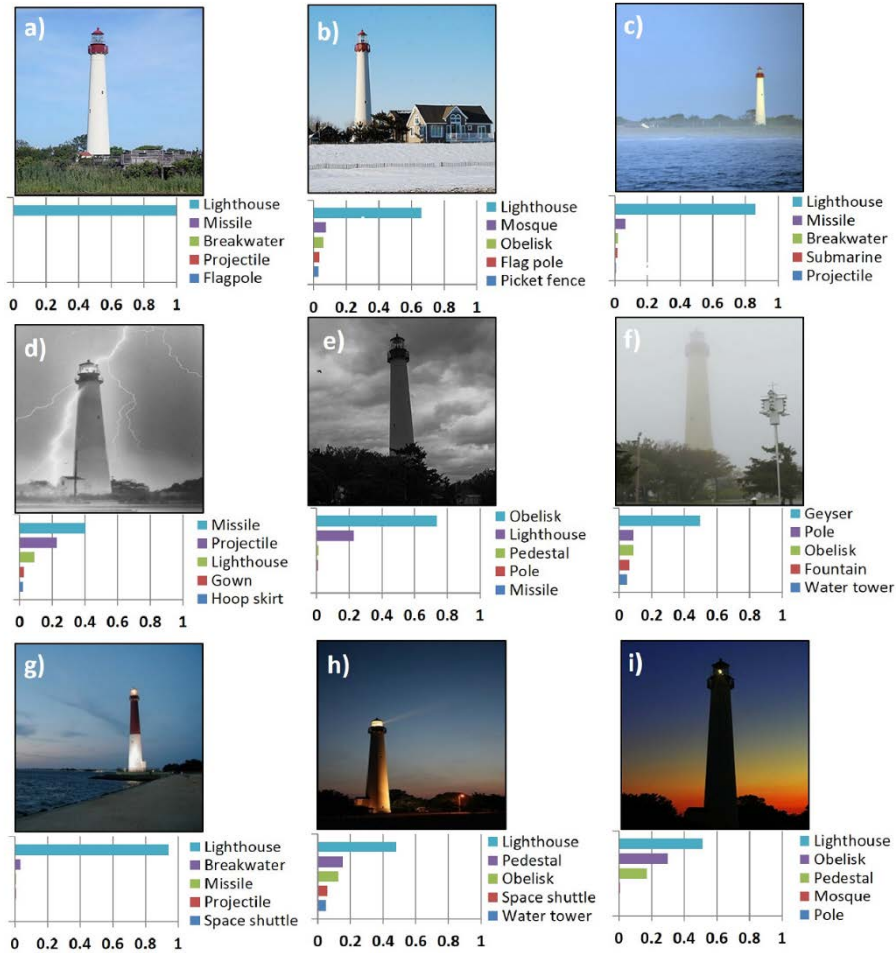


Fig. 3 Neural network model results for images of a lighthouse in clear sky and unfavorable environmental conditions: a) clear sky–daytime, b) clear sky–snow covered field, c) daytime–fog and haze, d) lightning, e) storm clouds, f) daytime–foggy, g) clear sky–sunset, h) clear sky–twilight, and i) nighttime–silhouette

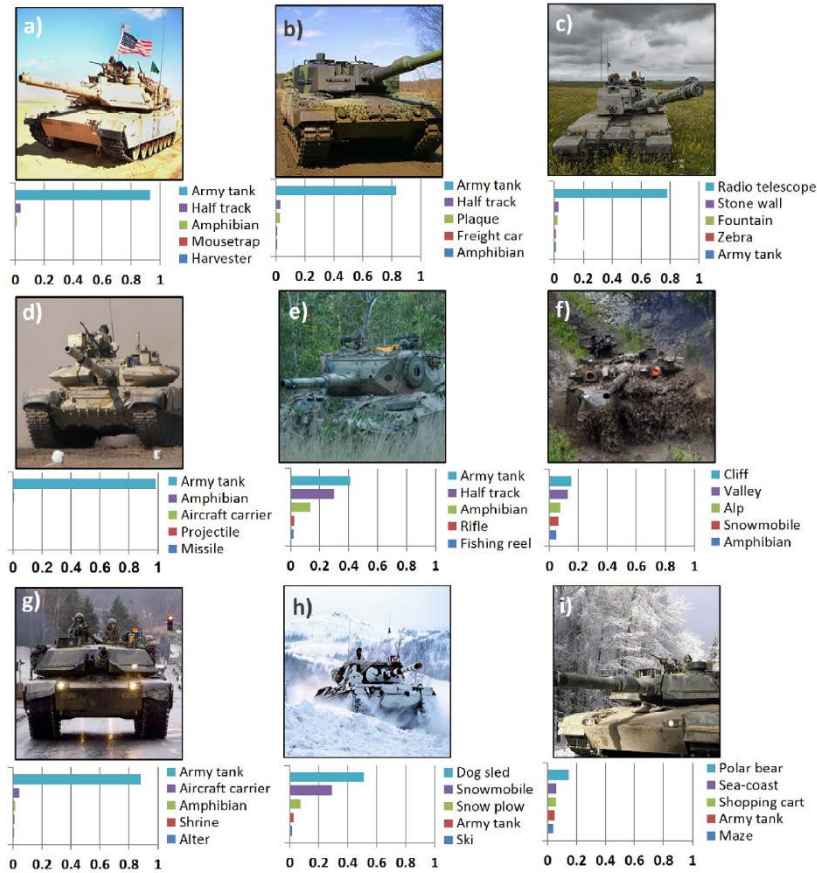


Fig. 4 Neural network model results showing images of Army tanks in clear sky and unfavorable environmental conditions: a) clear sky–sandy soil, b) clear sky–dirt field, c) cloudy sky–daytime, d) sand/dust storm, e) tall/dense vegetation, f) rain/muddy soil, g) rain/city street, h) snow–mountains, and i) snow–forest

Figures 3 and 4 show our CNN model results for several images of a lighthouse and an Army tank, respectively. Again, the top-5 most likely class labels and corresponding top-5 confidence levels are annotated below each figure. In Figs. 3 and 4, it is shown that the CNN model often identified the lighthouse and Army tank correctly and with fairly high confidence.

When the images also depict dynamic weather or terrain features, such as dark storm clouds, fog, lightning, rain, mud, and snow, this led to partially or completely unusable model results (i.e., either the lighthouse or Army tank were completely misidentified or they were identified with much lower probabilities). For instance, the lighthouse with a lightning strike (Fig. 3d) and the lighthouse under dark storm clouds (Fig. 3e) were identified by the model, but with very low probabilities (e.g., 8.9% and 2.3%, respectively). In Fig. 3f, none of the top-5 class labels correctly identified the lighthouse shown in dense fog, albeit a few alternate structures were predicted.

In Fig. 4c, the model predicted the class label for the Army tank under dark cloudy skies with only 1.2% confidence. In contrast, the model identified a visually obscured tank in a forest with tall and dense vegetation (Fig. 4e) with about 40% confidence. In Figs. 4f, 4h, and 4i, the Army tank in rainy weather and muddy terrain was completely misidentified, and the tanks in snow were predicted with very low confidences (i.e., 2.4% and 4.9%, respectively).

Interestingly, the test image in Fig 4d shows an Army tank (in the near-field) with a tan background indicative of a sand storm. In this case, the object trained CNN predicted the correct class label for the principal object, yet overlooked a relevant environmental setting. Certainly, the indication of a sand storm event is an essential feature that can directly impact navigation, visual tracking, or a related outdoor task. Similarly, the rain, mud, and snow shown in Figs. 4f and 4h are indicative of difficult and challenging terrain, which can impact autonomous navigation.

In summary, these results show that dynamic weather and terrain features can impact how a neural network model, such as our object-trained CNN software implementation, interprets outdoor scenes. In the following sections, we show that identifying both salient objects, in addition to relevant environmental settings, can be combined to autonomously search for and recognize targeted scenarios depicted in outdoor scenes.

4. Places-Centric Neural Network Model

In this section, comparative test image analyses are presented using a places trained CNN model (Zhou et al., 2014), which is similarly based on Krizhevsky et al. (2012), but is trained instead on a places/locations image database. In this study, we used the Places-CNN online scene recognition demo (i.e., <http://places.csail.mit.edu/demo.html>) to predict the most likely semantic categories and scene understanding attributes for several test images taken from Figs. 2–4. The semantic categories were based on a 2.5 million image subset of the Places data set (<http://places.csail.mit.edu/>) and the list of scene understanding attributes were based on the data set discussed by Xiao et al. (2010) and Patterson and Hays (2012). Figure 5 presents the places-trained CNN model results, where in many cases, the semantic categories (i.e., classification labels) and augmented scene attributes provided several cues related to environmental settings and features that were not available previously with the object trained CNN model predictions shown in Section 3 (e.g., “snowy mountain”, “direct-sun-sunny”, “clouds”, “foliage”, “trees”, “dirt-soil”, “cold”, “snow”, and “ice”). Interestingly, the places-trained CNN model predictions for the images of a lighthouse shown in Figs. 5d through 5f had much higher top-5 probabilities than the earlier object trained CNN results shown in

Fig. 3d through 3f. In contrast, the places-trained CNN model predictions for the “railway train” shown in Fig. 5c had a much lower probability than the earlier object trained CNN model results shown in Fig. 2d. Table 1 provides a summary of selected CNN results from the object-centric versus place-centric image training. Although, variations can arise due to different training data set domains (You et al. 2015; Zhang et al. 2016), the results in Table 1 demonstrate that scene understanding model software can be improved by neural network training on multiple image databases.

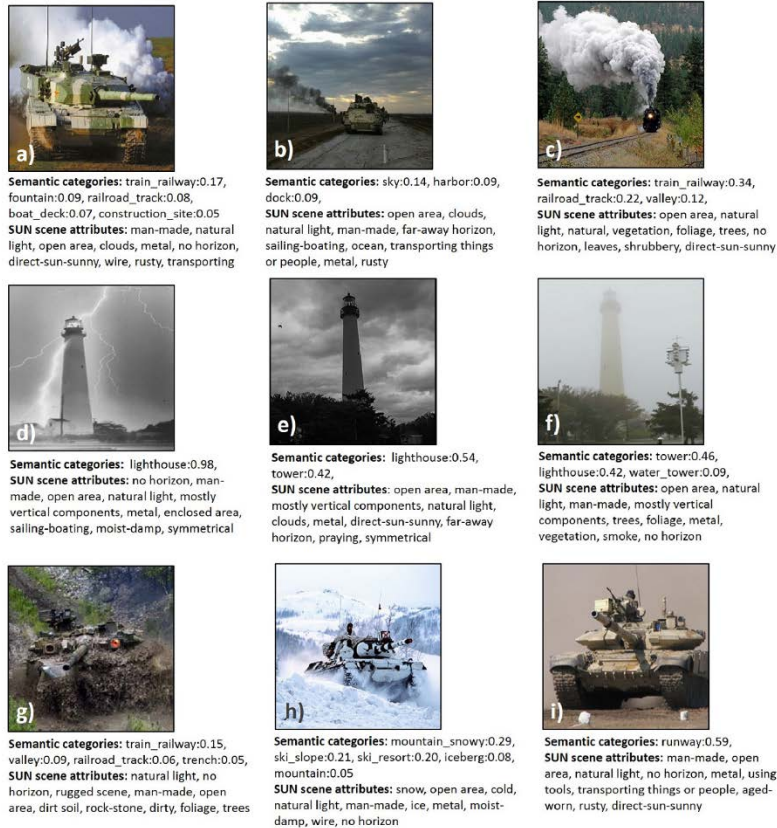


Fig. 5 Places-CNN scene recognition demo results for selected test images from Figs. 2, 3 and 4: a) Fig. 2a, b) Fig. 2c, c) Fig. 2d, d) Fig. 3d, e) Fig. 3e, f) Fig. 3f, g) Fig. 4f, h) Fig. 4h, i) Fig. 4d

Table 1 Comparison of neural network model results

Analyzed image		Classification label		Top-5 probability	
Object	Place	Object	Place	Object	Place
Fig. 2d	Fig. 5c	Steam locomotive	Railway train	0.92	0.34
Fig. 3d	Fig. 5d	Lighthouse	Lighthouse	0.09	0.98
Fig. 3e	Fig. 5e	Lighthouse	Lighthouse	0.23	0.54
Fig. 4f	Fig. 5g	Valley	Valley	0.13	0.09

5. Autonomous Detection of Key Scenarios

Proof-of-principle experiments were conducted to demonstrate the recognition of key scenarios by identifying both salient objects and relevant environmental settings characteristic of the targeted scenario. In this study, we use our object-trained neural network model software implementation together with the places-trained CNN (Zhou et al. 2014) to autonomously detect specific scenarios. Here, a “scenario” is signified by salient objects and environments providing mutual context (i.e., a primary or key object in an outdoor scene embedded in a realistic environmental setting or place). A scenario is “detected” when the primary object is correctly labeled as the most likely top-1 semantic category by the object-trained CNN and the environmental setting is correctly labeled as most likely top-1 semantic category by the places-trained CNN. Table 2 presents a summary of the combined neural network model results. A total of 114 images were analyzed, to include 6 additional images for each environmental setting where the Army tank was not embedded in the scene. As a result, 15 scenarios of interest were detected with no false positives. This result shows that the use of neural network models trained on multiple data sets has applicability for reliable detection of scenarios for a wide class of problems. Representative examples of the test images and the CNN model results that contributed to the data presented in Table 2 are presented in Fig. 6. In our analyzed data set, only instances where both the object of interest and place of interest are predicted as the top-1 semantic category are counted in the “scenarios detected” column shown in Table 2.

Table 2 Neural network model results for the autonomous detection of scenarios


Scenario	No. of images analyzed	No. of tanks recognized	No. of places recognized	No. of scenarios detected	No. of false positives
1. tanks in the desert	30	16	19	8	0
2. tanks on the beach	16	6	7	1	0
3. tanks in the snow	16	2	3	0	0
4. tanks in the forest	18	2	7	0	0
5. tanks in a grass field	18	11	8	4	0
6. tanks in the city	16	6	7	2	0
TOTAL	114	43	51	15	0

	TANKS IN THE DESERT	TANKS ON THE BEACH	TANKS IN THE SNOW	TANKS IN THE FOREST	TANKS IN A GRASS FIELD	TANKS IN THE CITY	
OBJECT CNN	army tank 0.68	army tank 0.91	amphibious vehicle 0.37	half track 0.33	army tank 0.97	army tank 0.21	
	camel 0.04	amphibious vehicle 0.03	army tank 0.36	army tank 0.25	half track 0.01	half track 0.13	
	ibex 0.04	plow 0.02	submarine 0.10	amphibious vehicle 0.09	harvester 0.01	palace 0.07	
	ship wreck 0.03	oxcart 0.01	cannon 0.02	reel 0.05	amphibious vehicle 0.01	monumental archway 0.06	
	gazelle 0.02	---	half track 0.02	rifle 0.04	---	horse cart 0.05	
ANALYZED IMAGE							
	PLACES CNN	desert sand 0.71	ocean 0.33	snowy mountain 0.11	rain forest 0.11	pasture 0.51	medina 0.12
		excavation 0.06	coast 0.24	ski resort 0.08	fountain 0.11	wild field 0.11	slum 0.09
		---	sand bar 0.07	boardwalk 0.07	swamp 0.11	wind farm 0.07	castle 0.07
		---	harbor 0.06	valley 0.05	trench 0.05	runway 0.06	plaza 0.07
---		boat deck 0.05	---	---	---	palace 0.07	


Fig. 6 Representative neural network model results for the autonomous detection of targeted scenarios. Top-5 image classification labels and top-5 probabilities are annotated for the object-centric trained CNN (above) and location/places-centric trained CNN (below). A scenario is “detected” when the top-1 classification correctly identifies both the object and place of interest.

Ostensibly, the image and CNN model data for “tanks in the forest” shown in Fig. 6 were also presented in Fig. 4e; however, the classification labels appeared in a different order with different probabilities. It was later found that 2 separate copies of the original image were used in our analysis, wherein each copy was resized and/or cropped independently of the other. Although there are only slight differences between the 2 images (Fig. 7), the demonstrated differences in the neural network model results highlight an important issue; even small changes in the input data such as brightness, blurring, and cropping can affect the performance of a machine learning classifier (Kurakin et al. 2016). Clearly, this issue will need to be addressed in future developments and implementations of enhanced scene understanding software.

army tank	0.41	half track	0.33
half track	0.30	army tank	0.25
amphibious vehicle	0.14	amphibious vehicle	0.09
rifle	0.03	reel	0.05
reel	0.02	rifle	0.04



a)



b)

Fig. 7 Visual demonstration that slight differences in the image input due to brightness, blurring, and/or cropping can result in differences in the top-5 image classification: a) image shown in Fig. 4 and b) image shown in Fig. 6

A further demonstration of the autonomous detection of scenarios using neural network models separately trained on images of places and objects is provided as follows. The object of interest is a “baseball player” and the place/location of interest is a “baseball field”. Table 3 summarizes the combined CNN model results. A total of 18 images were analyzed and 5 scenarios of interest were detected (Fig. 8). Significantly, the analysis of image data that included photographs of persons playing sports other than baseball on grass fields (Fig. 9) presented no false positives.

Table 3 Neural network model results for the autonomous detection of the “baseball game” scenario

Scenario	No. of images analyzed	No. of objects recognized	No. of places recognized	No. of scenarios detected	No. of false positives
1. baseball game	18	6	7	5	0

DETECTED "BASEBALL GAME" SCENARIO


OBJECT CNN	baseball player	0.93	baseball player	0.59	baseball player	0.91	baseball player	0.96	baseball player	0.56	
	baseball	0.07	sundial	0.21	baseball	0.06	baseball	0.32	baseball	0.38	
	---	---	baseball	0.04	hockey puck	0.01	---	---	racket	0.02	
	---	---	solar dish	0.03	---	---	---	---	---	---	
	---	---	racket	0.02	---	---	---	---	---	---	
ANALYZED IMAGE											
	PLACES CNN	baseball field	0.63	baseball field	0.67	baseball field	0.52	baseball field	0.50	baseball field	0.73
		baseball stadium	0.37	baseball stadium	0.21	baseball stadium	0.48	baseball stadium	0.50	baseball stadium	0.27
		---	---	outdoor track	0.11	---	---	---	---	---	---
		---	---	---	---	---	---	---	---	---	---
---		---	---	---	---	---	---	---	---	---	

Fig. 8 Neural network model results for the autonomous detection of the “baseball game” scenario. Top-5 image classification labels and top-5 probabilities are annotated for the object-centric trained CNN (above) and location/places-centric trained CNN (below). A scenario is “detected” when the top-1 classification correctly identifies both the object and place of interest.

REPRESENTATIVE EXAMPLES FROM THE ANALYZED IMAGE DATA SET

OBJECT CNN	chain armor	0.34	football helmet	0.32	cheetah	0.61	lawn mower	0.12	rapeseed	0.12	
	croquet ball	0.10	umbrella	0.15	cobra	0.04	baseball player	0.09	chain link fence	0.11	
	shield	0.08	baseball player	0.06	bathing suit	0.03	soccer ball	0.06	rail fence	0.11	
	soccer ball	0.07	balloon	0.05	terrier	0.03	croquet ball	0.06	scoreboard	0.07	
	racket	0.04	parachute	0.04	baseball player	0.02	parachute	0.05	seashore, coast	0.06	
ANALYZED IMAGE											
	PLACES CNN	fairway	0.33	football stadium	0.58	fairway	0.44	playground	0.32	football stadium	0.28
		golf course	0.22	baseball field	0.18	baseball field	0.32	topiary garden	0.08	racecourse	0.17
		baseball field	0.11	racecourse	0.17	golf course	0.21	football stadium	0.08	playground	0.14
		yard	0.07	---	---	---	---	botanical garden	0.08	picnic area	0.07
football stadium		0.07	---	---	---	---	yard	0.06	runway	0.07	

Fig. 9 Neural network model results for images of persons playing sports other than baseball on a grass field. Top-5 image classification labels and top-5 probabilities are annotated for the object-centric trained CNN (above) and location/places-centric trained CNN (below). In these examples, there were no neural detections of the “baseball game” scenario. Significantly, this sports image series presented no false positives.

In summary, proof-of-principle experiments demonstrated the recognition of key scenarios by identifying both salient objects and the environmental settings characteristic of the outdoor scene. The autonomous detections of 20 scenarios of interest were demonstrated with no false positives using neural network models separately trained on images of objects and places.

5.1 Computer Processing and Timing

Fast computing speeds are important for realistic autonomous outdoor missions. For the object-centric trained CNN described previously, it took approximately 19 s for the code to initialize and then 5.4 s per test image to produce the top-5 image classification labels and top-5 probabilities. For real-time future implementations of scene understanding software tools, increased computing speeds can be gained by exploiting more efficient algorithms and faster computer processors as they become available. Super computers and parallel processing will enable faster training and validation of neural network codes. In a compact form factor, such capabilities will allow for real-time implementation of neural network scene understanding software tools in such autonomous assets as robots, drones, and self-driving vehicles.

6. Comprehensive Approach: Adding Dynamic Environmental Data and Physics Based Modeling

In the previous sections, we presented the implementation of neural network models separately trained on 2 different image data sets, which resulted in the development of a prototype scene understanding software tool to autonomously detect specific scenarios in diverse outdoor scenes. We found that changing weather and terrain features along the path in realistic scenarios can introduce a variability in the neural predictions not addressed by the neural network training sets. In this section, we show that adding dynamic environmental data and physics-based modeling can minimize such unpredictability by constraining neural predictions to physically realizable solutions, which could augment neural network training and enhance autonomous decision making.

6.1 Dynamic Environmental Data Retrieval

Tunick (2016c) importantly proposed incorporation of space and time varying (i.e., dynamic) environmental data from the very beginning of the autonomous data collection process so that the recorded images can be more effectively indexed and retrieved for operational use and analysis. A systematic characterization of the collected images could be useful for improving scene descriptions and could help end users (e.g., Soldiers for military applications) develop improved course of action strategies for their autonomous robotic assets. As mentioned previously, the retrieval of current terrain and weather data in combat can significantly optimize mission success.

There are many key pieces of information that can be identified as new image data are being recorded that are important and accessible, but are usually overlooked or left undocumented. For example, images can have a timestamp relative to the sun’s angle or relative to a world clock. Recorded images can also be characterized by the GPS position, the prevailing environmental and weather conditions, and the field of view, depth of view, and image resolution (Table 4). The first group, shown in Table 4, focuses on dynamic environmental data, such as the GPS position and altitude above ground level (AGL), prevailing weather, cloud cover, ground and road conditions, and visibility (e.g., fog, smoke, haze, obscurants, or optical turbulence).

Table 4 Time- and space-varying elements of scene understanding

Dynamic environmental data
GPS position and altitude AGL
Location: geographical context
Timestamp
Weather conditions, sky, and cloud cover
Sun/moon angle
Ground/road conditions
Visibility
Vegetation
Buildings, parking lots, people, or crowds
Image/camera information
Image resolution
Pixel size and pixel separation
Scene color or shading variations
Field of view and depth of view
Shutter exposure time
Time interval between image frames
Time over which images are captured in a sequence

Identifying the key environmental and terrain conditions can provide location and geographical context information to help categorize image scenes recorded in diverse regions (e.g., coastal, mountain-valley, desert, forest, urban, rural, ocean, and arctic). Detailed terrain characteristics, such as muddy, sandy, gravelly, wet, dry, or icy, and reports of the most current weather conditions available, such as rain, snow, fog, or haze, can be retrieved and annotated to help describe images used to support the planning or execution of outdoor tasks. Changing weather conditions, cloud cover, and visibility bring about changes in the illumination of a scene, which can affect image contrast and resolution (Narasimhan and Nayar 2002; Lalonde et al. 2012). Retrieval of time of day and sun angle information is useful to indicate when glare, shadows, or silhouettes may cause difficulties for automated computer vision processes (Shafer 1985; Reddy and Veeraraghavan 2014). Taking

note of optical turbulence conditions is important because these effects can significantly degrade and blur image quality due to spatial smearing (Roggemann et al. 1996).

The second group in Table 4 lists elements related to the camera specifications and the image data measurements themselves (e.g., the spatial and temporal image resolutions, field of view, depth of view, and scene color or shading variations). Together with the environmental information, these camera/image elements can provide further details for scene description and image indexing. Note that, in most cases, the image/camera information can be annotated based on the camera type, lensing, pixel array, and timing specifications. Also, for example, co-located range finder instrumentation could provide effective depth and field of view measurements for this purpose.

Next, when communications are available, the most current environmental information available can be extracted from several accessible resources, such as those shown in Table 5. Obtaining the most current data available is advantageous since environmental conditions (e.g., weather and terrain) can change over very short temporal and spatial intervals. For example, access to data from Department of Defense (DOD) GPS (<http://www.gps.gov/governance/agencies/defense/>) can provide latitude and longitude or Universal Transverse Mercator (UTM) location and timestamp information, commonly reported as Greenwich Mean Time (GMT) or Coordinated Universal Time (UTC). Data from the US Naval Observatory (USNO) (<http://www.usno.navy.mil/USNO>) can provide precise timing information as well as solar and lunar elevation/azimuth angles. Similarly, terrain and geographical location and context information are provided by satellite and aerial imagery for military operations from the US Army Corps of Engineers, Army Geospatial Center (USACE AGC) (<http://www.agc.army.mil/>) or from public Internet resources such as Google (<https://www.google.com/maps/>), MapQuest (<http://www.mapquest.com/>), Bing (<https://www.bing.com/maps/>), and Yahoo Maps (<https://maps.yahoo.com/b/>).

Table 5 Available and accessible dynamic environmental data

1	DOD GPS: Lat/long or UTM, altitude (AGL), GMT, or UTC
2	USNO: Precise time, sun/moon elevation/azimuth angle
3	Terrain and location: USACE AGC — Satellite/aerial imagery and terrain analysis
4	Terrain and location: Google, MapQuest, Bing, Yahoo Maps
5	Weather: US Air Force (USAF) 557th Weather Wing
6	Weather: National Weather Service (NWS) and National Centers for Environmental Information (NCEI)
7	Weather: Intellicast, AccuWeather, Weather Underground

Weather conditions and related oceanic, atmospheric, and geophysical data are available for the military through the USAF 557th Weather Wing (<http://www.557weatherwing.af.mil/>) (i.e., formerly the USAF Weather Agency) and for the civilian community through the NWS (<http://www.weather.gov/>) and the NCEI (<https://www.ncei.noaa.gov/>). Daily NWS weather reports that are found online contain hourly records citing the date, time, wind speed (miles per hour), visibility (miles), weather (i.e., rain, snow, fog, haze, etc.), sky/cloud condition (reported as overcast [OVC], broken [BRK], scattered [SCT] or clear [CLR] along with the cloud ceiling height in hundreds of feet AGL), air temperature, dew point temperature, relative humidity (%), pressure, and precipitation (in inches). Naturally, current weather and weather forecast information are readily found on Internet websites, such as Intellicast (<http://www.intellicast.com/>), AccuWeather (<http://www.accuweather.com/>) and Weather Underground (<http://www.wunderground.com/>). Note, however, that in areas where communications are either restricted or unavailable, the information needed to describe the scene (i.e., as outlined in Table 4) should instead be retrieved from co-located sensors on the robots themselves or gleaned from the recorded images using neural network models trained on dynamic environment features. In summary, we have shown that much dynamic environmental data can be retrieved to augment image data as they are being recorded for a better-organized, top-down approach to scene understanding.

6.2 Physics-Based Modeling

People are innately capable of making rapid physical inferences about objects or perceived activities in a scene and to answer the question “what happens next?” (Battaglia et al. 2013; Ullman et al. 2014; Wu et al. 2015; Lake et al. 2016). For computational scene understanding, such inferences can be achieved by combining neural network training with physics-based predictive modeling to help predict how objects in a scene interact with their surroundings. For instance, physics-based modeling can predict how the physical properties of the terrain (as affected by the weather) will impact the navigation of a robotic system. Zhang et al. (2016) discussed 2 types of physical scene understanding models. They conducted a study to compare “intuitive physics engines” with “memory-based models”. In this case, physics simulation engines approximate how objects in complex scenes interact under the laws of physics over short time periods (e.g., stability analysis for stacks of blocks). In contrast, memory-based neural networks make predictions of outcomes in a new scene based on “stored experiences” of encountered scenes and physical outcomes. Wu et al. (2015) also presented a model for detecting physical properties of objects by integrating a physics engine with deep learning neural

network predictions. Similarly, Fragkiadaki et al. (2016) reported on how an autonomous agent could be equipped with an internal predictive physics model of the surrounding environment, and how one could use the physics-based model in combination with neural network training to predict actions that previously have not been encountered by the agent. The combined approach of Fragkiadaki et al. (2016) was demonstrated by accurately predicting the required actions for an autonomous agent to play a simulated billiards game.

These physics-based models may seem complicated and difficult to implement. However, there are alternate types of physics-based models for terrain and weather forecasting already available (Chenery 1997; HQDA 1989, 2015; McDonald et al. 2016) that can be implemented directly on a robotic system or accessed via reach back network communications. One can also envision using weather forecast models together with terrain and morphology data to set up physics-based mission rehearsals and training. Furthermore, where terrain and weather data are available, they can be incorporated using nowcast techniques to interpolate and reconstruct local information needed to support the autonomous outdoor mission.

6.3 A Comprehensive Approach to Improve Scene Understanding Software

We have demonstrated that neural network model software trained on multiple domain image data sets can detect targeted scenarios; however, we understand that dynamic environmental conditions raise uncertainty in neural predictions. A more comprehensive approach to improve scene understanding software for robotic systems operating in realistic outdoor environments would be to add environmental data retrieval and physical modeling to neural network applications. To illustrate the advantages of an integrated scene understanding software approach, we present the following representative target scenario cases for real-world outdoor missions.

6.3.1 Case 1: Ground Search and Rescue

Ground search and rescue is an autonomous outdoor scene exploration mission (Table 6) that involves terrain and obstacle traversal, acoustic detection of endangered personnel, as well as robotic lifting of heavy objects (e.g., people and/or debris and rubble from fallen buildings) (Levinger et al. 2008). Neural network training that specifically addresses changing environmental dynamics can provide useful cues related to ground conditions, visibility, and illumination to benefit navigation and robotic search algorithms. Adding environmental data retrieval and physics-based weather forecasting can provide analyses of current and changing conditions (e.g., those that may affect traversability for robot route planning in uneven terrain). Also, with regard to robots operating over extended periods of

time, it would be important to consider changes in scene illumination from day to night and changes in the visual appearance of a scene due to changes in weather and the seasons (Pepperell et al. 2014; Sunderhauf et al. 2015; Neubert and Protzel 2016).

Table 6 Scene understanding for realistic autonomous outdoor missions (case 1)

Ground search and rescue	Neural network training				Physics based models										Dynamic environmental data retrieval					
	Object recognition	Building / location identification	Terrain recognition	Weather identification	Acoustic propagation	Optical propagation	Optical turbulence	Image enhancement Turbulence mitigation	Weather and terrain prediction	Robot / terrain / morphology interaction	Payload stability analysis	Aerodynamics	NBC hazard analysis	Retrieved local weather	Regional weather trends	GPS, timestamp, and sun angle	Co-located robot weather sensor data	Air quality data	NBC sensor data	Atmospheric turbulence / stability
<ul style="list-style-type: none"> • Terrain and obstacle traversal • Autonomous scene exploration via visible and IR imaging • Acoustic detection • Lifting heavy objects, e.g., people, rubble, and debris 	x	x	x	x			x	x	x					x	x	x				
	x		x		x									x	x	x				

6.3.2 Case 2: Aerial Reconnaissance

The aerial reconnaissance autonomous outdoor scene exploration mission (Table 7) involves the take off, landing, and in-flight control of aerial autonomous assets as well as obstacle avoidance (HQDA 2009; Korpela 2016). Real-time dynamic environmental data retrieval combined with physics-based weather forecast modeling can augment neural network training by providing critical updates on atmospheric stability conditions essential for aerial missions. Terrain and morphology data retrieval can be used to augment both navigation and areal search models. Environmental data such as temperature, air density, cloud cover, wind speed, wind direction, visibility, and precipitation can also be used to augment physics-based optical turbulence modeling to assess refractive index effects on aerial image quality (Roggemann et al. 1996).

Table 7 Scene understanding for realistic autonomous outdoor missions (case 2)

Aerial reconnaissance	Neural network training				Physics based models										Dynamic environmental data retrieval					
	Object recognition	Building / location identification	Terrain recognition	Weather identification	Acoustic propagation	Optical propagation	Optical turbulence	Image enhancement Turbulence mitigation	Weather and terrain prediction	Robot / terrain / morphology interaction	Payload stability analysis	Aerodynamics	NBC hazard analysis	Retrieved local weather	Regional weather trends	GPS, timestamp, and sun angle	Co-located robot weather sensor data	Air quality data	NBC sensor data	Atmospheric turbulence / stability
<ul style="list-style-type: none"> • Take off, landing, and in-flight control • Obstacle avoidance • Autonomous scene exploration via visible and IR imaging 	x	x	x	x					x	x		x	x	x	x					
	x	x	x	x			x	x	x			x		x	x	x				

6.3.3 Case 3: Nuclear, Biological, and Chemical (NBC) Hazard Detection

The NBC hazard detection autonomous outdoor scene exploration mission (Table 8) strives to identify and quantify NBC hazards. It involves robotic NBC sensors, real-time air quality data retrieval, and NBC hazard analysis (HQDA 1986; Scott 2003). Clearly, real-time turbulence and atmospheric stability data are vital for downwind NBC hazard prediction, to include identifying toxicity levels and assessing the pervasiveness and persistence of the threat. Integrating current and forecasted weather elements is equally important because in combat, for example, the weather can alter terrain features and traversability; low visibility can impede reconnaissance or alternately conceal friendly forces maneuvers and activities; and wind speed and direction can favor upwind personnel in the event of an NBC attack or decrease the effectiveness of downwind personnel and equipment due blowing dust, smoke, sand, rain, or snow. Alternately, NBC hazard detection missions can be those associated with chemical spills from overturned trucks or trains, or from accidents at industrial plants. In this situation, neural network training on objects, places, and changing environmental features can provide many critical components for the scene understanding image analysis.

Table 8 Scene understanding for realistic autonomous outdoor missions (case 3)

NBC hazard detection	Neural network training				Physics based models								Dynamic environmental data retrieval								
	Object recognition	Building / location identification	Terrain recognition	Weather identification	Acoustic propagation	Optical propagation	Optical turbulence	Image enhancement	Turbulence mitigation	Weather and terrain prediction	Robot / terrain / morphology interaction	Payload stability analysis	Aerodynamics	NBC hazard analysis	Retrieved local weather	Regional weather trends	GPS, timestamp, and sun angle	Co-located robot weather sensor data	Air quality data	NBC sensor data	Atmospheric turbulence / stability
<ul style="list-style-type: none"> NBC sensors data retrieval NBC hazard analysis Autonomous scene exploration via visible and IR imaging 	x	x	x	x			x	x	x	x			x	x	x	x		x	x	x	x

6.3.4 Case 4: Cave and Tunnel Reconnaissance

The cave and tunnel reconnaissance autonomous scene exploration mission (Table 9) supports the exploration of underground spaces and facilities. It involves underground mapping, terrain and obstacle traversal, laser illumination, and visible and IR imaging (Magnuson 2013; Eshel 2014). Neural network training can provide image classification of salient and meaningful features related to open or concealed objects as well as recognizing subterranean morphology and ground cover conditions. Physics-based models can help predict the outcome and mission impact of relevant robot, terrain, and morphology interactions (e.g., those associated with traversability). Also, in a potentially communications limited region, retrieval of

co-located temperature, pressure, and humidity sensor data can supplement physics-based models, for instance, to assess microclimate effects, such as condensation or frost on robotic vision systems.

Table 9 Scene understanding for realistic autonomous outdoor missions (case 4)

Cave and tunnel reconnaissance	Neural network training				Physics based models									Dynamic environmental data retrieval							
	Object recognition	Building / location identification	Terrain recognition	Weather identification	Acoustic propagation	Optical propagation	Optical turbulence	Image enhancement Turbulence mitigation	Weather and terrain prediction	Robot / terrain / morphology interaction	Payload stability analysis	Aerodynamics	NBC hazard analysis	Retrieved local weather	Regional weather trends	GPS, timestamp, and sun angle	Co-located robot weather sensor data	Air quality data	NBC sensor data	Atmospheric turbulence / stability	
<ul style="list-style-type: none"> • Terrain and obstacle traversal • Laser illumination • Autonomous scene exploration via visible and IR imaging 	x		x	x		x		x		x				x			x				

6.3.5 Case 5: Sniper Detection

The sniper detection autonomous outdoor scene exploration mission (Table 10) searches for difficult to find weaponized threats. It involves acoustic detection of gunfire, laser illumination of identified signal/source locations, and visible and IR imaging to identify potential threats from a distance (Crane 2006). In this situation, neural network training can provide person, object, building, and place recognition capabilities as well as identifying vital weather and terrain features, such as ground conditions for navigation and visibility for reconnaissance. Physics-based acoustic and optical models can provide analysis of propagation conditions for optimal sensor and equipment performance. Physics-based weather forecast modeling and terrain assessment can predict whether ground or visibility conditions will change if the weather worsens during operations. Concurrently, environmental data retrieval, physics-based modeling, and neural network training can provide enhanced situational awareness to support the mission.

Table 10 Scene understanding for realistic autonomous outdoor missions (case 5)

Sniper detection	Neural network training				Physics based models									Dynamic environmental data retrieval							
	Object recognition	Building / location identification	Terrain recognition	Weather identification	Acoustic propagation	Optical propagation	Optical turbulence	Image enhancement Turbulence mitigation	Weather and terrain prediction	Robot / terrain / morphology interaction	Payload stability analysis	Aerodynamics	NBC hazard analysis	Retrieved local weather	Regional weather trends	GPS, timestamp, and sun angle	Co-located robot weather sensor data	Air quality data	NBC sensor data	Atmospheric turbulence / stability	
<ul style="list-style-type: none"> • Acoustic detection • Laser illumination • Autonomous scene exploration via visible and IR imaging 	x	x	x	x	x	x	x	x	x					x	x	x					

In summary, 5 representative outdoor mission scenarios were investigated to explore the types of components needed to bring about rapid and robust autonomous scene exploration. Our analysis indicates that adding environmental data and physics-based modeling to neural network training will improve the next generation of scene understanding software. The integrated scene understanding approach is summarized in Appendix C and its table.

7. Conclusions

A neural network model software implementation has been designed for improving scene understanding of realistic autonomous missions in dynamic outdoor environments. Proof-of-principle experiments demonstrate the recognition of key scenarios by identifying both salient objects in addition to environmental settings characteristic of the targeted scenario. Neural network models separately trained on images of objects and places demonstrated the autonomous detections of 20 scenarios of interest with no false positives. This result shows that the use of neural network models trained on multiple data sets has applicability for reliable detection of scenarios or activities for a wide class of problems. Furthermore, our analysis of 5 representative real-world scenarios illustrated the advantages of incorporating environmental data and physical modeling to improve the next generation of scene understanding software. Addressing changing environmental dynamics in this manner can augment neural network model predictions and lead to the progressive development and implementation of a comprehensive scene understanding approach for autonomous robotic systems supporting realistic outdoor missions.

8. References

- Aminoff EM, Tarr MJ. Associative processing is inherent in scene perception. *PLoS ONE*. 2015;10(6):e0128840.
- Aminoff EM, Toneva M, Shrivastava A, Chen X, Misra I, Gupta A, Tarr MJ. Applying artificial vision models to human scene understanding. *Frontiers in Computational Neuroscience*. 2015;9(8):1–14.
- Army Research Laboratory (US). Army Research Laboratory S&T campaign plans 2015–2035: system intelligence and intelligent systems, 98–99. Adelphi (MD): Army Research Laboratory (US); 2014.
- Bahrampour S, Ramakrishnan N, Schott L, Shah M. Comparative study of deep learning software frameworks. *arXiv:1511.06435v3*; 2016.
- Bar M. Visual objects in context. *Nature Reviews Neuroscience*. 2004;5(8):617–29.
- Battaglia PW, Hamrick JB, Tenenbaum JB. Simulation as an engine of physical scene understanding. *Proc National Academy of Sciences*. 2013;110(45):18327–18332.
- Biederman I. Recognition-by-components: A theory of human image interpretation. *Psychological Review*. 1987;94:115–148.
- Chenery JT. The terrain model: a miniature battlefield. *Military Intelligence Professional Bulletin*. 1997;23(4).
- Crane D. Anti-sniper/sniper detection/gunfire detection systems at a glance. *Defense Review*; 2006 July 19 [accessed 14 June 2017]. <http://defensereview.com/anti-snipersniper-detectiongunfire-detection-systems-at-a-glance/>.
- Ding W, Wang R, Mao F, Taylor G. Theano-based large-scale visual recognition with multiple GPUs. *arXiv:1412.2302v4*; 2015.
- Eshel T. Operating robots underground. *Defense Update*; 2014 July 27 [accessed 14 June 2017]. http://defense-update.com/20140727_underground-robots.html.
- Fragkiadaki K, Agrawal P, Levine S, Malik J. Learning visual predictive models of physics for playing billiards. *arXiv:1511.07404v3*; 2016.

- Gori I, Aggarwal JK, Matthies L, Ryoo MS. Multi-type activity recognition in robot-centric scenarios. *IEEE Robotics and Automation Letters*. 2016;1(1): 593–600.
- Headquarters, Department of the Army. Field behavior of NBC agents (including smoke and incendiaries). Washington (DC): Headquarters, Department of the Army; 1986. Field Manual No.: FM 3–6. Appendix C was written by Meyers RE.
- Headquarters, Department of the Army. Weather support for Army tactical operations. Washington (DC): Headquarters, Department of the Army; 1989. Field Manual No.: FM 34-81.
- Headquarters, Department of the Army. Army unmanned aircraft system operations. Washington (DC): Headquarters, Department of the Army; 2009. Field Manual No.: FM 3-04.155.
- Headquarters, Department of the Army. Weather support and services for the US Army. Washington (DC): Headquarters, Department of the Army; 2015. Field Manual No.: AR 115-10.
- Howard N, Cambria E. Intention awareness: improving upon situation awareness in human-centric environments. *Human-centric Computing and Information Sciences* 2013; 3(1):1–17.
- Itti L, Koch C, Niebur E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans Pattern Analysis and Machine Intelligence*. 1998;20(11):1254–1259.
- Judson J. US Army putting finishing touches on autonomous systems strategy. *Defense News*; 2016 Mar 17 [accessed 14 June 2016]. <http://www.defensenews.com/story/defense/show-daily/ausa-global-force/2016/03/17/army-autonomous-system-strategy/81897736/>.
- Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions. In: *CVPR 2015. Proc IEEE Conference on Computer Vision and Pattern Recognition*. 2015. Boston, MA.
- Korpela C, Root P, Kim J, Wilkerson S, Gadsden SA. A framework for autonomous and continuous aerial intelligence, surveillance, and reconnaissance operations. *Proc SPIE 9828, Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications XIII*. 2016 May 17; p. 982803. Baltimore, MD.

- Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Proc Advances in Neural Information Processing Systems*. 2012;25.
- Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world. *arXiv:1607.02533v3*; 2016.
- Lake BM, Ullman TD, Tenenbaum JB, Gershman SJ. Building machines that learn and think like people. *Behavioral and Brain Sciences*. *arXiv:1604.00289*; 2016.
- Lalonde J-F, Efros AA, Narasimhan SG. Estimating the natural illumination conditions from a single outdoor image. *International Journal of Computer Vision*. 2012;98:123–145.
- Levinger J, Hofmann A, Theobald D. Semiautonomous control of an emergency response robot. In: *Proc Association for Unmanned Vehicle Systems International (AUVSI) Unmanned Systems North America Conference, 2008 June 10–12; San Diego, CA*.
- Lim K, Hong Y, Choi Y, Byun H. Real-time traffic sign recognition based on a general purpose GPU and deep-learning. *PLoS ONE*. 2017;12(3): e0173317.
- Ma H. University of Guelph, Ontario, Canada. Personal communication, 2016.
- Magnuson S. Reconnaissance robots' place on battlefields still unsettled. *National Defense*; 2013 Oct [accessed 2017 June]. <http://www.nationaldefensemagazine.org/archive/2013/October/>.
- Malcolm GL, Groen II, Baker CI. Making sense of real-world scenes. *Trends in Cognitive Sciences*. 2016;20(11):843-856.
- McDonald EV, Bacon SN, Bassett SD, Amit R, Enzel Y, Minor TB, McGwire K, Crouvi O, Nahmias Y. Integrated terrain forecasting for military operations in deserts: Geologic basis for rapid predictive mapping of soils and terrain features. In: *Military Geosciences and Desert Warfare*. New York (NY): Springer; 2016. p. 353–375.
- Narasimhan SG, Nayar SK. Vision and the atmosphere. *International Journal of Computer Vision*. 2002;48(3):233–254.
- Narasimhan SG, Wang C, Nayar SK. All the images of an outdoor scene. In: *Proc. 7th European Conference on Computer Vision*; 2002; Copenhagen, Denmark. p. 148–162.

- Neubert P, Protzel P. Beyond holistic descriptors, keypoints, and fixed patches: multiscale superpixel grids for place recognition in changing environments. *IEEE Robotics and Automation Letters*. 2016;1(1).
- Patterson G, Hays J, Sun attribute database: discovering, annotating, and recognizing scene attributes. In *Proc CVPR, Proc IEEE Conference on Computer Vision and Pattern Recognition*; 2012; Providence, RI.
- Pepperell E, Corke PI, Milford MJ. All-environment visual place recognition with SMART. In *ICRA 2014. Proc IEEE International Conference on Robotics and Automation*; 2014 May 31–June 7; Hong Kong, China.
- Perazzi F, Krahenbuhl P, Pritch Y, Hornung A. Saliency filters: contrast based filtering for salient region detection. *Proc IEEE Conference on Computer Vision and Pattern Recognition*; 2012 June 16–21; Providence, RI.
- Piekniewski F, Laurent P, Petre C, Richert M, Fisher D, Hylton T. Unsupervised learning from continuous video in a scalable predictive recurrent network. *arXiv:1607.06854v3*; 2016.
- Qiao K, Chen J, Wang L, Zeng L, Yan B. A top-down manner-based DCNN architecture for semantic image segmentation. *PLoS ONE*. 2017;12(3):e0174508.
- Reddy D, Veeraraghavan A. Lens flare and lens glare. In *Computer Vision: A Reference Guide*. New York (NY): Springer; 2014. p. 445–447.
- Roberts R, Ta D-N, Straub J, Ok K, Dellaert F. Saliency detection and model-based tracking: a two part vision system for small robot navigation in forested environment. *Proc. SPIE 8387, Unmanned Systems Technology XIV Conference*; 2012.
- Roggemann MC, Welsh BM, Hunt BR. *Imaging through turbulence*. Boca Raton (FL): CRC Press; 1996.
- Robotics Collaborative Technology Alliance (RCTA). FY 2012 annual program plan. Adelphi (MD): Army Research Laboratory (US); 2012 [accessed 2017 June 14]. https://www.arl.army.mil/www/pages/392/RCTA_FY12_APP.pdf.
- Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L. ImageNet large scale visual recognition challenge. *arXiv:1409.0575*; 2014.

- Scott AJ, Mabesa JR, Hughes C, Gorsich DJ, Auner GW. Equipping small robotic platforms with highly sensitive more accurate nuclear, biological, and chemical (NBC) detection systems. *Proc SPIE 5083, Unmanned Ground Vehicle Technology V*; 2003. doi:10.1117/12.497541.
- Shafer SA. *Shadows and silhouettes in computer vision*. Berlin (Germany): Kluwer Academic Publishers; 1985.
- Socher R, Lin CC, Manning C, Ng AY. Parsing natural scenes and natural language with recursive neural networks. In: *Proc 28th International Conference on Machine Learning (ICML-11)*; 2011. p. 129–136.
- Sauve G, Harmand M, Vanni L, Brodeur MB. The probability of object–scene co-occurrence influences object identification processes. *Experimental Brain Research*. 2017;1–13.
- Sofer I, Crouzet SM, Serre T. Explaining the timing of natural scene understanding with a computational model of perceptual categorization. *PLoS Computational Biology*. 2015;11(9):e1004456.
- Sunderhauf N, Shirazi S, Dayoub F, Upcroft B, Milford M. On the performance of ConvNet features for place recognition. In *IROS 2015. Proc IEEE International Conference on Intelligent Robots and Systems (IROS)*; Sept. 28–Oct. 2; Hamburg, Germany.
- Sustersic J, Wyble B, Advani S, Narayanan V. Towards a unified multiresolution vision model for autonomous ground robots. *Robotics and Autonomous Systems*. 2016;75:221–232.
- Tai L, Liu M. Deep-learning in mobile robotics - from perception to control systems: a survey on why and why not. arXiv:1612.07139v3; 2017.
- Tunick A. Mission driven scene understanding: dynamic environment. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2016a. Report No.: ARL-TN-0764.
- Tunick A. Mission driven scene understanding: candidate model training and validation. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2016b. Report No.: ARL-TN-0781.
- Tunick A. Space–time environmental image information for scene understanding. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2016c. Report No.: ARL-TR-7652.

- Ullman TD, Stuhlmuller A, Goodman ND, Tenenbaum JB. Learning physics from dynamical scenes. In CSS 2014. Proc. 36th Annual Conference of the Cognitive Science Society; 2014 Jul 23–26; Quebec City, Canada.
- Valada A, Oliveira G, Brox T, Burgard W. Towards robust semantic segmentation using deep fusion. In: Robotics: Science and Systems (RSS 2016) Workshop, Are the Sceptics Right? Limits and Potentials of Deep Learning in Robotics; 2016.
- Warnell G, David P, Chellappa R. Ray saliency: bottom-up visual saliency for a rotating and zooming camera. *International Journal of Computer Vision*. 2016;116(2):174–189.
- Wigness M, Draper BA, Beveridge JR. Efficient label collection for unlabeled image data sets. In: CVPR 2015. Proc IEEE Conference on Computer Vision and Pattern Recognition. 2015:4594–4602. Boston, MA.
- Wigness M, Rogers III JG, Navarro-Sermenty LE, Suppey A, Draperz BA. Reducing adaptation latency for multi-concept visual perception in outdoor environments. In: IROS 2016. Proc IEEE/RSJ International Conference on Intelligent Robots and Systems; 2016 Oct 9–14; Daejeon, Korea.
- Wohler C, Anlauf JK. Real-time object recognition on image sequences with the adaptable time delay neural network algorithm - applications for autonomous vehicles. *Image and Vision Computing*. 2001;19:593–618.
- Wu J, Yildirim I, Lim JJ, Freeman B, Tenenbaum J. Galileo: perceiving physical object properties by integrating a physics engine with deep learning. *Advances in Neural Information Processing Systems*. 2015:127–135.
- Xiang T, Gong S, Parkinson D. Autonomous visual events detection and classification without explicit object-centred segmentation and tracking. In: Marshall D, Rosin PL, editors. Proc British Machine Conference; 2002. 21.1 – 21.10. doi:10.5244/C.16.21.
- Xiao J, Hays J, Ehinger K, Oliva A, Torralba A. SUN database: large-scale scene recognition from abbey to zoo. In: CVPR 2010. Proc IEEE Conference on Computer Vision and Pattern Recognition; 2010 Jun 13–18; San Francisco, CA.
- Xu J, Yang Z, Tsien JZ. Emergence of visual saliency from natural scenes via context-mediated probability distributions coding. *PLoS ONE*. 2010;5(12): e15796.

- Yeomans B, Shaukar A, Gao Y. Testing saliency based techniques for planetary surface scene analysis. In: ASTRA 2015. Proc 13th Symposium on Advanced Space Technologies in Robotics and Automation; 2015 May 11–13; Noordwijk, the Netherlands.
- You Q, Luo J, Jin H, Yang J. Robust image sentiment analysis using progressively trained and domain transferred deep networks. arXiv:1509.06041v1; 2015.
- Zhang R, Wu J, Zhang C, Freeman WT, Tenenbaum JB. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. arXiv:1605.01138v2; 2016.
- Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A. Learning deep features for scene recognition using Places database. Advances in Neural Information Processing Systems 27 (NIPS); 2014 Dec 8–13; Montreal, Canada.
- Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, Torralba A. Semantic understanding of scenes through the ADE20K data set. arXiv:1608.05442v1; 2016.

Appendix A. Installed Software and Dependencies

In this appendix, we present a description of the installed software and dependencies for our implementation of the convolutional neural network (CNN) Theano-AlexNet^{1,2} on a Windows notebook computer.

A-1 Prerequisite Software Installations to Implement Theano

To start, we outline the prerequisite software installations to implement the Theano program code³ on a Windows 10 notebook computer. Theano is a Python library that facilitates the efficient evaluation of mathematical expressions involving multidimensional arrays. Alternately, an online overview for installing Theano on Windows can be found at http://deeplearning.net/software/theano/install_windows.html#install-windows.

A-1.1 GIT for Windows

To access the GitHub software repository, download the 64-bit version of GIT from <https://github.com/git-for-windows/git/releases/tag/v2.7.1.windows.2> and extract the files into the folder C:\SciSoft\Git.

A-1.2 Visual Studio Community 2013

To access a C++ integrated development environment with 64-bit compilers, download Visual Studio Community 2013 from <https://www.visualstudio.com/en-us/news/vs2013-community-vs.aspx>. Installation and setup for this software is self-explanatory, although one does need to add the following 3 folders to the path:

1. C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\amd64
2. C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\lib\amd64
3. C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\include

A-1.3 Windows Software Development Kit for Windows 10

In addition to Visual Studio 12.0, download the Windows software development kit for Windows 10 from <https://dev.windows.com/en-us/downloads/windows-10-sdk> and extract the files into the folder C:\Program Files (x86)\Microsoft Visual Studio 12.0\VSSDK. The VSSDK folder should also be added to the path.

¹ Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proc Advances in Neural Information Processing Systems. 2012;25.

² Ding W, Wang R, Mao F, Taylor G. Theano-based large-scale visual recognition with multiple GPUs; arXiv:1412.2302v4; 2015.

³ Al-Rfou R et al. Theano: a Python framework for fast computation of mathematical expressions; arXiv:1605.02688v1; 2016.

A-1.4 CUDA v7.5

To provide a development environment for C++ programs implementing graphics processing unit (GPU)-accelerated applications, download CUDA v7.5 from <https://developer.nvidia.com/cuda-toolkit>. This software installation will require that a supported version Microsoft Visual Studio be found on the computer. If not completed automatically, the path can be updated to include the following 2 folders:

1. C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.5\libnvvp
2. C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.5\bin

A-1.5 TDM-GCC

The Theano code compiler requires TDM-GCC installation for either 32- or 64-bit platforms. Therefore, one needs to download the 64-bit version TDM-GCC software from <http://tdm-gcc.tdragon.net/> and extract the files into the folder C:\SciSoft\TDM-GCC-64.

A-1.6 Scientific Python v2.7.9.4

To provide the necessary Python components for both Theano and the CNN AlexNet and for all of their programs' software dependencies, such as numpy, hickle, pycuda, pylearn2, and zeromq, download and install the 64-bit version Python v2.7.9.4 from https://sourceforge.net/projects/winpython/files/WinPython_2.7/2.7.9.4/ and extract the files into the folder C:\SciSoft\WinPython-64bit-2.7.9.4.

A-2 Installing Theano v0.8.0

To provide the mathematical framework within which the CNN AlexNet compiles, download the most current 64-bit version of Theano (v0.8.0) from <https://github.com/Theano/Theano> and extract the files into the folder C:\SciSoft\Git\theano. Alternately, one can download and install the Theano files from a command window by typing the following at the prompt:

- C:\SciSoft\git> git clone <https://github.com/Theano/Theano.git>

A-2.1 Configuration of Paths

To configure the system path for Python and Visual Studio, save following shell script as C:\SciSoft\env.bat:

```
REM configuration of paths
set VSFORPYTHON="C:\Program Files (x86)\Microsoft Visual Studio
12.0"
set SCISOFT=%~dp0
REM add tdm gcc stuff
set PATH=%SCISOFT%TDM-GCC-64\bin;%SCISOFT%TDM-GCC-64\x86_64-w64-
mingw32\bin;%PATH%
REM add winpython stuff
CALL %SCISOFT%WinPython-64bit-2.7.9.4\scripts\env.bat
REM configure path for msvc compilers
CALL %VSFORPYTHON%\vcvarsall.bat amd64
REM return a shell
cmd.exe /k
```

Note that the file vcvarsall.bat, which is called within the env.bat shell script, should contain the following path information:

```
:amd64
echo Setting environment for using Microsoft Visual Studio 2013
x64 tools.
set VCINSTALLDIR=%~dp0VC\
REM set WindowsSdkDir=%~dp0WinSDK\
set WindowsSdkDir=%~dp0VSSDK\
if not exist "%VCINSTALLDIR%bin\amd64\cl.exe" goto missing
set PATH=%VCINSTALLDIR%Bin\amd64;%WindowsSdkDir%VisualStudioInteg
ration\Tools\Bin;%PATH%
set INCLUDE=%VCINSTALLDIR%Include;%WindowsSdkDir%VisualStudioInte
gration\Common\Inc;%INCLUDE%
set LIB=%VCINSTALLDIR%Lib\amd64;%WindowsSdkDir%VisualStudioIntegr
ation\Common\Lib\x64;%LIB%
set LIBPATH=%VCINSTALLDIR%Lib\amd64;%WindowsSdkDir%VisualStudio
Integration\Common\Lib\x64;%LIBPATH%
goto :eof
```

A-2.2 Test the Configuration of Paths

To test the path configuration, open the Python shell in a command window by typing C:\SciSoft\env.bat and then verify that the following programs are found by typing these lines at the prompt:

- C:\SciSoft> where gcc
- C:\SciSoft> where gendef
- C:\SciSoft> where cl
- C:\SciSoft> where nvcc

A-2.3 Link Library for GCC

To create a link library for GCC, open the Python shell in a command window by typing `C:\SciSoft\env.bat` and then type the following at the command window prompt:

- `C:\SciSoft> gendef WinPython-64bit-2.7.9.4\python-2.7.9.amd64\python27.dll`
- `C:\SciSoft> dlltool -dllname python27.dll -def python27.def -output-lib WinPython-64bit-2.7.9.4\python-2.7.9.amd64\libs\libpython27.a`

A-2.4 Setup/Install Theano

Finally, to set up and install Theano, open the Python shell in a command window by typing `C:\SciSoft\env.bat` and then type the following at the prompt:

- `C:\SciSoft\Git\Theano> python setup.py develop`

A-2.5 Test Theano: CPU

To test whether Theano works and is able to compile code for central processing unit (CPU) execution, create the following test file (e.g., filename = test.py):

```
import numpy as np
import time
import theano
A = np.random.rand(1000,10000).astype(theano.config.floatX)
B = np.random.rand(10000,1000).astype(theano.config.floatX)
np_start = time.time()
AB = A.dot(B)
np_end = time.time()
X,Y = theano.tensor.matrices('XY')
mf = theano.function([X,Y],X.dot(Y))
t_start = time.time()
tAB = mf(A,B)
t_end = time.time()
print("NP time: %f[s], theano time: %f[s] %(np_end-np_start,
t_end-t_start))
```

Then open the Python shell in a command window and type the following at the prompt:

- `C:\SciSoft\Git\Theano> python test.py`

The following is the example result:

```
NP time: 1.480863[s], theano time: 1.475381[s]
```

A-2.6 Test Theano: GPU

To test whether Theano works and is able to compile code for GPU execution, create the file `.theanorc.txt` in `C:\SciSoft\WinPython-64bit-2.7.9.4\settings` as follows:

```
[global]
device = gpu
REM device = cpu
floatX = float32
[nvcc]
flags=-LC:\SciSoft\WinPython-64bit-2.7.9.4\python\2.7.9.amd64\
libs
compiler_bindir=C:\Program Files (x86)\Microsoft Visual Studio
12.0\VC\bin
```

Then, rerun the `test.py` file shown in Section A-2.5.

A-2.7 Additional Theano Test

As an additional test of the Theano code, open the Python shell in a command window and type the following at the prompt:

- `C:\SciSoft\Git\Theano> python C:\SciSoft\Git\Theano\bin\theano-nose -batch=3000`

The following is the example result:

```
#####
# COLLECTING TESTS #
#####
# RUNNING TESTS IN BATCHES OF 3000 #
#####
100% done in 604.919s (failed: 0)
#####
# ALL TESTS PASSED #
#####
```

A-3 AlexNet CNN Implementation with Theano

In this section, we outline all of the prerequisite software installations to implement the AlexNet CNN program code^{Error! Bookmark not defined.} within Theano on a Windows 10 notebook computer. Alternately, an online overview of configuring the paths for the AlexNet CNN, preprocessing image data, and running the Python code can be found at https://github.com/uoguelph-mlrg/theano_alexnet.

A-3.1 PIP

An alternate way to install the Python site packages (e.g., pycuda) is to download `get-pip.py` from <https://pip.pypa.io/en/stable/installing/>, which can be extracted into the folder `C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Scripts`. Then to install PIP, open the Python shell `C:\SciSoft\env.bat` in a command window and type the following:

- `C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\ Scripts> python get-pip.py`

A-3.2 Pycuda

To install this dependent Python site package, download the file “`pycuda-2015.1.3+cuda7518-cp27-none-win_amd64.whl`” from <http://www.lfd.uci.edu/~gohlke/pythonlibs/#pycuda> and copy it to the folder `C:\SciSoft\WinPython-64bit-2.7.9.4\settings\pipwin\`. Then to install pycuda, open the Python shell in `C:\SciSoft\env.bat` and then at the command prompt type the following:

- `C:\SciSoft> pip install C:\SciSoft\WinPython-64bit-2.7.9.4\settings\pipwin\pycuda-2015.1.3+cuda7518-cp27- none-win_amd64.whl`

A-3.3 Boost v1.59.0

It is necessary to install several required C++ libraries prior to completing the steps for installing pycuda, as outlined previously. In this case, one must download `boost_1_59_0-msvc-12.0-64.exe` from <https://sourceforge.net/projects/boost/files/boost-binaries/> and then double click on the file to install boost in the folder `C:\local\boost_1_59_0`.

A-3.4 Hickie

To install this dependent Python site package, download hickie from <https://github.com/telegraphic/hickle> and then open the Python shell in `C:\SciSoft\env.bat` and then type the following at the command window prompt:

- `C:\SciSoft> cd C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\hickle`
- `C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\hickle> python setup.py install`

A-3.5 Pylearn2

To install this dependent Python site package, download pylearn2 from <https://github.com/lisa-lab/pylearn2>, open the Python shell in C:\SciSoft\env.bat, and type the following at the command window prompt:

- C:\SciSoft> cd C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\pylearn2
- C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\pylearn2>
python setup.py install

A-3.6 Theano-Alexnet

Download Theano-Alexnet from https://github.com/uoguelph-mlrg/theano_alexnet and extract files into the folder: C:\SciSoft\Git\theano_alexnet\.

A-3.7 Prepare and Preprocess ImageNet Data

To prepare and preprocess ImageNet data, register and download the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) image data .tar files and the 2014 development kit from <http://www.image-net.org> into the following 3 folders:

- C:\SciSoft\Git\theano_alexnet\mnt\data\data sets\ilsvrc_2014\ILSVRC2012_DET_train
- C:\SciSoft\Git\theano_alexnet\mnt\data\data sets\ilsvrc_2014\ILSVRC2012_DET_val
- C:\SciSoft\Git\theano_alexnet\mnt\data\data sets\ilsvrc_2014\ILSVRC2014_devkit

After downloading the image data, open the Python shell C:\SciSoft\env.bat and in the command window run the script C:\SciSoft\Git\theano_alexnet\preprocessing\generate_data.sh, which will call 3 Python scripts. This program runs for about 1–2 days. Alternately, for a short trial of the AlexNet code, run the script C:\SciSoft\Git\theano_alexnet\preprocessing\generate_toy_data.sh, which takes about 10 min.

A-3.8 Set Configurations Paths for AlexNet

Prior to preprocessing the image data, modify the path information in the file `C:\SciSoft\Git\theano_alexnet\preprocessing\path.yaml` as follows and be sure to make similar path annotations in the file `C:\SciSoft\Git\theano_alexnet\spec_1gpu.yaml`:

```
# dir that contains folders like n01440764, n01443537, ...
train_img_dir: 'C:\SciSoft\Git\theano_alexnet\mnt\data\data sets\
ilsvrc_2014\ILSVRC2012_DET_train\'
# dir that contains ILSVRC2012_val_00000001~50000.JPEG
val_img_dir: 'C:\SciSoft\Git\theano_alexnet\mnt\data\data sets\
ilsvrc_2014\ILSVRC2012_DET_val\'
# dir to store all the preprocessed files
tar_root_dir: 'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\'
# dir to store training batches
tar_train_dir: 'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\
train_hkl\'
# dir to store validation batches
tar_val_dir: 'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\
val_hkl\'
# dir to store img_mean.npy, shuffled_train_filenames.npy,
train.txt, val.txt
misc_dir: 'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\misc\'
meta_clsloc_mat: 'C:\SciSoft\Git\theano_alexnet\mnt\data\data
sets\
ilsvrc_2014\ ILSVRC2014_devkit\data\meta_clsloc.mat\'
val_label_file: 'C:\SciSoft\Git\theano_alexnet\mnt\data\data sets\
ilsvrc_2014\ILSVRC2014_devkit\data\ILSVRC2014_clsloc_validation_
ground_truth.txt\'
# training labels
valtxt_filename: 'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\
misc\val.txt\'
# validation labels
traintxt_filename: 'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\
misc\train.txt\'
```

In addition, in the file `C:\SciSoft\Git\theano_alexnet\make_labels.py`, add “import `os.path`” at the top of the file and replaced the line containing “`filename = filename.split('/')[1]`” with “`filename = os.path.basename(filename)`”. Also replace the line containing “`key = train_filename.split('/')[1]`” with “`key = os.path.basename(train_filename)`”. These corrections are necessary because the python `.split` delimiter “/” is not compatible with MS Windows path notations.

A-3.9 Training Theano-AlexNet

Theano-AlexNet was trained using the file C:\SciSoft\Git\theano_alexnet\train.py as follows:

```
C:\SciSoft\Git\theano_alexnet>  
python train.py THEANO_FLAGS=mode=FAST_RUN, floatX=float32.
```

A-3.10 Testing Theano-AlexNet

Theano-AlexNet was tested using a modified version of the file C:\SciSoft\Git\theano_alexnet\alex_net.py to include the inference model recommended by Ma,⁴ wherein one can extract the desired top-5 label and top-5 probability results (i.e., the p_y_given_x output from the CNN softmax layer). For additional details see https://github.com/uoguelph-mlrg/Theano-MPI/blob/master/lib/base/models/alex_net.py.

⁴ Ma H. University of Guelph, Ontario, Canada. Personal communication, 2016.

**Appendix B. Representative Convolutional Neural Network
(CNN) Deep Learning Libraries and Open-Source
CNN Model Codes**

Convolutional neural network (CNN) deep learning methods have influenced and advanced many applications in computer vision, especially those related to image classification^{1,2}. A recent paper by Bahrampour et al.³ presented a comparative study of 5 current deep learning software frameworks with regard to their capability to incorporate different types of CNN architectures, their hardware usage (central processing unit [CPU] and graphics processing unit [GPU]), and an evaluation of their training/testing speed. In this Appendix, we present a summary of these open-source libraries, as well as 3 additional frameworks, in Table B-1, to include a listing of the principal software developers, the primary programming language used, and key reference citations. Similarly, Table B-2 presents a summary of representative CNN open-source codes to include the CNN program⁴ and the Places-CNN⁵ that we used in our study. Note that the top-5 performance for the Places-CNN was evaluated with 2 data sets (i.e., Places 205 and SUN 205⁵).

¹ Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proc Advances in Neural Information Processing Systems; 2012;25.

² Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L. ImageNet large scale visual recognition challenge; arXiv:1409.0575; 2014.

³ Bahrampour S, Ramakrishnan N, Schott L, Shah M. Comparative study of deep learning software frameworks; arXiv:1511.06435v3; 2016.

⁴ Ding W, Wang R, Mao F, Taylor G. Theano-based large-scale visual recognition with multiple GPUs; arXiv:1412.2302v4; 2015.

⁵ Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A. Learning deep features for scene recognition using Places database. Advances in Neural Information Processing Systems 27 (NIPS); 2014 Dec 8–13; Montreal, Canada.

Table B-1 Convolutional neural network deep learning libraries: open source frameworks

Name	Developer	Language	Computation	Key reference
Caffe	Berkeley Vision and Learning Center	C++, Python/Matlab	CPU, GPU	a
Torch	Collobert, Farabet Kavukcuoglu, Chintala	Lua	CPU, GPU	b
Theano	The Theano Development Team	Python	CPU, GPU	c
TensorFlow	Google	C++, Python	CPU, GPU	d
CNTK	Microsoft (Computational Network Toolkit)	C++	CPU, GPU	e,f
Neon	Nervana Systems	Python	CPU, GPU	g
Deeplearning4j	Skymind	Java, Scala	CPU, GPU	h
VLFeat	Vedaldi, Fulkerson	C, Matlab	CPU, GPU	i

^a Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T. Caffe: Convolutional architecture for fast feature embedding; arXiv:1408.5093v1; 2014.

^b Collobert R, Kavukcuoglu K, Farabet C. Torch7: A MATLAB-like environment for machine learning. Proc Advances in Neural Information Processing Systems; EPFL-CONF-192376; 2011.

^c Al-Rfou R et al. Theano: A Python framework for fast computation of mathematical expressions; arXiv:1605.02688v1; 2016.

^d Abadi M et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems; arXiv:1603.04467v2; 2016.

^e Yu D et al. An introduction to computational networks and the computational network toolkit. Redmond (WA): Microsoft; Report No.: MSR-TR-2014-112; 2014.

^f Yu D, Yao K, Zhang Y. The computational network toolkit. IEEE Signal Processing Magazine. 2015;23-126.

^g Neon deep learning library (DLL). San Diego (CA): Nervana Systems Inc.; [accessed 2017 June]. <http://neon.nervanasys.com/docs/latest/index.html>.

^h Deeplearning4j. Open-source distributed deep learning for the JVM. San Francisco (CA): Skymind; [accessed 2017 June]. <http://deeplearning4j.org>.

ⁱ Vedaldi A, Fulkerson B. VLFeat: An open and portable library of computer vision algorithms. Proc ACM Int Conf on Multimedia. 2010.

Table B-2 Convolutional neural network open source codes

Name	Developer	Language	Top-5 accuracy	Reference
Cuda-Convnet (in Caffe)	Krizhevsky, Sutskever, Hinton	Python	81.8 % (2 GPUs)	a
AlexNet (in Theano)	Ding, Wang, Mao, Taylor	Python	80.1% (2 GPUs)	b
Places – CNN (in Caffe)	MIT	C++, Python/Matlab	81.1% / 91.9% (1 GPU)	c
GoogLeNet (in Caffe)	Google	C++, Python/Matlab	93.3 % (CPU)	d
VGG (in Caffe)	Simonyan, Zisserman	C++	92.5% (4 GPUs)	e
OverFeat (in Torch)	NYU	C++, Python/Lua	86.4% (1 GPU)	f
Matconvnet (in VLFeat)	Vedaldi, Lenc	Matlab	~80% (1 GPU)	g

^a Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proc Advances in Neural Information Processing Systems. 2012;25.

^b Ding W, Wang R, Mao F, Taylor G.. Theano-based large-scale visual recognition with multiple GPUs; arXiv:1412.2302v4; 2015.

^c Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A. Learning deep features for scene recognition using Places database. Advances in Neural Information Processing Systems 27 (NIPS); 2014 Dec 8–13; Montreal, Canada.

^d Szegedy C, Liu W, Yangqing J, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: CVPR 2015. Proc IEEE Conference on Computer Vision and Pattern Recognition; 2015; Boston (MA).

^e Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. Ithaca (NY); arXiv:1409.1556v6; 2015.

^f Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun T. Overfeat: Integrated recognition, localization and detection using convolutional networks; arXiv:1312.6229v4; 2014.

^g Vedaldi A, Karel L. Matconvnet: Convolutional neural networks for MATLAB. Proc 23rd ACM International Conference on Multimedia. 2015.

Appendix C. Integrated Scene Understanding Approach

In this appendix, we present a single Table C-1 to summarize our integrated scene understanding approach for the real-world mission scenarios described in Section 6.3 of the main report.

Table C-1 Scene understanding for realistic autonomous outdoor missions (summary)

Real-World Mission Scenario	Neural network training				Physics based models										Dynamic environmental data retrieval						
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	
1. Ground search and rescue <ul style="list-style-type: none"> • Terrain and obstacle traversal • Autonomous scene exploration via visible and IR imaging • Acoustic detection • Lifting heavy objects, e.g., people, rubble, and debris 	x	x	x	x				x	x	x					x	x	x				
2. Aerial reconnaissance <ul style="list-style-type: none"> • Take off, landing, and in-flight control • Obstacle avoidance • Autonomous scene exploration via visible and IR imaging 	x	x	x	x					x	x		x		x	x	x					
3. NBC hazard detection <ul style="list-style-type: none"> • NBC sensors data retrieval • NBC hazard analysis • Autonomous scene exploration via visible and IR imaging 	x	x	x	x				x	x				x	x	x	x		x	x	x	
4. Cave and tunnel reconnaissance <ul style="list-style-type: none"> • Terrain and obstacle traversal • Laser illumination • Autonomous scene exploration via visible and IR imaging 	x		x	x		x								x				x			
5. Sniper detection <ul style="list-style-type: none"> • Acoustic detection • Laser illumination • Autonomous scene exploration via visible and IR imaging 	x	x	x	x	x		x	x		x				x	x	x					

Key:	
a. Open / concealed object recognition b. Building / location identification c. Terrain features / ground cover recognition d. Weather feature identification e. Acoustic propagation, refraction, and scattering f. Optical propagation, refraction, and scattering g. Optical turbulence h. Image enhancement / turbulence mitigation i. Weather / terrain forecasts j. Robot / terrain / morphology interaction k. Payload stability analysis	l. Aerodynamics m. NBC downwind hazard analysis n. Local surface and lower atmosphere wind, temperature, pressure, air density, and humidity o. Regional scale weather trends p. GPS / timestamp / sun angle q. Robot weather sensors: co-located temperature, pressure, and humidity in communications denied region r. Air quality measurements s. NBC sensor measurements t. Atmospheric turbulence / stability

List of Symbols, Abbreviations, and Acronyms

AGC	Army Geospatial Center
AGL	above ground level
ARL	US Army Research Laboratory
BRK	broken
CLR	clear
CNN	convolutional neural network
CPU	central processing unit
DOD	Department of Defense
GMT	Greenwich Mean Time
GPS	global positioning system
GPU	graphical processing unit
ILSVRC2012	ImageNet Large Scale Visual Recognition Challenge 2012
NBC	nuclear, biological, and chemical
NCEI	National Centers for Environmental Information
NWS	National Weather Service
OVC	overcast
SCT	scattered
USACE	US Army Corps of Engineers
USAF	US Air Force
USNO	US Naval Observatory
UTC	Coordinated Universal Time
UTM	Universal Transverse Mercator

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

2 DIR ARL
(PDF) RDRL CIO L
IMAL HRA MAIL & RECORDS MGMT

1 GOVT PRINTG OFC
(PDF) A MALHOTRA

7 ARL
(PDF) RDRL CII
MA THOMAS
RDRL CII A
S YOUNG
D BARAN
A TUNICK
RDRL CIN T
B RIVERA
R MEYERS
K DEACON