ARL-TN-0781 ● SEP 2016

US Army Research Laboratory

Mission Driven Scene Understanding: Candidate Model Training and Validation

by Arnold Tunick

Approved for public release; distribution unlimited.

NOTICES

Disclaimers

ARL-TN-0781 ● SEP 2016

US Army Research Laboratory

Mission Driven Scene Understanding: Candidate Model Training and Validation

by Arnold Tunick

Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* September 2016	**2. REPORT TYPE** Technical Note	**3. DATES COVERED (From - To)** 10/2015 – 08/2016

4. TITLE AND SUBTITLE Mission Driven Scene Understanding: Candidate Model Training and Validation	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Arnold Tunick	**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 2800 Powder Mill Road Adelphi, MD 20783-1138	**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-0781
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 unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Army missions take place in dynamic environments, where changing illumination, precipitation, and vegetation can modify saliency and context of an outdoor scene, obscure features, and degrade object recognition. For Army missions, scene understanding tools need to account for dynamic environments that change as a function of space and time and should be tested in mission simulating conditions. In addition, the impact of dynamic environments should be included in the scene understanding approach. At this stage, we are evaluating different computational frameworks that may be useful to incorporate dynamic environments into mission driven scene understanding. One of the candidate engines that we are evaluating is a convolutional neural network (CNN) program installed on a Windows 10 notebook computer. In this report, we present progress toward the proof-of-principle testing of the candidate model to examine the impact of dynamic environments on scene understanding model results.

15. SUBJECT TERMS

computer vision, context, saliency, convolutional neural network, dynamic environments

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Arnold Tunick
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER (Include area code)**
Unclassified	Unclassified	Unclassified	UU	18	(301) 394-1233

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

I thank RE Meyers, G Warnell, P David, 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

Rapid and robust scene understanding is a critically important goal for the development of Army autonomous intelligent systems to support the Army mission.[1] Army missions take place in dynamic environments, where changing illumination, precipitation, and vegetation can modify saliency and context of an outdoor scene, obscure features, and degrade object recognition. For Army missions, scene understanding tools need to account for dynamic environments that change as a function of space and time and should be tested in mission simulating conditions. In addition, the impact of dynamic environments should be included in the scene understanding approach.[2] Image features that can potentially help the mission are relevant. For example, important image features may be related to space-time coordinates, weather conditions and trends, visibility, terrain, scene descriptors, anomalies, and other salient features.[3–5]

To explore the impact of dynamic environments on scene undersanding, we need a computational engine for scene exploration of new images. At this stage, we are evaluating different computational frameworks that may be useful to incorporate dynamic environments into mission driven scene understanding. One of the candidate engines that we are evaluating is a convolutional neural network (CNN) program (i.e., Theano-AlexNet[6,7]) installed on a Windows 10 notebook computer. 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.

In this report, we present progress toward the proof-of-principle testing of the candidate CNN model to examine the impact of dynamic environments on scene understanding model results. While we found previously[5] that the CNN was able to determine the correct class labels for images taken from the 2,560 image training data set, the validation process did not appear to provide optimal results for images not previously seen. As a result, we performed additional trials and analysis using the larger ImageNet[8] data set containing approximately 1.2 million images (Fig. 1). In Section 3, we show that the CNN achieved 79.7% validation accuracy for the top-5 class labels, which is in close agreement with results published by its developers.

We start our discussion by presenting an overview of representative deep learning libraries (i.e., available open-source computational engines/frameworks) as well as a summary of several current CNN open-source codes.

Fig. 1 For Army mission activities, the impact of dynamic environments should be included in the scene understanding approach (e.g., space-time coordinates, weather conditions and trends, visibility, terrain, scene descriptors, anomalies, and other salient features) (data from the ImageNet[8] Large Scale Visual Recognition Challenge 2012 [ILSVRC2012])

2. CNN Deep Learning Libraries and Open Source Codes

CNN deep learning methods have influenced and advanced many applications in computer vision, especially those related to image classification.[6,8] A recent paper by Bahrampour et al.[9] 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 graphical processing unit [GPU]), and an evaluation of their training/testing speed. We present a summary of these open-source libraries, as well as 3 additional frameworks, in Table 1, to include a listing of the principal software developers, the primary programming language used, the Internet location of the open-source codes, the Internet location of installation and user's guide documentation, and key reference citations. Similarly, Table 2 presents a summary of representative CNN open-source codes to include the candidate CNN program[7] that we trained and validated. Note that in Table 2, some CNN codes achieve better validation accuracy or train at greater speeds than AlexNet[6,7] in Theano[10], particularly those associated with the Caffe[11] and Computational Network Toolkit (CNTK)[14,15] frameworks. Nevertheless, Bahrampour et al.[9] commented that the Theano-based libraries and codes benefit from the flexibility and ease in development using the Python language. In contrast, the primary programming language for Caffe and CNTK is C^{++}.

Table 1 CNN deep learning libraries: open source frameworks

Name	Developer	Language	Availability	Documentation	Computation	Key Reference
Caffe	Berkeley Vision and Learning Center	C++ with Python/MATLAB wrappers	github.com/BVLC/caffe	Tutorial and installation guide: • caffe.berkeleyvision.org/tutorial/ • caffe.berkeleyvision.org/installation.html	Support for CPU, GPU	Jia et al.[11]
Torch	R Collobert C Farabet K Kavukcuoglu S. Chintala	Lua	github.com/torch/torch7	Tutorials, demos, examples, developer guide: • www.torch.ch	Support for CPU, GPU	Collobert et al[12]
Theano	The Theano Development Team	Python	github.com/Theano/Theano	Tutorial and installation guide: • deeplearning.net/software/theano/	Support for CPU, GPU	Al-Rfou et al.[10]
TensorFlow	Google	C++, Python	github.com/tensorflow/tensorflow	Installation and user's guide: • tensorflow.org/	Support for CPU, GPU	Abadi et al.[13]
CNTK	Microsoft (Computational Network Toolkit)	C++	github.com/Microsoft/CNTK	Tutorial, setup, examples: • www.cntk.ai/ • github.com/Microsoft/CNTK/wiki	Support for CPU, GPU	Yu et al.[14,15]
Neon	Nervana Systems	Python	github.com/nervanasystems/neon	Tutorial: • neon.nervanasys.com/docs/latest/tutorials.html	Support for CPU, GPU	Neon DLL[16]
Deeplearning4j	Skymind	Java, Scala	github.com/deeplearning4j/deeplearning4j	Tutorial and user's guide: • deeplearning4j.org/	Support for CPU, GPU	Deeplearning4j[17]
VLFeat	A Vedaldi B Fulkerson	C with Matlab interface	www.vlfeat.org/install-matlab.html	User's manual: • http://www.vlfeat.org/matconvnet/matconvnet-manual.pdf	Support for CPU, GPU	Vedaldi and Fulkerson[18]

Table 2 CNN open-source codes

Name	Developer	Language	Availability	Top-5 Accuracy	Reference
Cuda-Convnet (in Caffe)	A Krizhevsky I Sutskever GE Hinton	Python	github.com/akrizhevsky/cuda-convnet2	81.8 % (2 GPUs)	Krizhevsky et al.[6]
AlexNet (in Theano)	W Ding, R Wang, F Mao, G Taylor	Python	github.com/uoguelph-mlrg/theano_alexnet	80.1% (2 GPUs)	Ding et al.[7]
GoogLeNet (in Caffe)	Google	C++ with Python/MATLAB wrappers	github.com/BVLC/caffe/tree/master/models/bvlc_googlenet	93.3 % (CPU)	Szegedy et al.[19]
VGG (in Caffe)	K Simonyan A Zisserman	C++	www.robots.ox.ac.uk/~vgg/research/very_deep/	92.5% (4 GPUs)	Simonyan and Zisserman[20]
OverFeat (in Torch)	NYU Computational Intelligence, Learning, Vision, and Robotics Lab	C++ with Python/Lua wrappers	github.com/sermanet/OverFeat cilvr.nyu.edu/doku.php?id=software:overfeat:start	86.4% (1 GPU)	Sermanet et al.[21]
Matconvent (in VLFeat)	A Vedaldi K Lenc	Matlab	github.com/vlfeat/matconvnet	~80% (1 GPU)	Vedaldi and Karel[22]

3. Candidate CNN Model: Training and Validation

In this section, we present the candidate CNN model training and validation results that were achieved implementing the program code on a Windows 10 notebook computer using a single GPU. A description of the installed software and dependencies was given in in a previous report[5] and is therefore not repeated here.

The CNN was executed for 65 epoch (i.e., cycles), wherein 5,004 mini-batches of 256 images were processed for each training cycle. Here, we used image data from the ImageNet[8] Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). A few example images from the training data set are shown in Fig. 1. On average, training on 20 mini-batches (or iterations) took approximately 172 s. As a result, the entire 65 cycles took approximately 32 days to complete.* Even though the training time was long, the CNN achieved 56.6% validation accuracy for the top-1 class labels and 79.7% accuracy for the top-5 class labels (Fig. 2). These results are in close agreement with those reported by Krizhevsky et al.[6] (i.e., a top-5 accuracy of 81.8%) and Ding et al.[7] (i.e., a top-5 accuracy of 80.1%). Thus, in the next section, we show our initial top-5 class label results achieved from testing the CNN with 4 single images gleaned from the training data set.

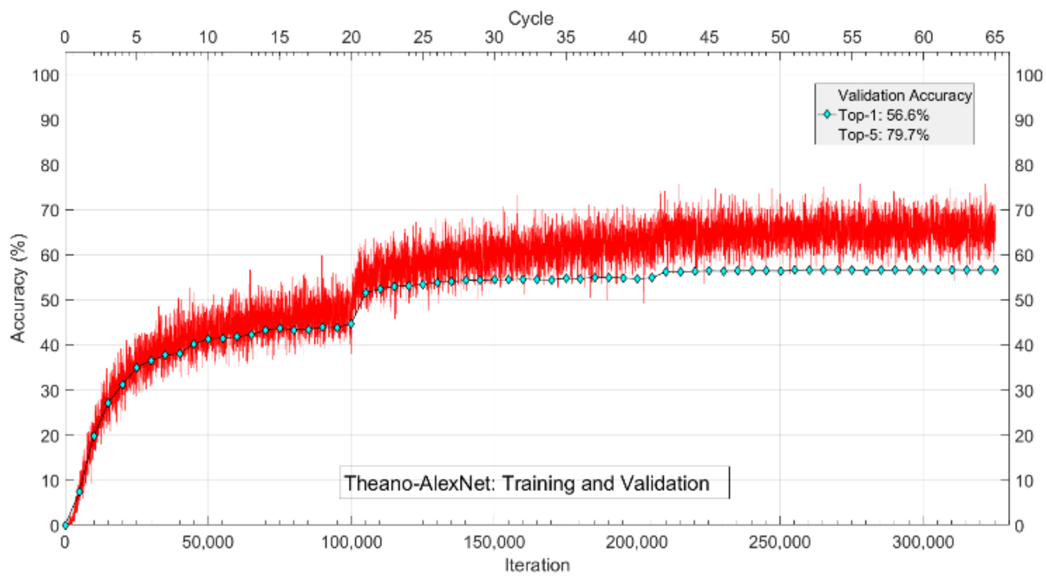

Fig. 2 **Candidate CNN training and validation. Top-1 training accuracy (red line). Top-1 validation accuracy (blue diamonds).**

4. Candidate CNN Model: Results

In this section, we test the candidate model to examine the impact of dynamic environments on scene understanding model results. For the example shown in Fig. 3, we had the model output the top-5 most likely classification labels and corresponding confidence levels (i.e., top-5 probabilities) for the 4 images shown in Fig. 1. To do this, we modified the model code and incorporated an inference calculation[23] to extract the desired results. We found that the CNN predicted the

* For comparison, Ding et al.[7] 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.

correct class label for the principal object(s) shown in the test images, generally with high confidence. Nevertheless, a person viewing these images (e.g., a Soldier-in-the-loop) would likely see several additional features, such those related to the environment that were not identified (e.g., clouds, haze, smoke plumes, sandy soil, rocky terrain, mountains, river water, trees, and forests). More importantly though, we noticed that, in Fig. 3c, low light and visibility conditions negatively affected the candidate model results (i.e., much lower probabilities). Hence, it is this kind of adverse impact on scene understanding model results that require further testing (e.g., with sets of new images that contain similar objects, but depict a wide variety of relevant dynamic environment features).

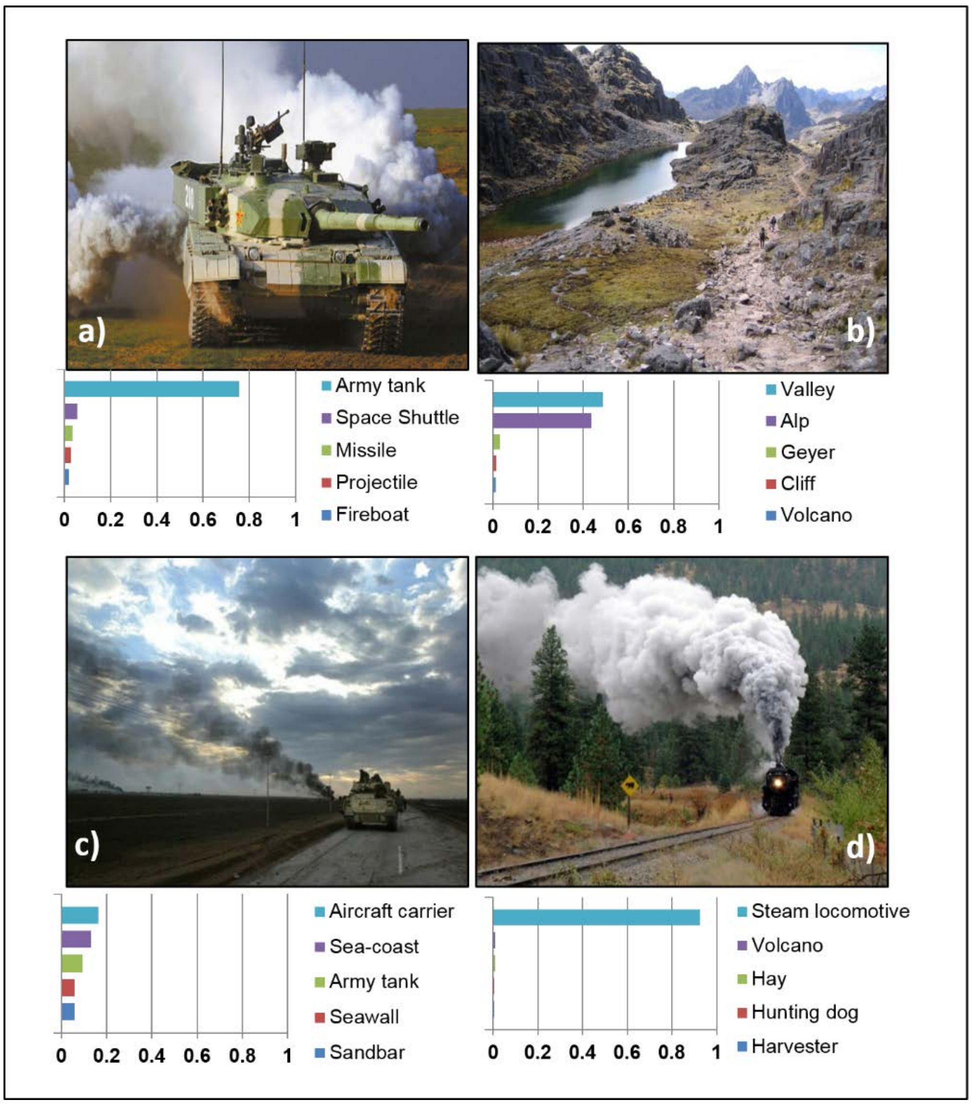

Fig. 3 Candidate CNN results showing the top-5 most likely classification labels and corresponding top-5 confidence levels

5. Summary and Conclusions

Two key aspects of scene understanding modeling are readily apparent from our research so far:

1) Scene understanding tools need to account for dynamic environments to better support Army missions performed by autonomous intelligent systems, and

2) Images depicting adverse dynamic environment features (e.g., low visibility and illumination) tend to negatively impact the scene understanding model results.

We can conduct further testing of candidate models to quantify these aspects in more detail. Nevertheless, it is clear that improved or retrained models are needed to better address the impact of dynamic environments on mission driven scene understanding.

6. References

1. Army Research Laboratory (US). US Army Research Laboratory S&T Campaign Plans 2015–2035. Adelphi (MD): Army Research Laboratory (US); 2014. System Intelligence & Intelligent Systems; p. 98–99.

2. Meyers RE. Army Research Laboratory (US), Adelphi, MD. Personal communication, 2014.

3. Tunick A. Space and time scale characterization of image data in varying environmental conditions for better scene understanding. Adelphi (MD): Army Research Laboratory (US); 2015 Sep. Report No.: ARL-TR-7419.

4. Tunick A. Space–time environmental image information for scene understanding. Adelphi (MD): Army Research Laboratory (US); 2016 Apr. Report No.: ARL-TR-7652.

5. Tunick A. Mission driven scene understanding: Dynamic environment. Adelphi (MD): Army Research Laboratory (US); 2016 Jun. Report No.: ARL-TN-0764.

6. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proc. Advances in Neural Info Processing Sys. 2012;25.

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

8. 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.

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

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

11. 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.

12. Collobert R, Kavukcuoglu K, Farabet C. Torch7: A MATLAB-like environment for machine learning. Proc Advances in Neural Info Processing Sys. 2011.

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

14. Yu D, Yao K, Zhang Y. The computational network toolkit. IEEE Signal Processing Magazine. 2015:123–126.

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

16. Neon deep learning library (DLL). San Diego (CA): Nervana Systems; 2016 Jul. 15. [accessed 2016 Jul]. http://neon.nervanasys.com/docs/latest/index.html.

17. Deeplearning4j: Open-source distributed deep learning for the JVM. San Francisco (CA): Skymind; 2016 [accessed 2016 Jul]. http://deeplearning4j.org.

18. Vedaldi A, Fulkerson B. VLFeat: An open and portable library of computer vision algorithms. Proceedings of ACM International Conference on Multimedia; 2010; Firenze, Italy.

19. 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. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 2015; Boston, MA.

20. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556v6; 2015.

21. 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.

22. Vedaldi A, Karel L. Matconvnet: convolutional neural networks for MATLAB. Proceedings of 23rd ACM International Conference on Multimedia; 2015; Brisbane, Australia.

23. Ma H. University of Guelph, Ontario. Personal communication, 2016.

1 (PDF)	DEFENSE TECH INFO CTR DTIC OCA
2 (PDF)	US ARMY RSRCH LAB IMAL HRA MAIL & RECORDS MGMT RDRL CIO L TECHL LIB
1 (PDF)	GOVT PRNTG OFC A MALHOTRA
9 (PDF)	US ARMY RESEARCH LABORATORY RDRL CII B BROOME M THOMAS RDRL CII A S YOUNG A TUNICK D BARAN G WARNELL P DAVID RDRL CIN T R MEYERS K DEACON

INTENTIONALLY LEFT BLANK.