LSTM Recurrent Neural Networks for High Resolution Range Profile Based Radar Target Classification

V Jithesh
Electronics & Radar Development Establishment, DRDO,
C. V. Raman Nagar, Bangalore, India, jithesh_lrde@rediffmail.com

Justin Sagayaraj M,
Member, IEEE, Electronics & Radar Development Establishment, DRDO,
Bangalore, India,
justin.sagayaraj.m@lrde.drdo.in

Dr. KG Srinivasa,
CBP Govt. Engineering College,
Jaffarpur, New Delhi,
kgsrinivasa@gmail.com

Abstract— **Positive and timely identification of targets is critical in any military scenario. Target identification from backscattered electromagnetic energy is an evolving area. The objective of this paper is to study the applicability of Long Short-Term Memory Recurrent Neural Network (LSTM RNN) for High Resolution Range Profile (HRRP) based Radar target classification. Simulated Radar Range Profile data is used here. Three Different Target models are considered in this study. The classification is performed using a LSTM RNN.**

Keywords—Radar, HRRP, Neural Network; Deep Learning; DNN; CNN; RNN; LSTM-RNN; Computational Network; CNTK;

I. Introduction

Artificial Neural Networks (ANN) have evolved into mature cognitive techniques which can be applied to variety of applications. The advances in ANN made deep learning techniques feasible in many applications. Deep Learning methods have become popular cognitive approach to impart computational intelligence to machines that perceive and understand the world. These methods use deep architectures to learn high-level feature representations. Convolutional Neural Network (CNN) is an improvement over DNN [12]. But CNNs cannot make use of dependencies and correlations between adjacent samples in an input sequence. Recurrent Neural Networks (RNN) addresses this deficiency, but are difficult to train and have difficulty modelling long-range dependencies. The LSTM RNN solves this problem by employing a set of gates.

A study on target classification using RNN is presented in [11]. The emphasis and contribution of the work presented here is on the usage and implementation aspects of LSTM-RNN for high range resolution profile (HRRP) based target classification. Simulated HRRP data from three different classes of targets are considered for evaluation.

Positive and timely identification of targets is critical in any military scenario. In case of a non-cooperative target, target identification must be performed purely based on the backscattered electromagnetic energy. Among the information from a typical radar signal, potential targets have to be classified correctly. A HRRP is the phasor sum of the time returns from different scatterers on the target located within a resolution cell. From a geometric point of view, a HRRP represents the projection of the apparent target scattering centers onto the range axis. The HRRP can provide information on the position and scattering strength of the targets scattering centres at that aspect [1], [2].

High Resolution Radar Range Profile can be used for automatic Radar target identification. HRRP represents a one dimensional range projection of a target's return onto the Radar Line of Sight and contains information about the geometry of the target. In other words HRRP plots scattered field versus range and shows the target's profile (in terms of scattering centres) in range dimension. The range profiles of a target are aspect dependent in the sense they vary with look angle. Thus different HRRPs can be generated for a target by varying the aspect angle. Classification of radar targets based on their HRRP has been studied in [3] - [11].

The remainder of this work is structured as follows. Section II reviews LSTM concepts and Section III discusses about the CN Toolkit. Simulated HRRP data is presented in Section IV. Implementation of the network and the results of the classification experiment are outlined in Sections V & VI respectively. In Section VII potential future work and conclusions are outlined.

II. LSTM RNN

Feedforward networks have no memory in the sense they won't remember any of their past inputs. They consider only the current data value they have been exposed to.

Recurrent networks, on the other hand, use information in the input sequence to perform tasks that feedforward networks can't. They also take what they perceived one step back in time. Thus recurrent networks have use present and the recent past inputs to determine response to new data. They do have memory, but lack long-term memory.

978-1-5090-6218-8/17/$31.00 ©2017 IEEE

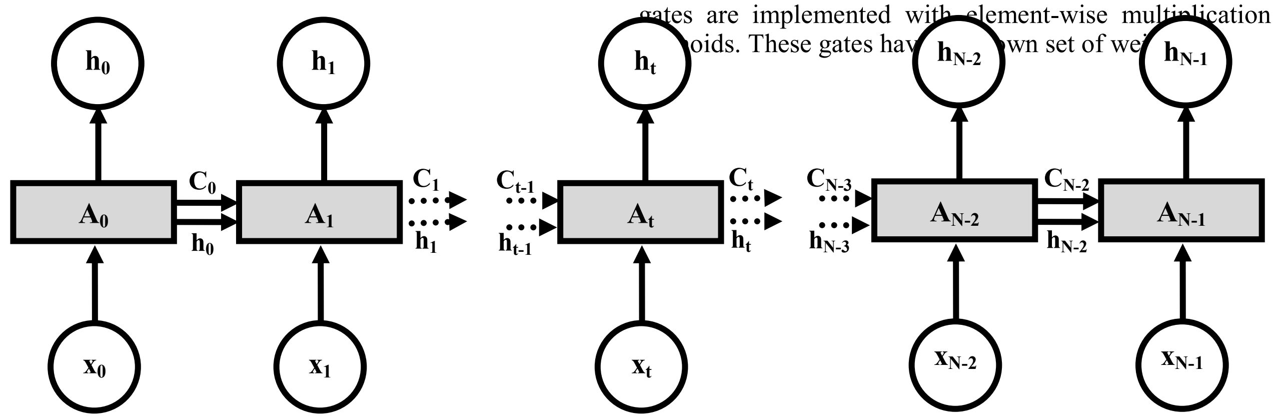

Fig. 1. A LSTM RNN [14]

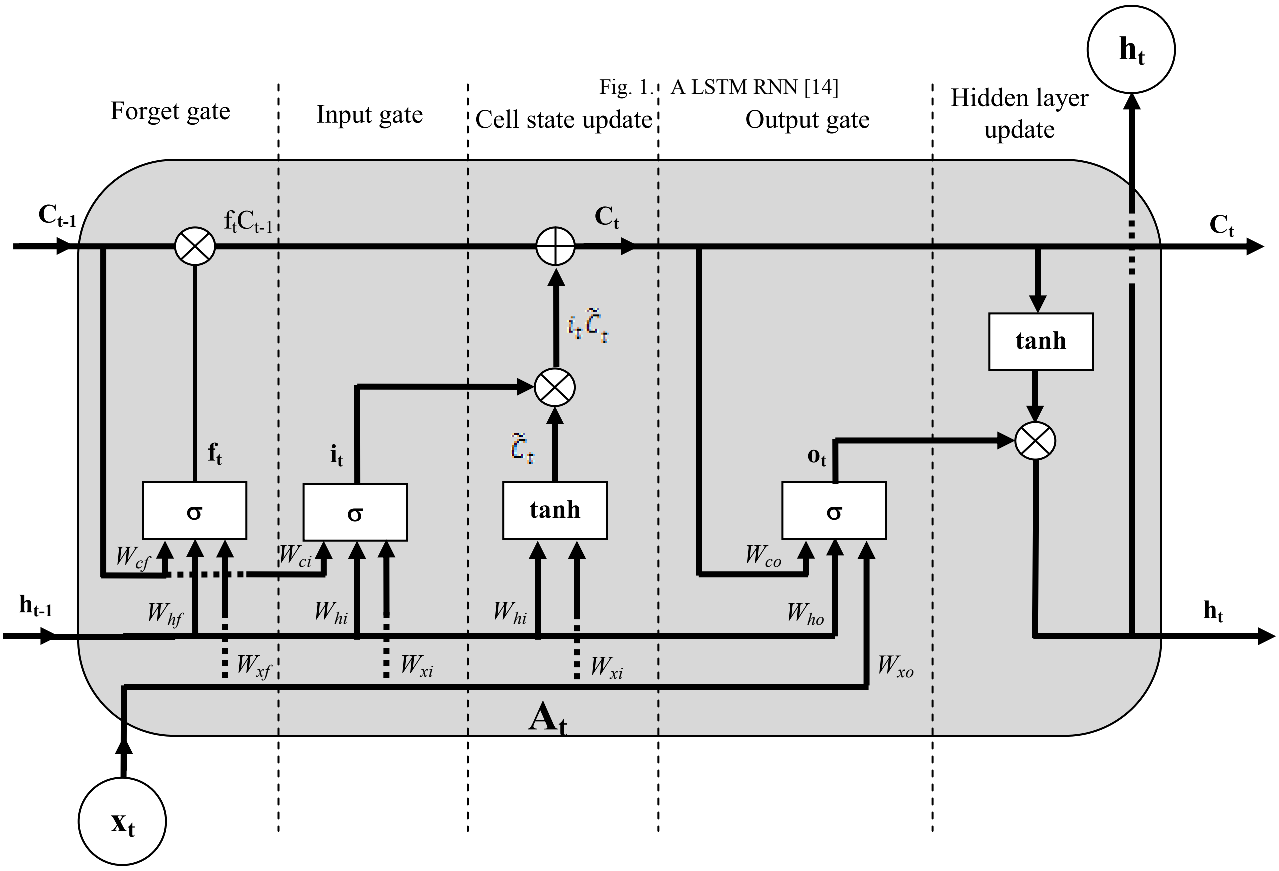

Fig. 2. A LSTM RNN Cell [14]

LSTMs can remember information for long time periods and help preserve the error that can be backpropagated through time and layers. Figure 1 shows a LSTM RNN with an input vector x of size N. The A_t's in this figure constitute the memory cell layer of the LSTM.

In LSTM, information get processed in a gated memory cell (the A_t's in figure 1). Information storing, writing, or reading operations are permitted in a cell. By controlling the gates a cell can make decisions about what data to be stored, and when to allow data reads, writes and deletions. These gates are implemented with element-wise multiplication by sigmoids. These gates have ... own set of weights

Figure 2 shows the different gates (forget, input & output gate) in a LSTM cell.

For each time step t, x_t is the input to the memory cell layer; i_t, f_t, o_t, C_t & h_t are values of the input gate (i-gate), forget gate (f-gate), output gate (o-gate), cell & the hidden layer respectively; W_{xi}, W_{xf}, W_{xc}, W_{xo} are the weights for the connections from the input layer to the i-gate, f-gate, cell & o-gate respectively; W_{hi}, W_{hf}, W_{hc}, W_{ho} are the weights for the connections from the hidden layer at time *t-1*to the i-gate, f-gate, cell & o- gate respectively; W_{ci}, W_{cf}, W_{co} are the weights for the connections from the cell at time *t-1* to the i-gate, f-gate & o-gate respectively; and b_i, b_f, b_c & b_o are the bias values for the i-gate, f-gate, cell & o-gate respectively.

Following update equations are used:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (2)$$

$$\tilde{C_t} = tanh(W_{xi}x_t + W_{hi}h_{t-1} + b_c) \quad (3)$$

$$C_t = f_t C_{t-1} + i_t \tilde{C_t} \quad (4)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (5)$$

$$h_t = o_t \tanh(C_t) \quad (6)$$

III. Computational Network Toolkit

Microsoft Computational Network Toolkit (CNTK) has been used in this work for modelling, training and evaluating the neural network. CNTK is a unified framework that makes it easy to design and test computational networks. A Computational Network (CN) is a directed graph in which each vertex, called a computation node, represents a computation and each edge represents a relationship between operator & operand. Each leaf node represents an input value or a model parameter and each non-leaf node represents a matrix operation upon its children. CNTK provides algorithms to carry out both forward computation and gradient calculation.

The goal of a computational network is to take feature data, transform the data through a network of simple computations, and then produce one or more outputs. The output is often some sort of decision based on the input features. A computational network can take many forms such as feed-forward, recursive or convolutional and includes various forms of computations and non-linearities. The network's parameters are optimized to produce the "best" possible outcome for a given set of data and an optimization criteria [13].

A CNTK model needs to have tasks for training and testing the network. The three main configuration blocks for training define the network itself and the parameters for the training algorithm and the data reader. These include the Network Builder, Stochastic Gradient Descent & the Reader block..

A configuration file is used for configuring these information. This configuration file has to be specified as the first argument when calling the CNTK executable. The configuration file uses a specific syntax.

A sample One hidden layer Computational Network is shown in figure 3 [13].

IV. HRRP Data Smulation

The Range Profile Data was generated through Electromagnetic Simulation (Asymptotic solver). Three different target models (Perfect Electric Conductor) available in the public domain were used in this simulation. Generic versions of these models are shown in Figures 4, 5 & 6.

For each target, simulations were carried out for 128 equidistant frequencies in the range 1 GHz to 2 GHz and 90 aspect angles (11520 simulations). The Range profiles thus generated were stored to text files. Each profile covers the entire range of frequencies and is for one aspect angle. These data files were then combined and reformatted as per the data format of CNTK input data using a MATLAB script and was stored in a text file.

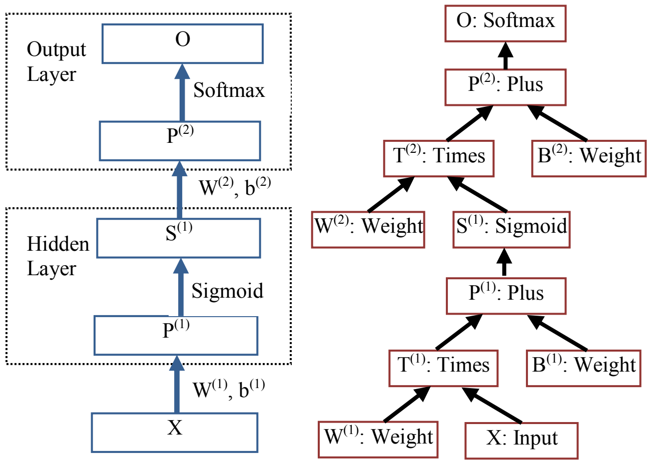

Fig. 3. A One Hidden Layer CN

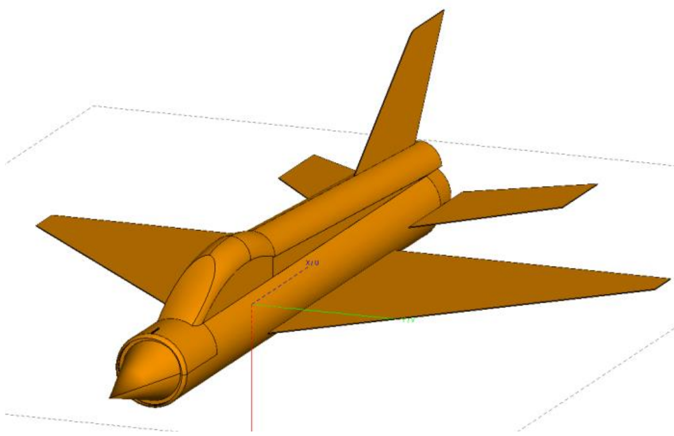

Fig. 4. Target Model - Class 1

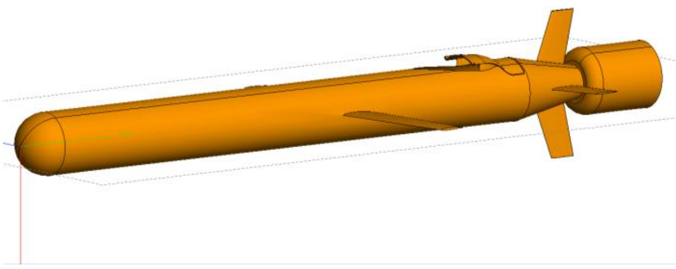

Fig. 5. Target Model - Class 2

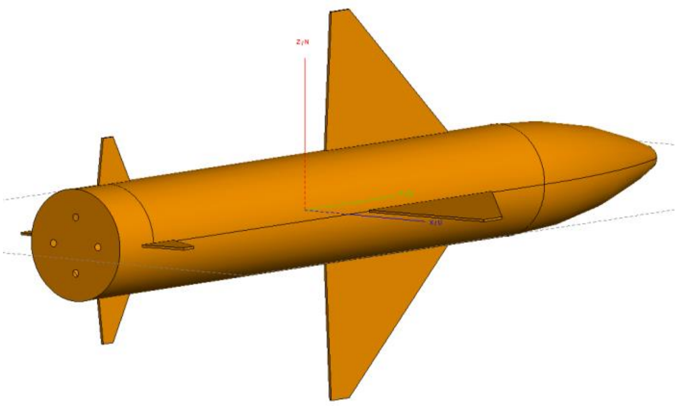

Fig. 6. Target Model - Class 3

|features	0.00142786	0.00049429	0.00039446
|features	0.00091088	0.00046459	0.00075628
|features	0.00053042	0.00075001	0.00080944
|features	0.00248232	0.0015191	0.00038259
|features	0.00150207	0.00063056	0.00159917
|features	0.00166206	0.00127518	0.0011356
|features	0.00154692	0.00129968	0.00041527
|features	0.00134937	0.00115759	0.00076522
|features	0.00100314	0.00048918	0.00162381
|features	0.00146144	0.00098491	0.00055125
|features	0.00074958	0.00151689	0.00190645
|features	0.00351138	0.00191006	0.00041826
|features	0.00209414	0.00153133	0.00125103
|features	0.00152648	0.00204124	0.00218804

Fig. 7. Formatted HRRP Profile - First 4 Column values

0.0012103	0.00137901	|labels 1	0	0
0.00205385	0.00202668	|labels 1	0	0
0.00171795	0.00155945	|labels 1	0	0
0.00099999	0.00061748	|labels 1	0	0
0.00207391	0.00359597	|labels 1	0	0
0.00054016	0.00172924	|labels 1	0	0
0.00115981	0.00109672	|labels 1	0	0
0.00088123	0.00104083	|labels 1	0	0
0.00219513	0.00325033	|labels 1	0	0
0.00187766	0.00182724	|labels 1	0	0
0.00135087	0.0002418	|labels 1	0	0
0.00105149	0.00050462	|labels 1	0	0
0.00113557	0.00110236	|labels 1	0	0
0.00068662	0.0002712	|labels 1	0	0

Fig. 8. Formatted HRRP Profile - End column values

Figure 7 shows a snapshot of the columns in the beginning of some of the HRRP profile data and figure 8 shows the end columns of the data set. Each row shows the HRRP for one aspect angle. The numbers 1, 0, 0 at the end of a row (after '*labels*') means that the target is of class 1. Here the '1' indicates a membership and '0' indicates a non-membership in a class. Thus 0, 1, 0 means that the target belongs to class 2 and 0, 0, 1 means it belongs to class 3.

V. IMPLEMENTATION

A Long Short-Term Memory RNN model is used here for classifying the HRRP profiles. The model has 128 node input layer, 3 node output layer and 50 node hidden layer. The BrainScript for the classification includes several commands like *train*, *test*, *write*, *plot* (for saving a CN as an image) & *dumpNode* (for displaying node information). CNTK 1.6 was used in this implementation.

The structure of a computation node is shown in figure 9. Details of the computation model in figure 9 is given below:

ce = CrossEntropyWithSoftmax (labels , z.z)
errs = ErrorPrediction (labels , z.z)
features = InputValue [128]
labels = InputValue [3]
P = Softmax (z.z)
z.b0 = LearnableParameter [50,1]
learningRateMultiplier = 1.000000
NeedsGradient = true
z.b1 = LearnableParameter [3,1]
learningRateMultiplier = 1.000000
NeedsGradient = true
z.r = RectifiedLinear (z.r._)
z.r._=Plus (z.r._.PlusArgs[0] , z.b0)
z.r._.PlusArgs[0] = Times (z.W0 , features)
z.W0 = LearnableParameter [50,128]
learningRateMultiplier = 1.000000
NeedsGradient = true
z.W1= LearnableParameter [3,50]
learningRateMultiplier=1.000000
NeedsGradient = true
z.z = Plus (z.z.PlusArgs[0] , z.b1)
z.z.PlusArgs[0] = Times (z.W1 , z.r)

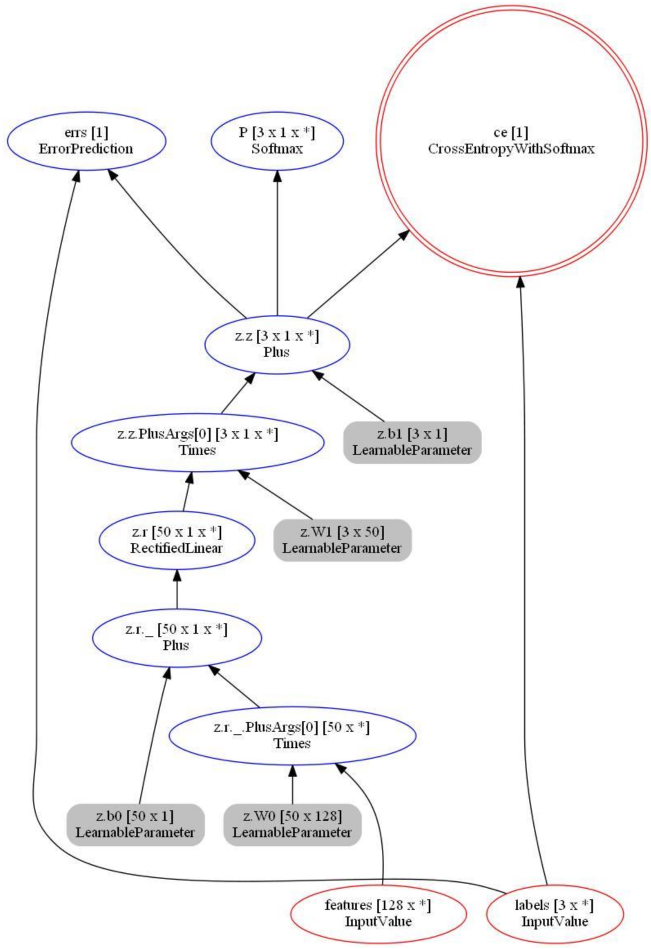

Fig. 9. CN for HRRP Classification

VI. Results

175 HRRP profiles were used for training the network and 100 profiles were used for testing (Holdout method, with 5 profiles common to both training & testing). Snapshots of the various stages in the execution are shown in figures 10, 11, 12 & 13. Figure 8 shows the training stage and figures 11 & 12 show the evaluation stage. In figure 12, *errs* represents the error percentage, *ce* represents the cross-entropy error and *perplexity* is *exp*(*ce*).

The values computed at the output layer are shown in figure 13. In this figure, the 3 columns in the data represent the 3 target classes. It should be noted that the values listed in this figure are either close to '0' or '1'. A value close to '1' in a column means that the target belongs to its class and a value close to '0' in a column indicates that the target does not belongs to its class. Thus value '1' in first column means that the target belongs to class 1 and so on.

```
##############################################################
#
# Action "train"
#
##############################################################

CNTKCommandTrainBegin: Train
Final model exists: Models/MC.dnn
No further training is necessary.
CNTKCommandTrainEnd: Train

Action "train" complete.
```

Fig. 10. Output of *train* action

```
##############################################################
#
# Action "test"
#
##############################################################


Post-processing network...

3 roots:
        P = Softmax()
        ce = CrossEntropyWithSoftmax()
        errs = ErrorPrediction()

Validating network. 14 nodes to process in pass 1.

Validating --> z.W1 = LearnableParameter() :  -> [3 x 50]
Validating --> z.W0 = LearnableParameter() :  -> [50 x 128]
Validating --> features = InputValue() :  -> [128 x *2]
Validating --> z.r._.PlusArgs[0] = Times (z.W0, features) :
Validating --> z.b0 = LearnableParameter() :  -> [50 x 1]
Validating --> z.r._ = Plus (z.r._.PlusArgs[0], z.b0) : [50
Validating --> z.r = RectifiedLinear (z.r._) : [50 x 1 x *2]
Validating --> z.z.PlusArgs[0] = Times (z.W1, z.r) : [3 x 50
Validating --> z.b1 = LearnableParameter() :  -> [3 x 1]
Validating --> z.z = Plus (z.z.PlusArgs[0], z.b1) : [3 x 1 x
Validating --> P = Softmax (z.z) : [3 x 1 x *2] -> [3 x 1 x
Validating --> labels = InputValue() :  -> [3 x *2]
Validating --> ce = CrossEntropyWithSoftmax (labels, z.z) :
Validating --> errs = ErrorPrediction (labels, z.z) : [3 x *

Validating network. 8 nodes to process in pass 2.


Validating network, final pass.
```

Fig. 11. Output of *test* action

```
Memory Sharing Structure:

0000000000000000: {[P Gradient[3 x 1 x *2]] [P Value[3 x 1 x *2]] [
00000000887D4D0: {[features Value[128 x *2]] }
00000000887D7F0: {[z.b0 Value[50 x 1]] }
00000000887D9D0: {[z.b1 Value[3 x 1]] }
00000000887DC50: {[z.W0 Value[50 x 128]] }
00000000887DCF0: {[z.W1 Value[3 x 50]] }
00000000887E010: {[errs Value[1]] }
00000000887E0B0: {[z.r._.PlusArgs[0] Value[50 x *2]] }
00000000887E150: {[z.z.PlusArgs[0] Value[3 x 1 x *2]] }
00000000887E1F0: {[z.z Value[3 x 1 x *2]] }
00000000887E290: {[z.r Value[50 x 1 x *2]] }
00000000887E330: {[labels Value[3 x *2]] }
00000000887E5B0: {[z.r._ Value[50 x 1 x *2]] }
00000000887E650: {[ce Value[1]] }
errs = 0.000% * 100; ce = 0.00048031 * 100; perplexity = 1.00048042
```

Fig. 12. Final Output after *test* action

```
18  0.000000 0.000000 1.000000
19  0.000000 0.000000 1.000000
20  0.000000 1.000000 0.000000
21  0.000000 1.000000 0.000000
22  0.997830 0.000847 0.001324
23  0.000000 0.000000 1.000000
24  0.000090 0.000000 0.999910
25  0.000000 1.000000 0.000000
26  0.998855 0.000520 0.000625
27  0.998719 0.000579 0.000702
28  0.999363 0.000314 0.000323
29  0.999467 0.000272 0.000261
30  0.999257 0.000356 0.000387
31  0.998314 0.000703 0.000983
32  0.998447 0.000723 0.000830
33  0.998183 0.000738 0.001079
34  0.000000 1.000000 0.000000
```

Fig. 13. Outcome of the Output Layer

It is evident from figure 12 that the network executed the test data without any error. Thus all the 100 target profiles (containing profiles of the 3 different targets) in the test data set were classified without any ambiguity.

VII. Conclusion

A Long Short-Term Memory RNN model has been implemented in this work to classify Radar targets based on their HRRP profile. Simulated profiles were used and the network was implemented using the CNTK framework. It has been observed that the model could classify the targets without any ambiguity. The results were compared against a simple RNN implementation (based on CNTK) and were matching. However, the performance of LSTM-RNN in classifying more complex targets and/or noisy data is yet to be explored and will be considered in future studies.

Future studies include classification of larger target classes and target classification based on Synthetic Aperture Radar (SAR) images. Applicability of Deep Learning techniques to other Radar related problems may also be considered in future studies.

Acknowledgement

The authors would like to thank Dr. Anil Kumar Singh & Mr. Suryaprakash K.S., Associate Directors, LRDE, for their valuable help, encouragement and motivation during the implementation of the work described in this paper.

References

[1] William L. Melvin, James A. Scheer (Editors), "Principles of Modern Radar - Vol. II: Advanced Techniques," SciTech Publishing, 2013.

[2] Caner Ozdemir, "Inverse Synthetic Aperture Radar Imaging with MATLAB Algorithms,"John Wiley & Sons, Inc., 2012.

[3] R. Williams, D. Gross, A. Palomino, J. Westerkamp, and D. Wardell, “1D HRR Data Analysis and ATR Assessment,” Air Force Research Laboratory Technical Report, 1998.

[4] Hsuch-Jyh Li, “Using Range Profiles as Feature Vectors to Identify Aerospace Objects,” IEEE Transactions on Antennas and Propagation, pp.261-268, March 1993.

[5] Anthony Zyweck, Robert E. Bogner, “Radar Target Classification of Commercial Aircraft,” IEEE Transactions on Aerospace and Electronic Systems, pp.598-606, April 1996.

[6] Xiao-Dan Wang, Chong-Ming Wu, “Uning Improved SVM Decision Tree to Classify HRRP,” Proceedings of the Fourth International Conference on Machine Learming and CyberneticsIEEE Transactions on Antennas and Propagation, pp.4432-4436, August 2005.

[7] Xiao Huaite, Guo Lei, Fu Qiang, "Radar Target Recognition Method Using Improved Support Vector Machines Based on Polarized HRRPs," International Conference on Computational Intelligence and Security, pp. 702-707, Nov 2006.

[8] Liu Mingjing, Zou Zhefeng, Hao Ming, "Radar Taret Recognition Based on Combined Features of High Range Resolution Profiles," Secon Asian-Pacific Conference on Synthetic Aperture Radar, pp. 876-879, Oct 2009.

[9] F. Benedetto, F. Riganti Fulginei, A. Laudani, G. Albanese, "Automatic Aircraft Target Recognition by ISAR Image Processing based on Neural Classifier," International Journal of Advanced Computer Science and Applications, pp. 96-103, Vol. 3, No.8, 2012.

[10] D. Chen and Z. Bao, “High Range Resolution Radar Target Identification Using Multilayer Feedforward Neural Network,” IEEE CIE International Conference of Radar Proceedings 1996, Piscataway, NJ, pp. 215–218.

[11] Xiao Huaitie; Zhuang Zhaowen; GuoGuirong, "Aircraft Target Recognition Based on Radar Range Profile Sequences Using Recurrent Neural Network," Journal of Electronics & Information Technology, pp. 386-391, Vol. 20, Issue (3), 1998.

[12] Dong Yu, Kaisheng Yao, and Yu Zhang, "The Computational Network Toolkit," IEEE Signal Processing Magazine, pp.123-126, November 2015.

[13] Amit Agarwal, et, al, “An Introduction to Computational Networks and the Computational Network Toolkit," MSR-TR-2014-112 (DRAFT v1.0: Feb 17, 2016.

[14] Christopher Olah, Understanding LSTM Networks, Posted on August 27, 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs.