Projectile Launch Point Prediction via Multiple LSTM Networks

Wisit Wiputgasemsuk

Abstract Launch point prediction of projectile targets is the main task of ground-based air surveillance radars in military services. Traditionally, the radar captures the portion of the projectile's trajectory to obtain time series data of position and velocity, and then uses an extrapolation method to determine the launch point. This work aims to enhance the accuracy of launch point prediction from the radar measurement data through the development of multiple Long-Short Term Memory (LSTM) networks. The proposed method consists of three stages. First, an LSTM-based filtering model is built to filter the noisy radar data and estimate the flight trajectory states. Second, an LSTM-based classifier model is created to identify the class of munitions based on their flight trajectories. Finally, multiple LSTM networks are developed to serve as the launch point predictor, in which each network is trained on different sets of munition classes. A switching law based on the classification result was established to select the best launch point predictor network for a specific class of munitions. The prediction results demonstrated the promising performance of the proposed method in launch point prediction. Furthermore, the correlation between the data distribution and the predictive performance of the network was explored.

Keywords Launch Point Prediction · Multiple Networks · Long-Short Term Memory

1 Introduction

Counter battery radars [1] play an important role in counter-fire operations. Their main function is to classify and forecast the launch point of hostile artillery, which typically includes rocket launchers, howitzers, and mortars. This facilitates the identification of the type of hostile artillery and the determination of its firing position through analysis of the flight trajectory of its shells. Radar is often used to capture the

W. Wiputgasemsuk (✉)
Aeronautical Engineering Division, Defence Technology Institute, Nonthaburi, Thailand
e-mail: wisit.w@dti.or.th

P. Meesad et al. (eds.), *Proceedings of the 19th International Conference on Computing and Information Technology (IC2IT 2023)*, Lecture Notes in Networks and Systems 679,
https://doi.org/10.1007/978-3-031-30474-3_10

portion of the flight trajectory of a projectile, allowing for the acquisition of time-history profiles of its position and velocity. Traditionally, to determine the launch point, the noisy radar measurement data is processed using a filtering technique to estimate the trajectory states of the projectile target. This information, along with a kinematic model of the projectile, is then used in a regression analysis to calculate the launch point [2–4].

In recent years, the advancement of neural networks has led to an increasing number of research studies that use this approach for the classification and launch point prediction of projectile targets. Carpenter et al. [5, 6] utilized the feed-forward neural network to identify the types of munitions, including 70 mm, 107 mm, and 122 mm, from radar measurement data. They also used the neural network as a functional representation instead of the piecewise solution from the traditional numerical calculation. Eckert et al. [7] employed a deep learning neural network to classify the variations of missiles. They built 12 distinct classes of missiles and added the simulated radar noise to each simulated trajectory within each class. Their results showed that the deep learning neural network achieved nearly 100% classification accuracy, while shallow neural networks did not perform well. Kim et al. [8] used an LSTM network for predicting the trajectory and launch point of ballistic objects from radar measurement data. The LSTM model was trained using true simulated trajectories. They used an Unscented Kalman Filter (UKF) to reduce measurement noise, and then used the filtered data for testing the performance of the LSTM model. Hou and Liu [9] utilized an LSTM network to estimate the real position of projectile trajectories from noisy radar measurements and developed a Mixture Density Network (MDN) for trajectory extrapolation and launch point prediction. They integrated the LSTM and MDN into a single end-to-end network, and trained it using radar measurement samples and their corresponding ground truth launch points. They also compared the traditional Extended Kalman Filter (EKF), a vanilla LSTM, and the LSTM + MDN for predictive performance.

In the literature, the use of neural network has been demonstrated to be effective in classifying and predicting the launch points of projectile targets. However, considering the munitions in the real world, they have various classes, each with unique flight trajectory characteristics. If the trajectory data for all classes were combined into a single training dataset, the resulting data would be highly diverse. Additionally, the radar measurement data is often noisy and restrictively available in practice, so training the network on the such dataset can lead to poor generalization performance. Indeed, some techniques, such as regularization, cross-validation, and early stopping callback, can help to improve the generalization. However, since the training data in this particular case is full of noise and contains outliers, more than those techniques might be required to achieve accuracy at an acceptable level. One such approach is to build separate networks for each class of munitions. By having prior knowledge that the characteristics of flight trajectories between classes are quite different, building several networks by training each on a certain class of munitions, could further improve the overall predictive abilities of the networks. This idea is not new. It was previously studied and analyzed in the literature. Jacops et al. [10] introduced a supervise learning procedure using the mixtures of local experts. Their idea is that

if we know in advance that a set of training cases may be naturally divided into subsets that correspond to distinct subtasks, interference can be reduced by using a system composed of several different "expert" networks plus a gating network that decides which of the experts should be used for each training case. Nguyen and Chan [11] presented multiple neural networks for long term time series prediction, with each model forecasting at a different time horizon. They showed that using multiple networks could yield better result than using a single network, as it could help reduce the propagation errors that grew during recursive prediction for long-range forecasts.

To this end, this paper proposes the method of utilizing multiple LSTM networks to enhance the accuracy of launch point prediction. The launch point predictor model consists of multiple LSTM networks which are trained on different sets of trajectory data. The LSTM-based classifier is built to identify the class of munition from its flight trajectories. The switching law based on the classification result is established to select the best launch point predictor network for each class of munitions. The main contributions of this paper are:

- Introducing the use of multiple LSTM networks to forecast the launch points of projectile targets from noisy radar measurement data, and
- Using multiple LSTM networks plus a switching law can improve the accuracy of launch point predictions compared to using a single LSTM network.

2 Methodology

2.1 Proposed Method

The basis of the proposed method is to make full use of LSTM's ability in sequence pattern recognition for processing, classifying and, predicting tasks. The framework of the proposed methodology is illustrated in Fig. 1. The method consists of three parts: the filter, the classifier, and the launch point predictor.

The first part of the proposed method is the filtering model. It is used to reduce the measurement noise and to estimate the real trajectory states. The LSTM network has shown effectiveness in filtering noise errors in measurement devices. For example, Jiang et al. [12] used the LSTM network to reduce random noise in MEMS IMU for navigation applications, which outperformed the conventional autoregressive moving average (ARMA) model. Hou et al. [9] used the LSTM network to predict the trajectory states of projectiles from noisy radar measurement data. In a similar way, this work uses the LSTM model to filter the noisy data in the first step. A segment of noisy radar data is fed into the filter model, which then outputs the estimated trajectory data of the same length as the input data.

The second part of the method is the classifier model, which predicts the munition class based on their trajectories. The estimated trajectory data from the filter is fed into the LSTM network, which then estimates the possibility to which of the class

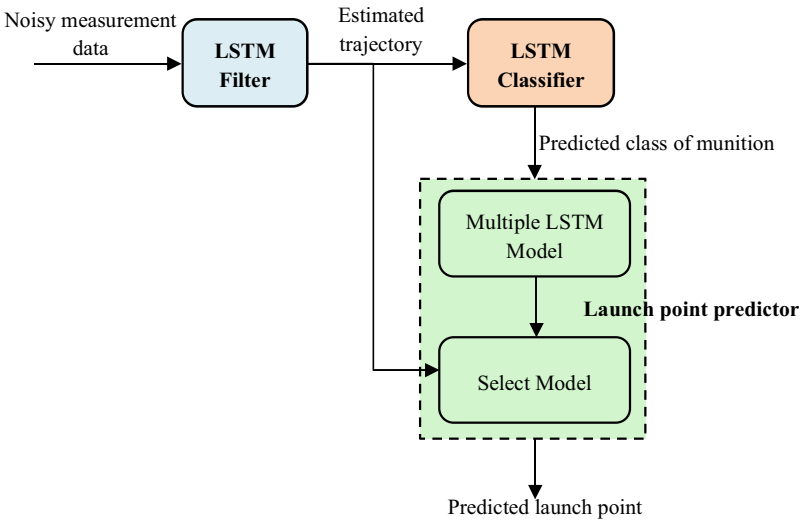

Fig. 1 Workflow of the proposed method

that the trajectories belong. The output of the model is the most probable class of munitions.

The third part is the launch point predictor model, which consists of multiple LSTM networks, each trained on a different set of munition classes. The predicted munition class is input into the switching law to select the best model for launch point prediction. The estimated trajectory data is then fed into the selected model, and the output is the predicted launch point.

2.2 Dataset

The radar measurement data used in this study was generated from the trajectory simulation program in the fire control system [13]. This program used the kinematic model of artillery rockets to create trajectories by varying simulation parameters such as launch elevation, launch azimuth, initial spin rate, air density, wind speed, wind direction, and others. The variation of these parameters was distributed according to a normal distribution with specified standard deviations. The resulting trajectory data included three-dimensional positions (x, y, z) at the update rate of 5 Hz (0.2 s per data point).

Four different classes of artillery rockets were created by configuring differently in thrust and aerodynamic drag. These differences resulted in distinct down range and apogees. Assume that rockets were launched from nine different launch points, with the radar station set as the origin coordinate (0, 0). To provide a clearer illustration, some samples of the projectile trajectories were picked to show in Fig. 2 (left). Table 1 summarizes the statistics of the trajectories in the training dataset. For each class in the dataset, there were 4,500 trajectories for training and an additional set of 900

trajectories for both validation and testing. The training, validation and testing data were in the ratio of 72:14:14, respectively. This ratio was appropriate since the validation and testing data were substantial enough to evaluate the model's performance on unseen data. Actually, the choice of ratio between the training and validation data depends on the size of dataset and the complexity of the model. Determining the optimal ratio for a specific problem and dataset requires an empirical process to minimize both the validation error and the error rate on the training set [14].

In general, a radar system can maintain tracking of a targets as long as the characteristics of the target and the operational environment are within the capabilities of the radar system. However, there are various factors that may cause a radar system to lose tracking such as target being out of range, target moving at high speeds, or the presence of interference like jamming. To represent the partially available radar data, the entire trajectory was segmented into multiple windows. By having longer window lengths, the accuracy of launch point prediction has the potential to increase. Assuming that the radar could track a target for 10 s during the time interval from 25 to 37 s after launch, and with the update rate of 5 Hz, the resulting window had a length of 50 time steps, starting from time step 125 and ending at time step 185. Shifting the window by one time step resulted in 11 overlapping windows for each trajectory. To make the data more realistic, additive white Gaussian noise with a

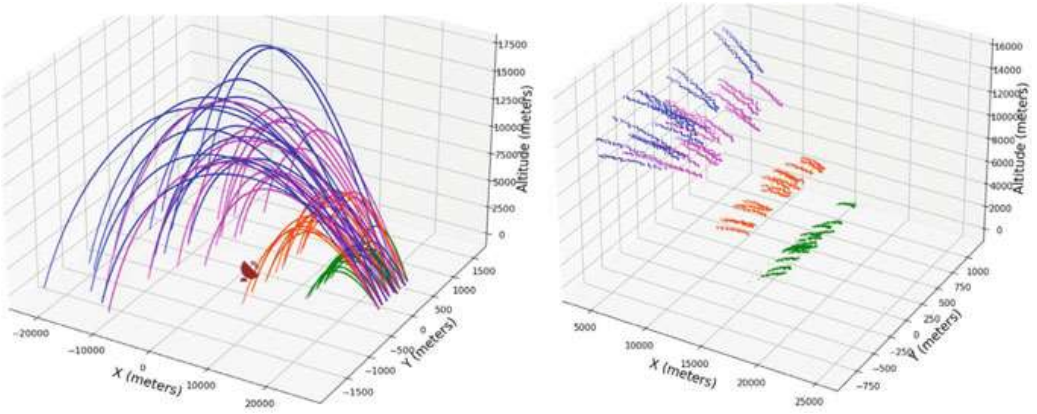

Fig. 2 (left) Projectile trajectories and (right) portions of noisy radar data

Table 1 Flight Statistics of 4 classes of rocket (***A***, ***B***, ***C***, ***D***)

Classes	Trajectories	Down Range (meters)			Cross Range (meters)			Apogee (meters)		
		Mean	Min	Max	Mean	Min	Max	Mean	Min	Max
A	4500	18,662	17,345	19,532	−18	−1909	1805	6512	3733	9453
B	4500	10,729	9910	11,131	6	−784	819	3379	1917	4794
C	4500	41,290	31,584	50,587	−21	−4344	3968	13,668	7082	22,255
D	4500	33,301	28,193	39,725	17	−3188	3687	11,476	6378	18,195

standard deviation of 10 m for each axis was applied to data points. Some examples of incomplete and noisy radar data are shown in Fig. 2 (right).

2.3 LSTM Networks and Training

All networks were implemented using the TensorFlow framework on a Windows10 system equipped with an Intel Xeon E3-1505 M 2.8 GHz CPU and 32 GB of RAM. Before training, the datasets were standardized using the StandardScaler function provided by Scikit-Learn. In the training process, the number of layers and neurons were tuned by random search and adjusted manually around the best value found by random search. At each time, the early stopping callback was used to avoid overfitting and to restore the best weights. The training process and hyperparameters for each network are as follows:

Filter: The network consisted of four layers, with the first being a flatten layer of 150 neurons (representing 50 time steps $\times$ 3 input features). The hidden layers were two LSTM layers with 28 and 20 neurons, respectively. The output layer was a dense layer with 150 neurons, matching the number of neurons in the first layer. In the training phase, the input sequence was the noisy trajectory data (x, y, z) with a length of 50 time steps, and the target was the true trajectory data with the same length as the input sequence. The Adam optimizer was used, with a batch size of 50 and the mean square error (MSE) as the loss function.

Classifier: The network consisted of four layers, starting with a flatten layer with 150 neurons. The second layer was an LSTM layer with 4 neurons, followed by a dense layer with 4 ReLU activation functions. The final layer had 4 cells with a softmax activation function to output the class probabilities. The output of the network was the class label with the highest probability. During training, the Adam optimizer was used, with a batch size of 100 and categorical cross entropy as the loss function. Accuracy was used as the evaluation metric.

Launch Point Predictor: To find the best launch point predictor for each class of munition, seven alternative LSTM networks were developed and trained on a different dataset, as outlined in Table 2. All alternative networks had the same architecture but differed in the number of neurons in the hidden LSTM layers. The first layer was a flatten layer with 150 neurons, followed by three LSTM layers with varying numbers of neurons, as listed in the third column of Table 2. The final layer was a dense layer with 2 neurons corresponding to the target (x, y). In the training phase, the Adam optimizer was used, with a batch size of 50 and the MSE as the loss function. The input sequence for training was the true trajectory data over 50 time steps, and the target was the corresponding actual launch points (x, y). The MSE losses for training and validation were listed in the last two columns of Table 2. We can observe that the MSE value for validation was slightly lower than the value for training. This was

Table 2 The network architecture and the MSE for training/validation

Networks	Datasets	Neurons	MSE	
			Training	Validation
1	***A***	22,20,16	14.67e-6	13.08e-6
2	***B***	20,18,14	12.61e-6	5.46e-6
3	***C***	20,18,14	13.84e-6	8.68e-6
4	***D***	20,18,14	6.29e-6	2.19e-6
5	$\boldsymbol{A+B}$	22,20,16	20.46e-6	4.53e-6
6	$\boldsymbol{C+D}$	22,20,16	31.66e-6	5.30e-6
7	$\boldsymbol{A+B+C+D}$	22,20,16	9.95e-6	2.18e-6

due to the early stop callback that stopped the training process when the validation error reached its minimum and started to increase.

3 Experimental Results

To evaluate the effectiveness of the proposed method, a new set of 3,600 trajectories (900 trajectories per class) with added white Gaussian noise was generated to serve as the testing dataset. The same preprocessing steps were applied once for the testing dataset, which involved standardizing the data and dividing it into a window with a length of 50 time steps. The testing process started with feeding the segment of noisy data into the filter model, which then output the estimated trajectory. The root mean squared error (RMSE) was used as a metric to evaluate the deviation between the estimated and actual trajectories. The result showed the RMSE of 4.3 m across the entire testing dataset, as illustrated in Fig. 3, which showed the close proximity between the estimated and actual trajectories.

The estimated trajectories were then input into the classifier model. The result of the classification accuracy was found to be 100%, despite the network having a simple structure with few neurons. This was likely due to the clear separation between the trajectories of each class, making it easier for the network to classify accurately. After that, the estimated trajectories were fed into the alternative networks, as listed in Table 2. Once again, the RMSE was used as a metric to evaluate the performance of all alternative networks in comparison. The RMSE for launch point prediction is defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{2n_{\mathrm{sample}}} \sum_{i=1}^{n_{\mathrm{sample}}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2}$$

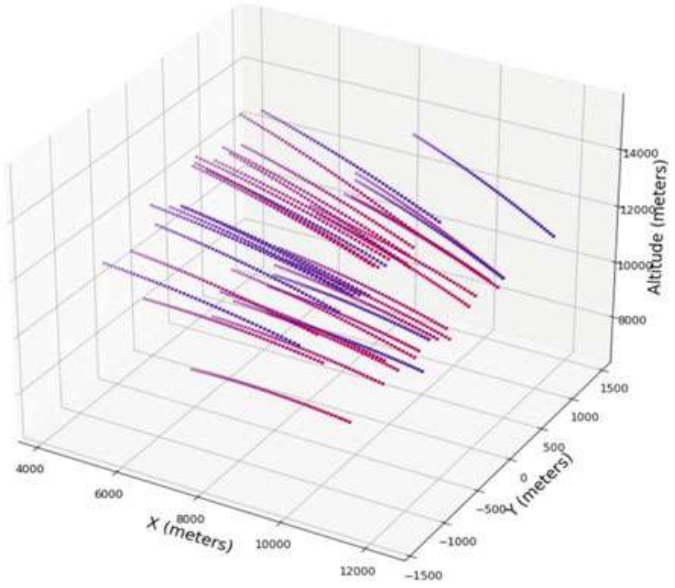

Fig. 3 Comparison of the estimated trajectories (blue dot) and actual trajectories (red cross)

where $\hat{x}_i$, $\hat{y}_i$ represent the i-th predicted launch point, and x_i, y_i represent the i-th actual launch point.

Table 3 compares the performance of all alternative networks on the launch point prediction. When comparing network ***7***, which was trained on all combined data, with the individual networks ***1***, ***2***, ***3***, and ***4***, which were trained on individual datasets ***A***, ***B***, ***C***, and ***D***, respectively, it can be seen that all the individual networks except for network ***3*** perform better than network ***7***. This result is consistent with the point of view that the data distribution in a specific dataset correlates with the network's predictive performance, regardless of how well the network is trained. Although all networks were comparably well-trained, network ***3*** which was trained on highly distributed ***C***-class data performs the worst, while network ***2*** which was trained on low distributed ***B***-class data performs much better. Combining the highly distributed ***C***-class data with lower distributed data (***A***, ***B***, ***D***) could give the trained networks (***6, 7***) better performance on ***C***-class prediction, but it worsened the prediction for other classes. It is clearly seen that the data distribution significantly affected the network's performance. Therefore, the multiple networks in implementation were the combination of networks that gave the least value of RMSE in each class, as indicated by the green boxes in Table 3. Based on these comparison results, the multiple networks then consisted of network ***1*** for ***A***-class prediction, network ***2*** for ***B***-class prediction, network ***7*** for ***C***-class prediction, and network ***4*** for ***D***-class prediction.

To implement a multiple-network scheme, the switching law was needed to select the network when receiving the predicted class from the classifier. For example, if the classifier predicts the ***B***-class, the switching law will select network ***2*** as the launch

Table 3 Comparisons of the predictive performance between all alternative networks in term of the RMSE values

Networks	Datasets	RMSE (meters)	RMSE (meters)			
			A	***B***	***C***	***D***
1	***A***	42.5	42.5	-	-	-
2	***B***	13.5	-	13.5	-	-
3	***C***	111.6	-	-	111.6	-
4	***D***	74.4	-	-	-	74.4
5	***A+B***	77.7	82.8	72.3	-	-
6	***C+D***	101.1	-	-	96.0	106.0
7	***A+B+C+D***	**80.4**	81.9	77.0	80.3	82.1
Multiple Networks		**59.1**	42.5	13.5	80.3	74.4

point predictor model, or if the classifier predicts the ***C***-class, the switching law will select network ***7***. The overall prediction results across the testing dataset could be obtained by combining the results of all selected networks. As shown in the last row in Table 3, the RMSE value of the multiple networks is 59.1 m, considerably reduced by 26% compared to the single network ***7***.

In the case where the classifier accuracy is not perfect, misclassification can deteriorate the overall predictive performance. For example, suppose the classifier misclassifies the actual B-class as the A-class, the switching law will select network ***1*** instead of the expected network ***2***, which leads to a worse prediction. Additionally, the accuracy of the launch point prediction can also be affected by the performance of the filter. Poor filtering will output a poorly estimated trajectory which deviates more from the actual trajectory and consequently gets worse in launch point prediction. The effect of the filter and the classifier on the results of launch point prediction were omitted here. However, it is said that all three parts of the proposed method - the filter, classifier, and launch point predictor - are critical to the accuracy of launch point prediction.

Figure 4 compares the launch point prediction of network ***7*** with the individual networks (***1***, ***2***, ***3***, ***4***). It can be seen that all the individual networks, except for network ***3***, make more accurate predictions for the launch points compared to the predictions from network ***7***. Notably, network ***2*** stands out from all other networks in terms of the dispersion of predicted launch points. Network ***2*** predicts most of the launch points close to the actual launch points, while the other networks predict more launch points located along the path between the true launch points.

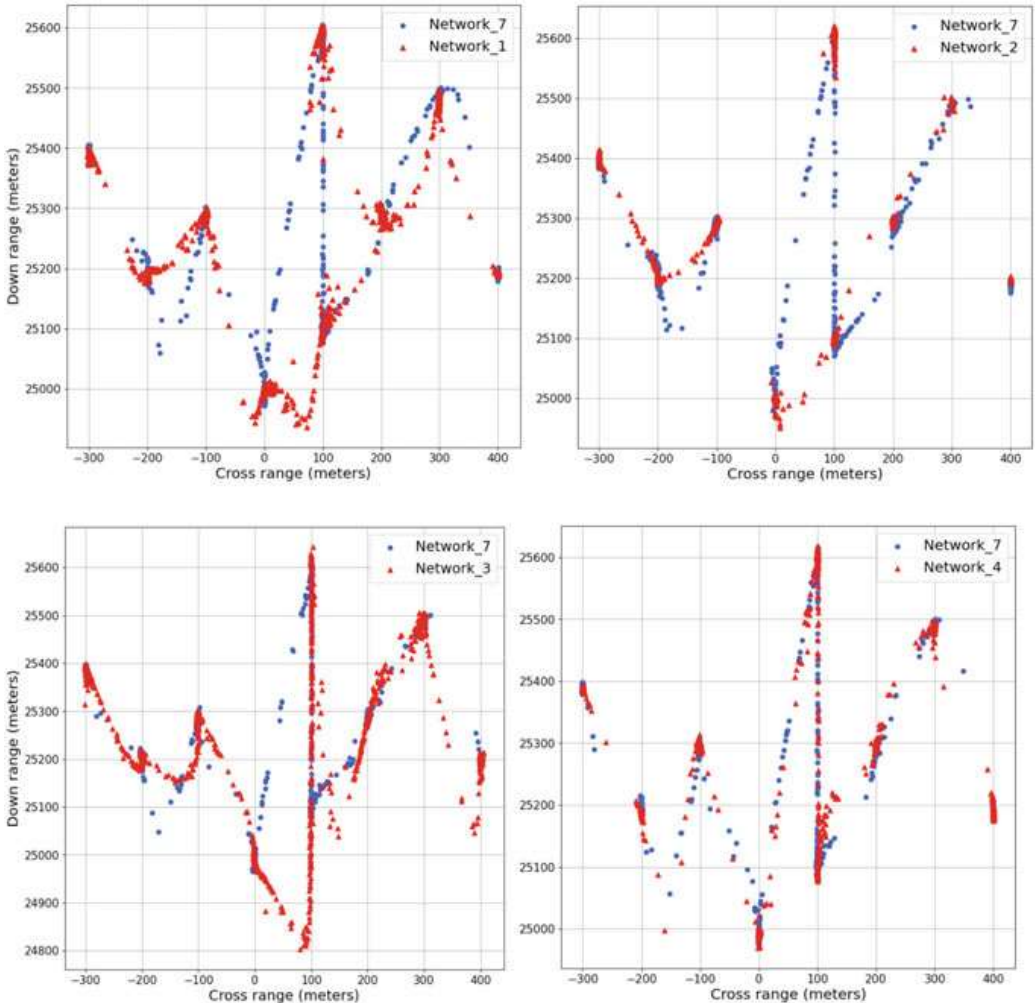

Fig. 4 Comparison of launch point predictions between network ***7*** and the individual networks (***1***, ***2***, ***3***, ***4***). The prediction by network ***7*** is represented as blue dots. The predictions by networks ***1***, ***2***, ***3***, and ***4*** are represented as red dots. The coordinates (x, y) of nine actual launch points are (25,000,0), (25,100,100), (25,200,400), (25,200,-200), (25,300,200), (25,300,-100), (25,400,-300), (25,500,300), and (25,600,100)

4 Conclusion

In this paper, the method for improving the accuracy of launch point prediction for projectile targets using multiple LSTM networks was presented. The method consisted of three parts. The first part involved using the LSTM-based filtering model to filter the noisy radar data and estimate the flight trajectory. The second part involved using the LSTM-based classifier model to identify the class of munitions based on its flight trajectory. Finally, multiple LSTM networks were developed as the launch point predictor model. The switching law was established based on the classification result, which enabled the selection of the most suitable launch point predictor network for each class of munitions. The results showed a considerable improvement in the

accuracy of launch point prediction when using multiple LSTM networks compared to a single LSTM network. Furthermore, the predictive performance was found to be inherently correlated to the distribution of the data.

Acknowledgement This original work forms part of a CS/AH2 light howitzer project of Defence Technology Institute, the support and encouragement of which is herein acknowledged.

References

1. Weapons Locating & Counter Battery Radars. https://www.defenseadvancement.com/suppliers/counter-battery-radars/. Accessed 28 Dec 2022
2. Eric, N., Meir, P., Stantio, M.: Projectile launch point estimation from radar measurements. In: 2005 American Control Conference, vol. 2. pp. 1275–1282 (2005)
3. Farina, A., Timmoneri, L., Vigilante, D.: Classification and launch-impact point prediction of ballistic target via multiple model maximum likelihood estimator (MM-MLE). In: 2006 IEEE Conference on Radar, pp. 802–806 (2006)
4. Ravindra VC, Bar-Shalom Y, Willett P (2010) Projectile identification and impact point prediction. IEEE Trans Aerosp Electron Syst 46(4):2004–2021
5. Carpenter, M., Speakman, N., Hartfield, R.J.: Rapid characterization of munitions using neural networks. In: AIAA Atmospheric Flight Mechanics Conference, American Institute of Aeronautics and Astronautics (2016)
6. Carpenter, M., Hartfield, R.J., Zhou, L., Speakman, N.: Statistical learning for munition trajectory prediction. In: AIAA Scitech 2019 Forum, American Institute of Aeronautics and Astronautics (2019)
7. Eckert, J., Carpenter, M., Hartfield, R., Cervantes, N.: Classification of intermediate range missiles during launch. In: AIAA Scitech 2020 Forum, American Institute of Aeronautics and Astronautics (2020)
8. Kim, J., Lim, M.C., Park, S.-S., Kim, I., Choi, H.-L.: Ballistic object trajectory and launch point estimation from radar measurements using long-short term memory networks. In: 2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA), pp. 26–31 (2019)
9. Hou, L., Liu, H.: An end-to-end LSTM-MDN network for projectile trajectory prediction. In: Cui, Z., Pan, J., Zhang, S., Xiao, L., Yang, J. (eds.) Intelligence Science and Big Data Engineering. Big Data and Machine Learning. IScIDE 2019. Lecture Notes in Computer Science, vol. 11936, pp. 114–125. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36204-1_9
10. Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. Neural Comput 3(1):79–87
11. Nguyen HH, Chan CW (2004) Multiple neural networks for a long term time series forecast. Neural Comput Appl 13(1):90–98
12. Jiang, C., et al.: A MEMS IMU De-noising method using long short term memory recurrent neural networks (LSTM-RNN). Sensors **18**(10), 3470 (2018)
13. Charubhun, W., Chusilp, P.: Development of automatic firing angle calculation for ground to ground MLRS. In: 2015 Asian Conference on Defence Technology (ACDT), pp. 17–26 (2015)
14. Guyon, I., Laboratories, T.B.: A scaling law for the validation-set training-set size ratio (1997)