CS230

Characterization and Localization of GNSS Interference Events

Zixi Liu
Department of Aeronautics
Astronautics
Stanford University
liu1322@stanford.edu

Abstract

Global Navigation Satellite System (GNSS) interference events happen around airports could cause severe safety issues such as denial of GNSS based landings. Current solutions such as radio direction finding technique are time-consuming. There exists no previous research on applying deep learning algorithms to this topic. We designed standard NN and ConvNets to take airplane's position reports as inputs and output a classification of whether these airplane has been jammed. We achieved 96.7% and 97.1% f_{beta} score for both models we designed.

1 Introduction

GNSS has become a safety-of-life system in aviation. Losing GNSS signals on approach to land could be catastrophic. Therefore, we want to design a system to detect the existence of GNSS interference event and provide Air Traffic Control (ATC) situational awareness. One way to detect the existence of jamming event is by monitoring the Automatic Dependent Surveillance—Broadcast (ADS-B) reports broadcast by the airplane. Figure 1 shows how interference event affects ADS-B outputs.

In this project, we designed one standard neural network and one convolutional neural network to help detect the existence of interference event. The input of standard NN is one ADS-B message which is a $(5, 1)$ vector, and the output is a binary classification of whether or not the given point has been jammed: $(\hat{y} = 1)$ for jammed point. The input of CNN is all ADS-B messages from one flight which is represented as a $(25, 25, 5)$ matrix, and the output indicates whether or not the flight has been jammed.

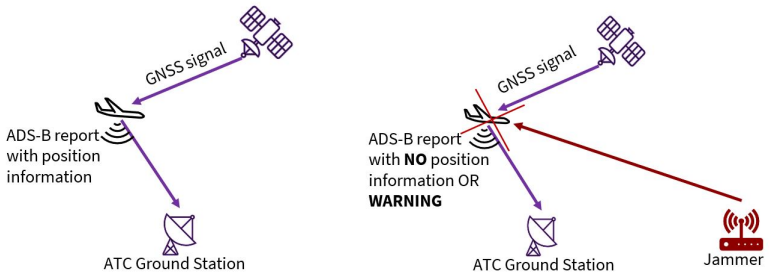


Figure 1: ADS-B performance under normal circumstance versus during interference event

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

There exists no previous research on applying deep learning algorithms to this topic. All prior works focus on solving the problem from signal transmissions perspective and performed some statistical analyses.

For instance, researchers from Czech Technical University analyzed the change of Navigation Accuracy Category – Position (NACp) probability distribution, which is an integrity feature of ADS-B operational status message [(1)]. Similar work has also been done by researchers from New York University [(2)]. EUROCONTROL has developed a grid probability model to calculate and generate heatmaps for possible location of the RFI source [(3)].

3 Dataset and Features

This project uses ADS-B data queried from OpenSky Network [(4)]. The data of this project is structured data shown in Figure2. Each row is one ADS-B message and we choose 5 features for this project including latitude, longitude, altitude, time and Navigation Integrity Category (NIC). NIC indicates the accuracy level of current position message, the higher NIC value means the more accurate the information is.

This project collected 4.52GB excel files which contains 8,491,752 ADS-B messages from 5,484 flights. The entire dataset is split into training/validation/test set with 80%/10%/10%. Data is labeled as jammed $(y = 1)$ based on definition of ADS-B anomalies from 14 CFR § 91.227(c)[(5)].

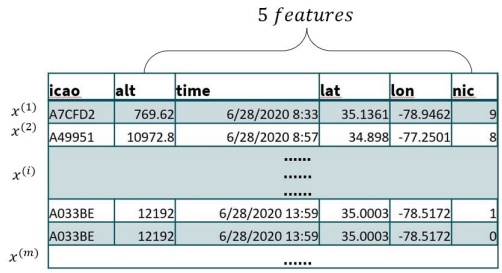


Figure 2: Information contained by ADS-B dataset

3.1 Data Preprocessing

Our dataset has class imbalance problem. Only 8% of data were jammed during each interference event. Specifically, there are 7,812,722 normal points and only 679,030 jammed points. Therefore, we solved this problem by down-sampling the false class to match with the numbers of true class data$\{False : 7,812,722 \;\; True : 679,030\} \rightarrow \{False : 679,030 \;\; True : 679,030\}$. Resulted in a balanced numbers of True/False class as 50%/50%.

This dataset is normalized into zero mean and one standard deviation because all parameters shown in Figure2 are on different scales. Parameter 'time' is converted into time differences by subtracting the first timestamp of the day from all timestamps. This step helps convert data type from datetime to seconds.

3.2 Data representation for CNN

We represent the structured data of one flight into a 3 dimensional matrix in order to feed into a ConvNet. This process is shown in Figure3. We separated each flight path into multiple parts such that each part contains exact 625 ADS-B data points. This number of 625 is picked based on the size of airspace, average length of flights, and expert advice. This $(625, 1)$ vector is then converted into $(25, 25)$, and because each ADS-B message has 5 features, the input data has 5 input channels similar to RGB channels in an image. Therefore, that leads to the $(25, 25, 5)$ matrix shown in Figure3.

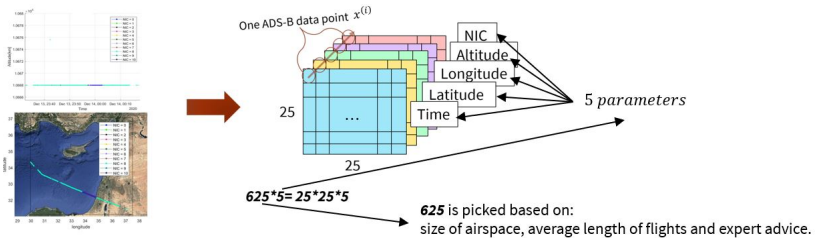


Figure 3: Data representation from vector to 3D matrix for CNN

4 Methods

In this project we explored two different models to perform GNSS interference event detection. Standard neural network can be used to identify whether any specific location has been affected. ConvNet can be used to identify whether one specific flight has been jammed.

4.1 Standard Neural Network

The final model architecture is shown in Figure4. Each training example is one ADS-B data $\in \mathrm{R}^{5*1}$ which contains 5 parameters. The output is 1 if that point has been jammed and 0 otherwise.

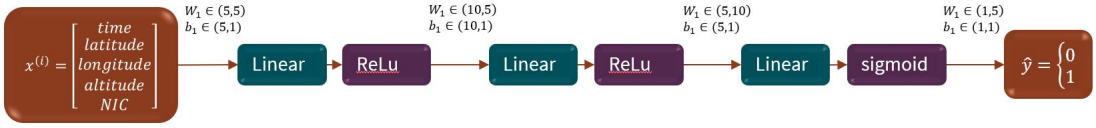


Figure 4: Model architecture of Standard Neural Network

This model is trained by minimizing the difference between true label y and predicted result $\hat{y}$ using equation 1. One important thing we noticed is that the cost of our model tends to reach saturation during the first few iterations. Therefore, we implement HE initialization scheme shown in equation 2, and learning rate decay from $\alpha = 0.1 \rightarrow \alpha = 0.001$. These helped make sure the cost does not stop at local minimum and converges in the end.

$$J = \frac{-1}{m} \sum_{i=1}^{m} [y^{(i)} log(\hat{y}^{(i)}) - (1 - y^{(i)}) log(1 - \hat{y}^{(i)})] \tag{1}$$

$$W_{i,j}^{[l]} = \mathcal{N}(0, \frac{2}{n^{[l-1]}}) \tag{2}$$

4.2 ConvNet

We are inspired by the idea of using ConvNet to classify images. We designed a ConvNet to classify whether an interference event exists by looking at different types of jammed or non-jammed flights. Figure5 illustrates the final model architecture. The data representation process for this model is mentioned in section 3.2.

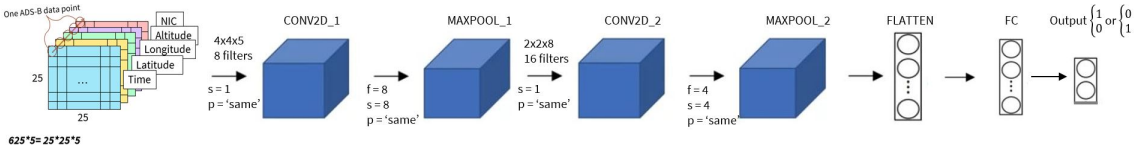


Figure 5: Model architecture of CNN

5 Experiments/Results/Discussion

Figure6 shows summary of key steps of experiments and corresponding results for both models. For **standard NN** shown on the left hand side, each experiment is done with 950,642 training examples and 407,418 testing examples. We decided to evaluate the result using $\mathbf{F}_2$ **score** in order to put more weight on recall. That is because we do not want any false negatives since failing to reveal the existence of jamming event could cause severe issue. training with 3,839 examples and testing with 1,645 flights. For **CNN** shown on the right hand side, each experiment is done with 3,839 training flights and 1,645 testing flights. We decided to evaluate the result using $\mathbf{F}_{0.5}$ **score** in order to put more weight on precision. Results from confusion matrix of CNNs indicates most false points from our CNN models come from false positives.

Experiment Detail	F_beta (β =2) (Train, Test)	AUROC
NN with 1 hidden layer, 5 hidden units. α = 0.01. ***Random initialization***.	0.776, 0.778	0.8186
NN with 1 hidden layer, 5 hidden units. α = 0.01. ***He initialization***.	0.912, 0.913	0.9396
NN with 1 hidden layer, 5 hidden units. $\boldsymbol{learning\ rate\ decay\ \alpha = 0.1 \rightarrow \alpha = 0.001}$. He initialization.	0.943, 0.944	0.9552
NN with 2 hidden layer, each has 5 hidden units. learning rate decay $\alpha = 0.1 \rightarrow \alpha = 0.001$. He initialization.	0.918, 0.912	0.8968
NN with 3 hidden layer, 1st layer has 5 hidden units, 2nd layer has 10 hidden units, 3rd layer has 5 hidden units. learning rate decay $\alpha = 0.1 \rightarrow \alpha = 0.001$. He initialization.	0.960, ***0.967***	***0.9635***

Experiment Detail	F_beta ($\beta = 0.5$) (Train, Test)	AUROC
CNN with [conv (4x4x8) → conv (2,2,16)]. One fully connected layer. 100 epochs, batch size = 32.	0.714, 0.706	0.7399
CNN with [conv (4x4x8) → conv (2,2,16)]. One fully connected layer. 100 epochs, ***batch size = 64***.	0.939, 0.936	0.9177
CNN with [conv (4x4x8) → ***max pooling (8,8,8)*** → conv (2,2,16)]. One fully connected layer. 100 epochs, batch size = 64.	0.963, 0.9572	0.9536
CNN with [conv (4x4x8) → max pooling (8,8,8) → conv (2,2,16) → max pooling (4,4,16)]. One fully connected layer. 100 epochs, batch size = 64.	0.976, ***0.971***	***0.984***

Figure 6: Table of key experiments and results (standard NN on the left, CNN on the right)

Few important things we noticed during experimentation is that both two models are sensitive to learning rate and batch size. That is because of the cost of our models tend to reach stagnation at first few iterations, need to **make sure the learning step is not too small** otherwise the cost will stay at the local minimum and learn nothing. Another thing we noticed is that the training and testing set always have similar f_β score no matter how we modify the model architecture. This is the reason why we **do not add regularization term**, since the model is hardly to become overfitting. The details about parameters in our final model archtecture is shown in the last row in Figure6. We achieved **96.7%** and **97.1%** $\mathbf{f}_{beta}$ **score** for both models we designed.

The qualitatively results of our final design is shown in Figure7. The predicted result from NN is shown on the left hand side, and it **matches quite well** with humanly labeled result which is shown on the right hand side. Red points are jammed points and green points are non-jammed/regular points. The overall impact area of the jammer is indicated by the crowd of red jammed points.

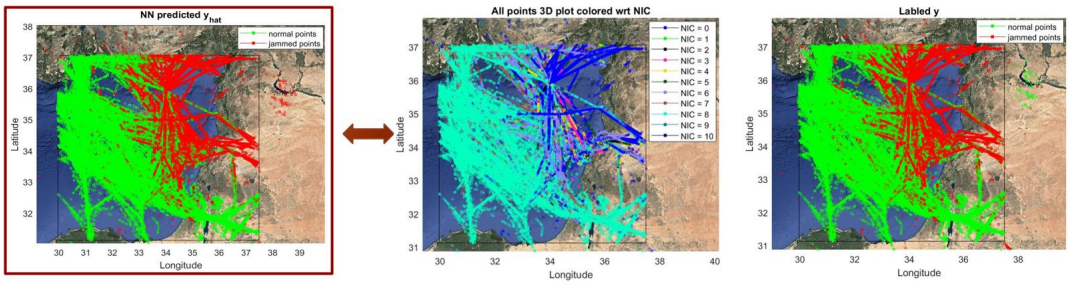
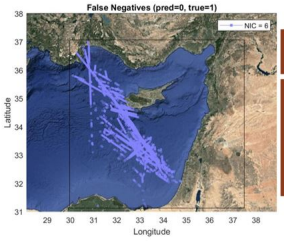


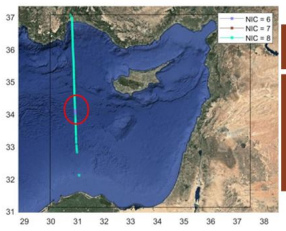
Figure 7: 3D plot (top view) of all ADS-B data in testing set

Another interesting result we noticed is that most of the false positives and false negatives from both models, shown in Figure8, are also points which we were not sure whether or not they were jammed. We struggled a lot during labeling those data points. For instance, we commonly use NIC = 7 as a threshold, for points with NIC < 7, we believe the accuracy level is low and the point should be affected. However, 7 is not a clear cut, it could not indicate for sure that NIC = 6 means point has

been jammed and NIC = 8 means the point is good. Noticed on the left hand side of Figure8, all points have NIC = 6 and all of them are far away from the possible impact area of the jammer. This means we do not even know whether saying those points were not been jammed is a false statement. Similar to the false negative flight shown on the right hand side.



	Predicted Label 0	1
Actual Label 0	TN 133908	FP 2200
Actual Label 1	FN 9580	TP 124270



	Predicted Label 0	1
Actual Label 0	TN 759	FP 47
Actual Label 1	FN 6	TP 833

Figure 8: Sampled false negatives from both models

6 Conclusion/Future Work

In this project, we designed standard NN and ConvNets to take airplane's position reports as inputs and output a classification of whether these airplane has been jammed. We achieved 96.7% and 97.1% f_{beta} score for both models we designed.

In the future, we would hope to perform more experiments on current models by performing further analysis on false positives and false negatives. In addition, we would hope to design a DL model to be able to output highest possible location of the jammer. This requires obtaining dataset from interference event that contains information about specific location of the jammer. This requirement is difficult to achieve since all GPS interference testing events hardly provide information about the interference source. But once obtained the data, what we have already trained could still be helpful, we could apply transfer learning to building new DL models for this topic.

References

[1] P. Lukeš, T. Topková, T. Vlček and S. Pleninger, "Recognition of GNSS Jamming Patterns in ADS-B Data," *2020 New Trends in Civil Aviation (NTCA)*, Prague, Czech Republic, 2020, pp. 9-15, doi: 10.23919/NTCA50409.2020.9291039

[2] Darabseh, A., Bitsikas, E., and Tedongmo, B., "Detecting GNSS Jamming Incidents in OpenSky Data,". *EPiC Series in Computing, Volume 67, 2019, Pages 97-108.*

[3] Jonáš, P., and Vitan, V., "Detection and Localization of GNSS Radio Interference using ADS-B Data,". *2019 International Conference on Military Technologies (ICMT)*, Brno, Czech Republic, 2019, pp. 1-5, doi: 10.1109/MILTECHS.2019.8870034.

[4] Schäfer, M., Strohmeier, M., Lenders, V., Martinovic, I., and Wilhelm, M., "Bringing up OpenSky: A large-scale ADS-B sensor network for research". *In Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, Berlin, Germany, 15–17 April 2014.

[5] Doc. No. FAA-2007-29305, 75 FR 30194, May 28, 2010; Amdt. 91-314-A, 75 FR 37712, June 30, 2010; Amdt. 91-316, 75 FR 37712, June 30, 2010

[6] Syd, Busyairah., Schuster, W., Ochieng, W., and Majumdar, A., "Analysis of anomalies in ADS-B and its GPS data". *GPS Solutions (2015)*. 20. 10.1007/s10291-015-0453-5.