GNSS Interference Detection Using Machine Learning Algorithms on ADS-B Data

Zixi Liu, *Stanford University*
Sherman Lo, *Stanford University*
Todd Walter, *Stanford University*

BIOGRAPHY (IES)

Zixi Liu is a PhD candidate at the GPS Laboratory at Stanford University. She received her B.Sc. degree from Purdue University in 2018 and her M. Sc. degree from Stanford University in 2020.

Sherman Lo is a senior research engineer at the GPS laboratory at Stanford University.

Todd Walter is a Professor of Research and director of the GPS laboratory at Stanford University.

ABSTRACT

Global Navigation Satellite System (GNSS) interference events that occur near airports can cause severe safety issues by denying GNSS based approaches and landings. Current solutions for interference detection and localization (IDL) such as using radio direction finding are generally costly and time-consuming. The approach described in this paper uses Automatic Dependent Surveillance—Broadcast (ADS-B) reports for IDL and applies machine learning algorithms to this data. We utilized a standard neural network (NN) and a convolutional neural network (CNN) to detect GNSS interference event in a given airspace. These models take airplane's ADS-B reports as inputs and output a classification of whether this airplane has experienced jamming. With this approach, we achieved 83.6% and 90.4% accuracy for corresponding model. We used logistic regression as a baseline model which achieved an 75.1% accuracy.

This paper has two main objectives. First, our NN model is used to determine the size and shape of the jammer impact region. By achieving this purpose, we are able to identify the complicated environmental factors from local airspace such as the signal blockage caused by the mountains, which are commonly difficult to identify or represent using mathematical models. Second, our CNN model is used for identifying the most likely location of the interference source which only requires ADS-B data collected within few hours' time window from target airspace. The key significant step to this approach is that it is mimicking the way of how human identifies the location of interference source which is by looking at the overall picture of the airspace. In addition, it also learns the complicated reasons of how interference source could cause impact on the overall picture of ADS-B data in current airspace. In order to feed ADS-B data into CNN, we designed a method to convert 1-D structured ADS-B data into higher-dimensional matrix. This data preprocessing allows us to apply convolutional neural networks to this topic.

INTRODUCTION

GNSS has become a safety-of-life system in aviation. Losing GNSS signals on approach or landing could be catastrophic. Therefore, a system that can quickly detect the existence of GNSS interference event and provide Air Traffic Control (ATC) situational awareness can be highly useful to protect the safe use of GNSS in aviation. One way to detect the existence of a jamming event is by monitoring the Automatic Dependent Surveillance—Broadcast (ADS-B) reports broadcast by the airplane. Figure1 shows how interference event affects ADS-B outputs.

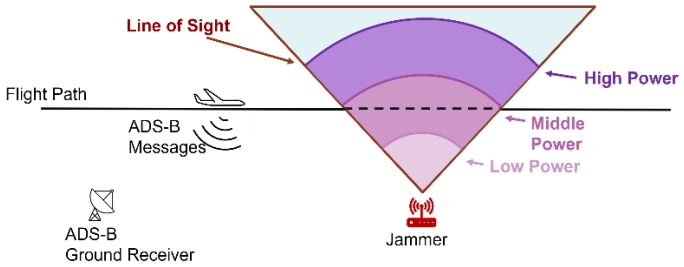


Figure 1: ADS-B performance under normal circumstance versus during interference

There are two important factors we need to consider during interference event. First is the signal line of sight of the interference source. Imagine an aircraft flying along the path, the jammer could only cause impact on the aircraft when the aircraft is flying on the part of the flight path that is within the line of sight. Once the aircraft is within the line of sight, the second factor we need to consider is the power level of the jammer. For instance, in Figure1, only the jammer with middle or high-level power could cause impact on the aircraft.

Figure 2 is an overview of the two models we built and trained in this project. The standard NN is trained by feeding one ADS-B message at a time, where each ADS-B data point contains latitude, longitude, and altitude information of the aircraft's current position. The output is a determination of whether the given point has been jammed: $\hat{y} = 1$ for jammed point or $\hat{y} = 0$ for a nominal (unjammed) point. The application of this model is to fill the entire target airspace with data points and feed all points into the trained NN model. The trained model will then determine which points have been jammed. All the points which are marked as being jammed, together shows the impact region of the jammer. The CNN model is trained by inputting the overall picture of flight tracks that are within the target airspace, in other words, each training example contains all ADS-B messages we collected during a few-hours' time window. The CNN outputs the most likely location of the jammer. Both models output the final decision based on which category has the maximum probability, therefore, we could also observe the overall trend in probability such as the probability contour of whether each location contains interference source.

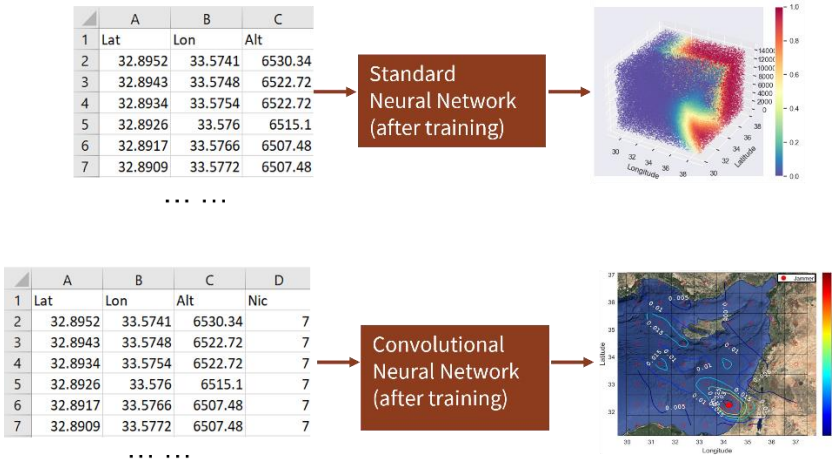


Figure 2: Overview of NN and CNN applications in this project

RELATED WORK

There exists no previous research on applying deep learning algorithms to ADS-B data for interference detection and localization (IDL). Most of the prior works on ADS-B for assessing GNSS interference focus on solving the problem from a signal transmission perspective combined with statistical analyses. For instance, researchers from Czech Technical University analyzed the change of Navigation Accuracy Category – Position (NACp) probability distribution [1]. NACp measures the horizontal and vertical position

uncertainty of provided ADS-B position solution. There are some other commonly seen accuracy and integrity level indicators in ADS-B messages such as Navigation accuracy category–velocity (NACv), Navigation integrity category (NIC), and Surveillance integrity level (SIL). In this project, we use NIC as one of our input parameters. This decision is made based on some of our previous study in characterizing ADS-B performance during interference events [2].

As for localization, EUROCONTROL has investigated the use of ADS-B to localize GNSS interference. They developed a grid probability model to calculate and generate heatmaps for possible location of the RFI source [3]. We proposed using convex optimization to identify possible location of interference source [4]. During our previous study, we noticed that it is difficult to represent or identify complicated environmental factors such as signal blockage by buildings or terrain elevation. The resultant mathematical model is overly idealistic when used for identifying possible location of interference source in real life. In this project, we tried to understand the environmental factors by identifying the size and shape of the impact area using decision boundary of NN.

In this project, we collected ADS-B data from interference events happened in Cypriot airspace based on the ongoing significant and known GNSS interference that occurs in that airspace [5]. In addition, the possible location of interference source was identified by C4ADS using satellite data [6]. This report aids on our analysis by providing an educated guess on the location of the interference signal.

DATASET

This project uses ADS-B data queried from OpenSky Network which is a community-based receiver network that collects ADS-B data and stores historical data [7]. The data of this project is structured data, and each data point is one ADS-B message. For each ADS-B message, we selected four features including latitude (Lat), longitude (Lon), altitude (Alt), and Navigation Integrity Category (NIC). NIC is the integrity level indicator of current position message. It indicates the size of 95% accuracy bound on current reported position. Imagine drawing a circle centered at reported position, the actual position should be somewhere within the circle. Therefore, the smaller the radius of the circle is, the higher the accuracy the information has. Figure 3 shows the size of the containment radius corresponding to each possible NIC value, noticed that the higher NIC value means the more accurate the information is.

NIC	Containment Radius
0	Unknown
1	$R_C < 37.04$ km (20nm)
2	$R_C < 14.816$ km (8nm)
3	$R_C < 7.408$ km (4nm)
4	$R_C < 3.704$ km (2nm)
5	$R_C < 1852$ m (1nm)
6	$R_C < 1111.2$ m (0.6nm)
	$R_C < 926$ m (0.5nm)
	$R_C < 555.6$ m (0.3nm)
7	$R_C < 370.4$ m (0.2nm)
8	$R_C < 185.2$ m (0.1nm)
9	$R_C < 75$ m
10	$R_C < 25$ m
11	$R_C < 7.5$ m

Figure 3: NIC value and corresponding size of containment radius

Real Data

This project collected $4.52GB$ of 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 based on ADS-B equipment performance requirements defined within Title 14 of the Code of Federal Regulations (CFR) Part 91 [8]. 14 CFR Part 91 defines general operating and flight rules including the performance requirements for ADS-B system under normal circumstances. One of the requirements is that the aircraft's NIC must be less than 0.2 nautical miles (NIC ≥ 7), under normal circumstances. Therefore, for training set, we labeled each data point as: jammed/anomaly ($y = 1$) if the NIC value is < 7 or unjammed/normal ($y = 0$) if the NIC value is ≥ 7.

Figure 4 shows the overall picture of the ADS-B flight tracks in Cypriot airspace for one day. Each flight track is shown by dotted line and each dot corresponds to one ADS-B message or one data point. The color of each dot is showing the corresponding NIC

value. The pink star-shaped marker is showing the possible location of the interference source identified by C4ADS. Noticed that most of the airplanes that were flying near the jammer location had NIC values dropped down to 0. As the distance between the aircraft and the jammer gets larger, the NIC value gets higher which means more accurate position information.

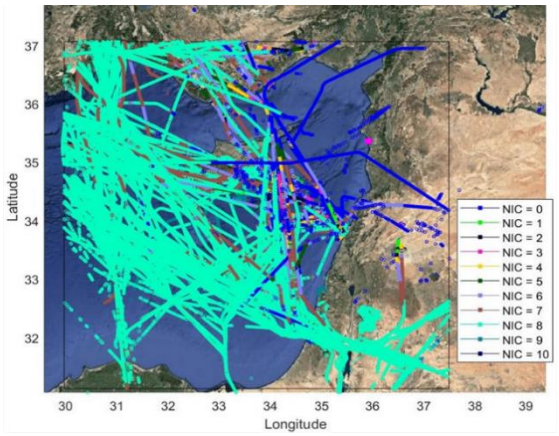


Figure 4: Overall picture of the ADS-B flight tracks in Cypriot airspace for one day.

Simulated Data

As for CNN, the purpose is to identify the possible location of the interference source given corresponding overall picture of the flight tracks from one day. Therefore, we need to train the model with different overall pictures caused by different jammer locations. However, in Cypriot airspace, there is only one interference source with one constant location among different days. That means no matter how many days of datasets we collected; there is only one class label which corresponds to that one constant location of the interference source. Therefore, we need to use simulated data which has different overall pictures caused by different jammer locations. Figure5 shows two sampled plots of the simulated data.

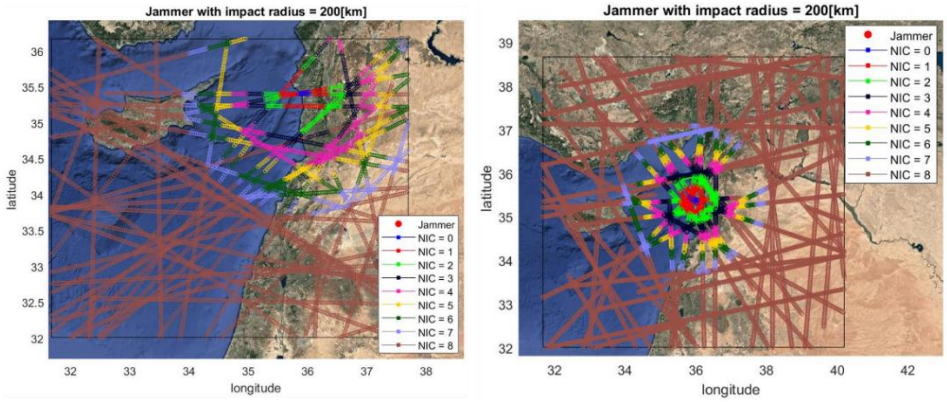


Figure 5: Two sampled overall pictures of the simulated ADS-B flight tracks for one day.

METHODS

In this project we explored two different algorithms: standard neural network and convolutional neural network. In this section, we will discuss the mathematical operations behind each model. The detailed information of corresponding hyperparameters and how we obtained the model architecture is presented in the EXPERIMENTS AND REULTS section.

Logistic Regression

Detecting the existence of interference event is a classification problem. Logistic regression is a proper first candidate to be used in this type of problem. Therefore, we used logistic regression as a baseline for this project. It inputs one ADS-B message and outputs a prediction of whether this point has been jammed ($\hat{y} = 1$) or not ($\hat{y} = 1$). Logistic regression outputs this classification based on the probability of whether the input point experienced jamming. This probability is calculated by $h_\theta(x)$ shown in equation1. x in equation1 is our input vector of one ADS-B data $\in R^{3*1}$ which contains 3 parameters including latitude (Lat), longitude (Lon), and altitude (Alt). Parameter vector θ in equation1 is obtained by performing gradient descent on cost function shown in equation2 which is summing over all m numbers of training examples. In other words, the chosen value for θ in equation1 maximizes the chance that: for each training example, the predicted result $\hat{y}^{(i)}$ is the same as labeled result $y^{(i)}$.

$$\begin{cases} \hat{y} = 0 \; if \; h_\theta(x) < 0.5 \\ \hat{y} = 1 \; if \; h_\theta(x) \geq 0.5 \end{cases} \text{ where } h_\theta(x) = \frac{1}{1+e^{-(z)}} \text{ and } z = \theta^T x \tag{1}$$

To better understand equation 2, assumed that for the i_{th} training example, the logistic regression predicted a result of $\hat{y}^{(i)} = 0$ but the true labeled result is $y^{(i)} = 1$. In that case, $-[y^{(i)} \log(\hat{y}^{(i)}) - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] = -[1 * \log(0) - 0 * \log(1)] = \infty$. Therefore, the cost function penalizes this result heavily which makes the algorithm learn new θ to avoid making incorrect predictions. In contrast, if the predicted result is the same as labeled result: $\hat{y}^{(i)} = y^{(i)} = 1$, the cost value becomes $-[1 * \log(1) - 0 * \log(0)] = 0$. In that case, the cost function will not penalize the result at all since the algorithm made a correct prediction.

$$J = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log(\hat{y}^{(i)}) - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \tag{2}$$

Standard Neural Network

The model architecture we used in this project for standard NN is shown in Figure6. The detailed information of how we obtained this model architecture is presented in the EXPERIMENTS AND REULTS section. Similar to logistic regression, the input layer of NN is ADS-B data $\in R^{3*1}$. The output layer of NN is a binary classification of the given input point: $\hat{y} = 1$ for jammed point or $\hat{y} = 0$ for a nominal (unjammed) point.

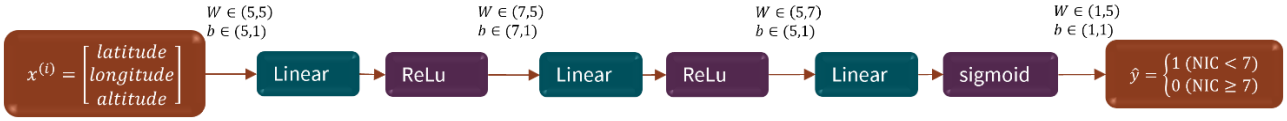


Figure 6: Model architecture of standard neural network

Between neurons of each layer shown in Figure6, W and b are weights and biases been applied to the result from previous layer. The detailed information of how we determined the size of W and b is shown in the EXPERIMENTS AND REULTS section. These parameters were trained in the same way as parameter θ in logistic regression. The only difference is that instead of performing linear operation on input x using parameter θ: $z = \theta^T x$, within each layer, the standard NN performs linear combination on result a from previous layer using parameter W and b (see equation3). In addition, instead of using same activation function throughout entire model, the standard NN applies two different types of activation function $\sigma(z)$: Rectified Linear Unit (ReLu) and Sigmoid (see equation3). Sigmoid activation function is used in output layer because the model needs to output a probability value between 0 and 1, and sigmoid function by default, outputs a value between 0 and 1. ReLu activation function is used in each hidden layer because this activation function always has a slope even when z is too large, this prevents learning process from being slowed down.

$$\begin{gathered} for \; l = 1 \; to \; L\text{: } (L \text{ is the total numbers of layers [L=4 in our model]}) \\ z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}; \text{ and } \; a^{[l]} = \sigma(z^{[l]}); \end{gathered} \tag{3}$$

$$\text{Where } \sigma(z^{[l]}) = \begin{cases} \max(0, z) & ReLu\ for\ l < L \\ \frac{1}{1+e^{-z}} & Sigmoid\ for\ l = L \end{cases}$$

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 [9] which initializes or samples the weights based on the distribution shown in equation4. The goal of He Initialization is to initialize the weights such that the variance of the activations is the same across every layer. This helps prevent the gradient of our models from vanishing. In addition, we use learning rate decay from $\alpha = 0.1 \rightarrow \alpha = 0.001$ which helps make sure the cost function minimization does not stop at local minimum and does converge in the end.

$$w_{i,j}^{[l]} = \mathcal{N}(0, \frac{2}{n^{[l-1]}}) \tag{4}$$

Convolutional Neural Network

In this project we use CNN to identify possible location of the interference source based on the overall picture of the flight tracks in the airspace. Since CNN are commonly used in image classification, the structured ADS-B data cannot be input into the model directly. Therefore, we converted 1-D flight ADS-B data into high-dimensional matrix before input into the CNN. Figure7 illustrates this data representation process.

As for input data, we started by drawing grids onto the target airspace where the size of the sub grid is 0.1 by 0.1 degree in the latitude and longitude. Imagine an aircraft flying through the airspace with the flight track shown by the yellow dots in Figure7, each dot represents one position point, and each point should be contained by one of the sub grids. Knowing which sub grid contains position point is like having the pixel information of an image. Then we created two channels to contain information for each position point which are NIC and altitude. These two channels are similar to the Red/Green/Blue (RGB) channels of an image. Therefore, instead of showing color information, we are showing the altitude and NIC value of each given position point in the overall picture. As for the output, the algorithm identifies which sub grid contains the most likely location of the jammer. This is done by labeling each sub grid with indices and converting the corresponding index number into a one-hot vector.

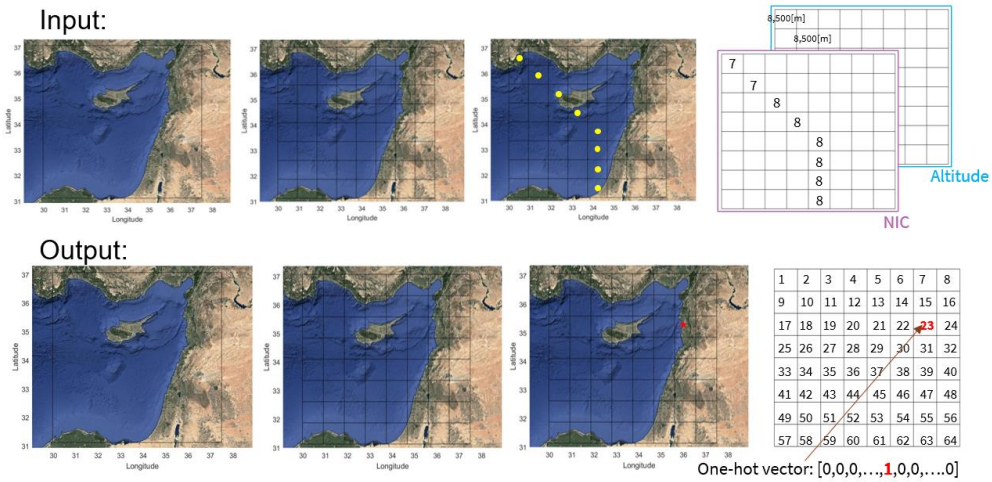


Figure 7: Data representation from vector to multi-dimensional matrix for CNN

The model architecture we designed for CNN is shown in Figure8. The detailed information of how we obtained this model architecture is shown in the EXPERIMENTS AND REULTS section.

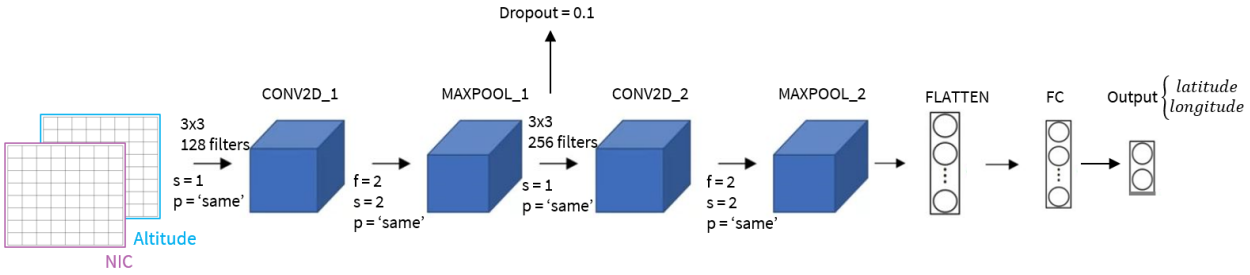


Figure 8: Model architecture of CNN

Within each of the first 2 layers in Figure5, we performed a convolution (CONV2D) followed by a max-pooling (MAXPOOL). These operations are done by sliding a (f, f) kernel filter over the matrix from previous layer with the step size of s. The activation function between each operation is ReLu shown in equation3. For better understanding, we illustrate how those operations could be done using simplified structure shown in Figure9, all the information shown in Figure9 is not related to the final model of this project, this figure is simply used for demonstration. The only difference between convolution and max-pooling is that convolution is performing elementwise multiplication and summation between the filter and the selected window from input matrix. However, max-pooling returns the maximum value from each selected window. Before each operation, we also performed a same padding which is adding extra 0 values to the input matrix so that output matrix has the same size as the input matrix.

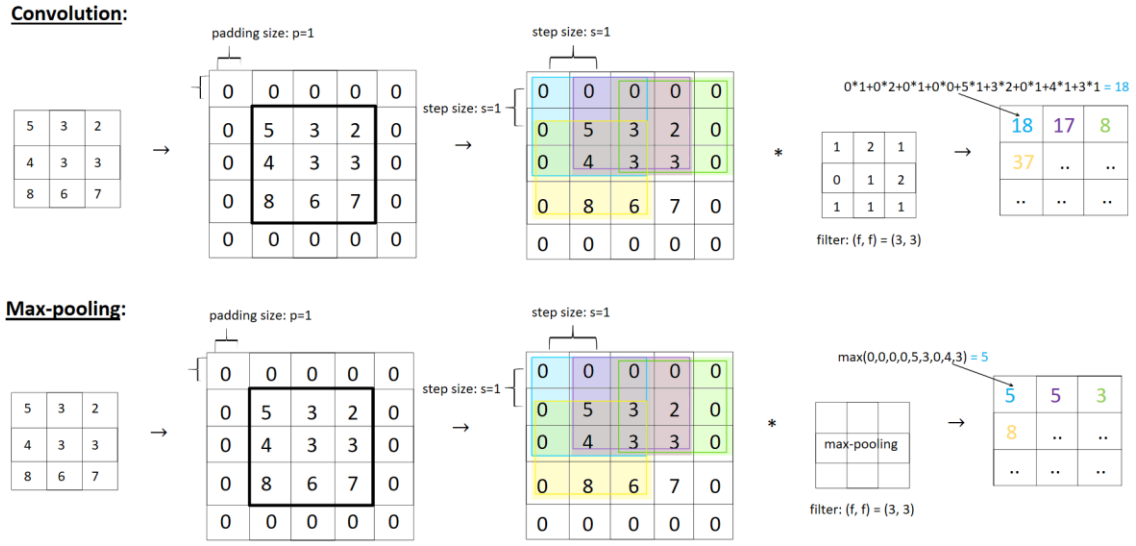


Figure 9: Convolution (top) and max-pooling (bottom) operation demonstrated using simplified structure (non-related to the model of our project)

After the second max-pooling layer, we added a layer (FLATTEN) to flatten the matrix into 1-D vector and followed with a fully connected layer (FC).

EXPERIMENTS AND RESULTS

In this project, we designed the overall structure of the models such as choosing proper activation functions and finding the optimal values for hyperparameters. As for implementing the mathematical algorithm described in the METHOD section, we used software packages from Keras [10].

Figure10 shows a summary of key steps of experiments and corresponding results for standard NN. Each experimental model uses 950,642 training examples and 407,418 testing examples. During experimentation, we noticed few significant properties. First,

model is sensitive to the size of learning steps. We noticed that the cost function of our models tends to stagnate after the first few iterations, we need to make sure the learning step is not too small. Otherwise, the cost will stay at the local minimum and the algorithms will learn nothing.

Experiment Detail	Accuracy (Train, Test)
NN with 1 hidden layer, 5 hidden units. $\alpha = 0.01$. ***Random initialization***.	0.798, 0.786
NN with 1 hidden layer, 5 hidden units. $\alpha = 0.01$. ***He initialization***.	0.833, 0.831
NN with 1 hidden layer, 5 hidden units. $\boldsymbol{learning\ rate\ decay\ \alpha = 0.1 \rightarrow \alpha = 0.001}$. He initialization.	0.811, 0.808
NN with 2 hidden layer, 1st layer has 5 hidden units, 2nd layer has 7 hidden units. learning rate $\alpha = 0.01$. He initialization.	0.835, 0.832
NN with 2 hidden layer, 1st layer has 5 hidden units, 2nd layer has 10 hidden units. learning rate $\alpha = 0.01$. He initialization.	0.836, 0.831
NN with 3 hidden layer, 1st layer has 5 hidden units, 2nd layer has 7 hidden units, 3rd layer has 5 hidden units. learning rate $\alpha = 0.01$. He initialization.	0.8361,0.8368

Figure 10: Table of key experiments and results for standard NN

Figure11 shows two sampled results we obtained using CNN to predict location of interference source. During each experiment, the experimental model uses $16,000$ sets of training data and $4{,}000$ sets of testing data, each set contains an overall picture of all flight tracks passing the airspace within one day based on simulated ADS-B data. The main problem we noticed during training is the low validation accuracy caused by large number of classes. Recall that each sub grid on the map is one class which indicates one possible location of the interference source. It is difficult to make a sufficiently complicated model such that it could perform classification among large numbers of classes. In addition, we also need to prevent overfitting of the training set since CNN tends to overtrain easily as the epoch number increases.

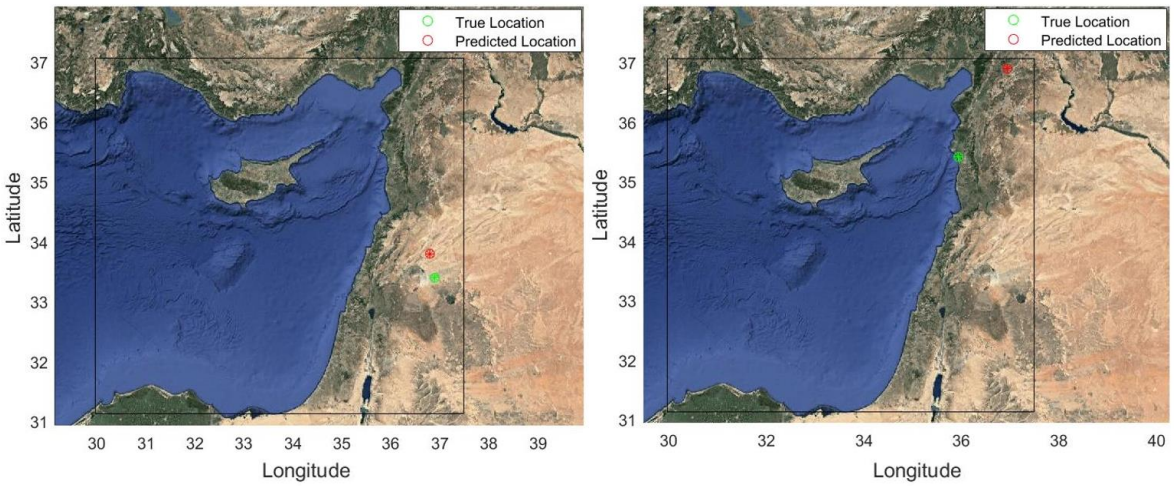


Figure 11: Sampled plot of prediction result using simulated (left) and real (right) data

The standard NN we designed has 3 hidden layers, each contains 5, 7, and 5 hidden units. This model was trained by performing gradient descent for 3,000 iterations with He initialization and learning rate $\alpha = 0.01$. The CNN we designed has an architecture of $CONV2D \rightarrow RELU \rightarrow MAXPOOL \rightarrow CONV2D \rightarrow RELU \rightarrow MAXPOOL \rightarrow FLATTEN \rightarrow FULLY\ CONNECTED$. The filer sizes of $CONV2D$ and $MAXPOOL$ are shown in Figure8. This model was trained by performing mini-batch gradient descent on cost function for 100 epochs with batch size of 32.

DISCUSSION

In this project, we built and trained a standard NN model to identify whether a given ADS-B position point has been jammed in the airspace. Being able to complete this task implies another important application which is to identify the size and shape of the impact region formed by the interference source. This application is quite useful for identifying the power level of the jammer as well as considering the effect of complicated environmental factors such as signal blockage caused by mountains. Figure12 illustrates how we perform this application. Plot on the left is showing the impact region caused by the interference source. This result is generated by first filling the entire target airspace with position points, then feeding all those data points into the trained NN and let the NN determine which ones are jammed. The final result shows the decision boundary which should be the boundary of the impact region caused by the interference source. Plot on the right shows how the interference source affects the true ADS-B data points. The green star-shaped marker is showing the possible location of interference source identified by C4ADS using satellite data. The color of each point in Figure12 indicates the probability of whether the point has been jammed. Red means 100% probability of being jammed and blue means 0% probability of being affected.

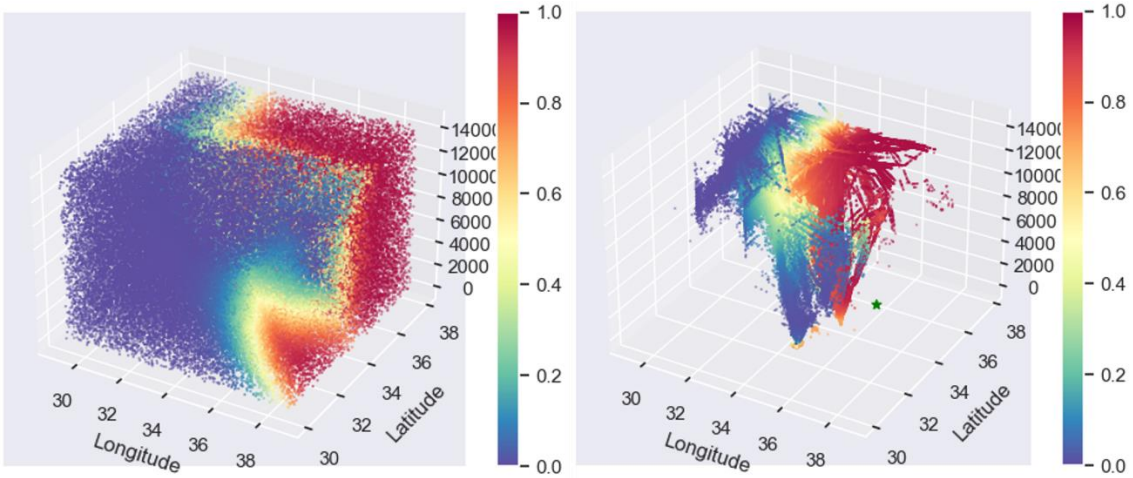
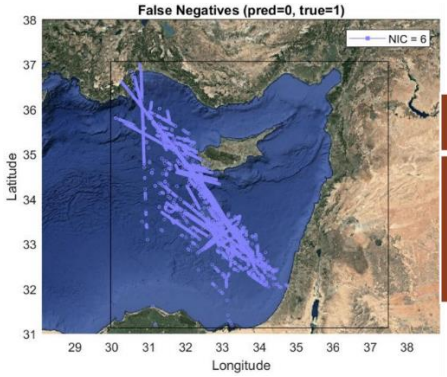


Figure 12: Impact region of the interference source in Cypriot airspace

Noticed that the existence of NIC==0 data points affects the model a lot. The plot on the right in Figure12 has few data points that are closed to the ground and being marked by orange color under the main cluster of blue points. These points have NIC = 0 in real life and NN model labels them as 80% probability of being jammed. These NIC==0 points lead to an overall decision boundary that has part of impact region closed to the ground on the left plot. Noticed that when filling the airspace with position points, we extended size of the overall airspace by one degree from 32°-37° to 30°-38° in altitude and similar in longitude. Therefore, the NN model, instead of learning to close the decision boundary around the possible location of the jammer, extends the impact region to the unknow area. This result is as expected because in current airspace, most of the data we collected was from southwest side of the jammer due to the lack of coverage from publicly accessible data. This gives us an imbalanced data distribution within airspace surrounding the jammer. That leads to the open area to the northeast of the jammer in Figure12.

Another interesting result we noticed is that most of the false positives and false negatives from both models, shown in Figure13, are also points which we were not sure whether they were jammed. We struggled during labeling whether to declare those data points as jammed or not. 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 threshold, we are not certain that NIC = 6 means that the point has been jammed or that NIC = 8 means the point is good. Noticed on the left-hand side of Figure11, all points have NIC = 6 and all of them are far from the possible impact area of the jammer. We are not certain whether saying those points were not jammed is a false statement. Similar to the false negative flight shown on the right-hand side.



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

Figure 13: Sampled false negatives from standard NN

CONCLUSION AND FUTURE WORK

In this project, we designed and trained a standard NN and a CNN model for interference event detection and localization. Given an airspace in which there exists interference events, by only collecting and feeding ADS-B data from the airspace into those trained models, we are able to identify the size and shape of the impact region as well as identifying the location of the interference source.

In the future, we would like to solve two problems we encountered during this project. First, we would like to further improve our standard NN model by observing the overall shape of the decision boundary learned by the model. This requires us to test our model in other airspaces where we could collect ADS-B data evenly from all directions to the interference source. Recalled that in current airspace, most of the data we collected was from southwest side of the jammer which gives us an imbalanced data distribution within airspace surrounding the jammer. Second, we would like to test our CNN model with real ADS-B data instead of simulated data. This requires obtaining dataset from interference event that contains information about specific location of the jammer. This requirement is difficult to satisfy because 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 machine learning models for this topic.

ACKNOWLEDGMENTS

The authors thank the Federal Aviation Administration (FAA) for sponsoring this research. The authors also thank OpenSky Network for providing ADS-B data for this study.

REFERENCES

1. Lukeš, P., Topková, T., Vlček, T., and Pleninger, S., "Recognition of GNSS Jamming Patterns in ADS-B Data," *2020 New Trends in Civil Aviation (NTCA), Prague, Czech Republic, 2020,* pp. 9-15.
2. Liu, Z., Lo, S., and Walter, T., "Characterization of ADS-B Performance under GNSS Interference," *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020), September 2020,* pp. 3581-3591.
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.
4. Liu, Z., Lo, S., and Walter, T., "GNSS Interference Characterization and Localization Using OpenSky ADS-B Data," *Proceedings 2020, 59,* 10. https://doi.org/10.3390/proceedings2020059010
5. EUROCONTROL (2021, March 01). "Does Radio Frequency Interference to Satellite Navigation pose an increasing threat to Network efficiency, cost-effectiveness and ultimately safety?" *Aviation Intelligence Unit,* Think Paper #9.

https://www.eurocontrol.int/sites/default/files/2021-03/eurocontrol-think-paper-9-radio-frequency-intereference-satellite-navigation.pdf
6. Center for Advanced Defense Studies. *Above Us Only Stars: Exposing GPS Spoofing in Russia and Syria.* Washington, DC: C4ADS, 2019. https://www.c4reports.org/aboveusonlystars.
7. 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,* April 2014.
8. Office of the Federal Register, National Archives and Records Administration. (2013, December 31). *14 CFR § 91.227 - Automatic Dependent Surveillance-Broadcast (ADS-B) Out equipment performance requirements.*. [Government]. Office of the Federal Register, National Archives and Records Administration.
9. He, K., Zhang, X., Ren, S., and Sun, J., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *In Proceedings of the IEEE international conference on computer vision, 2015,* pp. 1026-1034.
10. Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras.