CIS 522 – Final Project – Technical Report

AirTraj

April 2022

Team Members:

- Team Member 1; Shivani Reddy Rapole; Email: srapole@seas.upenn.edu
- Team Member 2; Ajay Anand; Email: anandaj@seas.upenn.edu
- Team Member 3; Bharathrushab Manthripragada; Email: mbharath@seas.upenn.edu

---

Abstract

Aircraft emissions are one of the largest sources of major sources of air pollution, with the carbon, nitrogen and sulphur oxides emitted by airplane engines causing major impacts to global warming, erosion of the ozone, acid rain and depletion of air quality. The use of optimal flights are meant to minimize these emissions while also providing the economic benefit of reducing fuel consumption and thereby reducing costs the cost of air travel. Fuel costs are generally measured in amount of fuel per passenger per mile flown. While some of these costs are unavoidable, deviations from optimal flight paths due to adverse weather conditions, changing restricted flying zones and proximity to other aircraft increase fuel consumption, delays in air travel and an increase in these emissions drastically increase emissions and cost airlines millions of dollars in additional fuel costs and disruptions to scheduled landing times at busy airports. These deviations can also lead to a cascade of effects with aircraft being held in holding patterns for hours due to missing their assigned landing times. Being able to predict optimized flight paths taking the current and predicted weather into consideration can help minimize all of these effects as well as minimize flight schedule disruptions by allowing scheduling systems to be predictive instead of reactive. In this work we propose a model that would generate an optimized flight path needing minimal deviations from planned paths by using planned flight paths, previous deviated flight tracks and past weather data. This would allow for commercial aircraft to always be on the optimal flight path even with adverse weather patterns minimizing costs and reducing harmful emissions.

1 Introduction

Environmental pollution caused by human actions and its effects on our planet are some of the largest social, economic and technological challenges faced in the modern era. Environmental pollution comes in many forms but one of the primary means human beings are directly effected by it are in the loss of air quality and the global rise in temperatures leading to more adverse weather patterns. The Aviation industry contributes to this trend by the release of oxides of carbon, nitrogen and sulphur from aircraft emissions. The aviation industry contributes more to global warming than is commonly appreciated because of the mix of climate pollutants it generates, this is shown by the fact that despite being responsible for only 2.4% of global annual emissions of CO2, aviation contributed approximately 4% to observed human-induced global warming to date(5). The sulphur and nitrogen also contribute heavily to the production of smog in busy urban centers. Smog is also one of the primary factors leading to the formation of acid rain which causes infrastructure and soil degradation. These effects only get exacerbated by the fact that airlines frequently have to deviate from optimized flight paths due to adverse conditions such as weather, maintaining safe distance from other aircraft and restricted flight zones.

2 Related Work

There has not been a large body of work done in this area, a large part of which is the difficulty in obtaining the data required to train models in this domain. Aircraft designs and fuel consumption figures largely remain proprietary and security concerns make obtaining FAA flight path and track data difficult. The largest body of work in this area is in the form of competition entries to GE Flight Quest Challenge(2014). This yearlong challenge provided entrants with a large amount of data in the form of flight path, flight track, aircraft weather and meteorological prediction information to generate models which help better optimize aircraft paths. Another work done in this domain was by Liu et al. titled Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach(1) which we primarily base our current work on due to its shorter training times and more lightweight nature. We improved on their work by modernizing the libraries used, creating entirely new pytorch based tools and adding additional features to improve performance.

3 Dataset and Features

Flight data is unexpectedly tough to get due to the sensitive nature of aircraft data and the additional security measures implemented after 2001. The Federal Aviation Administration (FAA) grants researchers secure access to flight data for research purposes; nevertheless, each proposal must be manually verified. We used datasets from a previous study in this domain (2013) as well as the GE flight quest challenge dataset in this study (2014). Flight tracks, flight plans,

atmospheric weather, and convective weather data are among the data set.

The FAA Traffic Flow Management System (TFMS) flight tracks file comprises 4D locations of each aircraft during its flight - latitude, longitude, altitude, and time, with resolutions of around 1 minute, 1 minute, 100 feet, and 1 minute, respectively. We restrict our scope for this analysis to flights from George Bush International Airport (IAH) to Logan International Airport (BOS) in 2013.

The flight plan dataset, also obtained from FAA TFMS, provides the most recently filed 2D flight plan coordinates (latitude and longitude) for each flight. While the lengths of flight plans range from 9 to 144 points, we may considerably minimize their dimensions by identifying distinctive points.

The atmospheric datasets, which contain the wind speed dataset and air temperature dataset,come from the North American Mesoscale (NAM) Forecast system. It produces high-resolution atmospheric information four times a day – 00:00, 06:00, 12:00, and 18:00 UTC.we cropped the original georeferencing system so that the new georeferencing is a box area with four corners: (130W, 22N), (130W, 52N), (64W, 22N), (64W, 52N). The new horizontal resolution is about 413 by 336, 6 which is roughly half of the original. Our final horizontal georeferencing system Gis shown by the blue raster in Figure we cropped the original georeferencing system so that the new georeferencing is a box area with four corners: (130W, 22N), (130W, 52N), (64W, 22N), (64W, 52N). The new horizontal resolution is about 413 by 336, 6 which is roughly half of the original. Our final horizontal georeferencing system is shown by the blue raster in Figure

National Convective Weather Forecast (NCWF) system provided the convective weather dataset. Every record in the collection provides the coordinates of convective weather polygons (the storm's borders and peak altitude) as well as the direction of movement at the moment of recording. Every 5 minutes, the dataset was routinely updated.

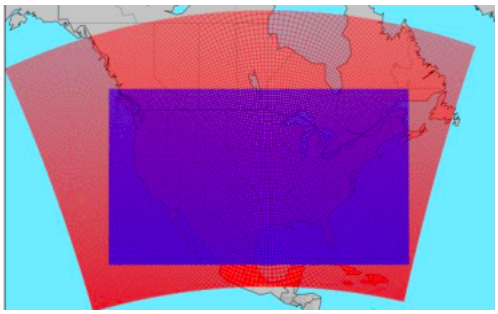

Figure 1: Atmospheric data raster

4 Methodology

4.1 Baseline Model:

Our baseline model for comparison is created using the flight plan database, the associated API allows us to view filed flight plans as well as associated flight tracks and weather information from a publicly maintained database. This allows us to compare the results of our model with real world efforts at generating alternative flight plans in case of adverse weather.

4.2 Deep Learning Model:

We base our model on the paper - Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach - in which the author develops an encoder-decoder LSTM-based generative model to predict aircraft 4D trajectories.

4.2.1 Model Inputs:

We predict aircraft trajectories based on three kinds of information:

- Flight's last filed flight plan prior to departure: the last filed flight plan is an important indicator of what the flight trajectory will be. But it is common for an aircraft to deviate from their flight plans rather than "fly as filed", so last filed flight plan is not determinative.
- Convective weather information: Strong updrafts and downdrafts are seen within a convection region, causing severe and unfavorable turbulence. As a result, planes nearly always avoid such areas, either strategically by taking a path forecast to be clear of convective weather or tactically by navigating around convective weather cells.
- Atmospheric Conditions: This refers to the temperature of the air and the speed of the wind. Trajectories can be influenced by the conditions in at least two ways. For starters, flights prefer routes with strong tailwinds because they save fuel and time. Finally, high temperatures are linked to increased turbulence, which pilots may try to avoid. Finally, because aircraft engines can produce greater power in cold and thick air, pilots prefer to fly through it.

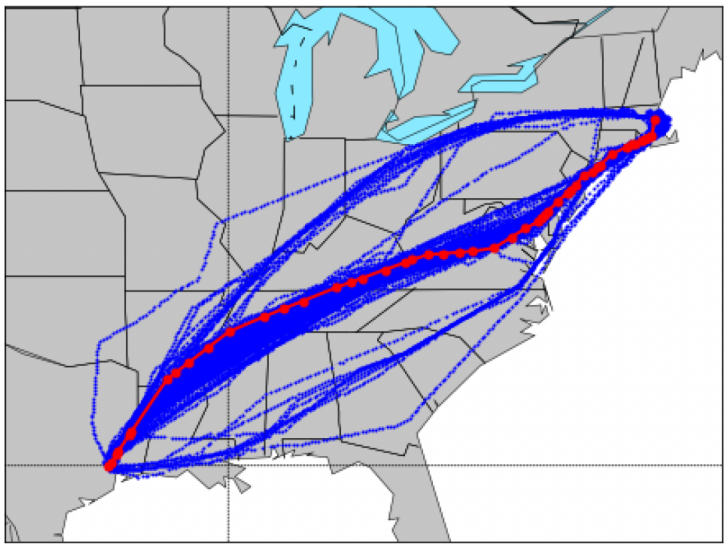

Figure 2: Red - Last filed flight Plan, Blue - Actual Flight trajectories; Source: Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach - Yulin Liu (2018)

4.2.2 Feature Engineering:

A 4D referencing system associated to a certain track point is utilized to match aircraft trajectories with the required raw meteorological and atmospheric datasets. For each track point, we start by creating a grid square around it. The grid square's width and height are dx and dy degrees in latitude/longitude, respectively, and the square's resolution is nx by ny points. Because each flight is made up of a series of track points, it has a reference grid path that is the feature cube grid path. Based on the time stamp and geographic location, the feature cubes can then be matched with the meteorological and atmospheric databases.

After this feature engineering and matching, the following information if captured:

- Track points of a trajectory
- The corresponding weather data for each track point
- The corresponding wind speed and air temperature data for each track point

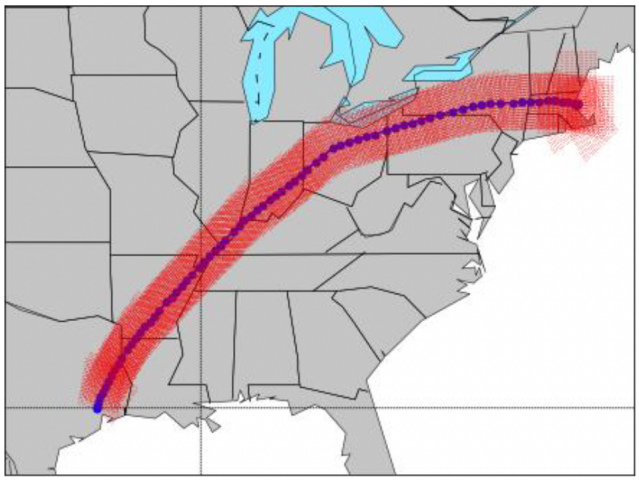

Figure 3: Feature Cubes

4.2.3 Model Architecture:

Based on the different reasons discussed above, we can infer that aircrafts frequently deviate from their last filed flight plan. To capture such deviations and predict the actual flight trajectory, this problem can be thought of as a combination of sequence generation and translation. We try to generate the next "sequence" of air trajectory. But, we also use the weather and atmosphere data to "translate" the trajectory.

For this purpose, we use the model which mainly consists of three components:

- Encoder LSTM - Embed flight plans into fixed-size feature vector.
- Decoder LSTM - Maps fixed-size feature vector to target flight trajectory sequence
- CNN Layers - The feature cube we generated earlier can be thought of images as they are 3-dimensional. So, we use CNN-layers to condense high-dimension weather-related feature cubes into fixed-size feature representations that can be sent as input into the decoder network.

Before feeding the flight plan coordinates to the LSTM, the encoder contains an embedding layer with a dimension of 32 for the flight plan coordinates. The encoder then has two further layers, each with a fixed-size hidden state of 128 dimensions. To represent the probabilistic distributions of actual flight tracks, Gaussian mixture components were employed. As a result, there are 45 decoder LSTM output parameters for each timestamp.

The decoder network's convolutional layers learn representations from the weather feature cubes. It has four layers, three of which are convolutional and one of which is fully connected. The feature cube is fed by the first layer, which has 16 filters of size 6*6*4 and stride 2. The second layer uses 16 smaller 3*3*16 filters with stride 1, and the third layer uses 32 smaller 3*3*16 filters with stride 1. There are 32 neurons in the completely linked layer. Because locational weather information is vital in our issue domain, no pooling or padding processes are used.

The decoder network is a two-layer LSTM with a 128-dimensional hidden state that starts with the last hidden state from the encoder LSTM. Before entering the decoder LSTM, feature representations from the convolutional layers and aircraft state variables are input into an embedding layer of 64 neurons. Using a dense layer, the LSTM outputs will be transferred to parameter estimates 18 of Gaussian mixtures (3 components). For all activations in our architecture, we use the Exponential Linear Unit (ELU) function.

The author in our base paper uses the Nesterov Momentum optimizer with gradient clipping and batch size 256 to train our neural networks. The learning rate is initialized as 0.001 and is decaying every 1,000 epochs. But, in our model we used Adam optimizer since it has better aspects like better decay of learning rate, etc.

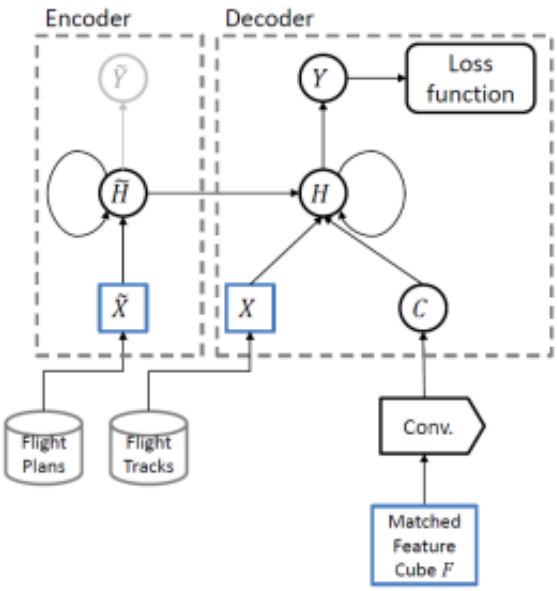

Figure 4: Model Framework; Source: Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach - Yulin Liu (2018)

The actual flight state at time t is represented by a list of variables that includes latitude, longitude, altitude, latitude speed and longitude speed at time t. At each timestamp, we assume that this state follows Gaussian mixtures which the decoder will learn.

4.2.4 Loss Function:

At timestamp t, the decoder network takes the hidden variable at time t 1, current aircraft state, some weather related feature representations from CNN layers, and the weight variables to be learned, and outputs the hidden variables that contain the information regarding Gaussian mixtures. All the Gaussian mixture parameters are the functions of this output from the decoder network.

So, the loss function for our model is to minimize the negative log likelihood of the probability density function of the Gaussian distribution that tries to estimate the current state given the mean and covariance of then distribution.

4.2.5 Inference:

During the inference time, we have last filed flight plan, observed flight state in the trajectory till tile T', their corresponding feature cubes till T'. We input these into the trained model and obtain the Gaussian mixture parameters from decoder outputs. Using this, we predict the flight state at $T'+1$ and then apply Adaptive Kalman Filter (AKF) recursively. Using this state, we can match the corresponding weather and atmosphere feature cube using the process described earlier. This new predicted state along with the matched feature cube can now be fed into the decoder to predict the next state. The same process can be followed again to predict the rest of the aircraft trajectory. Beam search and RTS smoother are used to further improve the predicted trajectory.

5 Results

We applied our method on a historical flight trajectory dataset from IAH to BOS in the year 2013. The preprocessed dataset contains 1679 flights and is split into two sets, with 80 percent in the training set and the rest in the evaluation set. In the inference process, we use the first 20 actual track points and their corresponding feature cubes for every flight on the evaluation set as the observed sequence and predict the rest of flight tracks. The results below are 5 samples in the test flights datasets on which we are predicting the flight tracks using our model, the magenta curve is the predicted flight tracks, the red curve is the filed flight plan, the green band indicates the path of 99.7 percent confidence intervals (three standard errors) around predicted flight tracks.

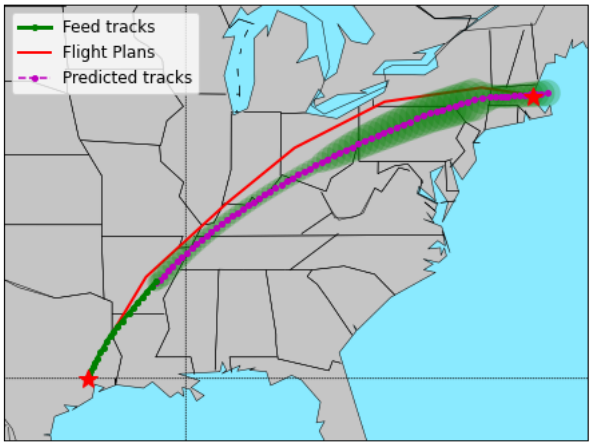

Figure 5: Sample prediction 1 of flight ID: 20130801438231 and fligh plan: FP0005

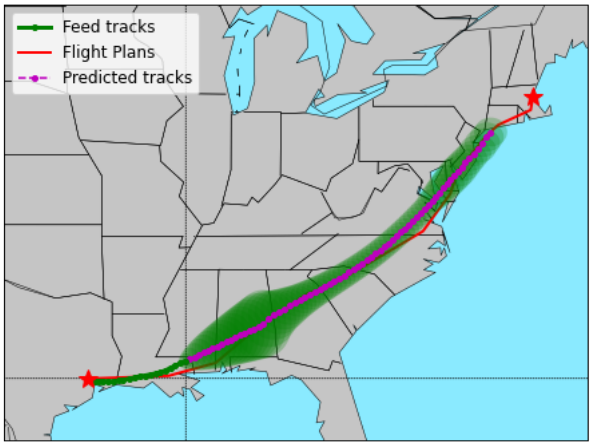

Figure 6: Sample prediction 2 of flight ID: 20130626387884 and flight plan: FP0003

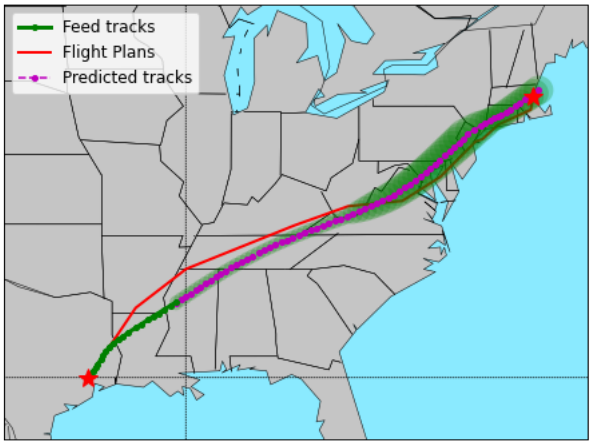

Figure 7: Sample prediction 3 of flight ID: 20130203017464 and flight plan: FP00001

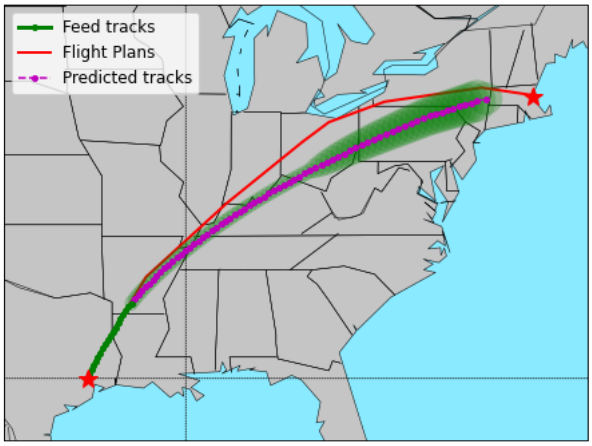

Figure 8: Sample prediction 4 of flight ID: 20130630760485 and flight plan = FP0002

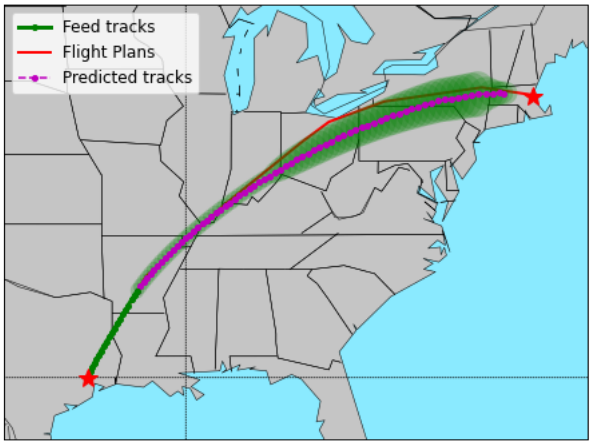

Figure 9: Sample prediction 5 of flight ID: 20130717182606 and flight plan:FP00002

5.1 Findings

We modernized the code from Tensorflow 1.15 to PyTorch for the ease of prototyping, achieving lower training times and since we have familiarity with the functions and methods in PyTorch. In this process, we encountered a lot of findings which we feel are useful and which matter in the field of Deep Learning. Learning new Deep Learning frameworks and having a cross-comparison based idea between different Deep Learning frameworks is a very good skill to

have in current research and corporate world.

None of us worked with Tensorflow earlier. So, we used this opportunity to learn Tensorflow and switch to the framework we already know i.e. PyTorch. This task was even more challenging since the Tensorflow version we tried to learn was 1.15 and not 2.0. We'd like to point out a few basic differences we encountered between the two frameworks while re-writing the entire Codebase in PyTorch.

- The basic workflow of building a DL model in Tensorflow can be described as creating objects (like structure - 'struct' in C/C++ lang) for the variables (inputs, hidden layer variables, etc) and the network, and finally making use of these objects. Variables are defined using Placeholders where we have to define the input dimension and then pass these variables into the Keras layers of Tensorflow.

- A deep neural network can be constructed as a Graph in Tensorflow by using multiple basic network components. Then, Tensorflow requires us to create a session for the Graph to perform any kind of computations.

- All the above process is extremely convenient and dynamic in PyTorch. The main advantage of Python over the other languages is its nature of being dynamic. PyTorch utilizes this particular advantageous aspect to the fullest. So, we don't have to define any of the variables in before-hand. Even while defining the layers, for example in a convolutional layer, we just have to mention the essential information like in_channels, out_channels, kernel_size, etc. Just this one point eliminates a ton of hassle while defining the network architecture.

- For example, consider the case of defining our encoder which has a stack of LSTM cells. In Tensorflow, we first make a list of (BasicLSTMCell, Dropout Wrapper), pass this to MultiRNNCell to create a structure for it and this is finally passed to dynamicRNN to enable computation on this structure. Pytorch has a function called LSTM where we can send the number of LSTM layers we want (which are followed by dropout layers) and this will directly return the encoder we desire.

- Another aspect where the difference was apparent where different mathematical computational functions like fill_triangular, mat_diag_transform, etc. There are no predefined functions in PyTorch that implement the same. So, we referred to the actual implementations of those functions on Github, understood and implement it again in PyTorch which was very challenging.

We also tried to replace Long short-term memory (LSTM) cells with Recurrent Neural Network (RNN) and Gated recurrent units (GRU) cells. LSTMs worked best for this architecture. RNNs work well for sequential data (like some kind of time-series data) and also its not suitable for deep networks since

vanishing gradient problem will occur, so it did not perform well for our model. GRUs and LSTMs were designed to tackle the issues with RNNs. LSTMs are more complex versions of GRUs and turned out to be the best choice in our scenario. With respect to the code implementation part, we used torch.nn.RNN for RNN and torch.nn.GRU for GRU instead of torch.nn.LSTM which was for LSTM.

5.2 Limitations and Ethical considerations

Due to our limitations in compute power, we limited our study to only two airports.Our model also does not take some factors like minimum altitude into consideration. Our Model also does not take nearby flights into account when generating paths due to which airspace congestion near busy airports might reduce model performance.

While on the surface no ethical considerations may clearly display themselves, using models such as ours can be biased towards creating more optimized flight paths in more well developed nations with busy airports. This is because of the fact that these airports have larger portions of our training data dedicated to them, putting flights taking off from smaller, less well developed airports at a disadvantage in having optimized flight paths generated to them. This can funnel more air traffic to already well serviced destinations, putting less affluent destinations on the back burner for airline route development.

By implementing debiasing techniques on our model we believe that less profitable routes can be made more appealing to airlines and improve connectivity to those who most need it.

5.3 Future Research Directions

Due to the specific nature of our dataset, we feel that our work can be improved by dedicating larger amounts of training time on more general datasets. This would allow our model to be able to better predict flight paths between any two airports. We also feel that the incorporation of more recent flight and weather data, especially in the light of global warming causing more and more adverse weather patterns would help future proof our model.

6 Conclusions

In this project, we start from the baseline model proposed by Yulin Liu and Mark Hansen for air trajectory prediction summarized in their paper titled - 'Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach'.

In this approach the flight state is modelled as Gaussian mixtures whose parameters are learned by our decoder. The hidden state generated by the encoder

is fed as the initial state for the decoder. The feature cubes which contain the weather and atmosphere information are processed by the CNN layers and fed into the decoder. The decoder takes all these inputs and predicts the Gaussian mixture parameters using the negative log likelihood as loss function. We apply and test this model on 2013 flight datasets from Houston to Boston.

We find that using models such as ours can create better optimized flight paths for aircraft and minimize deviations from filed plans, reducing emissions, costs and schedule delays for aircraft.

7 References

1. Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach. Yulin Liu, Mark Hansen

2. GE Flight Quest 2- Kaggle Challenge: https://www.kaggle.com/c/flight2-final

3. C. Kiss-Tóth and G. Takács, "A dynamic programming approach for 4D flight route optimization," 2014 IEEE International Conference on Big Data (Big Data), 2014, pp. 24-28, doi: 10.1109/BigData.2014.7004427.

4. What does aviation contribute to climate change: https://www.transportenvironment.org/challenges/planes/airplane-pollution/

5. Quantifying aviation's contribution to global warming. M Klöwer et al 2021 Environ. Res. Lett. 16 104027