Predicting Airport Configurations Using Ensemble Learning

Spencer McDonald *, Marek Travnik† and R. John Hansman ‡
Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139, USA

The National Aerospace System (NAS) is one of the most complex systems operating in the world which is shaped by many centralized and decentralized decisions made daily. One of the most impactful decentralized decisions made in the NAS is each airport's specific runway configuration at any given point in time (i.e., the combination of arrival and departure runways used at a specific time). The choice of a configuration directly affects the capacity envlope of an airport and the physical airspace utilization surrounding the airport. Due to increasing flight volumes and the introduction of Advanced Air Mobility (AAM), there is a desire to properly predict these airport configurations. Having a forecast of the probability that an airport will be in a given runway configuration could inform air traffic coordination decisions and improve airport operation planning. This paper presents an approach to forecast runway configuration using ensemble machine learning. The approach uses the airport configuration history, arrival/departure logs, and a weather forecasts to predict runway configurations for the following 6 hours from prediction time using an XGBOOST - Neural network ensemble architecture. A novel concept of "Minimum Configuration Support" is implemented in the neural network architecture to enforce runway configuration inertia. The algorithms are trained and tested on one year of data for 10 different airports in the US. The solution was the winning submission to the "Runway Functions" competition hosted by NASA on `drivendata.org`*.

I. Nomenclature

NAS = National Aerospace System
AAM = Advanced Air Mobility

II. Introduction

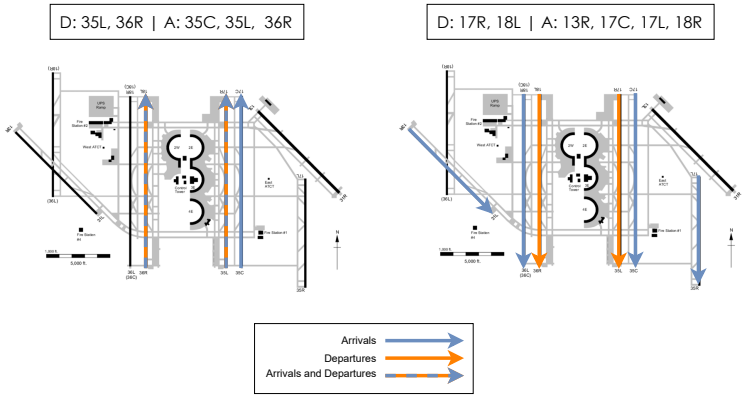


Fig. 1 Two example runway configurations at KDFW (Dallas-Fort Worth).

*Graduate Student, Aeronautics and Astronautics Engineering, MIT, 77 Mass Ave, Cambridge MA, 02139, AIAA Student
†Graduate Student, Aeronautics and Astronautics Engineering, MIT, 77 Mass Ave, Cambridge MA, 02139, AIAA Student
‡T. Wilson Professor, Aeronautics and Astronautics Engineering,MIT, 77 Mass Ave, Cambridge MA, 02139, AIAA Fellow
*https://www.drivendata.org/competitions/92/competition-nasa-airport-configuration-prescreened/

THE National Aerospace System (NAS) is becoming increasingly complex. While some decisions about operations in the NAS are centralized, others are not. One of the most crucial decentralized decisions made throughout the NAS is the choice of runway configuration at a given airport (the combination of runways used for arrivals and departures and the flow direction on those runways (see examples in Figure 1).

The choice of a specific runway configuration is made by air traffic coordinators at the airport; the configurations are chosen based on a number of features including: weather, time, and departure/arrival rates just to name a few [1, 2]. This choice affects many aspects of the operations in the NAS. Firstly, runway configurations have differing capacity envelopes - the number of aircraft that can depart and land in a given period of time differs depending on configuration. The capacity envelopes for the two configurations from Figure 1 are shown in Figure 2 and Figure 3 respectively.

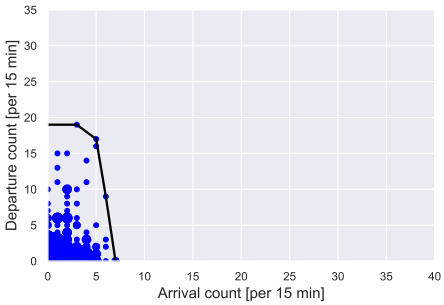


Fig. 2 Capacity envelope for the KDFW D: 35L, 36R | A:35C, 35L, 36R configuration

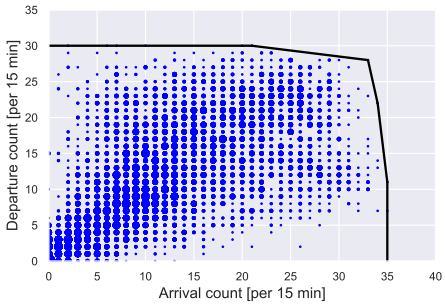


Fig. 3 Capacity envelope for the KDFW D: 17R, 18L | A: 13R, 17C, 17L, 18 configuration

In addition, the choice of runway configuration will significantly impact operations of other vehicles in the airspace around the airport (these could be smaller vehicles that do not follow legacy procedures at the airport - e.g. helicopters). With the emergence of Advanced Air Mobility (AAM), this issue becomes ever more relevant. For AAM procedures to be implemented into an already congested airspace it is essential to be able to evaluate the risk posed to the AAM vehicle by legacy aircraft (i.e. aircraft operating to and from the airport) and the risk posed by the AAM to those aircraft [3, 4]. The risk can be vastly different depending on the runway configuration used - as this affects the procedures used by legacy aircraft. One can see an example of what legacy procedures may look like for the two KDFW configurations in Figure 4. It can be seen that different portions of the airspace are used depending on the configuration. In addition, the arrival/departure rates on these procedures can be vastly different as shown previously in Figure 2 and Figure 3. Both of these factors will affect the risk calculation.

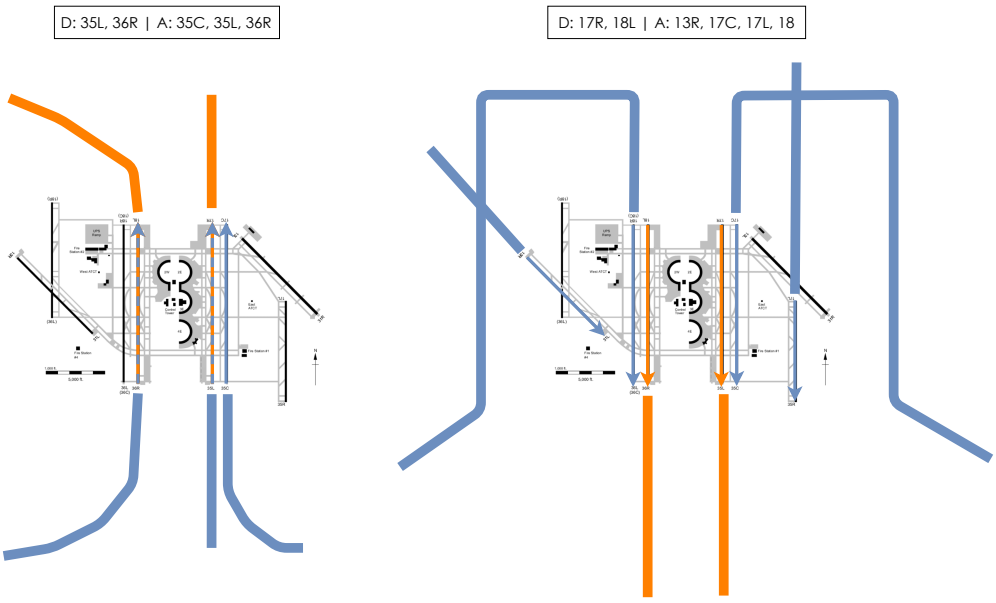


Fig. 4 Arrival and departure procedures for two configurations at KDFW. Note: this is an illustration - these are not real procedures at KDFW

The difference in capacity and airspace utilization are among the reasons why it is pertinent to build models that predict runway configurations for the near future. A probabilistic forecast of runway configurations can be converted to a capacity forecast which can be used as input into a ground holding program [5]. Similarly, this probabilistic runway configuration forecast, converted to an airspace utilization forecast, could be used to plan AAM procedures.

There are two broad types of approaches that have previously been developed to address the runway configuration problem: prescriptive and descriptive models. Prescriptive models try to use features to make recommendations as to what configuration should be selected. These approaches often use optimization methods to minimize some cost to "select" the best configuration for different time periods. One such example is the Enhanced Preferential Runway Advisory System (ENPRAS) [6] which tries to optimally select the best runway configuration. More recent methods take into account predictive methods for weather and capacity predictions [7–11]. The other approach is descriptive models, which seek to use data mining and machine learning approaches in an effort to predict runway configurations based off of historical data. These models do not seek to provide a recommendation, but rather seek to predict future decisions based off prior decisions. Previous approaches of this type have shown accuracy values of up to 75% using many different types of predictive methods [1, 12–17]. This paper presents a descriptive model for creating runway configuration forecasts for every 30 minute interval in the 6 hours after prediction time. The model is an ensemble of XGBOOST submodels and a neural network, using weather forecasts, runway configuration history and air traffic information as inputs. It was developed for the NASA competition "Runway Functions" hosted on `drivendata.org`, for which it was the winning submission.

III. Problem Description

The goal was to build a model that forecasts the probability that an airport will be in a given configuration, looking ahead every 30 minutes for the next 6 hours - 12 lookahead predictions for each prediction time. As the model is to be used in a real-time application, only data that was available at a given prediction time can be used. Figure 5 shows a simple illustration of the prediction task at a given airport. As an example, one can take a closer look at time=0; the task is to make 12 predictions about what the configuration will be for the 30 through 360 minute lookahead, only using data available at or before time=0. This task is then repeated for all the requested prediction times.

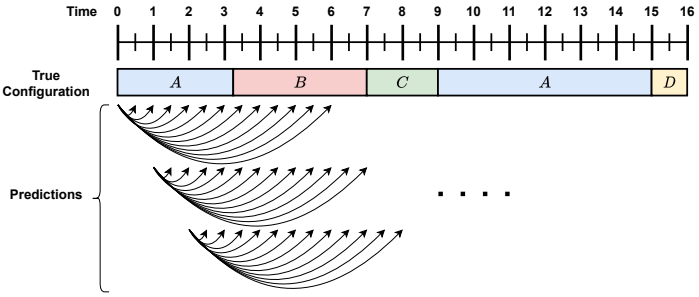


Fig. 5 Diagram showing predictions to be made: 12 predictions per prediction timestamp for 12 lookahead values - 30 through 360 minutes.

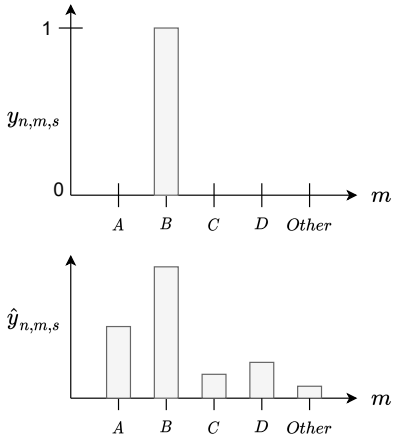


Fig. 6 Example true label distribution $y_{n,m,s}$ and prediction distribution $\hat{y}_{n,m,s}$

Denote by S the number of prediction times, by N the number of predictions to be made for a prediction time (e.g. 120 when requesting 12 lookahead predictions at 10 airports) and by M the set of configurations at a given airport. A single prediction (one arrow in Figure 5) is then a valid categorical probability distribution over set of configurations $m \in M$. The probability predicted for a single configuration is denoted by $\hat{y}_{n,m,s}$, where s is a given prediction time, n is an airport-lookahead combination and m is a given configuration at that airport. Note that for the probability distribution to be valid one needs to make sure that $\sum_{m \in M} \hat{y}_{n,m,s} = 1$ and $\hat{y}_{n,m,s} \geq 0$, for all n and s. Figure 6 shows an example prediction distribution for a given s and n (time-airport-lookahead combination). It also shows the "true" configuration distribution for that s and n where $y_{n,m,s}$ is 1 if configuration m was active and 0 otherwise. The loss function that is used to evaluate the quality of the predictions is shown in Equation 1.

$$L = -\frac{1}{S \times M \times N} \sum_{s=1}^{S} \sum_{m=1}^{M} \sum_{n=1}^{N} y_{n,m,s} \log(\hat{y}_{n,m,s}) + (1 - y_{n,m,s}) \log(1 - \hat{y}_{n,m,s}) \tag{1}$$

One can see that this loss essentially calculates the average binary log-loss between $y_{n,m,s}$ (truth) and $\hat{y}_{n,m,s}$ (prediction probability) across all prediction times, airports, lookaheads, and importantly, configurations. This, in turn, means that all configuration predictions are weighted the same, no configurations are more or less similar to each other in the eyes of the loss function. The problems that this loss definition may create are discussed further in the paper.

Raw feature data (weather forecast, air traffic and configuration history) and truth labels (actual configurations) was used for analysis, including data 10 airports for one whole year: November 2020 to November 2021. Figure 7 shows the sets of prediction times used in the analysis, namely the training, validation and test set.

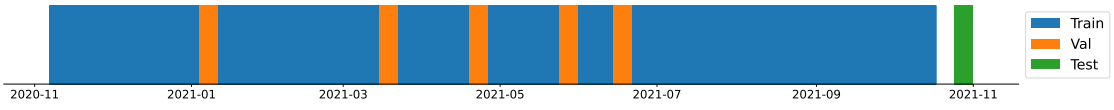


Fig. 7 Timestamps with available data. Train, Val and Test timestamps are shown.

Note that some timestamps were left out completely (not used for train, val or test). A week was left out to create a temporal buffer between the train and the test set to avoid overfitting. Another week was left out at the very beginning of the dataset to allow for feature engineering involving data from previous days. Raw features that were used in training the models are summarized below:

- **Actual departure time and runway logs:** Actual departure time an runway for all flights at a given airport.
- **Actual arrival time and runway logs:** Same as above, but for arrivals.
- **Weather Forecast:** Provided by the LAMP (Localized Aviation Model Output Statistics Program) service. Provides a weather forecast for each hour in the following 24 hours from each prediction time. The weather features include: temperature, wind, cloud ceiling, visibility, cloud coverage, lightning probability and precipitation.

- **Airport Configuration History:** The configuration in which the airport was actually in at a given time.

IV. Methodology

The overall approach is summarized in Figure 8. It contained three major components: 1. Data Processing & Feature Engineering, 2. Submodels and 3. Final Model. In the first step, the raw data is processed into useful features. In the second step submodels are built which take these features and complete some relevant tasks of the final problem, such as predicting whether the airport configuration was likely to switch in a given time frame. Finally, in the final step, a model was built that learns how to map processed features and submodel predictions into predictions $\hat{y}_{n,m,s}$ to minimize the loss in Equation 1.

The idea of using submodels and a final model that aggregates predictions from these submodels is one used in Ensemble Machine Learning[18].

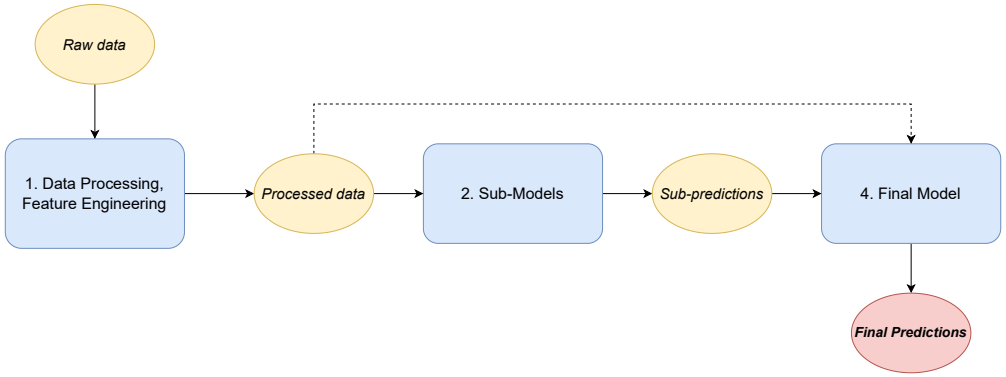


Fig. 8 Overall model approach

A. Data Processing and Feature Engineering

The first essential part of the process was Data Processing and Feature Engineering, which took the raw data files as input to produce features useful for the machine learning models used down the line, for example, instead of individual aircraft arrivals and departures a feature was created to indicate the arrival and departure count in a given period. Figure 9 shows a diagram of the data processing and feature engineering pipeline.

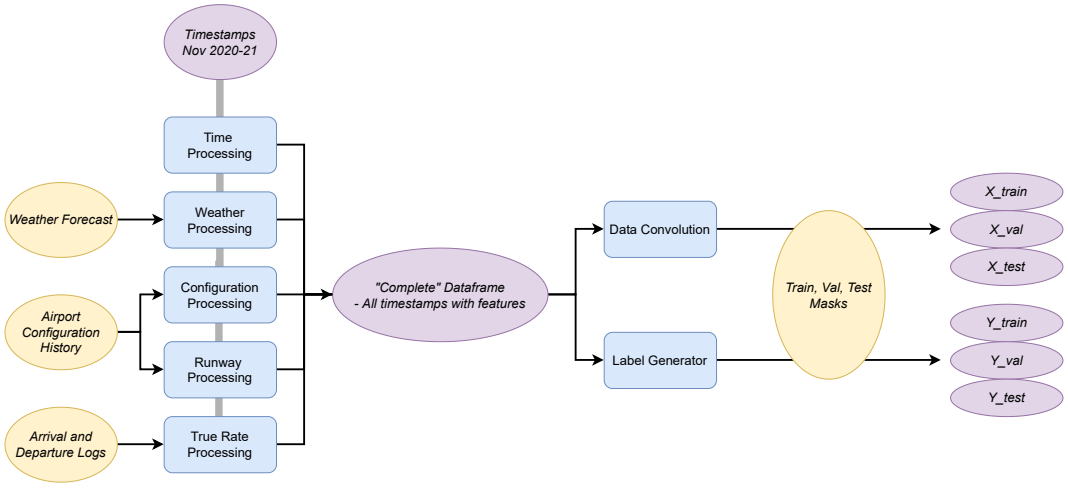


Fig. 9 Data Processing and Feature Engineering workflow

In some cases, after features were processed, data convolution was applied - rows with older timestamps were concatenated to rows with newer timestamps, i.e. the row for 6pm on May 5th may contain features from 5pm on

that day as well. In addition, some submodels required labels that were not explicitly provided. For this reason, each submodel contained an additional label generation processing step. Once all features and labels were generated for all timestamps, the timestamps were divided into train, validation, and test feature sets X and label sets Y.

B. Submodels

Once features were created, they were fed into a number of submodels. Each submodel would complete come sub-task of the final task. This means that the submodels differed in the labels they were predicting. These submodels essentially served as informed feature engineering and dimensionality reduction for the problem. The final models later used submodel predictions to make the final prediction about runway configurations. All submodels designed and trained were classifiers. Specifically, the XGBOOST classifier was used for all of them [19].

Four types of submodels were designed: Binary Runway Submodels (whether or not specific a specific runway would be used), Binary Configuration Submodels (whether or not a specific configuration will be used), Categorical Configuration Submodels (which configuration will be used) and Configuration Change Submodels (whether a configuration change will occur during a time period). Each submodel was trained for a specific time period and aggregated time periods. Figure 10 shows an illustration of the different submodels and their outputs for all lookahead values at a given airport.

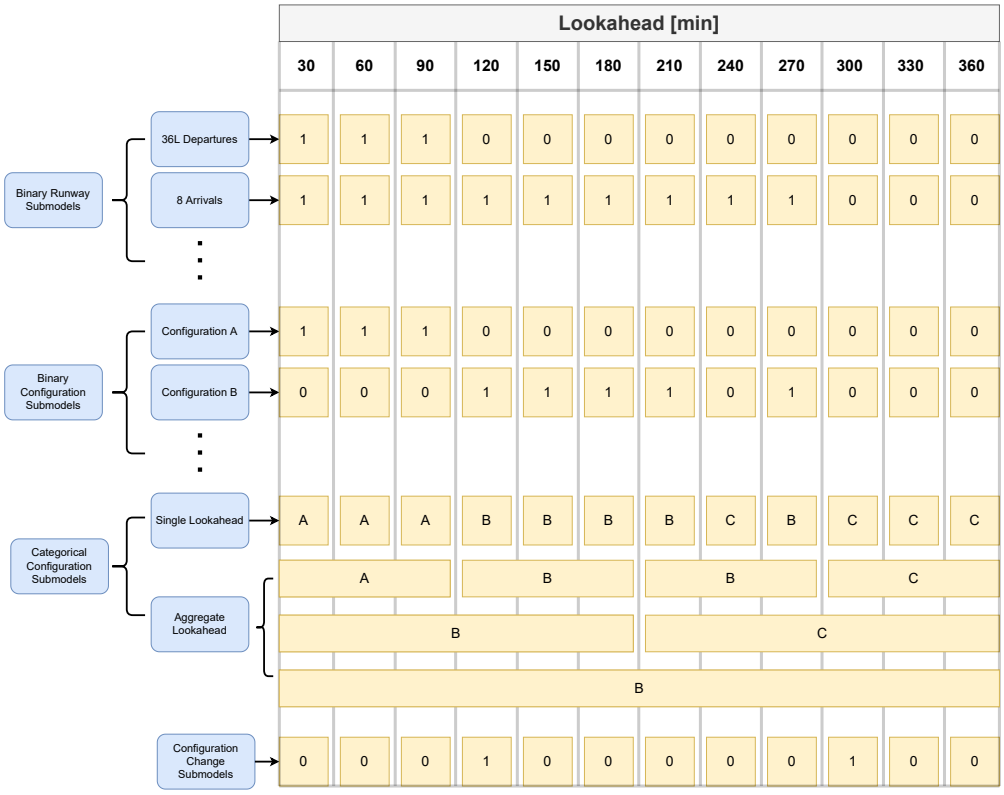


Fig. 10 Example of submodel predictions. For illustration, binary or categorical predictions are shown. Note that the actual submodel outputs are probabilities, not binary or categorical values.

C. Ensemble Network

The final model in this methodology would output the probability distributions for each runway configuration. Since predictions for each airport and lookahead combination have to be made, there is final model constructed for each airport and lookahead combination. Since there are 10 different airports and 12 lookaheads (30 min increments over 6 hours), a total of 120 final models are used to make the probability distributions. In this methodology, the final model is a dense neural network that takes in the submodel predictions as inputs and outputs probability distribution across configurations for each airport and lookahead combination. The purpose of the final model is to leverage the knowledge

of each of the submodels and aggregate them together into a final probability prediction. This ensembling structure has been widely successful for many machine learning applications due to its ability to leverage the strengths of each submodel and mitigate the weaknesses. The overall architecture for the final model is shown in Figure 11 below.

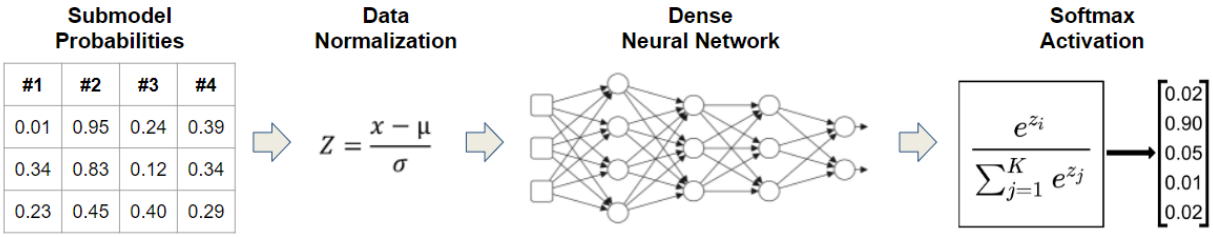


Fig. 11 Final Model Architecture

As mentioned above, the final model takes in the submodel probability outputs (described in the previous section). The submodel probabilities are first normalized to have zero mean and a standard deviation of 1 (using the mean and variance from the training data). From there, the processed submodel outputs are passed into a dense neural network. The final layer of the dense neural network has the same number of units as the number of configurations predicted. The final activation function is a softmax function, which effectively makes the last layer output a probability distribution with respect to the configurations. From there, the probability values can be used to make the submission predictions.

In designing the final model, one of the most important considerations is determining the hyperparameters of the dense neural network model. These include parameters such as the depth of the NN, the number of units, the optimizer, the learning rate, etc. Instead of just arbitrarily trying different hyperparameters and choosing the best combination, modern approaches are used to train the neural network optimization. Specifically, AutoKeras, an AutoML Python package that uses a sophisticated tuner is used to optimize the hyperparameters in order to minimize the validation loss. AutoKeras is configured so that it would spend 8 minutes training the final models for each Airport and Lookahead combination and output the best trained model.

D. Minimum Configuration Support

One of the most important parts of predicting future airport configurations is the concept of inertia. Configuration inertia is an airports resistance to changing configurations. Changing from one configuration to another often results in the change of direction of the runway, number of runways used, etc. This change comes at the cost of downtime, lost capacity, and control effort. As a result, airports do not change configurations instantaneously, even if another configuration is more desirable than the current configuration. While the neural network does have the ability to “learn" these inertia values, as the current configuration is passed as an input, initial analysis illustrated that this implicit inertia consideration is not enough. Specifically, for rare airport configurations, the empirical prior that rare configurations will be active is so low that inertia is not even properly considered. To illustrate how strong inertia is refer to Figure 12. This figure illustrates the “diffusion curve" of configuration inertia for each airport. Effectively, this curve represents the probability that active configurations will persist into future time periods taken across all possible configurations. Obviously from examination, inertia remains quite high for short lookahead values and monotonically decreases as the lookahead increases.

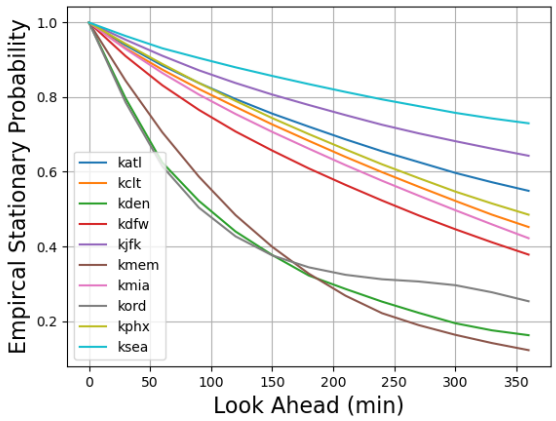


Fig. 12 Empirical Airport Configuration Diffusion Curve (Inertial Curve)

To properly account for inertia a novel method called Minimum Configuration Support is introduced. The idea is that final neural network layer will be redesigned to provide a minimum support probability for the current configuration based on different lookaheads and airports. In effect, this will explicitly bake inertia tendencies into the model, rather than just hoping that the model implicitly learns inertia from the features. The implementation of this method is shown in Figure 13.

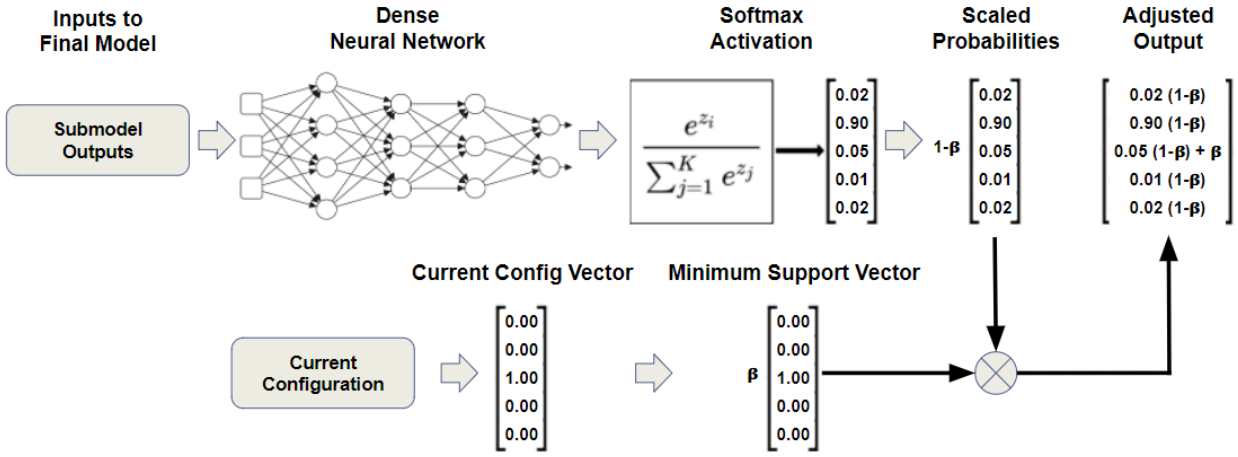


Fig. 13 Minimum Configuration Support Architecture

From above, one can see how the Minimum Configuration Support works. In the above, β is the Minimum Configuration Support parameter. Here β acts as the minimum probability that is provided to the current configuration for that specific prediction. The Minimum Configuration Support method first scales the NN outputted probabilities by a factor of $(1-\beta)$. It then adds β support to the current configuration. This ensures that a minimum probability is assigned to that specific configuration, explicitly making inertia part of the final model. The effects of this approach can be seen in Figure 14.

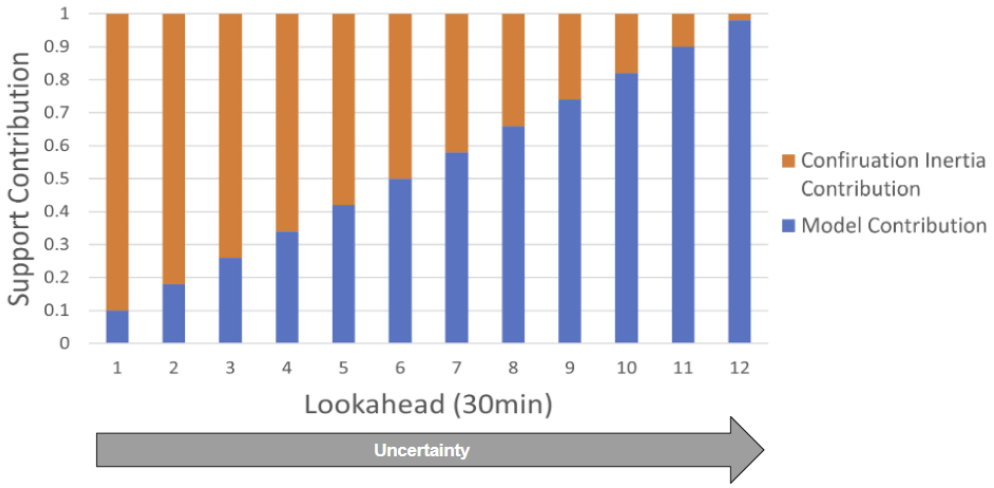


Fig. 14 Minimum Configuration Support Effects to Model Contributions

In simple terms, the Minimum Configuration Support method regulates how much the NN controls in terms of the total contribution to the final predictions. For shorter time periods, the effects of inertia will have the greatest contribution to the overall probability assignments. For future lookahead values, inertia does not play a major role in increasing the uncertainty values. As such, β can be larger for small lookaheads, and decreases as the lookaheads get larger.

As described above, the Minimum Configuration support method is possible to implement in a continuous matrix operations manner. As such, the ensemble neural network layer is redefined and trained with this method in place. In addition, the value β is learned for each airport and lookahead combination in the form of hyperparameter tuning with AutoKeras to minimize the validation loss.

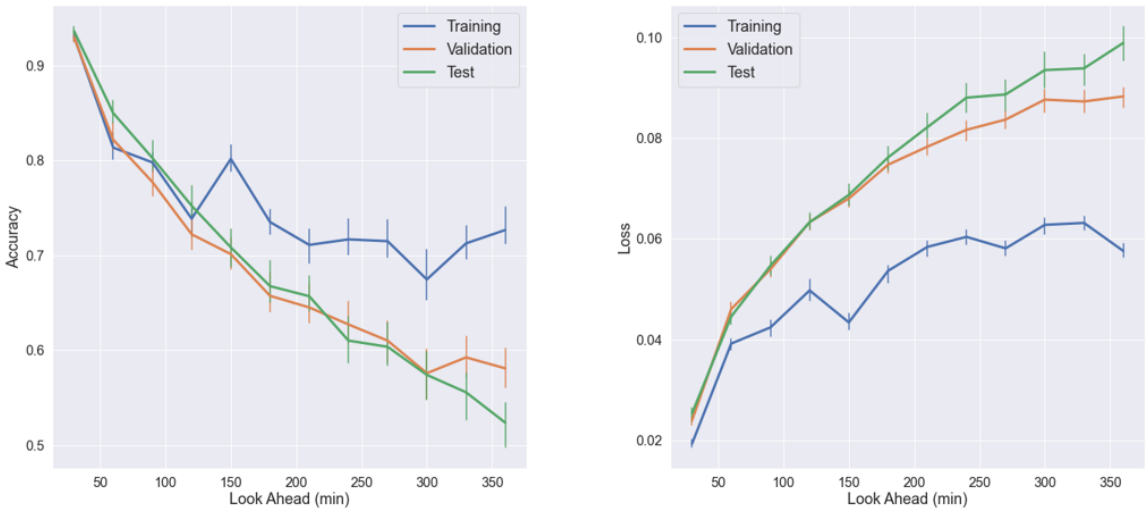
V. Results

This section will explore the results of the final submission to the competition. First the final training and validation loss is summarized in the table below Table 1.

Table 1 Final Submission - Training, Validation, and Test Loss

Training Loss	Validation Loss	Test Loss
0.06934	0.0723	0.0776

The next thing considered is the accuracy and loss as a function of the lookahead values. This analysis can be seen in Figure 15.



(a) Accuracy by Lookahead **(b) Loss by Lookahead**

Fig. 15 Accuracy and Loss vs Lookahead

When inspecting the figure, fluctuations in the training accuracy/loss as a function of lookahead can be seen, while validation and test change smoothly. This is because training is done independently, so different combinations of hyperparameters could cause training to perform better or worse for different values. The key thing is that a monotonic decrease in accuracy and correspondingly a monotonic increase in loss as the lookahead value increases is observed. Another positive sign is that the validation and test accuracy/loss values are comparable, which means the validation loss is serving as a reasonable proxy for the test loss. In addition, these accuracy and loss inspections are performed on a per airport/lookahead basis, as can be seen in Figure 16 and Figure 17 for accuracy and loss, respectively.

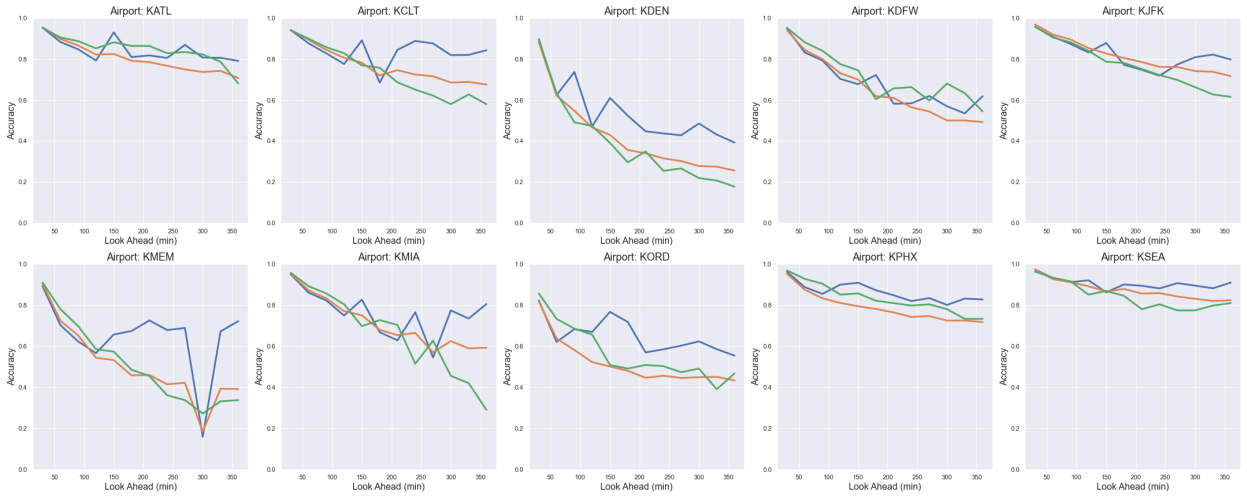


Fig. 16 Accuracy vs Airport and Lookahead

Above the accuracy of the model as a function of lookahead for all 10 different airports can be seen. All airports have relatively high accuracy's for low lookaheads, but diverge for larger lookaheads. Some airports such as ATL, PHX and SEA remain relatively high, even for the largest lookahead values, whereas other airports such as DEN and MIA drop significantly. Another important note that can be seen is that MEM has a significant drop in accuracy for one of

its lookahead values that appears to be an outlier. It is indeed an outlier, and the reason that this occurs is due to the hyperparameter tuning AutoKeras module. Due to some combination of hyperparmaeters, it thought that the minimum configuration support for that model should be much higher than what it should have been. As such, it significantly hurt its performance in training. However, with respect to the loss, the jump does not stand out as much. This is because loss measures probability values, not just the largest probability prediction.

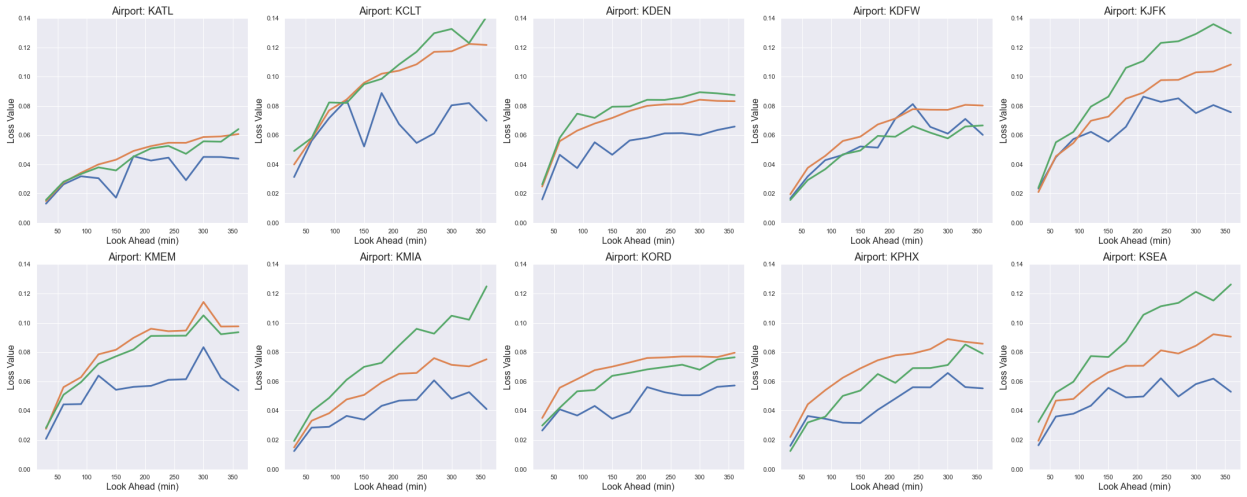


Fig. 17 Loss vs Airport and Lookahead

In Figure 17 the models performance with respect to the objective loss of the competition for each airport can be seen. The airports that were very accurate are not the airports that have the lowest losses. As an example, refer to SEA. Although it had the highest accuracy in terms of predictions, it can be seen that it contributes to some of the largest losses.

Next, one can expect the minimum configuration support term (β) as a function of lookahead and airport. Figure 18 shows the best β value (blue line) found through validation and the empirical diffusion curve for each airport (orange line).

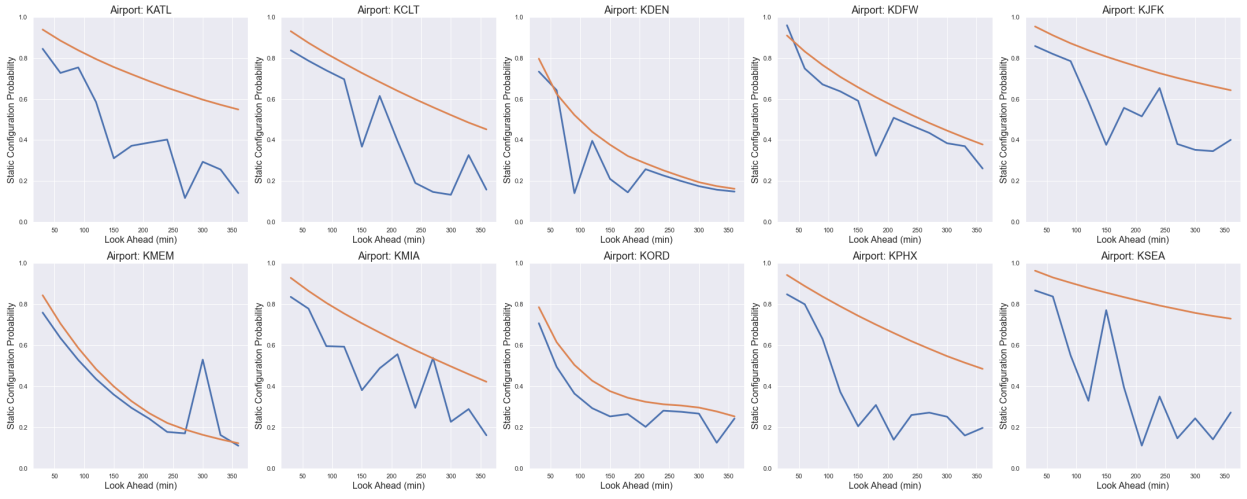
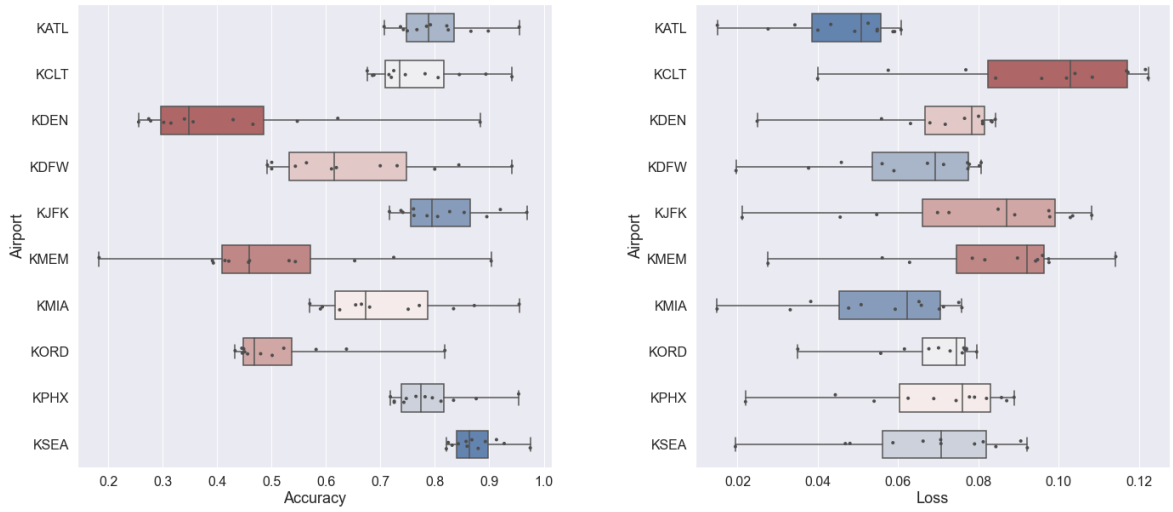


Fig. 18 Configuration Support (β) vs Airport and Lookahead. Learned β value shown in blue, empirical diffusion curve in orange

Overall, the β values found matched the general intuition of what was expected. Specifically, they match two different criteria. Number one, they are (mostly) monotonically decreasing with respect to the lookahead. The reason

they fluctuate so much is that each β value is trained independently across lookaheads; as such, noise and tuner choices will cause it to deviate from a monotonic decreasing nature. In future applications, other approaches can be considered to regulate the shape created by β across lookaheads. The second criterion they match is that they are below the empirical diffusion curve (as it is the minimum configuration support). One observation is that there is an outlier with respect to the MEM airport and the look-ahead. The β value chosen for that airport and the lookahead was significantly higher than it was supposed to be. This was probably just a coincidence in how AutoKeras selected the hyperparameters. However, this is what caused the outlier in MEM training accuracy and loss.

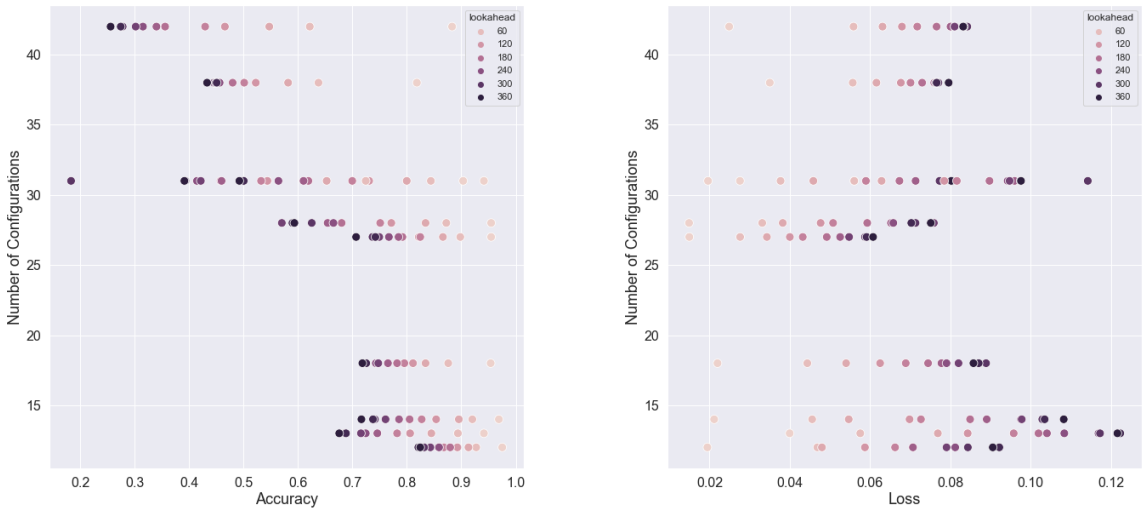
One can then move their attention to analyzing accuracy and loss metrics on an aggregate level vs airport. Figure 19 shows the accuracy and loss values for each of the different airports.



(a) Accuracy by Airport **(b) Loss by Airport**

Fig. 19 Accuracy and Loss by Airport

One can see from Figure 19 that the best performing airports in terms of accuracy are SEA, JFK, and ATL. However, the best performing airports in terms of loss contributions are ATL, DFW, and MIA. One thing is clear when inspecting the above graph; just because an airport does well with respect to accuracy, does not mean it does well with respect to loss. To further illustrate why this may be the case, refer to the figure below. Figure 20 shows the accuracy and loss values the same as above, but instead shows the number of configurations of each airport instead of labeling the airport.



(a) Accuracy by Number of Configurations

(b) Loss by Number of Configurations

Fig. 20 Accuracy and Loss by Number of Configurations

From above, it can be seen that the number of configurations is the predominant factor for accuracy being better or worse. Specifically, the configurations with the highest accuracy are the configurations with the fewest number of configurations and visa versa. On the other hand, the loss values do not share the same correlation pattern with respect to the number of configurations. This is why log-loss is used as a metric for classification problems, as it is more representative of learning the true problem as opposed to shortsightedly assuming that the airports with a small number of configurations perform better because they have higher accuracy.

VI. Conclusion

In summary, the team set out to win the `DrivenData.org` Runway Functions competition hosted by NASA with the objective of predicting future runway configurations using current and past features. The overall approach presented in this paper involved three main components. The first was Data Processing - performing feature engineering on the raw data to create understandable features for prediction models. The second was Submodels - train prediction models that learn relevant subtasks that are related to the overarching task of predicting runway configurations. The third and last component was the Final Model - train dense neural network that leverages the outputs of the submodels to make the final probability predictions. The fundamental idea behind the approach presented was that the submodels would find a smaller (latent) feature space that encapsulates the information that is relevant to the problem at hand. From there, the final model could use the wisdom from all the submodels to make its final prediction. In making submissions, it was found that the model could not adequately account for configuration inertia (the tendency for configurations not to change). As such, a novel approach we called Minimum Configuration Support was introduced to explicitly account for configuration inertia. After adding the Minimum Configuration Support method, the models performed significantly better. Presently (as of May 31st 2022), the final submission presented in this paper is tentatively in first place on the prescreened leaderboard for the Driven Data competition, with the final results expected to be released in June 2022. Overall, the authors are confident in their final design and expect their models to perform well on the final evaluation dataset.

VII. Future Work

In addition to redesigning the labels and loss function for the problem, there are more improvements that could be made make such that the models perform even better. First, there are features that were omitted in the current models, these include scheduled arrival and departure rates, taxi times, and airspace interactions. Airspace interactions, in

particular, are of interest, as they were shown to be highly important in deciding on runway configurations in metroplex areas with multiple airports by Murca in [20] for LGA, JFK, and EWR. The metroplex airspace is shown in Figure 21.

The final improvements to be made in the future concern the final model. Firstly, the current version of the model assumed the same minimum configuration support for all configurations at a given airport. However, looking at diffusion curves for specific configurations at a given airport showed that configurations will likely exhibit different diffusion behaviours. Figure 22 shows the diffusion curves for all configurations at KATL, with a clear distinction between what the authors call "stationary" configurations - configurations that tend to prevail, and “transitional" configurations - ones that are likely to switch more frequently. The idea of different diffusion behaviors could be implemented into the learning to improve performance.

Finally, using other neural network architectures that are more suited for time-series data, such as convolutional networks, recurrent networks, or transformers, could achieve higher performance as well.

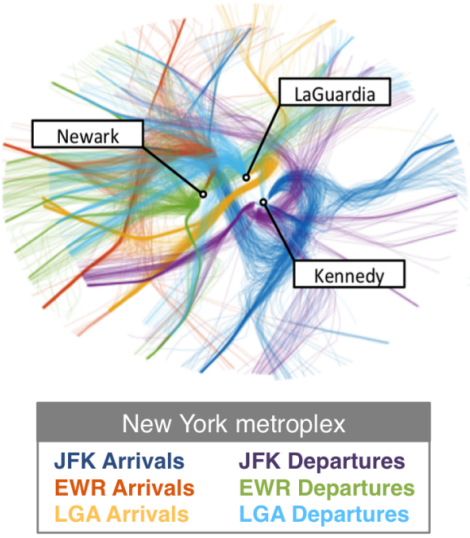


Fig. 21 New York airspace [20]

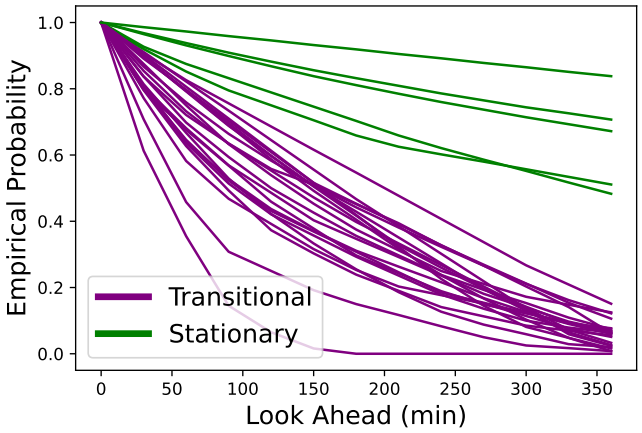


Fig. 22 Stationary versus transitional diffusion curves at KATL

References

[1] Avery, J., and Balakrishnan, H., “Data-Driven Modeling and Prediction of the Process for Selecting Runway Configurations,” *Transportation Research Record*, Vol. 2600, No. 1, 2016, pp. 1–11. https://doi.org/10.3141/2600-01, URL https://doi.org/10.3141/2600-01.

[2] Gilbo, E. P., “Airport capacity: Representation, estimation, optimization,” *IEEE Transactions on control systems technology*, Vol. 1, No. 3, 1993, pp. 144–154.

[3] Vascik, P. D., “Systems analysis of urban air mobility operational scaling,” Ph.D. thesis, Massachusetts Institute of Technology, 2020.

[4] Cavalheiro, G. V., Vascik, P. D., Travnik, M., Salgueiro, S., and Hansman, J., *Characterizing Effective Navigation Performance of Legacy Aircraft to Enable Reduced Procedural Separation in Support of Advanced Air Mobility*, 2021. https://doi.org/10.2514/6.2021-2389, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-2389.

[5] Richetta, O., and Odoni, A. R., “Solving Optimally the Static Ground-Holding Policy Problem in Air Traffic Control,” *Transportation Science*, Vol. 27, No. 3, 1993, pp. 228–238. URL http://www.jstor.org/stable/25768591.

[6] “Enhanced Preferential Runway Advisory System (ENPRAS),” *Flight Transportation Associates*, 2001.

[7] Southgate, D. G., and Sedgwick, S. J., “Time stamped aircraft noise prediction-Replacing the’Average Day’with the’Composite Year’,” *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, Vol. 2006, Citeseer, 2006, pp. 4538–4546.

[8] Li, L., and Clarke, J.-P., “A stochastic model of runway configuration planning,” *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 7697.

[9] Provan, C. A., and Atkins, S. C., "Optimization models for strategic runway configuration management under weather uncertainty," *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth, Texas*, 2010.

[10] Weld, C., Duarte, M., and Kincaid, R., "A runway configuration management model with marginally decreasing transition capacities," *Advances in Operations Research*, Vol. 2010, 2010.

[11] Oseguera-Lohr, R. M., Phojanamongkolkij, N., Lohr, G. W., and Fenbert, J. W., "Benefits assessment for tactical runway configuration management tool," *2013 Aviation Technology, Integration, and Operations Conference*, 2013, p. 4395.

[12] Liu, P.-c. B., Hansen, M., and Mukherjee, A., "Scenario-based air traffic flow management: From theory to practice," *Transportation Research Part B: Methodological*, Vol. 42, No. 7-8, 2008, pp. 685–702.

[13] Buxi, G., and Hansen, M., "Generating probabilistic capacity profiles from weather forecast: A design-of-experiment approach," *Proc. of USA/Europe Air Traffic Management Research & Development Seminar*, 2011.

[14] Hesselink, H., and Nibourg, J., "Probabilistic 2-Day Forecast of Runway Use," *Ninth USA/Europe air traffic management research and development seminar (ATM2011)*, Citeseer, 2011.

[15] Houston, S., and Murphy, D., "Predicting runway configurations at airports," Tech. rep., 2012.

[16] Balakrishnan, H. B., and Ramanujam, V. R., "Estimation of maximum-likelihood discrete-choice models of the runway configuration selection process," 2011.

[17] Ramanujam, V., and Balakrishnan, H., "Data-driven modeling of the airport configuration selection process," *IEEE Transactions on Human-Machine Systems*, Vol. 45, No. 4, 2015, pp. 490–499.

[18] Polikar, R., *Ensemble Learning*, Springer US, Boston, MA, 2012, pp. 1–34. https://doi.org/10.1007/978-1-4419-9326-7_1, URL https://doi.org/10.1007/978-1-4419-9326-7_1.

[19] Chen, T., and Guestrin, C., "XGBoost, A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. https://doi.org/10.1145/2939672.2939785, URL http://dx.doi.org/10.1145/2939672.2939785.

[20] Condé Rocha Murça, M., "Data-driven modeling of air traffic flows for advanced Air Traffic Management," Ph.D. thesis, Massachusetts Institute of Technology, 2018.