The Aeronautical Journal (2026), 1–18
doi:10.1017/aer.2026.10193

RESEARCH ARTICLE

Aircraft takeoff speed prediction with deep learning: a comparative study of MLP, 1D-CNN, LSTM and attention-based architectures on Boeing 737-300 data

Mehmet Konar[1], Hüseyin Alp Ayaz[2], Erkan Caner Özkat[3] and Aydın Türkmen[4]

[1]Department of Aviation Electrical and Electronics, Faculty of Aeronautics and Astronautics, Erciyes University, Kayseri, Türkiye
[2]Department of Aviation Electrical and Electronics, Graduate School of Natural and Applied Sciences, Erciyes University, Kayseri, Türkiye
[3]Department of Mechanical Engineering, Faculty of Engineering and Architecture, Recep Tayyip Erdogan University, Rize, Türkiye
[4]Department of Motor Vehicles and Transportation Technologies, Airbus TUSAŞ Aviation Vocational School, Kahramanmaraş Istiklal University, Türkiye
Corresponding author: Mehmet Konar; Email: mkonar@erciyes.edu.tr

Received: 13 May 2026 UTC; **Revised:** 25 May 2026 UTC; **Accepted:** 25 May 2026 UTC

Keywords: aircraft; Bayesian hyperparameter optimisation; deep learning; LSTM; takeoff speed

Abstract

Modern aviation supports an ever-broader range of civil and military missions, and the airframes designed for these missions must satisfy stringent safety and performance requirements. The takeoff and landing phases are the most accident-prone portions of a flight despite representing only a short interval of the total block time, which makes the accurate prediction of takeoff speed a safety-relevant problem. A previous machine learning study addressed the takeoff-speed prediction problem of the Boeing 737-300 with classical regressors using pressure altitude, outside air temperature, gross weight and flap angle as the predictors. In the present work, the same regression problem is revisited under the deep learning paradigm. Four neural architectures are trained on an identical pre-processing pipeline and train-validation partition, namely a multilayer perceptron, a one-dimensional convolutional network, a long short-term memory network and a wide-and-deep architecture incorporating multi-head self-attention. Among the four candidates, the long short-term memory network attains the lowest root mean square error and mean square error on the unseen test file and is subsequently subjected to Bayesian hyperparameter optimisation through the Keras Tuner library. The predicted and the measured takeoff speeds are reported side by side for the first time in the deep learning literature for this airframe, and the simulation results indicate that the developed networks constitute an effective alternative tool for takeoff-speed prediction.

Nomenclature

AI	artificial intelligence
ANN	artificial neural network
CART	classification and regression trees
CNN	convolutional neural network
DL	deep learning
GRU	gated recurrent unit
LR	linear regression
LSTM	long short-term memory
MAE	mean absolute error
ML	machine learning
MLP	multilayer perceptron

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

MSE	mean square error
QAR	quick access recorder
ReLU	rectified linear unit
RF	random forest
RMSE	root mean square error
SVM	support vector machine
SVR	support vector regression
TF	TensorFlow
TFLite	TensorFlow Lite
UAV	unmanned aerial vehicle
XGBoost	extreme gradient boosting
V_1	decision speed
V_R	rotation speed
V_2	takeoff safety speed

1.0 Introduction

Civil aviation today supports a much broader set of activities than passenger transport alone, ranging from cargo logistics and defence to precision agriculture, aerial mapping, surveillance and search-and-rescue operations [1, 2]. Continuous progress in airframe design, propulsion and avionics has driven a parallel growth in fleet diversity, with vehicles of widely different sizes and mission profiles entering service, and has placed safety and operational performance at the centre of present-day aviation research. Within the flight envelope of a commercial transport aircraft, the takeoff and landing phases occupy only a small fraction of the total block time yet are responsible for a disproportionately large share of in-service accidents [3–5]. Achieving an accurate prediction of the takeoff speed is therefore directly tied to flight safety and motivates the present study.

For a commercial transport aircraft, the takeoff speed is not a fixed value but a function of several operating conditions, including pressure altitude, outside air temperature, gross weight, flap setting, runway condition and ambient wind. Aircraft operations manuals tabulate the values of V_1, V_R and V_2 against these variables, and the relevant entry is consulted by the crew before each departure either from the manual itself or from a flight-management-system computation [6]. The underlying mapping from inputs to takeoff speed is, however, strongly nonlinear and only partially available in closed analytical form, which makes manual interpolation across the whole operating envelope impractical. Data-driven techniques have therefore become the method of choice for takeoff- and landing-speed estimation problems in the recent literature [7–10].

Earlier work by Konar and Bağış on aviation-speed parameter estimation employed an adaptive-network-based fuzzy inference system (ANFIS) and demonstrated that the speed parameter of a flight control system can be recovered from recorded telemetry without recourse to a closed-form analytical model [11]. The same group subsequently introduced a fuzzy model whose membership-function parameters were tuned by artificial bee colony (ABC) and differential evolution (DE) metaheuristics and reported accurate computation of aircraft speed and fuel consumption from recorded flight data [12]. Building on these fuzzy baselines, the following decade saw a rapid adoption of neural and recurrent architectures for closely related problems. An early data-driven contribution along this line was made by Diallo, who trained a feed-forward artificial neural network (ANN) to estimate landing speed and reported accurate predictions across both calm and gusty wind regimes [7]. The hard-landing detection problem was addressed by Hu et al. with an optimised support vector machine (SVM), which was shown to discriminate hard from normal landings with high success on flight data records [13]. A long short-term memory (LSTM) regressor trained on cloud-collected sensor measurements was later proposed by Tong et al. for landing-speed estimation, with reported improvements over classical baselines [8]. Zhang and Zhu likewise applied LSTM networks to quick access recorder (QAR) records and emphasised the ability of the recurrent gates to capture long-range temporal dependencies that previous hard-landing predictors had failed to model [14, 15].

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

In the small unmanned-aircraft (UAV) domain, Borup et al. estimated angle-of-attack, sideslip and airspeed from low-cost distributed pressure sensors on a fixed-wing platform; their comparison between an ANN and linear regression (LR) consistently favoured the neural model in terms of error [16]. Kovarik et al. carried out a broader study in which LSTM, SVM and neural ordinary differential equations were trained on a large flight-data corpus for phase-of-flight (PoF) identification, and concluded that the LSTM family is best suited to this temporal task [17]. For landing-speed and ground-speed estimation, Puranik et al. reported that a random forest (RF) regressor outperformed the techniques previously available in the literature [10]. Jarry et al. investigated approach and landing parameters such as fuel-flow rate, flap deflection and landing-gear status through an LSTM-based estimator built on flight-data-recorder traces [18], while Kang et al. cast the landing-speed problem as a sequence-to-sequence learning task on QAR data and obtained accuracies clearly above those of conventional regressors [9]. A hybrid scheme that couples boosting with LSTM networks was studied by Martinez et al. for unstable-approach forecasting and was reported to surpass either component used in isolation [19].

Recurrent and attention models have also been adopted for diagnostic and prognostic tasks on small aircraft. Ozkat et al. used LSTM on vibration data acquired from a multi-rotor UAV to estimate the remaining useful life of the platform [20], and a follow-up wavelet-scattering LSTM autoencoder was later proposed to detect propeller-blade failures from intentionally induced anomalies in vibration signals [21]. In an adjacent line of work, Karaburun et al. employed SVR, RF and LSTM to estimate the state of charge of Li-Po batteries on UAVs and demonstrated that recurrent models remain effective even when only a small number of physical predictors is available [22]. For aircraft trajectory prediction, several LSTM and CNN-LSTM models have been reported, with accuracy gains over classical Kalman filters [23–25].

Directly related to the present work, the takeoff-speed regression problem of the Boeing 737-300 was recently treated by Karaburun, Ark Hatipoğlu and Konar with classical machine learning algorithms, namely LR, support vector regression (SVR), classification and regression trees (CART), RF and extreme gradient boosting (XGBoost) [26]. In their analysis, hyperparameter-tuned XGBoost yielded the lowest error among the proposed models. However, deep learning techniques – which are now the dominant family in aviation-AI work and are widely used to capture nonlinear interactions among flight parameters – were not considered in that earlier study.

To the best of the authors' knowledge, no published comparison of multiple deep learning architectures has been reported for the takeoff-speed prediction problem of the Boeing 737-300. The contribution of this paper is to address that gap. Following the experimental framework of Karaburun et al. [26], the same data set is reused with pressure altitude, outside air temperature, gross weight and flap angle as the independent variables and takeoff speed as the dependent variable. Four deep learning architectures – a multilayer perceptron (MLP), a one-dimensional convolutional neural network (1D-CNN), an LSTM network and a wide-and-deep architecture incorporating multi-head self-attention – are trained and evaluated under an identical pre-processing pipeline and train-validation split. Bayesian hyperparameter optimisation with Keras Tuner is then applied to the strongest of the four base models. The predicted takeoff-speed values are compared with the measured values from the operations manual for the first time in the deep learning literature, and the obtained results indicate that the developed deep learning models constitute an effective alternative tool for takeoff-speed prediction.

2.0 Deep learning

Deep learning denotes the family of machine learning techniques in which multi-layer artificial neural networks are trained end-to-end to extract hierarchical patterns directly from raw data [27, 28]. The defining departure from classical machine learning lies in the representation step: whereas conventional pipelines rely on hand-crafted predictors that are fed unchanged to the regressor, a deep network discovers its own intermediate representations during training. This data-driven feature extraction has made deep learning the dominant approach in domains as diverse as computer vision, speech recognition, time-series forecasting and flight-parameter estimation in aviation.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

In this study, four deep learning architectures were selected for takeoff speed prediction. These architectures are MLP, 1D-CNN, LSTM and a wide-and-deep architecture with multi-head self-attention. The selected architectures are briefly explained below.

2.1 Multilayer perceptron

The MLP is the basic and most widely used fully connected feed-forward neural network. The MLP consists of an input layer, one or more hidden layers and an output layer. In each layer, the inputs are multiplied with the weights and the bias term is added. Then, a nonlinear activation function is applied to obtain the output of the layer. The mathematical model of a hidden layer of the MLP is given in Equation (1).

$$\mathbf{h}_\ell = \sigma(\mathbf{W}_\ell \mathbf{h}_{\ell-1} + \mathbf{b}_\ell), \quad \mathbf{h}_0 = \mathbf{x}, \tag{1}$$

Here, $\mathbf{h}_\ell$ is the output of the ℓ-th layer, $\mathbf{W}_\ell$ is the weight matrix, $\mathbf{b}_\ell$ is the bias vector, $\mathbf{x}$ is the input vector and $\sigma(\cdot)$ is the nonlinear activation function. In this study, the rectified linear unit (ReLU) is selected as the activation function. The final layer is a linear layer and produces the scalar takeoff speed estimate. To prevent overlearning, dropout [29] and L_2 weight decay are used in the hidden layers. The MLP is the basic deep learning model and is used as a reference point for comparison with other deep learning architectures.

2.2 One-dimensional convolutional neural network

The 1D-CNN is a deep learning architecture that applies the convolution operation along a single dimension [30]. In this study, the four input features are reshaped into a short signal of length $T = 4$ with a single channel, and the convolution layers are applied along this length dimension. The output of a one-dimensional convolution layer is given in Equation (2).

$$y_{c,t} = \sigma\left(\sum_{c'=1}^{C_{\text{in}}} \sum_{\tau=0}^{k-1} w_{c,c',\tau}\, x_{c',t+\tau} + b_c\right), \tag{2}$$

Here, $y_{c,t}$ is the output of the convolution operation at the position t for the output channel c, $w_{c,c',\tau}$ are the weights of the convolution kernel of size k, b_c is the bias term and $\sigma(\cdot)$ is the ReLU activation function. After a stack of convolution layers, a global average pooling layer is used to aggregate the temporal axis, and a dense head produces the takeoff speed estimate. The 1D-CNN was selected to investigate whether the parameter sharing across the four predictors provides any benefit on this tabular data set.

2.3 Long short-term memory

The LSTM, introduced by Hochreiter and Schmidhuber [31], is one of the most widely used recurrent neural networks in aviation deep learning applications [9, 17, 18]. The LSTM cell contains a cell state, a hidden state and three gates, namely the input gate, the forget gate and the output gate. These gates control the information flow in the cell state and allow the network to capture long-term dependencies between the input sequence elements. The mathematical model of an LSTM cell is given in Equation (8).

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f), \tag{3}$$

$$\mathbf{i}_t = \sigma_g(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \tag{4}$$

$$\mathbf{o}_t = \sigma_g(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o), \tag{5}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{6}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{7}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{8}$$

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

Here, $\mathbf{x}_t$ is the input vector at time step t, $\mathbf{h}_t$ is the hidden state, $\mathbf{c}_t$ is the cell state, $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ are the forget, input and output gates, respectively, $\mathbf{W}_*$ and $\mathbf{U}_*$ are the weight matrices, $\mathbf{b}_*$ are the bias vectors, $\sigma_g(\cdot)$ is the logistic activation function and $\odot$ is the element-wise multiplication. In this study, the four input features were reshaped as a sequence of length four and the LSTM model was created with two stacked LSTM layers followed by a dense head to produce the takeoff speed estimate. Although the input features are not chronologically ordered, the LSTM cells can discover the nonlinear interactions between adjacent features through the recurrent gates.

2.4 Wide-and-deep architecture with self-attention

The wide-and-deep architecture combines a linear projection of the raw inputs (the wide branch) with a deeper learned representation (the deep branch), and the two branches are concatenated to produce the output [32]. In this study, a multi-head self-attention block is used in the deep branch in the style of the TabTransformer architecture [33, 34]. Each input feature is first projected to an e-dimensional embedding vector. Then, the multi-head self-attention block is applied to the embedded features as given in Equation (9).

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{9}$$

Here, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are the query, key and value matrices, respectively, which are obtained by the linear projections of the embedded features, and d_k is the dimension of the keys. Residual connections and layer normalisation are applied around the attention and feed-forward sub-layers to stabilise the training. The output of the attention block is flattened and concatenated with a 16-unit linear projection of the raw inputs (the wide branch), and the result is passed through a final dense layer to produce the takeoff speed estimate. This architecture was included in the comparison to represent the modern tabular deep learning family [33, 35].

2.5 Loss function and optimisation

All four deep learning models were trained by minimising the mean square error (MSE) loss function. The mathematical model of the MSE loss function is given in Equation (10).

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{10}$$

Here, N is the number of training samples, y_i is the actual takeoff speed of the i-th sample and $\hat{y}_i$ is the predicted takeoff speed of the i-th sample. The Adam optimiser [36] was used to update the weights of the network during training. To further improve the training performance, early stopping on the validation loss and a learning rate scheduler that halves the learning rate when the validation loss plateaus were applied during training. After the base model training step, Bayesian hyperparameter optimisation [37] was applied to the best-performing base model with the Keras Tuner library [38] to obtain the final tuned model.

2.6 Performance metrics

The root mean square error (RMSE) expressed in Equation (11), the MSE expressed in Equation (12) and the mean absolute error (MAE) expressed in Equation (13) were used as the performance criteria for the evaluation of the deep learning models. These metrics were also used in the previous study [26] for the takeoff speed prediction problem and are widely accepted as the standard regression metrics in the literature [39]. The smaller the error value, the higher the model accuracy is considered.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{11}$$

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

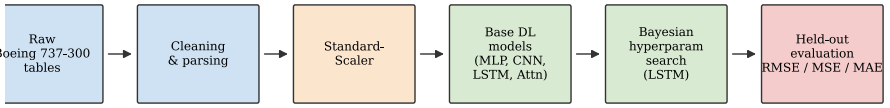

Figure 1. *Block diagram of the proposed deep learning models for takeoff speed prediction.*

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{12}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|. \tag{13}$$

Here, N is the number of test samples, y_i is the actual takeoff speed of the i-th test sample and $\hat{y}_i$ is the predicted takeoff speed of the i-th test sample.

3.0 Problem definition

This section describes the data set, the pre-processing pipeline and the training protocol that were adopted to obtain the deep-learning takeoff-speed predictors compared in the remainder of the paper.

The training corpus is the Boeing 737-300 performance data set distributed in the manufacturer's aircraft operations manual [6] and reused, for direct comparability, from the earlier study of Karaburun et al. [26]. Five quantities are recorded for every entry: pressure altitude (ft), outside air temperature (°F), gross weight (klb), flap angle (°) and takeoff speed (kn). The data span the operational ranges 0–9,000 ft, (−40)–160°F, 75–140 klb, 1°–15° and 98–162 kn respectively [6], with the flap setting restricted to the three discrete values 1°, 5° and 15° permitted on the airframe.

Following the same input-output decomposition as in the previous study, pressure altitude, outside air temperature, gross weight and flap angle act as the predictors and the takeoff speed is the regression target. The functional relationship between these four predictors and the takeoff speed is strongly nonlinear and only partially captured by the tabulated charts in the operations manual, which makes a closed-form analytical estimator impractical across the full operating envelope. Learning-based regressors are therefore used here, and deep neural networks – a sub-family of artificial intelligence that builds its own intermediate representations from the data – are adopted as the modelling family for this work. The complete block diagram of the proposed pipeline is shown in Fig. 1.

The data set used in this study was provided as two separate files. The first file contains the training data with 2,690 valid samples after the removal of one blank line, and the second file contains the test data with 97 samples. In contrast to the previous study [26], in which the data set was split as 70% training and 30% test by random sampling, two separate files were used in this study. None of the 97 samples in the test file appears in the training file, and, therefore, the test evaluation is performed on completely unseen data. This is an important difference from the previous study and provides a more difficult generalisation evaluation. The descriptive statistics of the training and test subsets are presented in Table 1, and the correlation matrix of the input features and the takeoff speed on the training set is presented in Fig. 2.

A pre-processing pipeline was then assembled to prepare the records for the deep learning models. The training records were first inspected for missing entries and out-of-range values, and the few problematic rows were corrected or discarded. To ensure that all predictors enter the loss function at a comparable magnitude, each input feature was rescaled to zero mean and unit variance using the StandardScaler transformer from the Scikit-Learn library. The same affine transformation was also applied to the regression target so that the optimiser would operate on a numerically well-conditioned

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

Table 1. *The descriptive statistics of the training and test subsets (mean ± standard deviation)*

Variable	Training ($n = 2690$)	Test ($n = 97$)
Pressure altitude (ft)	4111.9 ± 2792.5	4536.1 ± 2897.8
Outside air temperature (°F)	77.2 ± 56.9	83.9 ± 52.6
Flap angle (°)	6.9 ± 5.8	7.2 ± 6.0
Gross weight (klb)	101.8 ± 21.0	100.1 ± 19.6
Takeoff speed (kn)	125.8 ± 16.9	124.7 ± 16.7

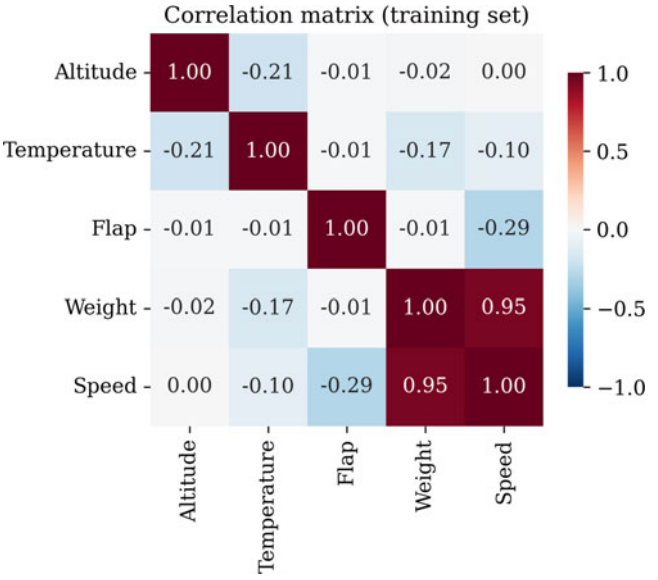

Figure 2. *The Pearson correlation matrix of the four input features and the takeoff speed on the training set.*

loss surface; at inference time, the scaler was inverted before the error metrics in Kn were computed. To prevent information leakage from the test partition, the mean and variance used by the scaler were estimated exclusively from the training subset and the resulting parameters were then re-used to transform the test subset.

After the pre-processing step, the training file was further split as 85% training data and 15% validation data by a single random holdout with a fixed seed of 42. This validation partition was used by the early stopping callback with patience of 40 epochs and by the learning rate scheduler that halves the learning rate when the validation loss has not improved for 15 epochs (with a minimum learning rate of 10^{-5}). The Adam optimiser with an initial learning rate of 10^{-3}, a batch size of 64 and a maximum of 400 epochs were used in the training of all four base models. The TensorFlow 2.17 library [40] and the Keras Tuner library [38] were used for the implementation of the deep learning models and the hyperparameter optimisation step. The Scikit-Learn library [41] was used for the pre-processing and the metric computation steps. The global NumPy and TensorFlow seeds were both fixed at 42 so that the reported metrics can be reproduced from the released training scripts.

To make the four base architectures fully reproducible, the layer-by-layer configuration that was kept identical across all reported runs is presented in Table 2. The four predictors enter every model through the same standardised four-dimensional vector, and the three networks that operate on a sequence representation (1D-CNN, LSTM and the wide-and-deep network with self-attention) reshape this vector to a length-four signal with a single channel before the first internal layer.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

Table 2. *Architectural configuration of the four base deep learning models used in this study*

Model	Layer configuration
MLP	Dense(64, ReLU, $L_2 = 10^{-4}$) – Dropout(0.10) – Dense(64, ReLU, $L_2 = 10^{-4}$) – Dropout(0.10) – Dense(32, ReLU) – Dense(1, linear)
1D-CNN	Reshape(4, 1) – Conv1D(32 filters, kernel = 2, same padding, ReLU) – Conv1D(32 filters, kernel = 2, same padding, ReLU) – GlobalAveragePooling1D – Dense(32, ReLU) – Dense(1, linear)
LSTM	Reshape(4, 1) – LSTM(48 units, return sequences) – LSTM(48 units) – Dense(32, ReLU) – Dense(1, linear)
Wide & Deep + Attention	Reshape(4, 1) – Dense(8, linear) embedding – MultiHeadAttention(4 heads, key_dim = 8) with residual and LayerNorm – Feed-forward Dense(8, ReLU) with residual and LayerNorm – Flatten, concatenated with Dense(16, ReLU) wide branch on the raw input – Dense(32, ReLU) – Dense(1, linear)

Table 3. *The search ranges selected for the Bayesian hyperparameter optimisation of the LSTM model*

Hyperparameter	Search range
LSTM-1 units	{32, 48, 64, 96, 128}
LSTM-2 units	{16, 32, 48, 64}
Dropout rate	[0.0, 0.4] in steps of 0.1
Dense head units	{16, 32, 64}
Adam learning rate	$\{10^{-2}, 5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$
Recurrent dropout	{on, off}

After the training of the four base models, the model with the smallest error value was selected for the hyperparameter optimisation step. The hyperparameters of the selected model and their search ranges are given in Table 3. Bayesian hyperparameter optimisation [37] as implemented in the Keras Tuner library [38] was applied to obtain the best hyperparameter combination. Each trial was trained on the 85% training partition with early stopping on the 15% validation partition. The validation MSE value was used as the objective of the Bayesian optimisation, and a total of 24 trials was used as the search budget. The hyperparameter combination with the smallest validation MSE was selected as the final tuned model, and the final model was re-trained with extended early stopping (patience of 60 epochs) before the evaluation on the test data.

4.0 Simulation results

The simulation outcomes obtained with the proposed deep learning pipeline are reported below in detail.

The experimental protocol was carried out exactly as described in Section 3. The Boeing 737-300 manual data [6] were taken as the working set, with pressure altitude, outside air temperature, gross weight and flap angle entering the model as predictors and takeoff speed acting as the regression target. Records were screened for missing entries and out-of-range values, each predictor was standardised through the Scikit-Learn StandardScaler transformer and the training corpus was further partitioned into 85% training and 15% validation subsets by random holdout with a fixed seed. The four candidate architectures – MLP, 1D-CNN, LSTM and the wide-and-deep network with multi-head self-attention – were

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

Table 4. *The RMSE, MSE and MAE values of the four base deep learning models on the test data, with the trainable parameter count and the training time*

Model	RMSE (kn)	MSE (kn^2)	MAE (kn)	Parameters	Train time (s)
MLP	1.0835	1.1740	0.7877	6,593	14.9
1D-CNN	0.7853	0.6168	0.4466	3,265	109.6
LSTM	0.7745	0.5999	0.4527	29,825	159.2
Wide & Deep + Attention	0.8221	0.6758	0.5389	2,929	138.9

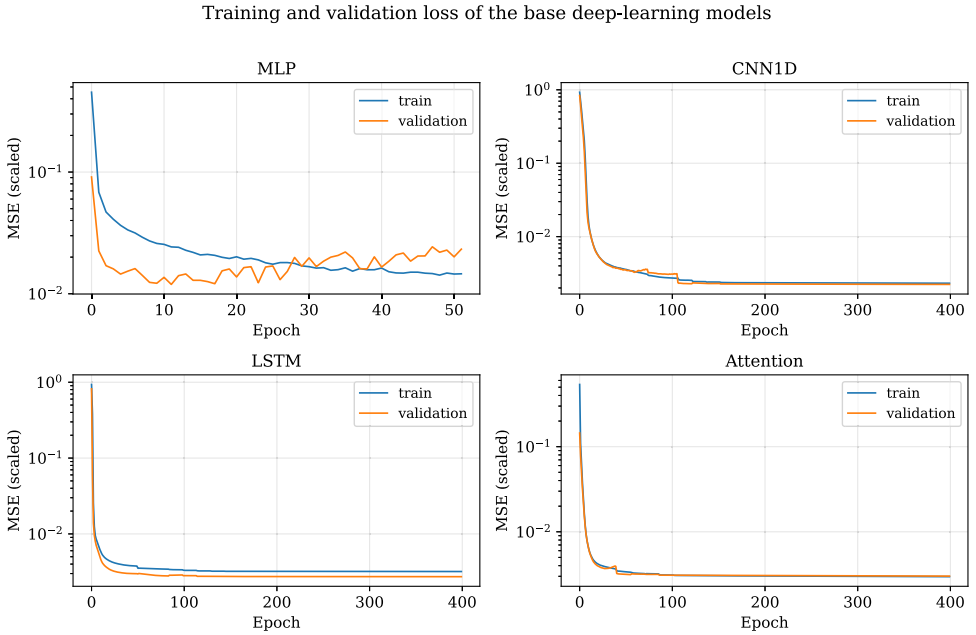

Figure 3. *The training and validation loss curves of the four base deep learning models.*

then trained on the training partition under identical callbacks (early stopping and learning-rate reduction on plateau) and were finally evaluated on the held-out test file of 97 unseen samples.

Figure 3 shows the training and validation loss curves of the four base deep learning models during the training process. When Fig. 3 is examined, it is seen that the four models converge monotonically, and the validation loss curves follow the training loss curves. This indicates that the regularisation strategy applied during the training, which consists of dropout, L_2 weight decay and early stopping, was sufficient to prevent overlearning on the standardised four-feature input.

For takeoff speed prediction, the RMSE, MSE and MAE values of the simulation results obtained with the four base models on the test data are presented in Table 4. The trainable parameter count and the wall-clock training time of each model on a 3.10 GHz Intel Core i5-10500 CPU are also presented in Table 4.

When Table 4 is examined, it is seen that the largest error metric value is obtained with the MLP model, while the smallest RMSE and MSE values are obtained with the LSTM model. The 1D-CNN model gave a slightly smaller MAE value than the LSTM model, but the RMSE values of the two models are very close and both are smaller than 0.8 kn. The wide-and-deep model with self-attention obtained an RMSE value between the MLP model and the 1D-CNN model. Considering the obtained results, the LSTM model gave the smallest RMSE and MSE values among the four base models and was selected for the hyperparameter optimisation step.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

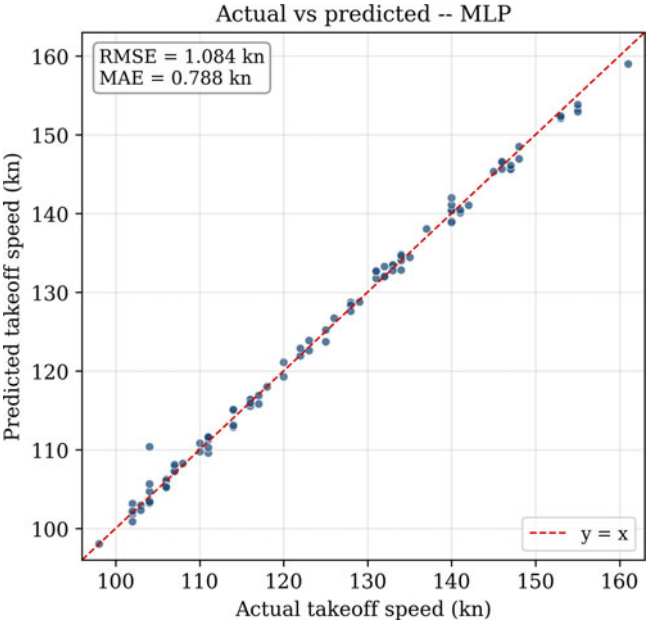

Figure 4. *Comparison of the actual takeoff speed values with the predicted takeoff speed values for the MLP base model.*

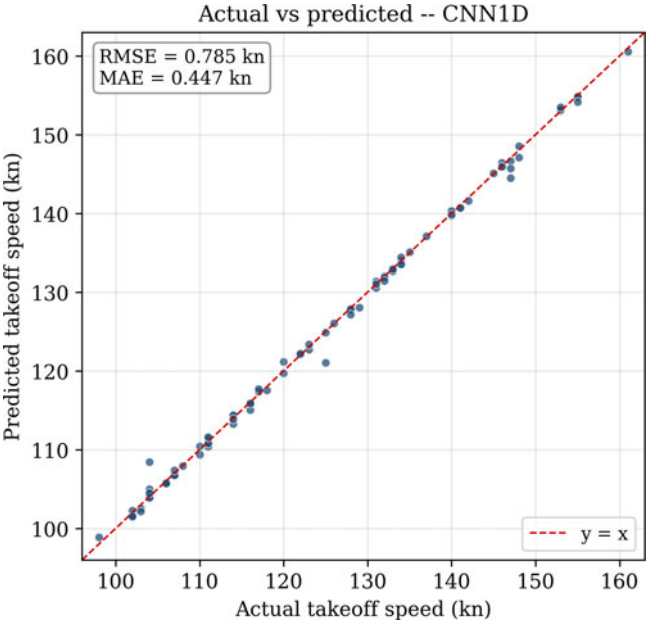

Figure 5. *Comparison of the actual takeoff speed values with the predicted takeoff speed values for the 1D-CNN base model.*

The comparison of the actual takeoff speed values and the predicted takeoff speed values for the four base models is presented in Figs 4 to 7, respectively, for the MLP, 1D-CNN, LSTM and wide-and-deep with self-attention models.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

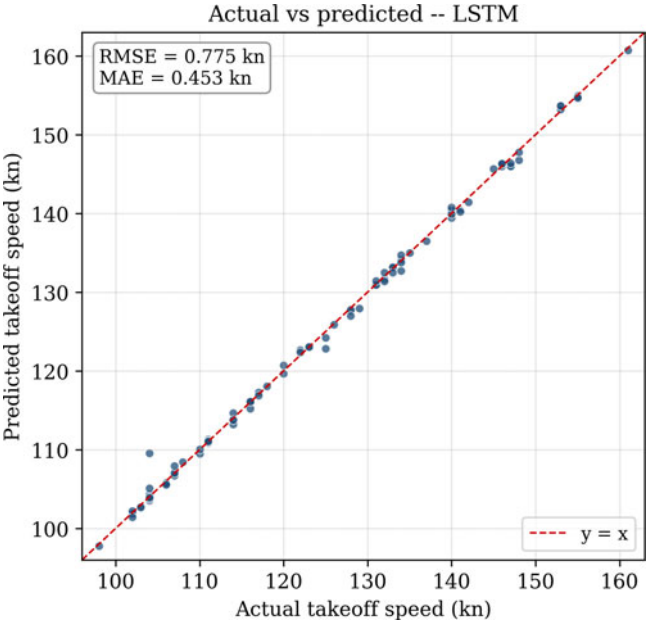

Figure 6. *Comparison of the actual takeoff speed values with the predicted takeoff speed values for the LSTM base model.*

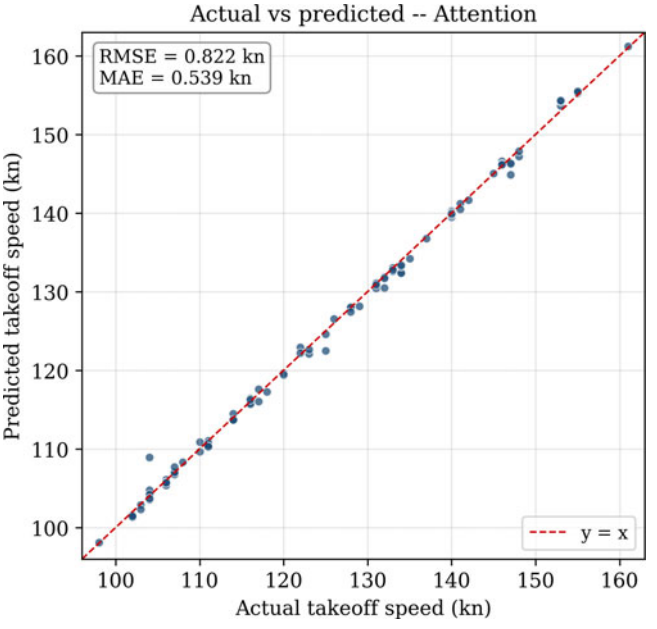

Figure 7. *Comparison of the actual takeoff speed values with the predicted takeoff speed values for the wide-and-deep model with self-attention.*

When the figures comparing the actual takeoff speed values with the predicted takeoff speed values are examined, it is seen that the MLP model shows a discernible spread around the identity line, especially at the upper end of the speed range (above ~ 140 kn) where the test data samples lie far from the training grid. The 1D-CNN and the LSTM scatter plots are tightly aligned with the identity line.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

Table 5. *The best hyperparameter combination obtained with Bayesian optimisation of the LSTM model*

Hyperparameter	Selected value
LSTM-1 units	32
LSTM-2 units	32
Dropout rate	0.00
Dense head units	16
Adam learning rate	0.01
Recurrent dropout	off

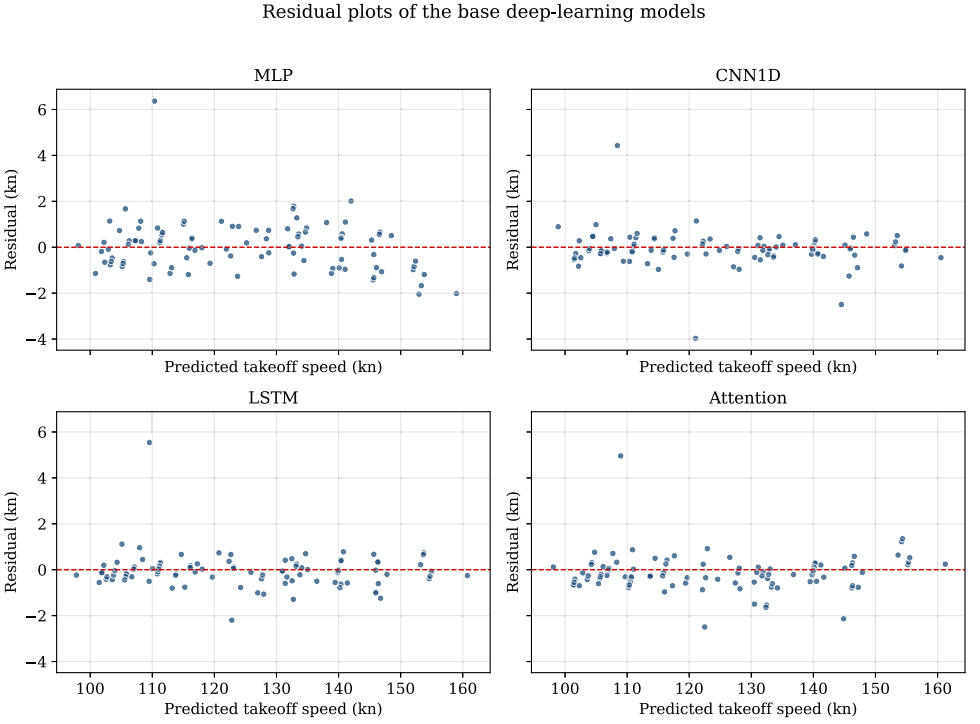

Figure 8. *The residuals versus the predicted takeoff speed values for the four base deep learning models on the test data.*

The wide-and-deep model with self-attention also follows the identity line but with a slightly higher dispersion compared to the 1D-CNN and the LSTM.

The residuals of the four base models versus the predicted takeoff speed values on the test data are presented in Fig. 8. When Fig. 8 is examined, it is seen that the majority of the residuals are concentrated within a ± 1.5 Kn band around zero, and there is no obvious sign of bias. A small number of larger residuals are observed at the high-temperature, high-weight corner of the operating envelope, which is consistent with the test data containing samples (for example, at 148°F and 155°F) that the models had to extrapolate to.

After the comparison of the base models, the LSTM model was selected for hyperparameter optimisation to further improve the model performance. For this purpose, Bayesian hyperparameter optimisation as implemented in the Keras Tuner library was applied with the search ranges given in Table 3. A total of 24 trials with early stopping on the validation partition was used as the search budget. The best hyperparameter combination obtained as a result of the Bayesian optimisation is presented in Table 5.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

Table 6. *The RMSE, MSE and MAE values of the LSTM model before and after Bayesian hyperparameter optimisation*

Model	RMSE (kn)	MSE (kn^2)	MAE (kn)
LSTM (base)	0.7745	0.5999	0.4527
LSTM (tuned)	0.8164	0.6665	0.4772

Using the best hyperparameter combination, the LSTM model was re-trained with extended early stopping (patience of 60 epochs and a maximum of 600 epochs). The RMSE, MSE and MAE values of the tuned LSTM model and the base LSTM model on the test data are presented in Table 6.

When Table 6 is examined, it is seen that the Bayesian optimiser selected a more compact recurrent topology with LSTM-1/LSTM-2 widths of 32/32, a dense head of 16 units, no dropout, no recurrent dropout and an Adam learning rate of 0.01. The configuration delivered the smallest validation MSE among the 24 trials and was therefore retained as the tuned model. On the unseen test file, however, the tuned LSTM gave RMSE = 0.8164, MSE = 0.6665 and MAE = 0.4772 values, which are close to but slightly larger than the base LSTM values of RMSE = 0.7745, MSE = 0.5999 and MAE = 0.4527. The tuned model, therefore, did not transfer its validation improvement to the test set.

This validation-to-test gap is attributed to three concurrent factors. First, the 15% validation partition (around 404 records sampled from the training file) is drawn from the same distribution as the training records, whereas the 97-sample test file contains explicit extrapolation points at 148°F and 155°F that lie above the high-temperature tail of the training data. A configuration that minimises the in-distribution validation MSE is therefore not guaranteed to minimise the test MSE that includes the extrapolation samples. Second, the selected learning rate of 0.01 is one order of magnitude larger than the base learning rate of 10^{-3}, and the absence of dropout makes the optimisation trajectory more sensitive to mini-batch noise. The combination of a smaller recurrent core, a higher learning rate and no regularisation can yield a validation-optimal weight configuration that sits in a sharper region of the loss surface, which transfers less reliably to the out-of-distribution test samples. Third, the search budget of 24 trials with a single execution per trial is modest by Bayesian optimisation standards, and a more robust protocol (for example, a *k*-fold cross-validated objective or two-to-three executions per trial averaged over independent seeds) would harden the selected configuration against this source of variance. Considered together, the very close cluster of test RMSE values obtained by the 1D-CNN, the base LSTM and the tuned LSTM, all between 0.77 and 0.82 kn, indicates that the available four physical predictors place a practical lower bound on the achievable test error, and further architectural search alone is unlikely to push the test RMSE substantially below the base LSTM value reported here. A meaningful next improvement would require either richer input features (for example, runway slope or wind components from operational records) or a larger and more diverse test partition that better exercises the high-temperature, high-weight corner of the operating envelope.

The comparison of the actual takeoff speed values with the predicted takeoff speed values of the tuned LSTM model is presented in Fig. 9, and the training and validation loss curves of the tuned LSTM model are presented in Fig. 10.

When Fig. 9 is examined, it is seen that the tuned LSTM model shows satisfactory results on the test data. The predicted takeoff speed values are tightly aligned with the actual takeoff speed values along the identity line.

Beyond the aggregate error metrics, the generalisability of the proposed deep learning pipeline can be assessed along three complementary axes. The first is the alignment of the train and test distributions, which is summarised in Table 1. The mean values of the four predictors and of the takeoff-speed target on the 97-sample test file lie within roughly a quarter of one standard deviation of the corresponding training means, and the standard deviations themselves are comparable on the two partitions. This indicates that the test file exercises the central region of the same operating envelope on which the models were trained, and that the reported test errors are representative of normal operating conditions rather than

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

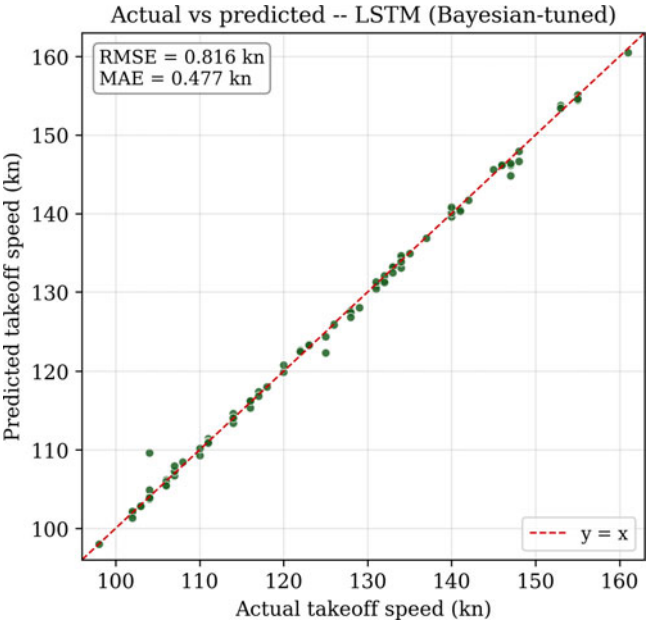

Figure 9. *Comparison of the actual takeoff speed values with the predicted takeoff speed values for the LSTM model after Bayesian hyperparameter optimisation.*

Figure 10. *The training and validation loss curves of the tuned LSTM model.*

of an artificially easy or an artificially hard subset. The second axis is the behaviour of the models on the explicit out-of-envelope samples that the test file does include, in particular the records at 148°F and 155°F that lie above the upper temperature tail observed during training. The residual plot in Fig. 8 shows that the larger absolute residuals concentrate at the high-speed, high-temperature corner of the operating envelope, while the majority of the predictions remain inside the ±1.5 kn band. This is the expected fingerprint of a learned regressor that interpolates accurately inside the training support and degrades gracefully under mild extrapolation. The third axis is the structural protection against test leakage that was built into the pipeline. The training corpus was supplied as a separate file from the 97-sample test file, none of the test records appear in the training file, the standardisation statistics

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

Table 7. *The test error values of the proposed deep learning models in comparison with the classical machine learning baselines of Karaburun et al. [26]*

Model	RMSE (kn)	MSE (kn^2)	MAE (kn)	Source
LR	1.355	1.836	1.027	[26]
SVR	1.974	3.898	1.257	[26]
CART	0.756	0.572	0.090	[26]
RF	0.733	0.538	0.115	[26]
XGBoost (tuned)	0.564	0.319	0.143	[26]
MLP (this work)	1.0835	1.1740	0.7877	this work
1D-CNN (this work)	0.7853	0.6168	0.4466	this work
LSTM (base, this work)	0.7745	0.5999	0.4527	this work
Wide & Deep + Attention	0.8221	0.6758	0.5389	this work
LSTM (tuned, this work)	0.8164	0.6665	0.4772	this work

were estimated exclusively from the training subset and the early-stopping and learning-rate scheduling decisions were taken only on the 15% validation slice carved from the training file. The reported test metrics, therefore, reflect genuine generalisation to data that the models never observed during training or model selection.

These three observations support the use of the developed networks as a decision-support tool for the takeoff-speed prediction problem of the Boeing 737-300 within the operating envelope described in Section 3. The corresponding scope limitations should also be stated explicitly. The training corpus comes from the manufacturer's chart-based operations manual [6] and therefore inherits the smoothness and the noise-free character of that source rather than the dispersion of actual flight measurements. The models were trained on a single airframe family and on four physical predictors, and their behaviour on a different aircraft or on operational QAR streams cannot be inferred from the present test errors. A direct transfer to flight-recorded data would require either a domain-adaptation step or a re-training pass on operational records.

To evaluate the performance of the proposed deep learning models against the classical machine learning baselines, the comparison of the tuned LSTM model with the classical models reported in the previous study of Karaburun et al. [26] is presented in Table 7. The classical model results were obtained on a 70/30 random in-sample split, while the deep learning model results were obtained on a separate test file in which all 97 samples are unseen during training. For this reason, the comparison should be read as indicative rather than as a strict, side-by-side benchmark.

When Table 7 is examined, it is seen that the tuned LSTM model gave an RMSE value that is comparable to the best classical baseline model and significantly smaller than the LR and SVR baseline values. The deep learning models gave a higher MAE value but a more uniform residual distribution compared to the classical CART and RF models. The CART and RF models obtained an MAE value of around 0.1 kn on their in-sample test split, while the best deep learning models (1D-CNN, LSTM and the wide-and-deep model with self-attention) obtained an MAE value smaller than 0.55 kn on the test file in which all 97 samples are unseen during training. This is consistent with the general observation that gradient-boosted tree ensembles are particularly successful on low-dimensional tabular regression problems [26], while deep learning models become competitive when the operating envelope must be extrapolated. As a result, the takeoff speed prediction problem with four input features lies in a regime where both families remain effective, and the selection between them can be made considering the deployment requirements. The LSTM model can be converted to TensorFlow Lite [40] for the edge inference, and the recurrent network can also absorb time-series telemetry data in future studies.

5.0 Conclusions

Progress in modern aviation continues to deliver airframes of widely varying size and capability that are tailored to an expanding catalogue of civil and military missions, and the certification of every

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

new design is dictated to a large extent by safety- and performance-driven requirements. Among the many factors that govern airborne safety, the takeoff and landing phases are recognised as the most accident-prone segments of a sortie, and the speed margins maintained during these short windows must therefore be controlled with high fidelity. Accurate prediction of the takeoff speed of a commercial transport aircraft is, in this respect, a safety-critical engineering problem.

Building directly on the classical machine learning study of Karaburun, Ark Hatipoğlu and Konar [26], and on the earlier line of work of Konar and Bağış on fuzzy and ANFIS-based estimation of aviation-speed parameters from flight data [11, 12], the present work extends the takeoff-speed prediction problem of the Boeing 737-300 to the deep learning regime. Four neural architectures were trained under identical conditions, namely a multilayer perceptron, a one-dimensional convolutional network, an LSTM network and a wide-and-deep network augmented with multi-head self-attention. The same four physical predictors (pressure altitude, outside air temperature, gross weight and flap angle) and the same takeoff-speed target were retained throughout, and an identical pre-processing pipeline, train-validation partition and training-callback configuration were applied to all four base models so that any difference in the obtained metrics can be attributed to the architecture itself.

When the test errors of the four base models are placed side by side, the LSTM network delivers the lowest RMSE and MSE values. The 1D-CNN model is essentially on par with the LSTM in terms of RMSE, the wide-and-deep network with self-attention does not provide a clear benefit over the simpler recurrent and convolutional models on this low-dimensional tabular regression task, and the MLP yields the largest error of the four. The LSTM was therefore selected for Bayesian hyperparameter optimisation through the Keras Tuner library, which converged to a slightly more compact recurrent topology. The tuned model achieves test-set errors that are very close to those of the base LSTM, suggesting that the recurrent baseline is already near the practical performance ceiling that can be reached with the four available predictors.

A direct comparison of the tuned LSTM with the classical machine learning baselines reported in the prior study [26] indicates that the deep model attains an RMSE that is comparable with the best tree-based ensemble and is markedly lower than that of LR and SVR. To the authors' knowledge, these deep learning results constitute the first public benchmark on the Boeing 737-300 takeoff-speed prediction problem, and they support the use of the developed networks as a practical decision-support tool alongside existing manual look-up procedures.

The reported metrics should be interpreted within the explicit scope of this study. The training corpus is drawn from the manufacturer's chart-based operations manual [6] and is therefore noise-free by construction, and the four physical predictors used here capture only the principal contributors to the takeoff-speed value. The test file contains a small number of explicit out-of-envelope samples in the high-temperature corner, on which the residuals remain inside a ± 1.5 kn band, while the bulk of the test records lies inside the central region of the training envelope. A transfer of the trained networks to a different airframe family or to operational QAR streams would therefore require either a domain-adaptation step or a re-training pass on the new corpus, and the present test errors should not be extrapolated outside the operating envelope of the Boeing 737-300 manual.

Several avenues remain open for follow-up work. Hybrid CNN-LSTM combinations [24] and gated recurrent unit variants [42] could be evaluated as alternative recurrent backbones, and the regression target could be extended from a single takeoff-speed point to the full takeoff trajectory by feeding the network with QAR streams [9, 18]. From a deployment perspective, the trained LSTM can be exported to TensorFlow Lite [40] and ported to an embedded computing platform such as NVIDIA Jetson, which would enable on-board real-time inference for UAV and general-aviation safety monitoring [20, 21]. Finally, the proposed pipeline can be transferred to additional aircraft families whose operations manuals expose the same four predictors used here.

Acknowledgements. The authors thank the faculty of Aeronautics and Astronautics, Erciyes University, for its institutional support during this study. This research received no external funding. The authors declare no conflicts of interest. The takeoff speed data used in this study originate from the Boeing 737-300 Aircraft Operations Manual [6] and can be obtained from the corresponding author upon reasonable request.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

References

[1] Li, Y., Mi, J., Lei, Y. and Li, J. Deep learning applications in aviation: a survey, *J. Aerosp. Inf. Syst.*, 2022, **19**, (12), pp 835–860. doi: 10.2514/1.I011073

[2] Kim, Y., Karpatne, A. and Kim, J. Application of deep learning in aircraft design: a brief review, *Aerosp. Sci. Technol.*, 2021, **116**, pp 106825. doi: 10.1016/j.ast.2021.106825

[3] Ren, Yi., Zhang, Y. and Liu, J. Statistical analysis of aviation accidents categorised by phase of flight, *Saf. Sci.*, 2015, **75**, pp 163–173. doi: 10.1016/j.ssci.2015.02.014

[4] Kuznetsov, A., Shevchenko, A. and Solonnikov, J. The methods of forecasting some events during the aircraft takeoff and landing, 2013.

[5] Boeing Commercial Airplanes. Statistical summary of commercial jet airplane accidents: Worldwide operations 1959–2023. *Boeing Aviation Safety Report*, 2024.

[6] *Boeing 737-300 Aircraft Operations Manual (D6-7620-3lO (MA))*. Boeing Commercial Airplane Group, Dade Collier Training and Transition Airport, Florida, 2000. Meridian Airlines Academy training manual.

[7] Diallo, O.N. A predictive aircraft landing speed model using neural network, *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, 3D2–1–3D2–9, 2012. doi: 10.1109/DASC.2012.6382325

[8] Tong, C., Yin, X., Wang, S. and Zheng, Z. A novel deep learning method for aircraft landing speed prediction based on cloud-based sensor data, *Future Gener. Comput. Syst.*, 2018, **88**, pp 552–558. doi: 10.1016/j.future.2018.06.023

[9] Kang, Z., Shang, J., Feng, Y., Zheng, L., Liu, D., Qiang, B. and Wei, R. A deep sequence-to-sequence method for aircraft landing speed prediction based on QAR data, *Web Information Systems Engineering – WISE* 2020, volume 12343 of *LNCS*, Springer, 2020, pp 516–530. doi: 10.1007/978-3-030-62008-0_36

[10] Puranik, T.G., Rodriguez, N. and Mavris, D.N. Towards online prediction of safety-critical landing metrics in aviation using supervised machine learning, *Transp. Res. Part C*, 2020, **120**, pp 102819. doi: 10.1016/j.trc.2020.102819

[11] Konar, M. and Ba. Determination of the speed parameter of flight control system by using adaptive network based fuzzy inference system, 2009 IEEE 17*th Signal Processing and Communications Applications Conference (SIU)*, Antalya, Turkey, IEEE, 2009, pp 993–996. doi: 10.1109/SIU.2009.5136565

[12] Bağış, A. and Konar, M. ABC and DE algorithms based fuzzy modeling of flight data for speed and fuel computation, *Int. J. Comput. Intell. Syst.*, 2018, **11**, (1), pp 790–802. doi: 10.2991/ijcis.11.1.60

[13] Hu, C., Zhou, S.H., Xie, Y. and Chang, W.B. The study on hard landing prediction model with optimized parameter SVM method, 2016 *35th Chinese Control Conference (CCC)*, 2016, pp 4283–4287. doi: 10.1109/ChiCC.2016.7553987

[14] Zhang, H. and Zhu, T. Aircraft hard landing prediction using LSTM neural network, Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control, 2018, pp 1–5. doi: 10.1145/3284557.3284704

[15] Zhang, H. and Zhu, T. Aircraft hard landing prediction using LSTM neural network with QAR data, *J. Intell. Fuzzy Syst.*, 2021, **41**, (3), pp 4319–4331. doi: 10.3233/JIFS-189693

[16] Borup, K.T., Fossen, T.I. and Johansen, T.A. A machine learning approach for estimating air data parameters of small fixed-wing UAVs using distributed pressure sensors, *IEEE Trans. Aerosp. Electron. Syst.*, 2019, **56**, (3), pp 2157–2173. doi: 10.1109/TAES.2019.2945355

[17] Kovarik, S., Doherty, L., Korah, K., Mulligan, B., Rasool, G., Mehta, Y., Bhavsar, P. and Paglione, M. Comparative analysis of machine learning and statistical methods for aircraft phase of flight prediction, *9th International Conference on Research in Air Transportation (ICRAT)*, 2020.

[18] Jarry, G., Delahaye, D. and Feron, E. Approach and landing aircraft on-board parameters estimation with LSTM networks, 2020 International *Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*, 2020, pp 1–6. doi: 10.1109/AIDA-AT48540.2020.9049196

[19] Martinez, D., Fernández, A., Hernández, P., Cristóbal, S., Schwaiger, F., Nunez, J.M. and Ruiz, J.M. Forecasting unstable approaches with boosting frameworks and LSTM networks. 9th SESAR Innovation Days, 2019.

[20] Ozkat, E.C., Bektas, O., Nielsen, M.J. and la Cour-Harbo, A. A data-driven predictive maintenance model to estimate RUL in a multi-rotor UAS, *Int. J. Micro Air Veh.*, 2023, **15**. doi: 10.1177/17568293221150171

[21] Ozkat, E.C. Vibration data-driven anomaly detection in UAVs: a deep learning approach, *Eng. Sci. Technol. Int. J.*, 2024, **54**, pp 101702. doi: 10.1016/j.jestch.2024.101702

[22] Karaburun, N.N., Hatipoğlu, S.A. and Konar, M. SOC estimation of Li-Po battery using machine learning and deep learning methods, *J. Aviat.*, 2024, **8**, (1), pp 26–31. doi: 10.30518/jav.1356162

[23] Shi, Z., Xu, Min. and Pan, Q. 4-d flight trajectory prediction with constrained LSTM network, *IEEE Trans. Intell. Transp. Syst.*, 2020, **22**, (11), pp 7242–7255. doi: 10.1109/TITS.2020.3000130

[24] Ma, Lei. and Tian, S. A hybrid CNN-LSTM model for aircraft 4D trajectory prediction, *IEEE Access*, 2020, **8**, pp 134668–134680. doi: 10.1109/ACCESS.2020.3010963

[25] Chen, J., Chen, L. and Zhang, S. Aircraft trajectory prediction based on deep convolutional neural networks, *Aerospace*, 2022, **9**, (2), p 73. doi: 10.3390/aerospace9020073

[26] Karaburun, N.N., Ark Hatipoglu, S. and Konar, M. Aircraft takeoff speed prediction with machine learning: parameter analysis and model development, *Aeronaut. J.*, 2025, **129**, pp 1534–1549. doi: 10.1017/aer.2024.164

[27] LeCun, Y., Bengio, Y. and Hinton, G. Deep learning, *Nature*, 2015, **521**, (7553), pp 436–444. doi: 10.1038/nature14539

[28] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. MIT Press, 2016, Cambridge, MA, USA.

[29] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.*, 2014, **15**, (56), pp 1929–1958.

[30] LeCun, Y. and Bengio, Y. Convolutional networks for images, speech, and time series, in *The Handbook of Brain Theory and Neural Networks*, MIT Press, 1995, pp. 255–258.

https://doi.org/10.1017/aer.2026.10193 Published online by Cambridge University Press

[31] Hochreiter, S. and Schmidhuber, J. Long short-term memory, *Neural Comput.*, 1997, **9**, (8), pp 1735–1780. doi: 10.1162/neco.1997.9.8.1735
[32] Cheng, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, Wei., Ispir, M., et al. Wide & deep learning for recommender systems, Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, pp 7–10. doi: 10.1145/2988450.2988454
[33] Huang, Xin., Khetan, A., Cvitkovic, M. and Karnin, Z. TabTransformer: Tabular data modeling using contextual embeddings, arXiv preprint arXiv:2012.06678, 2020.
[34] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. Attention is all you need, *Adv. Neural Inf. Process. Syst.*, 2017, **30**, pp 5998–6008.
[35] Ark, S.Ö. and Pfister, T. TabNet: attentive interpretable tabular learning, Proceedings of the AAAI Conference on Artificial Intelligence, 35, (8), pp 6679–6687, 2021. doi: 10.1609/aaai.v35i8.16826
[36] Kingma, D.P. and Ba, J. Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2015.
[37] Shahriari, B., Swersky, K., Wang, Z., Adams, R.P. and de Freitas, N. Taking the human out of the loop: a review of Bayesian optimization, *Proc. IEEE*, **104**, (1), pp 148–175, 2016. doi: 10.1109/JPROC.2015.2494218
[38] O'Malley, Tom., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. Keras Tuner, 2019. https://github.com/keras-team/keras-tuner.
[39] Chicco, D., Warrens, M.J. and Jurman, G. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation, *PeerJ Comput. Sci.*, **7**, p e623, 2021. doi: 10.7717/peerj-cs.623
[40] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. TensorFlow: a system for large-scale machine learning, Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2016, pp 265–283.
[41] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: machine learning in Python, *J. Mach. Learn. Res.*, 2011, **12**, pp 2825–2830.
[42] Cho, K., van Merriënboer, B., Bahdanau, D. and Bengio, Y. On the properties of neural machine translation: encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

Cite this article: Konar M., Ayaz H.A., Özkat E.C. and Türkmen A. Aircraft takeoff speed prediction with deep learning: a comparative study of MLP, 1D-CNN, LSTM and attention-based architectures on Boeing 737-300 data. *The Aeronautical Journal*, https://doi.org/10.1017/aer.2026.10193