Bayesian Recurrent Neural Networks for Monitoring Rotorcraft Icing from Aeroacoustics Time-Series Data

Hua Tong*
University of Science and Technology of China, Dynamics F1, 443 Huangshan Rd, Shushan District, Hefei, Anhui, China

Jeremiah M. A. Hauth† and Xun Huan‡
University of Michigan, 1231 Beal Ave, Ann Arbor, MI 48109-2133, USA

Beckett Y. Zhou§
University of Bristol, Queens Building, University Walk, Bristol, BS8 1TR, United Kingdom

Nicolas R. Gauger¶
TU Kaiserslautern, Bldg 34, Paul-Ehrlich-Strasse, 67663 Kaiserslautern, Germany

Myles Morelli‖ and Alberto Guardone**
Politecnico di Milano, Building B12 Campus Bovisa, Via La Masa, 34 20156 Milano, Italy

Motivated by the need for reliable on-board and real-time rotor blade icing detection with quantified uncertainty, we demonstrate the use of Bayesian recurrent neural networks, particularly the long short-term memory (LSTM), to predict time-series aerodynamic performance metrics from measurements of aeroacoustic time-series. The neural network models are trained using simulation data generated from physics-based models. To help improve robustness against measurement error in real life, we augment the data targets with artificial noise corruption. The training is done by minimizing the negative evidence lower bound (ELBO), taking a mean-field variational inference approach that finds independent Gaussian distribution most closely approximates the true Bayesian posterior. Promising performance is observed in the test set, where Bayesian LSTM achieved lower prediction errors compared to a Monte Carlo dropout version of the LSTM model.

I. Introduction

Rotorcraft are uniquely designed for vertical takeoff and landing, allowing them to navigate challenging flight conditions that may be unsuitable or unachievable by fixed-wing aircraft. Rotorcraft missions thus often involve low altitudes and are at the edge of the flight envelope. Further challenges emerge when these missions are conducted in extreme cold weather, such as during search-and-rescue operations. These cold environments bring the risk of highly dangerous icing, which can drastically impair the rotorcraft's flight performance [1, 2]. Unlike fixed-wing aircraft, rotorcraft cannot easily escape icing conditions by climbing to higher altitudes. Indeed, icing events have led to many recent rotorcraft accidents and with fatalities [3–7].

Implementation of ice protection systems is difficult for rotorcraft. Electrical resistance heating is both expensive and inefficient and many civil rotorcraft manufacturers forgo such deicing systems. As only about 5% of US rotorcraft are equipped with deicing systems, problems arise when rotorcraft without deicing systems encounter, often unintentionally, ice-forming conditions [8]. This necessitates an early warning system that can alert pilots of potentially dangerous ice accretion.

*Undergraduate Student, Theoretical and Applied Mechanics, gt2k01@mail.ustc.edu.cn
†PhD Candidate, Department of Mechanical Engineering, Student Member AIAA, hauthj@umich.edu
‡Assistant Professor, Department of Mechanical Engineering, Member AIAA, xhuan@umich.edu
§Lecturer in Aeroacoustics, Department of Aerospace Engineering, Member AIAA, beckett.zhou@bristol.ac.uk
¶Professor, Chair for Scientific Computing, Associate Fellow AIAA, nicolas.gauger@scicomp.uni-kl.de
‖PhD Candidate, Department of Aerospace Sciences and Technologies, mylescarlo.morelli@polimi.it
**Professor, Department of Aerospace Sciences and Technologies, alberto.guardone@polimi.it

A potential icing detection solution is by monitoring and analyzing the acoustic signature of the rotor blades. A severely iced blade emits a drastically different broadband noise component than a clean one. In fact, listening and feeling for this acoustic change is an effective skill that experienced pilots have learned to detect particularly severe ice formation [9, 10]. While both experiment and modeling efforts have progressed to connect aeroacoustics with icing [11–13], its utilization for on-board real-time ice detection remains impractical if physics-based aerodynamic and aeroacoustic simulations are needed. For example, an unsteady Reynolds-Averaged Navier Stokes (URANS) flow solve, even under a simplified two-dimensional representation of a pitching rotor blade, would take hours to complete despite being parallelized with tens of processors. This would typically not be possible under existing in-flight computation capabilities. The computational cost would only increase further when realistic three-dimensional rotor blades are simulated with scale-resolving methods.

We seek computational accelerations through machine learning (ML) techniques, to leverage their speed advantages to narrow the gap for real-time requirements. In particular, our approach trains a ML model offline using datasets of physics-based simulations. Once trained, the ML model can be deployed online to directly predict quantities of interest (QoIs), such as aerodynamic performance indicators, from an acoustic signal measured in-flight. In this paper, we investigate deep neural network (DNN) class of ML models, which are powerful function learners [14] and can handle large and high-dimensional datasets. While our previous efforts [15–17] investigated densely-connected DNN to predict mean, minimum, and maximum lift and moment coefficients over time from acoustic spectrum transformed in the frequency space, those models are unable to accommodate time-series quantities. In this paper, we will experiment with recurrent neural network (RNN) architectures that are particularly suited for analyzing time-dependent behavior.

Another important aspect of building and adopting DNN models, especially for mission- and safety-critical settings, is their uncertainty information. Traditional training of a DNN minimizes a loss function in order to find the best fitting DNN weight parameters to training data. Subsequently, when given an input, the DNN produces a single-value output prediction, without any uncertainty information to indicate the quality and credibility of that prediction. One approach to capture the parametric uncertainty for DNNs is to perform Bayesian inference on their parameters, and obtain a distribution of plausible values conditioned on the training data. As a result, such a DNN provides a distribution of predictions reflecting its uncertainty, and it is also known as a Bayesian neural network (BNN)[18–22]. We will combine the Bayesian training with RNN architecture in this work.

This paper is organized as follows. Section II briefly describes the governing physics as well as numerical solvers to simulate the ice accretion, fluid flow, and aeroacoustics for a rotor blade airfoil that together form the training data for our ML models. Section III presents formulation and computational techniques for constructing the Bayesian RNNs. Results for predictions using these ML models are presented in Sec. IV. The paper ends with conclusion remarks in Sec. V.

II. Physics-Based Simulation Data

We first build a dataset for training, validating, and testing of the ML models. The dataset entails solutions of physics-based aerodynamic, icing, and aeroacoustic simulations. These data-generating models are briefly introduced in this section.

A. Turbulent Flow Simulation

The aerodynamic flow solves are conducted using the SU2 software suite [23]. SU2 is a collection of open-source partial differential equation (PDE) analysis tools written in C++ and Python created for multi-physics design and simulation. It is built specifically for PDE constrained optimization using state-of-the-art algorithms and is particularly well suited for aerodynamic shape design. Finite volume method (FVM) is applied on arbitrary unstructured meshes using a standard edge-based data structure on a dual grid with control volumes constructed using a median-dual, vertex-based scheme. The core of the suite is a Reynolds-averaged Navier-Stokes (RANS) solver which is used in this study in conjunction with the Menter shear-stress transport (SST) turbulence model. SU2 uses a dual-time stepping strategy to solve implicitly both steady and unsteady problems with second order accuracy in both space and time.

B. Ice Particle Accretion

In this study, we consider the unsteady flow over a pitched airfoil. Each time step, the SU2 solver is seeded with super-cooled water droplets and their trajectories are computed. These particles are tracked using an in-house code called PoliDrop. PoliDrop is a Lagrangian based particle tracking solver designed primarily for icing simulation. This

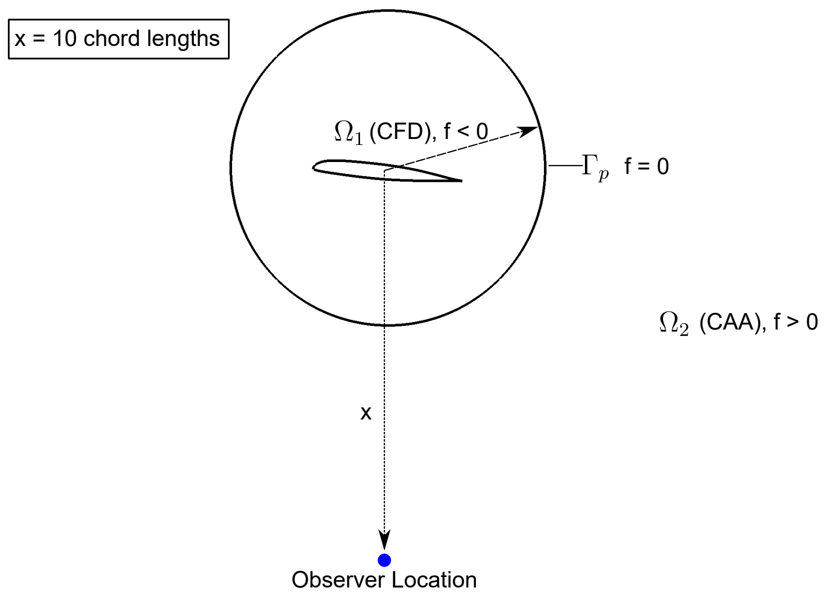


Fig. 1 Schematic of the permeable control surface Γ_p separating the CFD and CAA domains and the relative observer location.

approach was introduced in a previous study [24] and fully accounts for unsteady aerodyamic effects which might be present in the flow field of the oscillating airfoil. This is particularly important for mixed glaze-rime icing conditions. The super-cooled water droplet impingement locations and collection efficiency is then passed onto our in-house icing solver, PolyMIce. The PoliMIce software suite provides state-of-the-art ice formation models. The model used in this work uses the local exact solution of the unsteady Stefan problem for the temperature profiles within the ice layer in glaze conditions. Ice shapes are computed using an unsteady multi-step approach, in which non-linear ice accretion is accounted for by iteratively updating the surface solution on which the ice accretes.

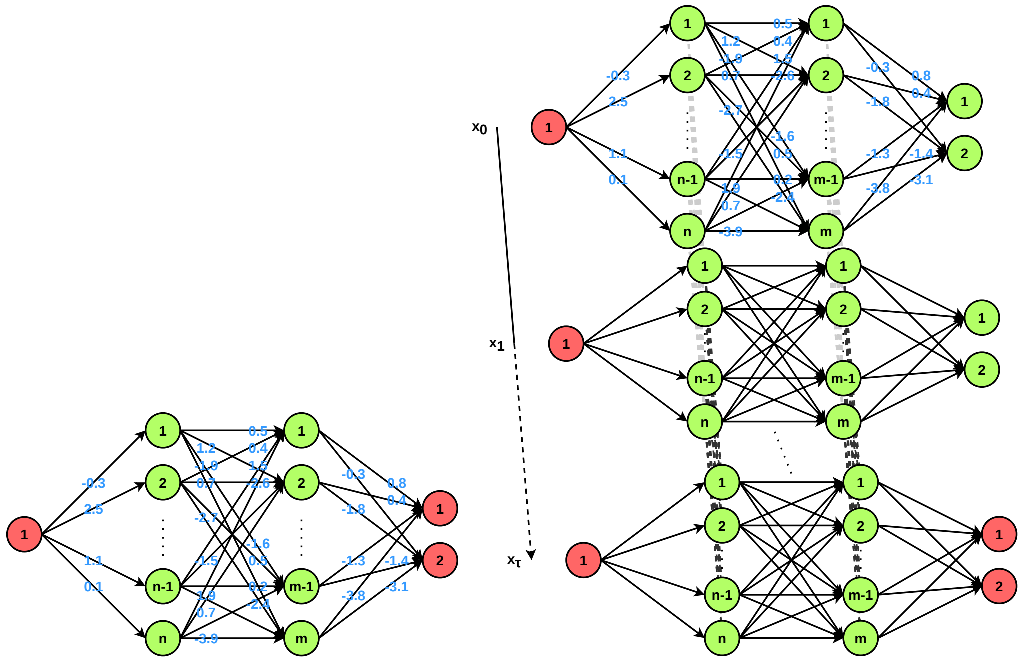
C. Noise Prediction in SU2

We again use SU2 to perform computational aeroacoustic (CAA) analysis on various accreted ice shapes using the permeable-surface Ffowcs-Williams-Hawkings (FWH) formulation [13]. This analysis can help predict early stages of ice accretion during flight and can in turn be used help inform the pilot of this accretion. The permeable FWH formulation allows fluid to flow through the fictitious surface Γ_p. Flow field information is then extracted and the noise is propagated to the far-field. This implementation is shown in the schematic from Figure 1. Therefore there are two separate computational domains in this problem: the near-field computational fluid dynamics (CFD) region Ω_1, and the far-field CAA region Ω_2. The permeable surface Γ_p can be described by the shape function $f = 0$, where $f < 0$ indicates the region inside the surface and $f > 0$ the region outside the surface. The FWH portion lies $\frac{3}{4}$ chord lengths from the airfoil trailing edge. The position of the observer locations is chosen based on where noise would likely be perceived on a conventional main rotor/tailrotor helicopter, which is directly below the main rotor. In accordance with the icing simulation, the acoustic analysis is likewise considered as a two-dimensional model, although it is understood the rotor blade noise is most definitely not two-dimensional in nature. This analysis, particularly with regard to the ML methodology, is therefor a proof-of-concept for the detection of rotor blade performance changes under different ice accretion geometries. In parallel, we have continued to develop and demonstrate three-dimensional physics implementations of this idea in [17].

III. Machine Learning Methodology

A. Recurrent Neural Networks and the Long-Short Term Memory

A DNN maps input x to output $\hat{y}$: we write $\hat{y} = f(x)$ where the hat in $\hat{y}$ denotes DNN *prediction*. For example, Fig. 2a shows a graphical representation of a densely connected DNN with an input layer, two hidden layers, and an



(a) Deep neural network (b) Recurrent neural network

Fig. 2 DNN and RNN, each with an input layer, two hidden layers, and an output layer. Weights are shown in blue, intermediate variables are shown in green, and input output variables are shown in red.

output layer, and its mathematical form is

$$\begin{aligned} h_0 &= x \\ h_1 &= \phi_1 \left(W_{01} h_0 + b_1 \right) \\ h_2 &= \phi_2 \left(W_{12} h_1 + b_2 \right) \\ h_3 &= W_{23} h_2 + b_3 = \hat{y}. \end{aligned} \tag{1}$$

where ϕ_i is a nonlinear activation function, h_{i-1} is the output for layer $(i-1)$, $W_{(i-1)i}$ is the weight matrix between layer $i-1$ and i, b_i is a biasing vector. Hyperparameters, such as the number of layers, number of neurons, and choice of activation functions, are usually selected via model selection techniques such as validation coupled with grid search [25], random search [26], or more sophisticated implementations such as Hyperopt [27]. Once the network architecture is decided, the collection of all weights and all bias terms (collectively denoted by w) constitutes the free parameters to be tuned. Accordingly, the notation is updated to $\hat{y} = f(x; w)$.

A densely connected DNN typically does not account for time-dependence of data. In contrast, a RNN is designed to accommodate time-series data (Fig. 2b) [28]. RNN propagation formulas are similar to that in DNN, except that the information from the previous time steps is also used:

$$\begin{aligned} h_0^t &= x^t \\ h_1^t &= \phi_1 \left(W_{01} h_0^t + W_{11} h_1^{t-1} + b_1 \right) \\ h_2^t &= \phi_2 \left(W_{12} h_1^t + W_{22} h_2^{t-1} + b_2 \right) \\ h_3^t &= W_{23} h_2^t + b_3 = \hat{y}^t. \end{aligned} \tag{2}$$

However, traditional RNN may suffer from gradient vanishing and explosion problems [29], caused by the repeated unrolling of terms in the back-propagation, leading to convergence difficulties. An improved network architecture, the Long Short-Term Memory (LSTM), adds a cell chain to memorize long-term information [30]. Fig. 3 provides a

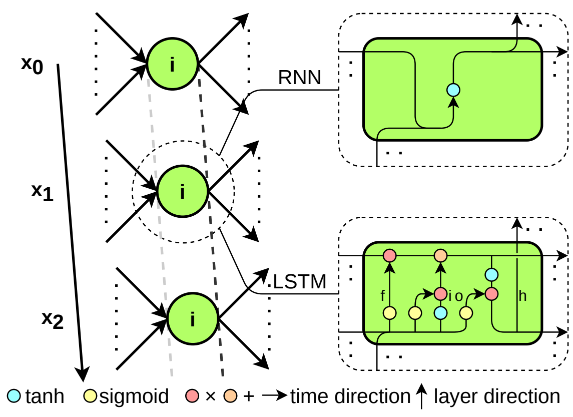


Fig. 3 Comparison of the simple RNN and LSTM nodal structure.

comparison of the simple RNN with LSTM. The LSTM shcematic contains a forget gate and an input gate:

$$f_i^t = \sigma\left(W_{f,i} \cdot \left[h_i^{t-1}, h_{i-1}^t\right] + b_{f,i}\right) \tag{3}$$

$$i_i^t = \sigma\left(W_{i,i} \cdot \left[h_i^{t-1}, h_{i-1}^t\right] + b_{i,i}\right), \tag{4}$$

where σ is the sigmoid activation function. The cell state C_i^t at the top orange circle is:

$$C_i^t = f_i^t \odot C_i^{t-1} + i_i^t \odot \tilde{C}_i^t \tag{5}$$

where $\odot$ denotes element-wise multiplication and the cell state $\tilde{C}_i^t$ at the bottom blue circle:

$$\tilde{C}_i^t = \tanh\left(W_{C,i} \cdot \left[h_i^{t-1}, h_{i-1}^t\right] + b_{C,i}\right), \tag{6}$$

Lastly, the output gate o_i^t:

$$o_i^t = \sigma\left(W_{o,i} \cdot \left[h_i^{t-1}, h_{i-1}^t\right] + b_{o,i}\right) \tag{7}$$

is incorporated to produce the final updated state h_i^t:

$$h_i^t = o_i^t \odot \tanh\left(C_i^t\right). \tag{8}$$

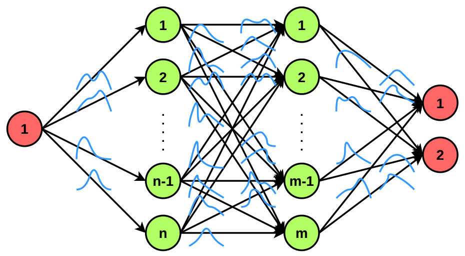
B. Bayesian Neural Networks

Classical RNN training chooses w in order to minimize a loss function, typically the mean square error (MSE), evaluated on a training set (x_T, y_T):

$$w^* = \underset{w}{\operatorname{argmin}}\, \mathcal{L}\left[f(x_T; w), y_T\right]. \tag{9}$$

Various gradient-based algorithms are often employed for this optimization problem, such as stochastic gradient descent [31, 32] and with momentum [33]. However, regardless of how Eqn. (9) is solved, its solution is a single-valued w^* and does not provide the uncertainty surrounding this solution. In contrast, a BNN treats w as random variables, where their associated probability density functions (PDFs) represent the uncertainty on w (Fig. 4). Given training data, these PDFs are updated through Bayes' rule:

$$p(w|x_T, y_T) = \frac{p(y_T|x_T, w)p(w|x_T)}{p(y_T|x_T)}, \tag{10}$$


(a) Bayesian neural network

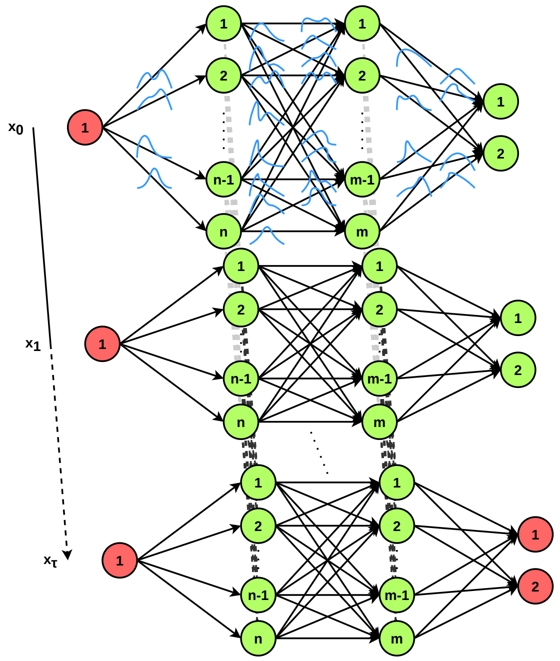

(b) Bayesian recurrent neural network

Fig. 4 BNN versions of a DNN and RNN, each with an input layer, two hidden layers, and an output layer. Probability distributions for the weights are illustrated in blue curves.

where $p(w|x_T)$ is the prior PDF on the weight parameters and can be reasonably simplified to $p(w)$, $p(y_T|x_T, w)$ is the likelihood, $p(w|x_T, y_T)$ is the posterior, and $p(y_T|x_T)$ is the model evidence. Solving the Bayesian inference problem entails computing the posterior $p(w|x_T, y_T)$, that is, the updated uncertainty on w given the training dataset (x_T, y_T).

Markov chain Monte Carlo (MCMC) [34, 35], the classical method to solve a Bayesian inference problem, becomes impractical for typical DNNs that involve thousands, even millions of weight parameters. Variational inference (VI) is an alternative to MCMC that seeks to approximate the posterior instead, thereby turning the sampling task into an optimization one, which is generally computationally less intensive. The aim of VI is to provide a locally-optimal, analytical form of a posterior approximation from within a family of distributions. Denoting the variational distribution by $q(w;\theta)$ parameterized by θ, we then want to find the best parameter θ^* such that $q(w;\theta)$ is closest to the true posterior $p(w|x_T, y_T)$. The Kullback-Leibler (KL) divergence is used to quantify the degree of closesness between the two distributions, leading to the following optimization statement:

$$\begin{aligned}
\theta^* &= \underset{\theta}{\operatorname{argmin}}\, D_{\mathrm{KL}}\left[q(w;\theta) \parallel p(w|x_T, y_T)\right] && (11)\\
&= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{q(w;\theta)}\left[\log q(w;\theta) - \log p(w|x_T, y_T)\right] && (12)\\
&= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{q(w;\theta)}\left[\log q(w;\theta) - \log p(y_T|x_T, w) - \log p(w)\right], && (13)
\end{aligned}$$

where in the last equality the $\log p(y_T|x_T)$ term is independent of w and thus can be dropped without changing the optimization location. The last expression is in fact the negative evidence lower bound (ELBO). The ELBO can be further estimated numerically using Monte Carlo sampling:

$$\theta^* \approx \underset{\theta}{\operatorname{argmin}} \left\{ \mathbb{E}_{q(w;\theta)}\left[\log q(w;\theta)\right] + \frac{1}{N}\sum_{n=1}^{N}\left[-\log p\left(y_T|x_T, w^{(n)}\right) - \log p\left(w^{(n)}\right)\right] \right\} \quad (14)$$

where $w^{(n)}$ are samples drawn from $q(w;\theta)$, and the first term (negative entropy of q) often can be computed analytically without Monte Carlo. Note that while the second and third terms appear to not contain θ, they indeed depend on θ since θ dictates the sampling distribution for $w^{(n)}$.

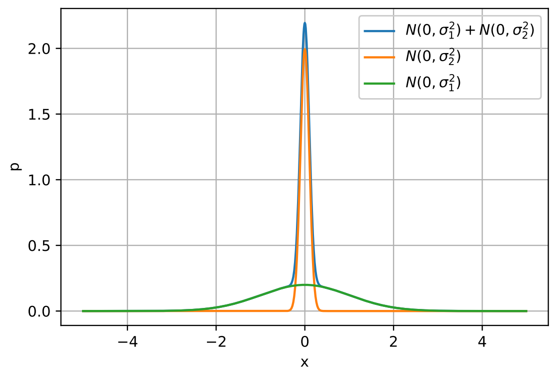


Fig. 5 Gaussian mixture prior distribution (blue line).

In the work, mean-field approximation is adopted for scalability, where $q(w;\theta)$ is assumed to be factorizable into independent components:

$$q(w;\theta) = \prod_{k=1}^{K} q(w_k;\theta_k) \tag{15}$$

with K being the total number of weight parameters in the DNN. Furthermore, univariate Gaussian is utilized for each component $q(w_k;\theta) = \mathcal{N}(w_k;\mu_k,\sigma_k^2)$, and hence $\theta_k = \{\mu_k,\sigma_k\}$ controls the variational distribution for w_k. Following [21], a prior distribution is selected also of an independent form, where each dimension involves (identically) a mixture of two preset Gaussian distributions (Fig. 5):

$$p(w) = \prod_{k=1}^{K} \left[\pi\mathcal{N}(w_k;0,\sigma_1^2) + (1-\pi)\mathcal{N}(w_k;0,\sigma_2^2)\right]. \tag{16}$$

with $\pi \in [0,1]$ and $\sigma_1 > \sigma_2 > 0$. A Gaussian mixture helps retain weight values near zero, thus favoring a starting model that is closer to linear. Lastly, our likelihood is chosen to reflect an additive Gaussian noise for the data labels $y = f(x;w) + \epsilon$ where $\epsilon \sim \mathcal{N}(0,\sigma_\epsilon^2)$. Hence,

$$p(y_T|x_T,w) = \prod_{i=1}^{n_T} \mathcal{N}\left(y_i; f(x_i;w), \sigma_\epsilon^2\right). \tag{17}$$

In this work, we set $\sigma_\epsilon = 1$ as an illustration, although one should select σ_ϵ to best reflect the anticipated magnitude of measurement noise in the data labels at hand.

C. Preprocessing, Training, Validation, and Testing

We implement a Bayesian LSTM model following the architecture described in Sec. III.A and the Bayesian formulation in Sec. III.B. Fig. 6 summarizes the overall preprocessing, training, validation, and testing procedure in a flowchart.

The procedure starts by loading all the CFD and CAA simulation data. In particular, the input feature are the acoustics time-series data, and output targets are the lift and moment coefficients (C_L and C_M) time-series. In an effort to increase robustness for our model, the original clean acoustic signals are augmented with (possibly multiple realizations of) additive Gaussian noise in order to mimic measurements corrupted by measurement error in real-life situations. The added noise are independent at every discrete time point with Gaussian form $\mathcal{N}\left(0, (\text{SNR})\frac{1}{\tau}\sum_{t=1}^{\tau} A_{st}^2\right)$ where (SNR) denotes a specified signal-to-noise ratio, and $\frac{1}{\tau}\sum\limits_{t=1}^{\tau} A_{st}^2$ denotes variation of acoustic signals. The entire dataset is then divided into training, validation, and testing. In particular, validation is performed for choosing the number of neurons in the hidden layers for the LSTM via Hyperopt [27]. During this process, Bayesian LSTM is trained per III.B under different hyperparameter settings, but a validation MSE loss is also computed based on a deterministic

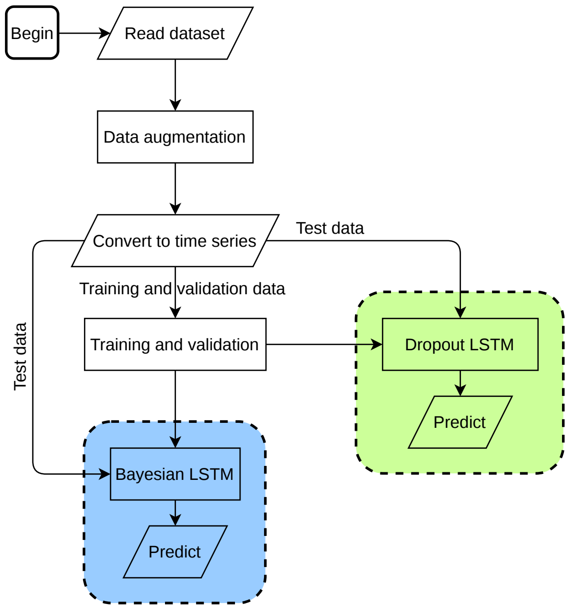


Fig. 6 The overall Bayesian LSTM workflow. The blue frame is the main Bayesian LSTM model, while the green frame is a Monte-Carlo Dropout LSTM model trained with same hyperparameters and used as a control case for comparison.

counterpart of this Bayesian LSTM where the weights are set at their posterior mean. We further track the best validation MSE throughout the training process. The optimal hyperparameter is then selected based on the lowest validation MSE.

Upon training the main Bayesian LSTM model, we compare its performance to a control case of Monte-Carlo Dropout LSTM [36]. Dropout is a regularization approach that is not derived from a Bayesian formulation but has a connection to a specific Bayesian prior and likelihood form under certain limiting conditions [37]. Dropout does not offer a way to adjust the dropout mechanism to remain consistent with, for example, when one wishes to use a different prior (e.g., uniform), different σ_1 and σ_2 values in our Gaussian mixture prior (Eqn. 16), or different σ_ϵ in the likelihood (Eqn. 17). In our setup, every hidden layer is enacted with the dropout mechanism, where each weight has a 50% chance of being set to zero (i.e. connection severed) at each iteration update during training. As a result, only around half of the weights will be back-propagated. During testing, following [36], the final trained deterministic LSTM (i.e. without applying any random dropping of the weights) is used for prediction.

IV. Results

The original simulation dataset available contains 101 far-field sound pressure signals sequenced from mild to severe icing cases. Each case has 1024 time-sample points spanning roughly 102 periods resulted from the forced vibration of the rotor blade. Fig. 7 presents some sample segment of the acoustic signal under different levels of icing.

The dataset is then augmented by creating two realizations of noise-injected acoustic signal, hence increasing to a total of 202 cases. After shuffling, 182 cases are then chosen for training, 10 cases for validation, and the last 10 cases are left for testing. The SNR is uniformly sampled from 0 to 60% for the training points, and summarized in Fig. 8. Since validation points are used for hyperparameter selection, noise is not injected. SNRs for testing points are set to 30% to simulate a scenario where this true noise level remains unknown when training the model.

Hypereparameters are optimized by the Hyperopt package [27]. Specifically, we explore different number of neurons in the first hidden layer $[40, 60, 80, 100, 120, 140, 160]$ and the second hidden layer $[20, 30, 40, 50, 60, 70, 80]$. The experiment is repeated 20 times (with limited computing resources) to find the nearly best hyperparameter pair. Fig. 9 presents all the validation loss under the hyperparameter settings explored by Hyperopt, with the two layer sizes $[160, 70]$ being the optimal. In the future, more hyperparameters may be included, such as number of layers, activation functions, and prior choices. The Bayesian LSTM and Monte-Carlo Dropout LSTM are then trained under the optimized hyperparameters for 5 epochs, with learning rate of 0.005, and gradient (L2 norm) clipping (i.e. limiting) of 0.1.

The trained models are then used to make predictions for the C_L and C_M time-series at the test cases, with Fig. 10

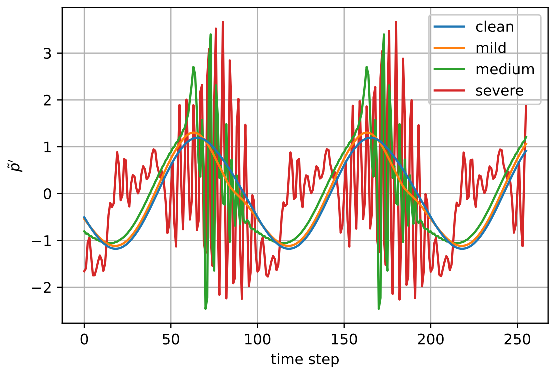


Fig. 7 Sample segments of the acoustic signals under different levels of icing.

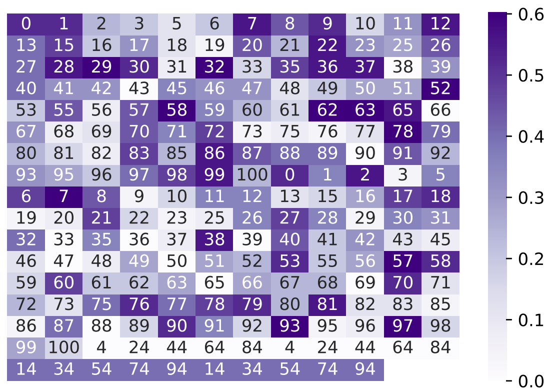


Fig. 8 Heatmap summarizing the sampled SNR. The first 182 cases are training, next 10 cases are validation, and the last 10 cases are testing.

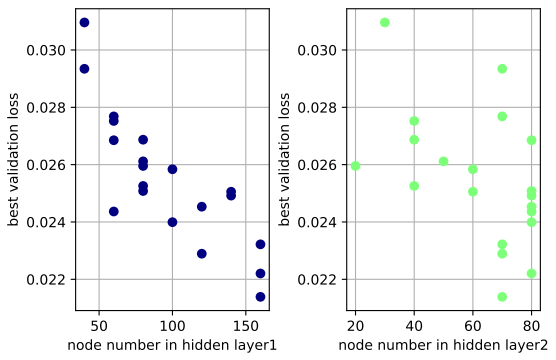


Fig. 9 Hyperparameter validation loss values.

Table 1 MSE Loss and time comparison between Bayesian LSTM and Monte-Carlo Dropout LSTM

Model	C_L MSE Loss	C_M MSE Loss	Prediction time [s]
Bayesian LSTM	0.001516	0.0001307	721
Monte-Carlo Dropout LSTM	0.01168	0.001152	560

showing sample time segments from select test cases. Uncertainty in the predictions are obtained by generating 100 samples from the Bayesian posteriors and then plotting the prediction mean and $\pm 2\sigma$ credible interval on the C_L and C_M quantities.

Table 1 summarizes the prediction MSE loss for the Bayesian LSTM and Monte Carlo Dropout LSTM. We also report the computational time for the prediction tasks on the 10 test cases from using a laptop with an Intel(R) Core(TM) i7-8750H CPU. The Bayesian LSTM appears to reach an MSE loss roughly ten times smaller than the Monte-Carlo Dropout LSTM model, suggesting a more accurate prediction. However, we note that the goal of a BNN is not to minimize error, but rather to produce the uncertainty as defined by the posterior distribution. Therefore, further assessment of its performance should be based on how close the computed posterior (from using VI in this case) is to the true posterior as dictated by the axioms of probability. Furthermore, the prediction times translate to predicting around 200 minutes of time-series window using only around 1 minute, showing promise for on-board usage. However, the computational requirement would also be affected by how often predictions need to be made.

V. Conclusions

In this work, we built a Bayesian LSTM model to predict aerodynamic performance indicators for rotorcraft blades from their acoustic signals. The use of LSTM accommodates time-series formats for both the input signal and output prediction (i.e. the coefficients of lift and moment as functions of time). The training, validation, and testing were performed using datasets generated from physics-based simulations, involving aerodynamic, ice accretion, and aeroacoustic computations. In an effort to increase model robustness against measurement noise in real-life settings, we augment the clean simulated acoustic signals with Gaussian noise. The model hyperparameters were explored using Hyperopt, and the model training was accomplished by minimizing the negative ELBO from a mean-field VI formulation. Preliminary results indicate good predictions along with posterior predictive credible intervals approximated by VI.

Acknowledgments

The computational resources provided by the RHRK high performance computing center via the ‘Elwetritsch’ high performance cluster at the TU Kaiserslautern is gratefully acknowledged. The work from Politecnico di Milano has received funding from the European Union’s H2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 721920. Further information can be found at the Network for Innovative Training on ROtorcraft Safety (NITROS) project website.

References

[1] Hartman, P., Narducci, R., Peterson, A., Dadone, L., Mingione, G., Zanazzi, G., and Brandi, V., “Prediction of Ice Accumulation and Airfoil Performance Degradation: a Boeing-CIRA Research Collaboration,” *American Helicopter Society 62nd Annual Forum*, 2006.

[2] Flemming, R., Alldridge, P., and Doeppner, R., “Artificial Icing Tests of the S-92A Helicopter in the McKinley Climatic Laboratory,” *42nd AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2004–737, Reno, NV, 2004.

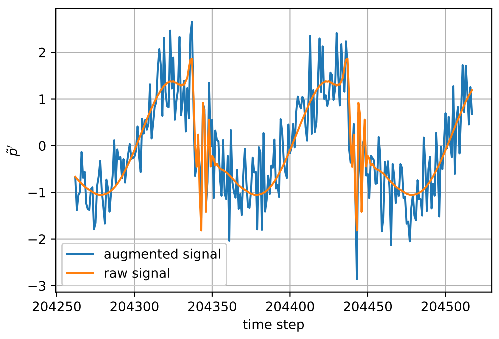
[3] NTSB, “National Transportation Safety Board Aviation Accident Final Report,” NTSB No. CEN13FA122, January 2013.

[4] NTSB, “National Transportation Safety Board Aviation Accident Final Report,” NTSB No. WPR17FA047, December 2017.

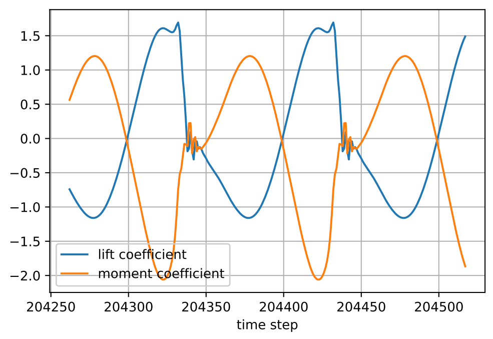
[5] NTSB, “National Transportation Safety Board Aviation Accident Final Report,” NTSB No. ANC16FA023, January 2018.

[6] NTSB, “National Transportation Safety Board Aviation Accident Final Report,” NTSB No. WPR16LA104, June 2018.

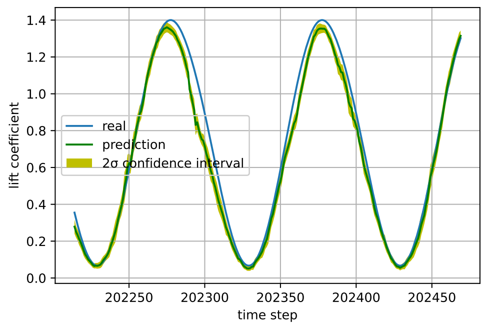
[7] NTSB, “National Transportation Safety Board Aviation Accident Final Report,” NTSB No. CEN19FA072, January 2019.



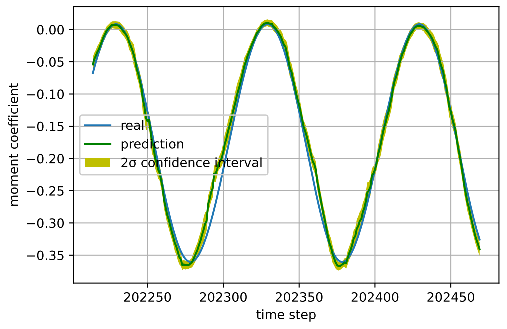
(a) Acoustic signals on test case ID 54 in Fig. 8



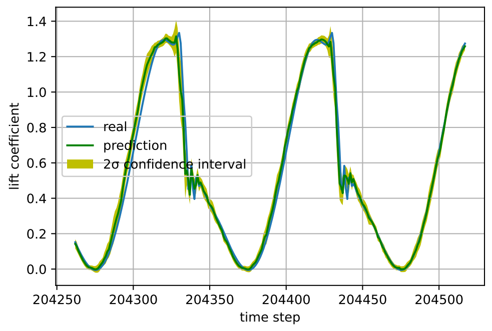
(b) Performance indicators on test case ID 54 in Fig. 8



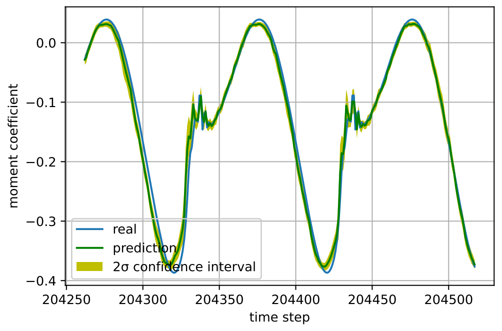
(c) C_L prediction on test case ID 14 in Fig. 8



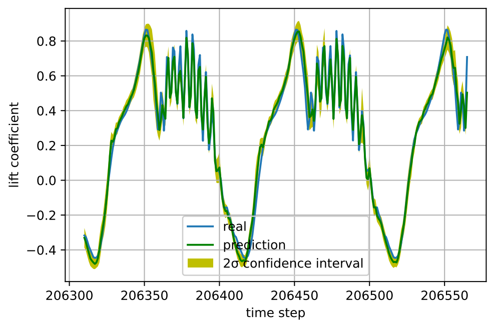
(d) C_M prediction on test case ID 14 in Fig. 8



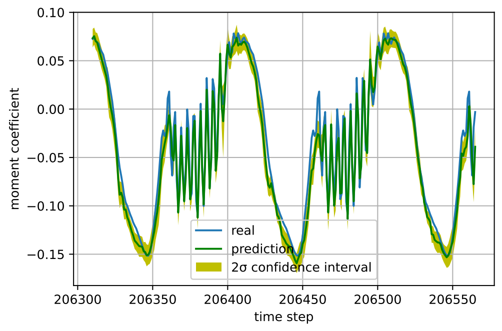
(e) C_L prediction on test case ID 54 in Fig. 8



(f) C_M prediction on test case ID 54 in Fig. 8



(g) C_L prediction on test case ID 94 in Fig. 8



(h) C_M prediction on test case ID 94 in Fig. 8

Fig. 10 (a)–(b): The raw acoustic signal, augmented signal after adding Gaussian noise, and true C_L and C_M corresponding to the case in (e)–(f). (c)–(h): Sample C_L and C_M predictions at select test points using the Bayesian LSTM.

[8] Heinrich, A., Ross, R., Zumwalt, G., Provorse, J., and Padmanabhan, V., “Aircraft Icing Handbook. Volume 1–3,” Tech. rep., Gates Learjet Corp Wichita KS, 1991.

[9] Lande, K., “Helicopter Flight in Icing Conditions–Operational Aspects,” *European Rotorcraft Forum*, 1996.

[10] Withington, T., “Bad Vibrations,” *Aerosafety World Magazine, Helicopter Safety, Flight Safety Foundation*, 2010, pp. 26–28.

[11] Cheng, B., Han, Y., Brentner, K. S., Palacios, J. L., and Morris, P. J., “Quantification of rotor surface roughness due to ice accretion via broadband noise measurement,” *American helicopter society 70th annual forum proceedings*, 2014, pp. 2680–2692.

[12] Chen, X., Zhao, Q., and Barakos, G., “Numerical Analysis of Rotor Aero-acoustic Characteristics for Ice Detection based on HMB Solver,” *American Helicopter Society International 73rd Annual Forum & Technology Display*, 2018.

[13] Zhou, B., Albring, T. A., Gauger, N. R., Ilario, C., Economon, T. D., and Alonso, J. J., “Reduction of airframe noise components using a discrete adjoint approach,” *18th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2017, p. 3658.

[14] Hornik, K., “Approximation Capabilities of Multilayer Neural Network,” *Neural Networks*, Vol. 4, No. 1989, 1991, pp. 251–257. https://doi.org/10.1016/0893-6080(91)90009-T.

[15] Zhou, B. Y., Gauger, N. R., Morelli, M., Guardone, A., Hauth, J., and Huan, X., “Towards a Real-Time In-Flight Ice Detection System via Computational Aeroacoustics and Bayesian Neural Networks,” *AIAA/ISSMO Multidisciplinary Analysis and Optimization at the AVIATION Forum*, Dallas, TX, 2019. https://doi.org/10.2514/6.2019-3103.

[16] Hauth, J. M. A., Huan, X., Zhou, B. Y., Gauger, N. R., Morelli, M., and Guardone, A., “Correlation Effects in Bayesian Neural Networks for Computational Aeroacoustics Ice Detection,” *AIAA Scitech 2020 Forum*, Orlando, FL, 2020. https://doi.org/10.2514/6.2020-1414.

[17] Zhou, B. Y., Gauger, N. R., Morelli, M., Guardone, A., Hauth, J., and Huan, X., “Development of a Real-Time In-Flight Ice Detection System via Computational Aeroacoustics and Bayesian Neural Networks,” *AIAA Scitech 2020 Forum*, Orlando, FL, 2020. https://doi.org/10.2514/6.2020-1638.

[18] MacKay, D. J. C., “A Practical Bayesian Framework for Backpropagation Networks,” *Neural Computation*, Vol. 4, No. 3, 1992, pp. 448–472. https://doi.org/10.1162/neco.1992.4.3.448.

[19] Neal, R. M., *Bayesian Learning for Neural Networks*, Springer-Verlag New York, New York, NY, 1996.

[20] Graves, A., “Practical Variational Inference for Neural Networks,” *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, Granada, Spain, 2011, pp. 2348–2356.

[21] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D., “Weight Uncertainty in Neural Networks,” *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37, 2015, pp. 1613–1622.

[22] Gal, Y., “Uncertainty in Deep Learning,” Ph.D. thesis, University of Cambridge, 2016.

[23] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., “SU2: An Open-Source Suite for Multiphysics Simulation and Design,” *AIAA Journal*, Vol. 54, No. 3, 2016, pp. 828–846. https://doi.org/10.2514/1.J053813.

[24] Morelli, M., Zhou, B. Y., Guardone, A., et al., “Simulation and Analysis of Oscillating Airfoil Ice Shapes via a Fully Unsteady Collection Efficiency Approach,” *American Helicopter Society 75th International Annual Forum*, 2019.

[25] Shekar, B., and Dagnew, G., “Grid search-based hyperparameter tuning and classification of microarray cancer data,” *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, IEEE, 2019, pp. 1–8.

[26] Bergstra, J., and Bengio, Y., “Random search for hyper-parameter optimization.” *Journal of Machine Learning Research*, Vol. 13, No. 2, 2012.

[27] Bergstra, J., Yamins, D., and Cox, D. D., “Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms,” *Proceedings of the 12th Python in Science Conference*, 2013.

[28] Schmidt, R. M., “Recurrent Neural Networks (RNNs): A gentle Introduction and Overview,” *arXiv preprint arXiv:1912.05911*, 2019.

[29] Hochreiter, S., “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 6, No. 02, 1998, pp. 107–116.

[30] Hochreiter, S., and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[31] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R., "Efficient BackProp," *Neural Networks: Tricks of the Trade*, edited by G. Montavon, G. B. Orr, and K.-R. Müller, Springer-Verlag Berlin Heidelberg, 2012, pp. 9–48. https://doi.org/10.1007/978-3-642-35289-8_3.

[32] Robbins, H., and Monro, S., "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, Vol. 22, No. 3, 1951, pp. 400–407. https://doi.org/10.1214/aoms/1177729586.

[33] Sutskever, I., Martens, J., Dahl, G., and Hinton, G., "On the importance of initialization and momentum in deep learning," *International conference on machine learning*, PMLR, 2013, pp. 1139–1147.

[34] Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I., "An Introduction to MCMC for Machine Learning," *Machine Learning*, Vol. 50, 2003, pp. 5–43.

[35] Brooks, S., Gelman, A., Jones, G., and Meng, X.-L., *Handbook of Markov Chain Monte Carlo*, Chapman and Hall/CRC, 2011. https://doi.org/10.1201/b10905.

[36] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, Vol. 15, 2014, pp. 1929–1958.

[37] Gal, Y., and Ghahramani, Z., "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *33rd International Conference on Machine Learning, ICML 2016*, Vol. 3, 2016, pp. 1651–1660.