# Development of a Machine-Learned Cruise Guide Indicator for Rotorcraft

**Mathew Boyer**
Computational Scientist
DoD HPCMP PET/GDIT
Aberdeen, MD, USA

**Wesley Brewer**
HPC/AI Research Scientist
Oak Ridge National Laboratory
Oak Ridge, TN, USA

**Jeff Finckenor**
Structural Engineer
DEVCOM AvMC
Redstone Arsenal, AL, USA

**Chris Brackbill**
Lead Aerospace Engineer
DEVCOM AvMC
Redstone Arsenal, AL, USA

**Daniel Martinez**
AI Engineer
DEVCOM AvMC
Moffett Field, CA, USA

**Andrew Wissink**
Aerospace Engineer
DEVCOM AvMC
Moffett Field, CA, USA

## ABSTRACT

This paper presents a machine-learned virtual cruise guide indicator (vCGI) for Chinook helicopters. Two temporal neural networks were trained and evaluated on measured data from 55 flight tests, one for the fore rotor and another for the aft rotor, to predict a vCGI value, which protects 23 components from fatigue damage during steady-state conditions. Three different classes of machine learning architectures were evaluated for prediction of the vCGI from time sequences: a temporal convolutional neural network with 1D dilated causal convolutions, a long short-term memory recurrent neural network, and an attention-based transformer architecture. The final average model accuracy on unseen flight data is currently greater than 93% for CGI values which could result in fatigue damage and 90% for normal operation CGI values. Model accuracy was improved through a series of advancements in: (1) selection of optimal training data using temporal collective variables and unsupervised learning, (2) dataset augmentation with maximum-entropy temporal collective variables, and (3) implementation of a mixture-of-experts classification-regression approach using an adversarial classification approach to assign maneuver labels. The results are presented for each advancement in model development along with lessons learned in training machine learning models on real-world, time-dependent rotorcraft data.

## INTRODUCTION

Virtual sensor methods have been proposed for future Army rotorcraft platforms or future upgrades as a means of monitoring or extending component lives. By constructing a virtual sensor from flight test data acquired from a heavily instrumented aircraft, stress loads can be monitored during standard aircraft operation without the need for additional physical sensors. Moreover, they could also enable mission planners in acquisition programs to assess performance before a new aircraft or upgrade is completed using digital models.

Cruise guide indicators (CGI), used on Chinook helicopters, provide pilots a visual indication of stress loads on critical components during flight, based on on-aircraft strain-gauges (Ref. 1). In contrast, a virtual CGI (vCGI) takes in standard flight data and virtually determines the stresses using a machine learning model trained on flight test data. Figure 1 shows an example of a cruise guide indicator available as a cockpit instrument in rotorcraft. The indicator is divided into three regions: normal operation (green), transient (yellow), and avoid (red). The CGI is computed as a function of both fore and aft fixed link oscillatory loads.

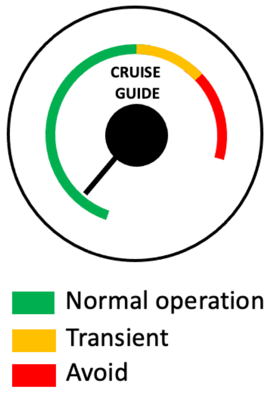

**Figure 1. Schematic of a Cruise Guide Indicator gauge for CH-47 rotorcraft, as described in Ref. 1.**

Presented at the Vertical Flight Society's 79th Annual Forum & Technology Display, West Palm Beach, FL, USA, May 16-18, 2023. This is a work of the US Government and is not subject to copyright protection. DISTRIBUTION STATEMENT A: Approved for public release: distribution unlimited.

Oldersma and Bos (Ref. 2) provided an overview of a system they developed called "CHAMP" which stands for Chinook Airframe Monitoring Program, which is used to develop a "virtual strain gauge" based on artificial neural networks (ANN). Their training data, based on 49 recorded flight parameters, monitors nine strain gauges on the CH-47D Chinook helicopter.

Isom et al. (Ref. 3) investigated the development of a technique for the virtual monitoring of loads (VML) based on flight tests of the U.S. Army RASCAL JUH-60 helicopter. Their model is based on 19 aircraft state parameters, and their dataset contains a series of maneuvers including turns, pull-ups, and pushovers. Their model uses proprietary regression algorithms developed by Sikorsky (Ref. 4).

Martinez et al. (Ref. 5) investigated the development of a virtual accelerometer that may be used to predict a vibration spectrum using deep neural-network-based regression, or deep regression, from which the health condition of an oil cooler may be predicted. Martinez et al. (Ref. 6) investigated the development of machine-learned aerodynamic models based on high-fidelity computational data from CREATE-AV Helios CFD software. Their neural networks are trained on simulated flight test maneuvers and demonstrated on the Utility Tactical Transport Aircraft System (UTTAS) pullup maneuver.

This study investigated the development of neural networks to predict the fore and aft fixed link oscillatory loads and, from the loads, the vCGI. A variety of neural network architectures were evaluated, including ones based on multi-layer perceptrons (MLP), temporal convolutional neural networks (TCNN), Long Short-Term Memory (LSTM) layers, and transformer/attention-based networks, as well as ensembles of these models. The final model developed achieves greater than 90% average accuracy for predicting the cruise guide indicator across a range of flight conditions. Several technical approaches enabled significant improvements in the model: (1) normalizing the input features before training or inferencing the neural network, (2) converting the input features to overlapping time sequences, typically with a window of four to eight time steps per model input, (3) using KerasTuner with the HyperBand technique for hyper-parameter optimization, (4) implementing various types of dataset culling and subsampling techniques, and (5) development of a mixture-of-experts classification-regression framework based on flight maneuvers. All training was performed using Nvidia V100 GPUs on the DoD HPCMP supercomputers: Onyx-MLA, SCOUT, and Vulcanite. The models were trained using data from 55 flight tests, resampled at a rate of approximately 16 Hz. The resampled data consists of 74 features routinely monitored on rotorcraft, such as: pitch angle, radar altitude, engine fuel flow, rate of climb, and engine torque. The 74 features and resample rate were chosen to mimic existing on-board data collection in fielded aircraft.

This paper presents a virtual cruise guide indicator, which protects 23 components from fatigue damage during steady-state conditions. The model uses 74 flight parameters and was trained and evaluated on 55 flights tests, covering more than 14 flight maneuver categories. Three classes of neural networks designed for temporal forecasting (Ref. 7) were evaluated for accuracy, with a focus on the yellow and red regimes of the CGI. The work advances the state-of-the-art in the areas enumerated below:

1. A feature-rich raw training data set was developed based on flight test data. The dataset contains a wide array of flight maneuvers performed at multiple gross weights and altitudes.
2. A novel framework was developed for selecting optimal subsets of flight test data for training neural networks using temporal collective variables and unsupervised learning. A maximum-entropy feature importance method was also developed and used to identify information-rich features. The collective variable representations of these features were incorporated as additional model inputs.
3. Various temporal forecasting neural network architectures were studied and optimal strategies for predicting the vCGI were explored. This included evaluation of model targets, whether to model the CGI instrument values directly versus modeling the oscillatory forces, the optimal sampling frequency, and whether the forces are averaged once or thrice (for the 3-bladed Chinook rotor) per rotor revolution.
4. A mixture-of-experts approach using two-stage classification-regression models was explored. A novel approach was developed to classify flight segments based on limited labeled data and maneuver-specific models were trained on the full dataset using the machine-generated labels. The final model uses a weighted average of the specialized models based on the classifier probabilities.
5. A production pipeline was developed for training and evaluating machine learned models on high performance computers (HPC) using multiple GPUs.

## TECHNICAL APPROACH

### Flight Test Data

Models were developed and tested based on data from 55 flight tests with a heavily instrumented CH-47 rotorcraft. A subset of features that would be readily available based on the standard (operational) instrumentation onboard H-47 rotorcraft was used for the inputs to models, such that the final models could be deployed without modification to existing rotorcraft. Several model targets were evaluated in this work, including direct prediction of the CGI and indirect prediction of the CGI via prediction of the mean or oscillatory fixed link loads. The cumulative dataset contained hundreds of hours of flight time sampled at 16 Hz, which corresponds to over six million data points. This real-world dataset contains a variety of flight maneuvers such as hover, turns, dives, climbs, doublets, and different types of landings, and includes different altitudes and gross weights. In addition to the targeted maneuvers, the dataset also contains all auxiliary operation within the test flights. Within the dataset, 98% of all samples correspond to a green CGI, 1.5% to a yellow CGI, and 0.5% to a red CGI.

The 72 input features utilized for the vCGI were first rescaled to prevent the models from overweighting features based on magnitude. Multiple scaling methodologies were explored, including simple approaches like rescaling the mean-centered features by one standard deviation (standardized scaling) and more complex approaches like the Yeo-Johnson power transformation (Ref. 8). While these more complicated approaches can provide some improvement to accuracy, this can come at the cost of stability if unseen cases exceed the bounds of the training sets. Ultimately, the standardized scaling approach was chosen for its stability and interpretability. Results presented were produced using the standardized scaling.

## Machine Learning Models

For the virtual sensor developed in this work, input features were considered as time histories, in order to account for their gradients. For example, the rate of change of air speed and its direction correspond to acceleration or deceleration; steady-state flight at a given airspeed can result in differing loads than acceleration or deceleration at the same air speed. To enable accounting for the time history of these features, neural networks commonly used in temporal forecasting were chosen. Instead of predicting a sequence of future values based on a sequence of past values, the models developed used a sequence before and including the current time to predict the target value at the current time.

Lim and Zohren (Ref. 7) present three different types of neural network architectures that are typically used in temporal forecasting: temporal convolutional neural networks (TCNNs), recurrent neural networks (RNNs) such as the long short-term memory (LSTM) network, and models based on the attention mechanism, like transformers. All models were implemented in TensorFlow (open source AI/ML software library) versions 2.6 or greater. RNN models were evaluated using a single LSTM layer followed by nine fully-connected layers. For the temporal CNN model, two one-dimensional convolutional layers precede a flattening operation and four fully-connected layers. Finally, the transformer architecture utilizes a multi-head attention layer, followed by two one-dimensional convolutional layers and nine fully-connected layers. For each architecture, the number of parameters (recurrent units, neurons, or filters) were varied and optimized. Multiple activations were evaluated, including hyperbolic tangent, rectified linear unit (ReLU), exponential linear unit (ELU), scaled exponential linear unit (SELU), Gaussian error linear unit (GELU), and Swish. Parameter optimization and activation selection is discussed later in this paper.

## Uncertainty Quantification

A common criticism of neural networks is the black box nature of their predictions, which can be biased or entirely inaccurate under certain circumstances (Ref. 9). For engineering applications, model interpretability is necessary to ensure end-users are aware of when the model is most likely to be inaccurate. To enable this, uncertainty quantification was incorporated into the vCGI through the use of Monte Carlo dropout, following the approach of Gal and Ghahramani (Ref. 10). Dropout (Ref. 11) is the process of setting a random collection of weights in a layer to zero and reweighting the remaining non-zero weights. Typically, dropout is only implemented during training to prevent overfitting. However, it can also be used during inference to perform an ensemble of inferences with a randomly varying neural network. This ensemble is then used to determine the mean prediction and its variance.

After initial development of the vCGI models, Monte Carlo dropout was added as an optional feature to all neural network architectures. Dropout was only applied before interior fully-connected layers and a dropout rate of 0.125 was used. An ensemble size of 16 was utilized for inference. As discussed in the work of Gal and Ghahramani (Ref. 10), the inference variance is dependent on the activation function; activations which bound the layer output, like hyperbolic tangent, can truncate the large variations and result in artificially small variances. This was considered in addition to model accuracy when selecting the optimal activation function for the vCGI.

## Hyper-parameter Optimization

An initial set of hyperparameters for each model architecture was obtained by automatic search using Keras Tuner (Ref. 12). Model parameterization was divided into a few parameters: the number of time-series-encoding layers (LSTM, multi-headed attention, or convolution), the number of neurons or filter for these encoding layers, the activation function used for all but the last layer, and the width of the first fully-connected layer. For the TCNN architecture, subsequent fully-connected layers decreased in size by powers of two, while for the transformer and RNN models a fixed width was used for all but the last fully-connected layer. The Hyperband technique was used for hyperparameter tuning, rather than random search or Bayesian optimization. Once the initial optimal parameterization was found, fine-tuning was performed by manual modification of the base set of parameters and comparing vCGI accuracy after fully training the neural networks. All architectures were found to produce optimal results using the Swish activation function and the Adamax optimizer with a learning rate of 0.0024. A batch size of 256 was utilized and all models were trained for 10 epochs with the learning rate set to decay by half based on plateau of the training loss. For the TCNN model, 112 filters and a kernel size of 2 were found to be optimal; the fully-connected layers used 224 and then halved the next two layers. For the LSTM, 256 units were used, and 256 neurons were used in the fully-connected layers. Finally for the transformer, 4 attention heads with a size of 72 were found to be optimal and fully-connected layers with 256 neurons were used.

## Data and Model Augmentation

The natural statistics (only 2% of data outside normal operation) of the dataset are not optimal for training a model to achieve the target objective: produce a virtual CGI with >95% accuracy for the red/avoid regime. A series of dataset augmentation strategies were explored to improve model accuracy for these conditions, which cause fatigue damage to rotorcraft components. The first approach explored was dataset culling (Ref. 13), where samples beneath a certain threshold were omitted from the training set to increase the relative population of higher values of CGI within the flights. After evaluating this thresholding approach, unsupervised learning methods were evaluated to construct more optimal training sets from the flight data.

To enable the application of common unsupervised learning methods, the CGI time series signal was divided into windows and then decomposed into four collective variable (CV) descriptions of the window: the mean, the slope, the variance, and the curvilinear distance. In this study, a window of eight time steps was used to construct the CVs. The four CV features were then used as the inputs for two different clustering algorithms: *K*-means and the self-organizing map (SOM) presented in Ref. 14. The clustering methods were used to segment the flight data into distinct groups, which can represent different flight modes in terms of strain on the rotorcraft. Figure 2 shows an example flight partitioned by a self-organizing map along with the pressure altitude; cluster indices were sorted from largest to smallest with the mapping shown in the inset. Training datasets were then constructed by randomly drawing an equal number of samples from each cluster. Data within smaller clusters was repeated when the sample size surpassed the cluster size.

The same approach can also be used to expand the input feature set by including the CV representations of certain input features as additional model inputs. Although the model architectures described in the previous section all process time series input features, they all decode a one-dimensional latent vector. To utilize both the time series and a vector of CV features, an additional fully-connected layer can be added to the model to encode the CV features and then the 1D encoding of the CV features can be concatenated with the 1D encoding of the time series features, before being processed by the decoding portion of the architecture. This methodology enables the expansion of the feature set without any additional data, can enable the model to more easily interpret temporal features like increasing altitude or deceleration, but also increases the computational cost of both training and inferencing the models. To maximize the benefit of the additional CV features, while minimizing the cost, an entropy-based feature-importance method was utilized to determine the *N* most optimal features to include as CV representations.

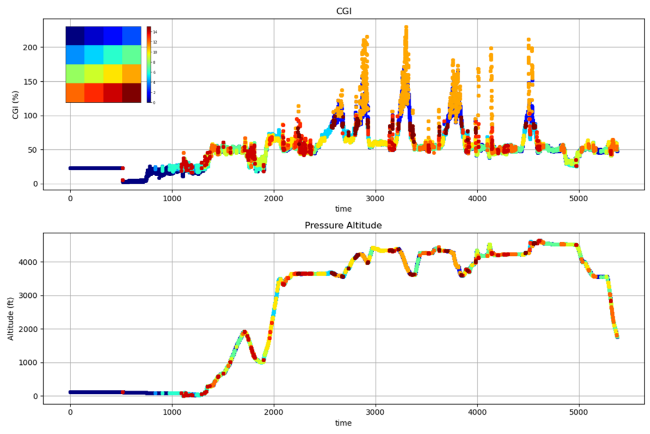

**Figure 2. CGI signal colorized according to self-organizing map cluster (top) and a single input feature, pressure altitude, colorized according to the same CGI clusters (bottom).**

The information entropy of the input time series features was measured by the following approach depicted in Figure 3. First, the time series representation of each input feature was converted to the same four CVs used in clustering the CGI. Next, the clusters generated were used to sort the input features into clusters. The population of each input feature CV in each cluster was then used to construct a probability distribution. For each CV feature, the minimum inter-cluster distance was found by computing the mean absolute sum of the Kullback-Leibler divergence (Ref. 15) between corresponding probability distributions. The entropy of each feature was then found by summing the four CV minimum distances. This process was performed for each flight separately and the maximum entropy of each feature in a flight was then used to rank the information entropy from greatest to least. The top *N* entropy features can then be included as CV representations, where *N* is an additional hyperparameter for model development. In this work, the top 20 entropy features were used when this approach was utilized.

## Mixture-of-Experts Models

In addition to developing general neural network models to predict the CGI, the development of a mixture-of-experts (MoE) approach was also explored. MoE approaches have been widely used in machine learning with methods like Gaussian process regression and support vector machines (Ref. 16) and, even more recently, for large language model neural networks (Ref. 17). As suggested by the name, a MoE uses a weighted sum mixture of specialized models. Typically, the weights for the sum are determined by another model like a probabilistic classifier. In this work, a two-stage classification-regression MoE approach was developed based on timestamp labeled maneuver data. A classification model was developed to predict the probability that a flight segment corresponded to each of nine maneuver classes: ground, low speed, level flight, acceleration, deceleration, climb, descent, turns, or dynamic operation (any transient condition not covered by another label). The same classifier was then used to divide the flight test data into maneuver-specific subsets. Specialized neural network models were then trained on each maneuver-labeled data subset following the same approach used to train the general models. Prediction of the vCGI was then made by taking the probabilities from the classifier and using them as weights for a sum of the predictions of the specialized models. A diagram demonstrating the application of MoE models to this problem is shown in Figure 4.

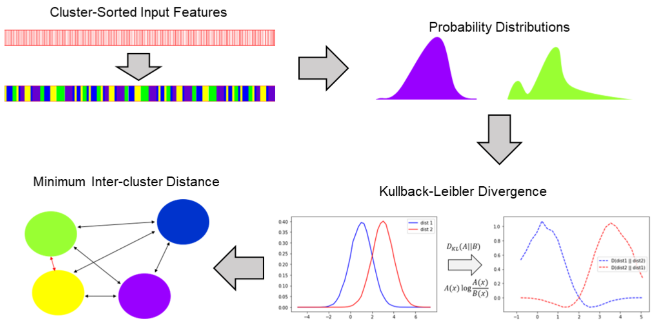
Cluster-Sorted Input Features
Probability Distributions
Minimum Inter-cluster Distance
Kullback-Leibler Divergence
$D_{KL}(A||B)$
$A(x)\log\frac{A(x)}{B(x)}$

**Figure 3. Schematic of entropy-based feature importance methodology using Kullback-Leibler divergence to compute distance between information in each cluster.**

### Adversarial Classification

Although the flight test data contained millions of samples for prediction of the CGI, only a few thousand samples were available with timestamped maneuver labels. In general, this process required hand-labeling by subject matter experts, which limited the ability to expand the dataset size. One key finding discussed within the landmark dropout paper (Ref. 11) was the tendency of neural networks to overfit on small datasets. In general, multiclass classification has issues with overfitting to noise or memorizing rare cases (Ref. 18). To contend with both the inclination of neural networks to overfit on small datasets and in multiclass classification tasks, a novel classification approach was developed to enable the MoE method described previously. Direct classification of flight segments with LSTM-based neural networks was also explored as a baseline methodology for comparison.

Instead of training a model to directly learn the maneuver label based on the input, neural networks were trained to rescale the inputs such that samples with the same label would be closer to one another in the high-dimensional mapped space than to samples with different labels, as illustrated in Figure 5. Once the input space is remapped, clustering can be performed through unsupervised methods, like the *K*-means algorithm. Once the labeled training samples are clustered, each cluster can be mapped to a maneuver based on the greatest label frequency. The label mappings and cluster model can then be applied to unlabeled data to predict the maneuver class.

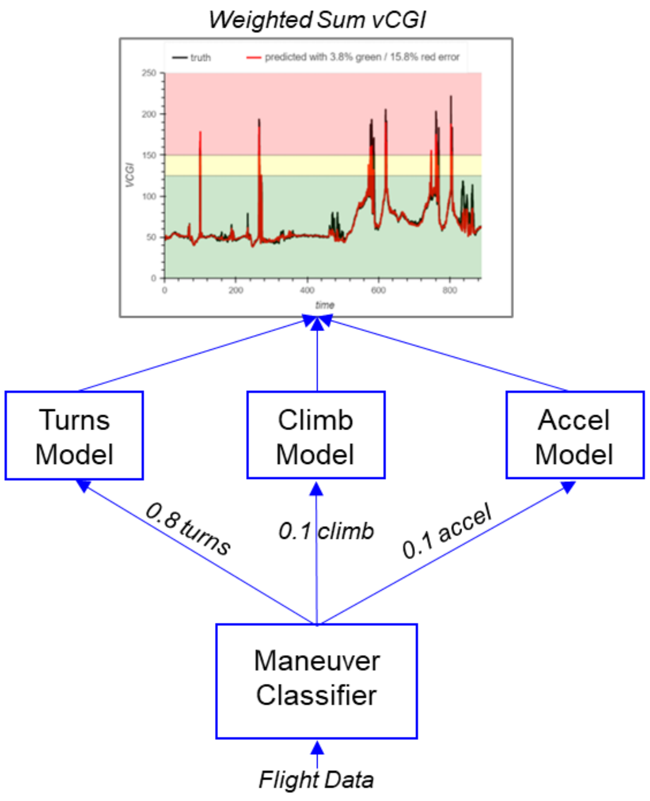

**Figure 4. Example of a mixture-of-experts classification-regression scheme used to predict the CGI.**

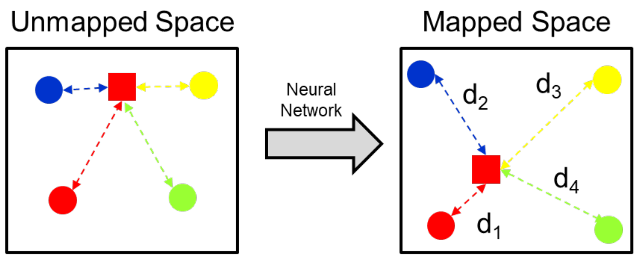

**Figure 5. Demonstration of a neural network learning high-dimensional mapping to a space where samples with the same label (color) are nearer to each other than to samples with different labels.**

The neural network used to rescale the inputs was trained on random groupings of one sample from the target class and one sample from each of the other classes. A custom loss function was implemented based on the squared Euclidean distances ($d^2$) between the target sample and each class sample. A vector of class probabilities was constructed where the probability, $P$, for each class, $i$, was given by:

$$P_i = \frac{1}{{d_i}^2 + 1}$$

The categorical cross-entropy (CCE) was then calculated based on this vector of probabilities and a one-hot encoding of the target maneuver label (target label = 1, other labels = 0). During training, $P$ is used directly, but for determination of MoE weights, the Softmax function is applied to the vector of probabilities.

By providing the model with both positive and negative examples during training, the effective dataset size increase from $N$ labeled samples to $\prod_M N_i$ for $N_i$ samples in each of the $M$ classes. This approach was inspired by the training of generative adversarial neural networks, where the generator is rewarded or penalized, depending on the discriminator. In this adversarial classification approach, the model is rewarded for decreasing distance between samples of the same class and penalized for decreasing the distance between samples of different classes. In direct prediction of the class label, only the first is explicitly done. The use of distance in a high-dimensional space allows for learning categorization in a continuous fashion, rather than the discrete assignment of labels. Furthermore, the distance mechanism allows for a clear and interpretable definition of an unknown sample: any test value that is nearly equally far from all reference centroids does not fall into any assigned class.

## RESULTS AND DISCUSSION

The vCGI model presented in this paper was developed and refined over a series of studies. It was found that the model architecture, the input features, and the relative sampling of different segments of the flight envelope within the training set all significantly affected the fidelity of the virtual sensor predictions. The vCGI model was based on two independent models trained to predict the fixed link oscillatory loads for the fore and aft rotors. This approach provides more information about specific component damage and avoids the need for the model to handle discontinuities from the piece-wise nature of the CGI. The oscillatory loads were averaged once per revolution, as this produced better results with less jitter than more frequent averaging, as shown in Figure 6.

Different neural network architectures were compared, with a primary focus on those which leverage a time history of the input features for prediction of the vCGI. Even after careful optimization of the training procedure and hyper-parameters, the vCGI trained on raw flight datasets was found to significantly overpredict and/or underpredict the analog CGI value. This led to optimization of the training dataset through subsampling based on temporal collective variables describing the CGI and its change over time. While this approach showed improvements to the model accuracy, most notably a significant reduction in over-prediction, additional avenues of refinement were explored to further improve accuracy.

The first leveraged an entropy-based feature importance method to down select the full feature set and then a model architecture modification to incorporate the temporal collective variables of these high-entropy features as additional model inputs. The second approach leveraged all previously mentioned developments within a mixture-of-experts framework based on classification of the flight segment based on maneuver. The intent was to develop models specialized towards certain input feature regimes representative of general flight maneuver categories such that the regression models did not need to accurately predict the CGI for the full flight envelope.

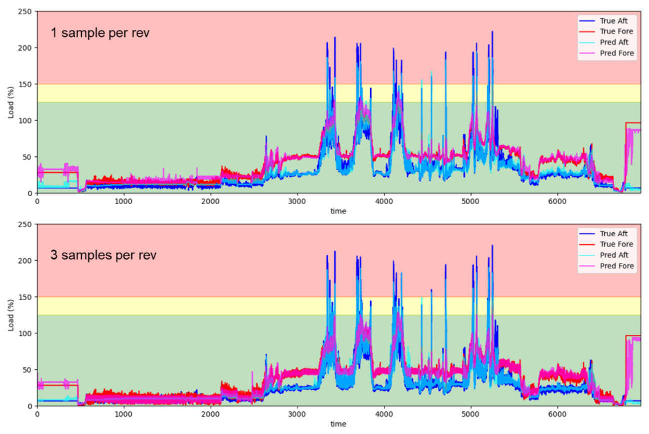

**Figure 6. Fore and aft fixed link oscillatory loads as a percent of component damage threshold and model predictions with sampling rates of once per revolution (top) and thrice per revolution (bottom).**

### Neural Network Architecture Evaluation

Initial attempts at developing a vCGI utilized flight data from a single instance in time to predict the CGI. This approach was found to be insufficient to predict quantities dependent upon gradients. For example, a model which does not interpret a time sequence or explicitly include feature gradients with respect to time cannot easily distinguish between steady flight and acceleration for a given airspeed. To demonstrate the importance of this time component, a multi-layer perceptron (MLP) neural network (equivalent to the LSTM-based model without the LSTM encoding layer) was trained and compared against the time series architectures. While an MLP can process a sequence by flattening the time dimension into the feature dimension, this does not necessarily interpret time in order. Table 1 shows selected results of the neural network models from evaluation on five unseen test flights; the models were trained on 37 test flights using sequence lengths of 1, 4, and 8. The mean percent error for instances when the true CGI is above the threshold for fatigue damage is shown (Y/R error). Additionally, the categorical overprediction ratio is reported: the number of false-positive predictions of CGI yellow/red divided by the number of true CGI yellow/red occurrences.

From these results, it can be seen that including a time history of the input features increases model accuracy for both the MLP and LSTM models, compared to their non-temporal equivalents. Furthermore, the architectures designed for time series data more accurately predict the CGI for values in the yellow or red, compared to the MLP model. Generally, there is an observed tradeoff between the yellow/red accuracy and the overprediction ratio, whereas when this accuracy increases, more green CGI samples are predicted to be in the yellow or red categories. The LSTM-based model exhibits the lowest tendency to overpredict the CGI category, while performing marginally worse than the TCNN model at predicting the CGI in the yellow/red.

**Table 1. Results of Architecture Evaluation.**

| Architecture | Sequence | Y/R Error (%) | Overprediction Ratio |
|---|---|---|---|
| MLP | 1 | 19.8 | 1.3 |
| MLP | 8 | 17.2 | 0.5 |
| LSTM | 1 | 18.0 | 2.6 |
| LSTM | 4 | 14.8 | 2.2 |
| LSTM | 8 | 15.5 | 1.7 |
| TCNN | 4 | 16.2 | 2.8 |
| TCNN | 8 | 14.4 | 3.8 |
| Transformer | 4 | 15.2 | 3.4 |
| Transformer | 8 | 12.4 | 4.2 |

To better understand the meaning of the two presented metrics, Y/R error and overprediction ratio, plots of the vCGI predictions for two of the unseen test flights are presented in Figures 7, 8, and 9, for the nontemporal MLP, the LSTM with sequence length 4, and the transformer with sequence length 8, respectively. The first unseen test flight contains pullups, partial power descents, and high-velocity level flight, while the second unseen test flight contains autorotation maneuvers. The baseline MLP model tracks the primary signal mode of the CGI with some spurious peaks and overpredictions for the second test flight. In general, this model underpredicts the magnitude of many of the peaks, even when accounting for the uncertainty estimate; this model only correctly predicts that the CGI is in the red/yellow 24.8% of the time and only 48% of underpredictions are within three standard deviations of the threshold for yellow CGI. By comparison, this model also predicts large uncertainty bands, which is likely due to the simpler description of the input features resulting in wide range of CGI values for many similar sets of inputs.

Utilizing the time history of the input features enables the models to better disambiguate by contextualizing flight conditions against the short-term history. This results in lower model uncertainty for the LSTM model, compared to the MLP model, with the exact same training data, as seen in Figure 8. The additional information enables the model to more accurately predict the CGI while in the yellow or red as well. However, even with the additional information, this model still underpredicts many of the CGI peaks and predicts spurious features, particularly in the second unseen flight with the autorotation maneuvers.

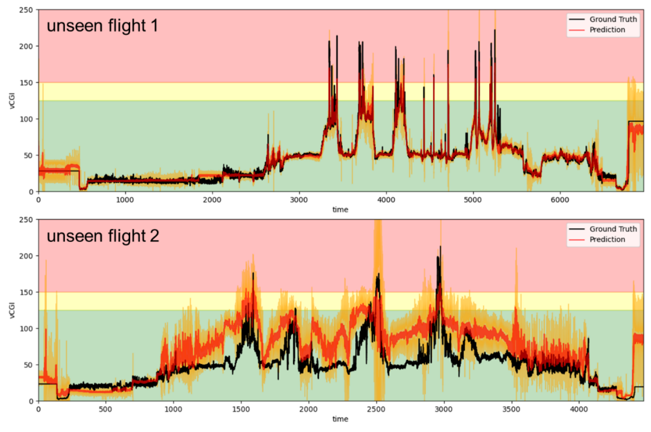

**Figure 7. Predictions on two unseen test flights simulating normal flight (top) and normal flight with autorotation (bottom) made using a multi-layer perceptron model. The model uncertainty is shown for ± 3 standard deviations in orange.**

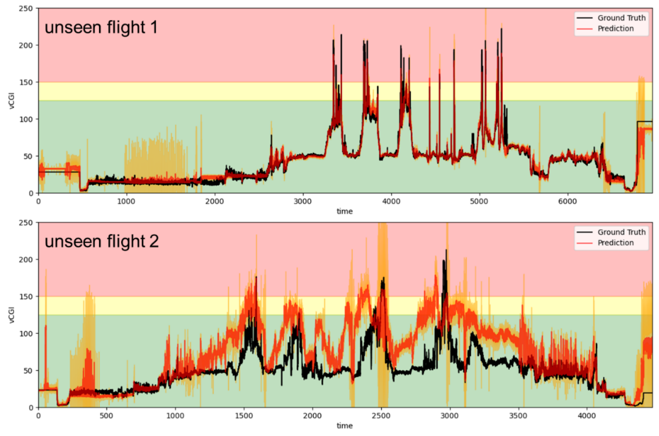

**Figure 8. Predictions on two unseen test flights simulating normal flight (top) and normal flight with autorotation (bottom) made using an LSTM neural network with a time sequence of four. The model uncertainty is shown for ± 3 standard deviations in orange.**

The transformer model demonstrated both the best accuracy in red CGI prediction and the highest overprediction ratio. Figure 9 demonstrates that these models capture many of the large peaks in CGI at the cost of predicting a large amount of spurious features, particularly for the second unseen test flight. In general, every model performed the worst on this flight, which suggests it is not well represented within the training set. However, comparing models with this flight demonstrates their ability, or lack thereof, to generalize. Both the MLP and the transformer appear to overfit the training data, as both models exhibit large uncertainties for the majority of the second unseen flight. While the MLP lacks the complex description of the input features provided by the time history, the transformer still appears to overpredict to some subset of features through the attention mechanism.

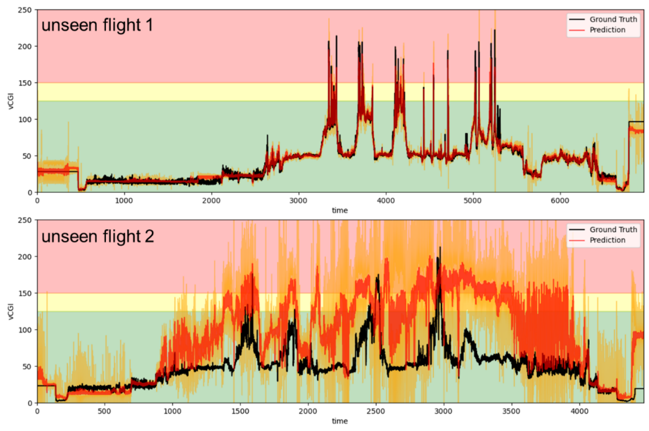

**Figure 9. Predictions on two unseen test flights simulating normal flight (top) and normal flight with autorotation (bottom) made using a transformer-based neural network with a time sequence of eight. The model uncertainty is shown for ± 3 standard deviations in orange.**

The results presented in this section were the final findings after automated tuning and hand-tuning of hyper-parameters, data scaling, and the training procedure. As mentioned previously, the raw dataset is heavily imbalanced towards lower values of CGI with the green regime. While it may be expected that this would result in underprediction of the CGI peaks, the poor generalization and spurious overpredictions were also expected to be caused by overfitting to noise within the imbalanced dataset. To further improve the vCGI model accuracy and generalization, dataset augmentation was explored.

## Dataset Augmentation

Initial investigations of dataset augmentation focused on dataset culling (Ref. 13) by thresholding or removing entire flight sections without any yellow/red CGI values. These approaches provided minor improvements to the yellow/red CGI error, but significantly degraded the model accuracy flight segments with green CGI. This finding led to the exploration of more complex methods based on unsupervised learning.

Using the methodology outlined in the Technical Approach section, the training dataset for the same 37 test flights was reduced from 4.5M samples to about 1M samples. Because the procedure includes the replication of samples within smaller clusters, it should be noted this does not amount to a removal of 3.5M samples; the data reduction was likely more significant. The clustering algorithm utilized 16 clusters and up to 2,000 samples from each cluster in each flight were drawn to construct the modified training set. Because of the training set size reduction, the time to train the vCGI models was also reduced by about a factor of four. Table 2 shows the results for the temporal forecasting architectures using sequence lengths of 4 and 8. The augmented dataset resulted in a reduction of the overprediction ratio across all models, while not significantly depreciating the accuracy for yellow/red CGI prediction. For some model formulations, such as the LSTM model with a sequence of 8, the accuracy increased along with the reduction in the overprediction ratio.

**Table 2. Model Performance with Subsampling.**

| Architecture | Sequence | Y/R Error (%) | Overprediction Ratio |
|---|---|---|---|
| LSTM | 4 | 15.6 | 1.4 |
| LSTM | 8 | 12.5 | 1.3 |
| TCNN | 4 | 15.7 | 1.7 |
| TCNN | 8 | 17.4 | 2.6 |
| Transformer | 4 | 14.6 | 1.8 |
| Transformer | 8 | 12.1 | 1.8 |

Accounting for both the yellow/red CGI accuracy and overprediction ratio, the LSTM model with the longer sequence produced the best overall results. Figure 10 shows the prediction with uncertainty for the same two unseen test flights as used in the previous section. It is apparent that the subsampling drastically improves the model generalization in the second flight containing the autorotation maneuvers, but careful inspection also indicates that the CGI peaks are also better resolved in the first unseen test flight.

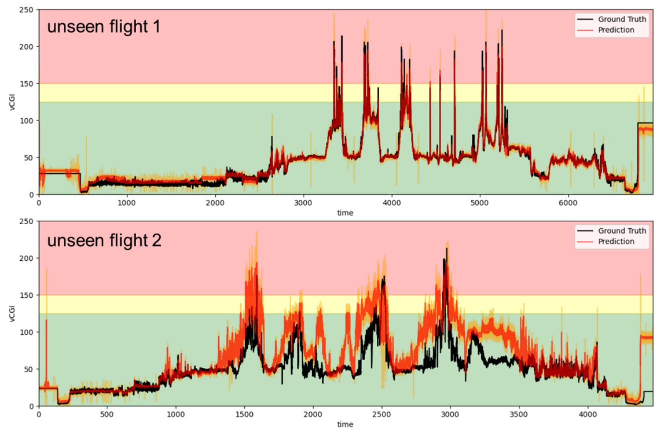

**Figure 10. Predictions on two unseen test flights simulating normal flight (top) and normal flight with autorotation (bottom) made using a LSTM-based neural network with a time sequence of eight trained on subsampled flight data. The model uncertainty is shown for ± 3 standard deviations in orange.**

This finding is significant, because it contradicts a common assumption in artificial intelligence research, which has been previously criticized in Ref. 19., that more data produces better models. For science and engineering applications, rare events are often the most important. For example, to ensure their model would accurately capture high-energy transitions states, the authors in Ref. 20 generated a synthetic dataset to train their machine-learned interatomic potential model. The synthetic data was generated through performing simulations with an external bias that made high-entropy configurations more likely to occur than under natural circumstances. Although the same approach cannot be adopted to experimental flight test data, the flight regime subsampling approach outlined in this work follows the same principle of placing more equal importance on events with unequal natural probabilities, where probability manifests as frequency of occurrence within the dataset. By constructing datasets with more uniform statistical representations of all salient phenomena, models can be trained which generalize better across the full flight envelope. To further leverage this concept, feature set augmentation was explored using a maximum entropy approach.

### Feature Augmentation and Architecture Modification

The methodology described in the Technical Approach was used to rank the input features from most information entropy to least. The top 20 entropy features were then selected for inclusion as temporal CVs. A single additional fully-connected network was added to each temporal forecasting architecture to process the 80 CV features. The encoded CV latent space was concatenated with the vector output of the first fully-connected layer following the time series encoder portion of each architecture. Table 3 shows the results for each architecture with sequence lengths of 4 and 8.

**Table 3. Model Performance with Subsampling and Maximum Entropy Collective Variables.**

| Architecture | Sequence | Y/R Error (%) | Overprediction Ratio |
|---|---|---|---|
| LSTM | 4 | 13.5 | 1.8 |
| LSTM | 8 | 12.0 | 1.0 |
| TCNN | 4 | 14.7 | 2.2 |
| TCNN | 8 | 14.3 | 2.4 |
| Transformer | 4 | 9.8 | 1.9 |
| Transformer | 8 | 12.9 | 2.4 |

In this approach, the time sequence length affects both the history interpreted by the time series encoder and the information basis for the CV features. For all three architectures, including the CVs and using a sequence length of 4 improved yellow/red CGI accuracy and increased the overprediction ratio. The extent of both these changes varied, but the transformer model showed the best improvement with only a slight increase in overprediction ratio and an almost 5% reduction in the red/yellow CGI mean error. In fact, the transformer performed better in both metrics with the shorter sequence, which was not true for either the TCNN or the LSTM. The sequence length was found to make little difference in the TCNN results, with a mild tradeoff between red/yellow accuracy and overprediction ratio, with the longer sequence improving high-CGI accuracy. The LSTM model produced superior results by both metrics with the longer sequence, further demonstrating the recurrent architecture's suitability for interpreting the time series features. Including the maximum entropy CV features reduced the overprediction ratio from 1.8 to 1.0 with a mild improvement to the high-CGI accuracy.

Although the transformer produced lower error in the yellow/red CGI, it still appeared to generalize poorly compared to the LSTM model. The best model would depend upon the error tolerance for overpredictions and for underpredicting the vCGI. Accounting for accurate high-CGI predictions and overpredictions, the LSTM model predicts the total time spend in red/yellow CGI to be 115% of the ground truth, while the transformer model predicts 224% of real time spent in yellow/red CGI. Based on this, the LSTM model with data and feature augmentation and a sequence length of 8 was determined to be the best overall model. Figure 11 shows the predictions of this model for the two unseen test flights. While determining the overall best model depends on the use-case needs, the LSTM model appears to be the best architecture for the purpose of monitoring fatigue damage-causing scenarios based on its accuracy and minimal net overprediction yellow/red CGI events.

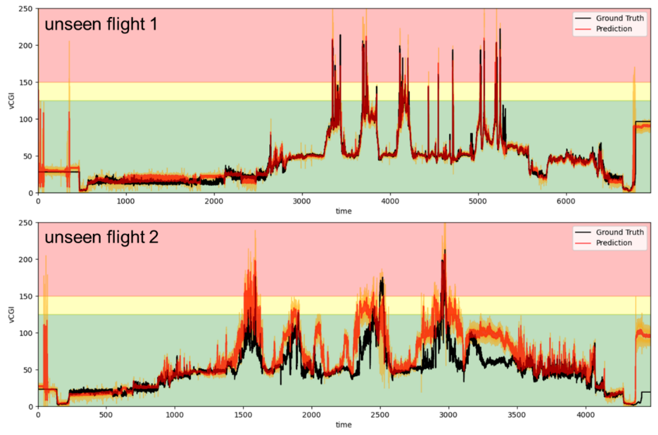

**Figure 11. Predictions on two unseen test flights simulating normal flight (top) and normal flight with autorotation (bottom) made using a LSTM-based neural network with a time sequence of eight, augmented with maximum entropy feature collective variables, and trained on subsampled flight data. The model uncertainty is shown for ± 3 standard deviations in orange.**

### Classification-Regression

The final avenue explored to improve vCGI accuracy was the implementation of a classification-regression MoE approach. Several classifiers were trained on five flights with manually labeled segments. Only about 20% of the flight time was associated with a maneuver label. The initial classifier model was an LSTM network used to perform logistic regression using SoftMax activations. However, because of the small dataset this model was found to overfit the training data. Although the model predicted the correct label for a subset of the labeled flight data with an accuracy of greater than 99%, the model predictions did not coincide with expectations about the dataset. When inferenced on unseen, unlabeled flights this model predicted that 38% of time in the air was spent in dynamic maneuvers, a general label associated with maneuvers like pullups and pushovers, which are short-lived (between a few seconds and 2 minutes in the training set). By contrast, it predicted only 2% of time in the air spent in level flight and 13% in hover, which are expected to be larger portions of these flights. Despite the apparent flaws in the model, it had a nearly 0% prediction rate of unknown class (where the difference between the maximum and minimum probability was not greater than the probability of guessing the correct label at random).

To address these issues, an alternative approach to maneuver classification was developed, as discussed in the Technical Approach. For the same five unseen flights, this classifier predicted that for the same flights the time spent in the air contained 1% unknown maneuvers, 12% dynamic maneuvers, 10% level flight, and 24% hover. Although the dynamic population may still be a bit high, it is more in line with expectations than the direct classification approach, which predicted this to be the most common maneuver label.

Two MoE models were trained using this classifier as a gate to split the flight data into maneuver-specific training sets and determine the expert model weights at inference. The first model used the entirety of the flight data in the maneuver category to train and the second used the same subsampling method as used with the general vCGI model. Both models used the top 20 entropy feature CVs as additional inputs. The MoE models yielded an average yellow/red CGI error of 14.8% with subsampling and 17.2% without. The overprediction ratios were 0.85 with subsampling and 0.65 without. Figure 12 shows the vCGI predictions and the adversarial classifier assigned maneuver label for the two unseen test flights discussed previously. Depending on the dataset used, the MoE models offer a tradeoff between the yellow/red CGI accuracy and overprediction ratio, compared to the general vCGI model. The version with subsampling predicts 75% of the real time spent in CGI yellow/red and the version without predicts 65% of real time for the same. Although these models have greater global accuracy when considering the green CGI regime, they do not perform as well for high-CGI regimes for the test flights evaluated.

To better assess the relative accuracies of the general vCGI and MoE vCGI models, a study was conducted using all 55 test flights with 10 randomly generated splits between 50 training flights and 5 evaluation flights. Both models were trained and evaluated independently, and the results were averaged. This was done to simulate deployment on unseen flight data after training on the entire dataset. From this study, it was found that the general vCGI model correctly predicted when the CGI was in the yellow/red 65.7% of the time and the MoE model predicted this correctly 70.4% of the time; additionally, the variance in this prediction was lower for the MoE model. The general model exhibited a lower overprediction ratio of 0.08 to the 0.10 of the MoE model. Finally, the general vCGI had a higher average yellow/red CGI error of 11% with a standard deviation of 15% for the ensemble. By comparison, the MoE model had an average error of 7.3% with a standard deviation of 3%, while maintaining a green CGI error of 10%. Based on this study, the MoE approach is able to more accurately and more consistently predict the CGI than the best general model presented, when considering randomly selected test flight instead of a hand-picked set of evaluation flights. These results also show the MoE model is able to obtain 93% accuracy predicting the yellow/red CGI and 90% accuracy for green CGI. This computational experiment indicates that the MoE model is better able to leverage the information from the additional flights used as training data and that the previous set of flights used for training were insufficient to yield a model with greater than 90% accuracy. For production deployment, all flights would be used for training, but this test demonstrates that the final MoE model architecture can consistently achieve 90% yellow/red CGI accuracy on unseen flights when trained on at least 50 of the test flights available.

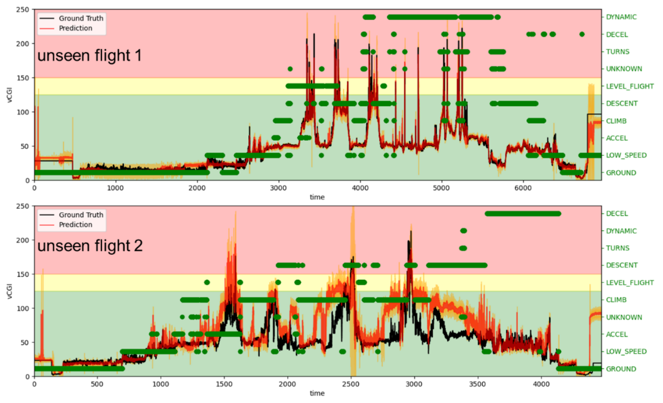

**Figure 12. Predictions on two unseen test flights simulating normal flight (top) and normal flight with autorotation (bottom) made using a mixture of experts approach with an adversarial classifier gate and the model design shown in Figure 10. The model uncertainty is shown for ± 3 standard deviations in orange.**

## CONCLUSIONS

A machine-learned virtual cruise guide indicator was developed based on features extracted from hours of test data recorded over 55 test flights conducted with CH-47 rotorcraft. During the development of the ML vCGI, a series of different neural network architectures were explored using accuracy of the predicted CGI yellow/red conditions and overprediction of CGI green conditions as key performance metrics. Important results and findings include:

1) The raw dataset is suboptimal for the intended goal of accurately predicting conditions which result in component fatigue damage, due to the relatively small population of these events. This is likely to be a common issue in developing machine learning models for engineering applications, particularly those focused on component failure, and must be addressed to advance this technology.
2) A dataset augmentation approach was developed to address the natural imbalance in the flight test data and was employed to improve model generalization. The input feature set was further augmented with temporal collective variables of the input features with the highest

information entropy, as determined by a novel approach presented in this study.

3) Multiple neural network architectures were evaluated for prediction of the vCGI from time-dependent input features. The model utilizing an LSTM encoder for temporal features was found to be the most reliable. This is believed to be the result of the time-dependent nature of the input features and prediction target. The recurrent layer interprets the directionality of time, while the TCNN is bidirectional and the transformer must learn the relationship between each element of the sequence.

4) Finally, an adversarial classification approach was developed to predict the maneuver category of flight segments from limited labeled data. This classifier was implemented in a classification-regression mixture-of-experts approach that achieved 93% accuracy predicting yellow/red CGI while maintaining 90% accuracy on green CGI.

A virtual sensor model, particularly for a component such as the CGI, has the potential to improve fleet management, reliability, and readiness by providing the Army with tools to track usage and loads. The issue of dataset suboptimality was in part the result of usage of existing datasets acquired for other purposes. This work can help inform future data acquisition efforts and provides approaches that can be leveraged during development of machine learning models across other rotorcraft applications. Future efforts aim to improve accuracy and robustness towards post-flight data processing for usage, anomalies and exceedances and shifting focus to AI/ML prediction of loads for component damage tracking.

Author contact: Mathew Boyer mathew.boyer@gdit.com and Wesley Brewer brewerwh@ornl.gov

## ACKNOWLEDGMENTS

This material is based upon work supported by, or in part by, the Department of Defense High Performance Computing Modernization Program (HPCMP) under User Productivity, Enhanced Technology Transfer, and Training (PET) contract # 47QFSA18K0111, Award PIID 47QFSA19F0058. Any opinions, finding and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DoD HPCMP.

## REFERENCES

1. Baskin, J. M., "CH-47C Fixed-System Stall-Flutter Damping," Boeing Vertol Co USAAMRDL-TR- 75-29. 1974.

2. Oldersma, A and Bos, M. J., "Airframe Loads and Usage Monitoring of the CH-47D 'Chinook' Helicopter of the Royal Netherlands Air Force," 26th Symposium of the International Committee on Aeronautical Fatigue (ICAF), Montreal, Canada, June 2011.

3. Isom, J. D., Davis, M. W., Cycon, J. P., Rozak, J., N., and Fletcher, J. W., "Flight Test of Technology for Virtual Monitoring of Loads," American Helicopter Society 69th Annual Forum Proceedings, Phoenix, Arizona, May 2013.

4. Isom, J. D. and Morris, B. E., "Rotor System Health Monitoring using Shaft Load Measurements and Virtual Monitoring of Loads," U.S. Patent 9,240,083, 2016.

5. Martínez D, Brewer W, Strelzoff A, Wilson A, Wade D., "Rotorcraft Virtual Sensors via Deep Regression," *Journal of Parallel and Distributed Computing,* Vol. 135, Jan. 2020, pp. 114-126. DOI: 10.1016/j.jpdc.2019.08.008.

6. Martinez, D., Sitaraman, J., Brewer, W., Rivera, P., and Jude, D., "Machine Learning based Aerodynamic Models for Rotor Blades," Vertical Flight Society (VFS) Transformative Vertical Flight Meeting, San Jose, CA, Jan. 2020.

7. Lim, B., and Zohren, S., "Time-series Forecasting with Deep Learning: a Survey," *Philosophical Transactions of the Royal Society A*, Vol. 379, (2194), Apr. 2021, pp. 20200209. DOI: 10.1098/rsta.2020.0209.

8. Yeo, I. K., and Johnson, R. A., "A New Family of Power Transformations to Improve Normality or Symmetry," *Biometrika*, Vol. 87, (4), Dec. 2000, pp. 954-959. DOI: 10.1093/biomet/87.4.954.

9. Holm, E. A., "In Defense of the Black Box," *Science*, Vol. 364, (6435), Apr. 2019, pp. 26-27. DOI: 10.1126/science.aax0051.

10. Gal, Y., and Ghahramani, Z., "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," International Conference on Machine Learning, New York, NY, June 2016.

11. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., "Dropout: a Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, Vol. 15, (1), Jan. 2014, pp. 1929-1958. DOI: 10.5555/2627435.2670313.

12. O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H. and Invernizzi, L., "Keras Tuner," *https://github.com/keras-team/keras-tuner,* 2019.

13. Yoshioka, K., Lee, E., Wong, S., and Horowitz, M., "Dataset Culling: Towards Efficient Training of Distillation-Based Domain Specific Models," IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, Sept. 2019.

14. Kohonen, T., "The Self-Organizing Map," *Proceedings of the IEEE*, Vol. 78, (9), Sept. 1990, pp. 1464-1480. DOI: 10.1109/5.58325.

15. Kullback, S., and Leibler, R. A., "On Information and Sufficiency," *The Annals of Mathematical Statistics*, Vol. 22, (1), Mar. 1951, pp. 79-86. DOI:

10.1214/aoms/1177729694.

16. Yuksel, S. E., Wilson, J. N., and Gader, P. D., "Twenty Years of Mixture of Experts," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 23, (8), June 2012, pp. 1177-1193. DOI: 10.1109/TNNLS.2012.2200299.

17. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J., "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," 5th International Conference on Learning Representations (ICLR), Toulon, France, Apr. 2017.

18. Sanyal, A., Dokania, P. K., Kanade, V., and Torr, P., "How Benign is Benign Overfitting?," 9th International Conference on Learning Representations (ICLR), Virtual, May, 2021.

19. Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I., "Deep Double Descent: Where Bigger Models and More Data Hurt," *Journal of Statistical Mechanics: Theory and Experiment*, Vol. 2021, Dec. 2021, pp. 124003. DOI: 10.1088/1742-5468/ac3a74.

20. Karabin, M., and Perez, D., "An entropy-maximization approach to automated training set generation for interatomic potentials," *The Journal of Chemical Physics*, Vol. *153* (9), Sept. 2020, pp. 094110. DOI: 10.1063/5.0013059.