Evaluating Image Classification Deep Convolutional Neural Network Architectures for Remaining Useful Life Estimation of Turbofan Engines

Nathaniel DeVol[1], Christopher Saldana[2], and Katherine Fu[3]

[1,2] *George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA*
ndevol3@gatech.edu
christopher.saldana@me.gatech.edu

[3] *Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, WI, 53706, USA*
kate.fu@wisc.edu

ABSTRACT

Accurate estimation of the remaining useful life (RUL) is a key component of condition-based maintenance (CBM) and prognosis and health management (PHM). Data-based models for the estimation of RUL are of particular interest because expert knowledge of systems is not always available, and physical modeling is often not feasible. Additionally, using data-based models, which make decisions based on raw sensor data, allow features to be learned instead of manually determined. In this work, deep convolutional neural network (CNN) architectures are investigated for their ability to estimate the RUL of turbofan engines. To improve the accuracy of the models, CNN architectures, which have proven successful in image classification, are implemented and tested. Specifically, the blocks used in the Visual Geometry Group (VGG) architecture, inception modules used in the GoogLeNet architecture, and residual blocks used in the ResNet architecture are incorporated. To account for varying flight lengths, the input to the models is a window of time series data collected from the engine under test. Window locations at the climb, cruise, and descent stages are considered. To further improve the RUL estimations, multiple overlapping windows at each location are used. This increases the amount of training data available and is found to increase the accuracy of the resulting RUL estimations by averaging the estimates from all overlapping segments. The model is trained and tested using the new Commercial Modular Aero-Propulsion System Simulation (N-CMAPSS) data set, and high prognosis accuracy was achieved. This work expands on the model developed and used in the 2021 PHM Society Data Challenge, which received second place.

https://doi.org/10.36001/IJPHM.2022.v13i2.3284

1. INTRODUCTION

Remaining useful life (RUL) is the remaining time for which an asset will be productive (Si, Wang, Hu, & Zhou, 2011), and the RUL is a key component of condition-based maintenance (CBM) and prognosis and health management (PHM) (Jardine, Lin, & Banjevic, 2006). Knowing the RUL of an asset allows for the optimal scheduling of maintenance and ordering of spare parts, which can reduce asset downtime and increase profitability. RUL estimation also has safety and environmental implications by preventing failures that could put users in danger and extending the life of an asset, thereby reducing the need for new equipment (Si et al., 2011).

While the RUL of an asset is affected by many variables and cannot be exactly known, there are several modeling techniques for estimating the RUL. The three primary approaches for estimating the RUL of an asset are: physics-based, data-based, and hybrid (Sikorska, Hodkiewicz, & Ma, 2011). Physics-based methods create a model of the system using an in-depth understanding of the underlying processes. While physics-based models have been shown to be successful (Bolander, Qiu, Eklund, Hindle, & Rosenfeld, 2009), they are often prohibitive due to the time and system understanding required to create them. Additionally, physics-based models tend to be specific to a failure mode, and failure modes must be well understood *a priori* (Sikorska et al., 2011). Data-based methods, including statistical and artificial intelligence (AI) methods, do not require knowledge of the underlying systems, but instead rely on the availability of a data set that captures the performance of the system. Finally, hybrid models merge these two approaches to reduce the underlying knowledge and data requirements of the two individual methods (Sikorska et al., 2011). While hybrid approaches work in some applications (Kong et al., 2020), they increase the modeling complexity by requiring the two

models to be developed and merged.

In this work, modeling methodologies for estimating the RUL of turbofan engines are investigated. This was originally completed as part of the 2021 PHM Society Data Challenge, which used the new Commercial Modular Aero-Propulsion System Simulation (N-CMAPSS) data set (Chao, Kulkarni, Goebel, & Fink, 2021). This data set contains a large amount of run-to-failure data, so this work takes a data-based approach. Past research on the original CMAPSS data set demonstrated the applicability of convolutional neural networks (CNNs) (Li, Ding, & Sun, 2018; H. Yang, Zhao, Jiang, Sun, & Mei, 2019), long short-term memory (LSTM) (da Costa, Akeay, Zhang, & Kaymak, 2019; Zheng, Ristovski, Farahat, & Gupta, 2017; Wu et al., 2020), and hybrid methods, which merge CNN and LSTM (Zhao, Huang, Li, & Iqbal, 2020) for RUL estimation. The use of convolutions allows for a reduction of the number of model parameters relative to a neural network without convolutions and aids in the extraction of features (Albawi, Mohammed, & Al-Zawi, 2017). Additionally, convolution layers have the ability to realize complex relationships between different sensor readings. Specifically, this work focuses on applying CNN architectures developed for image classification to RUL estimation. Image classification architectures are of interest because if they can be leveraged for RUL estimation, then they have the potential to quickly advance the field of prognostics. Additionally, the theoretical justifications for using CNN on images, including the ability for filters to be learned and applied across an entire image, also apply to sensor data.

The rest of the paper is organized as follows. In section 2, the problem is introduced and the data set is described. In section 3, the methodology is detailed, including the data preprocessing and model architecture. Section 4 details the results of each trained model. Finally, sections 5 and 6, contain discussion and conclusions.

2. PROBLEM DESCRIPTION

The work was initiated as part of the 2021 PHM Society Data Challenge. The challenge description is thus the same as the problem description presented in this section. This work builds on the result of the Data Challenge (DeVol, Saldana, Fu, & Woodruff, 2021) by including results from multiple model architectures and various model inputs including the flight stage and incorporating multiple windowed inputs.

2.1. Data Set Description

In this work, the new Commercial Modular Aero-Propulsion System Simulation (N-CMAPSS) data set is used to test a data-driven method for RUL estimation. This data set contains realistic run-to-failure data of turbofan engines (Chao et al., 2021). The N-CMAPSS data set offers higher fidelity data than the original CMAPSS data set by incorporating real recorded flight conditions and relating the degradation process to its operation history to extend the degradation model (Chao et al., 2021).

The data set models the failure trajectories of eight different failure modes that affect either the efficiency or flow of one or more of the sub-components of the engine. One of the failure modes included in the N-CMAPSS data set was not included in the 2021 PHM Society Data Challenge, so it was not used in the development of the tested models. The seven failure modes used in the data challenge are shown in Table 1. The failure modes span across all the rotating sub-components: fan, low-pressure compressor (LPC), high-pressure compressor (HPC), high-pressure turbine (HPT), and low-pressure turbine (LPT) and can affect either efficiency (E) or flow (F).

The flight durations are divided into three classes based on their length, but this work considers them together by using a window function, which is discussed in section 3. Each flight has an unknown initial condition and is run to failure, so the number of flights varies for each unit. Across all failure modes, the data set contained 90 units with data from a combined 6,825 flights. The data file for each flight contains scenario descriptors (w), measurements (x_s), an RUL label (y), and auxiliary data. The variables contained within the scenario descriptors, measurements, and auxiliary data are detailed in Tables 2, 3, and 4, respectively.

2.2. Problem Definition

The goal of this work is to develop a model that estimates the RUL of a turbofan engine given a data set $\mathcal{D} = \{w_i, x_{s_i}, y_i\}_{i=1}^{N}$, which contains run-to-failure data for N total flights of turbofan engines subject to different failure modes. The length of the sensory signals w and x_s is not constant between flights, so the model should be able to incorporate variable lengths of input data. The performance of the model is evaluated using a combination of the root-mean-square error (RMSE) and NASA's scoring function (Saxena, Goebel, Simon, & Eklund, 2008), calculated from the actual (y) and predicted ($\hat{y}$) RUL values. RMSE is calculated following Eq. (1), where m_{v*} is the number of test samples. NASA's scoring function (s_c) is calculated using Eq. (2), where α is defined in Eq. (3). With α defined this way, under-estimations are preferable to over-estimations. The values calculated in Eqs. (1) and (2) are combined using Eq. (4) to get the final score. The values of each of these equations as a function of the difference between the actual and predicted RUL values is shown in Figure 1.

$$RMSE = \sqrt{\frac{1}{m_{v*}} \sum_{j=1}^{m_{v*}} (y^{(j)} - \hat{y}^{(j)})^2} \tag{1}$$

Table 1. Overview of the data set.

Name	# Units	Fan		LPC		HPC		HPT		LPT	
		E	F	E	F	E	F	E	F	E	F
DS01	10							✓			
DS03	15							✓		✓	✓
DS04	10	✓	✓								
DS05	10					✓	✓				
DS06	10			✓	✓	✓	✓				
DS07	10									✓	✓
DS08	25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2. Scenario descriptors (w)

Symbol	Description	Units
alt	Altitude	Units
Mach	Flight Mach number	-
TRA	Throttle-resolver angle	%
T2	Total temperature at fan inlet	°R

Table 3. Measurements (x_s)

Symbol	Description	Units
Wf	Fuel flow	pps
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T48	Total temperature at HPT outlet	°R
T50	Total temperature at LPT outlet	°R
P15	Total pressure in bypass-duct	psia
P2	Total pressure at fan inlet	psia
P21	Total pressure at fan outlet	psia
P24	Total pressure at LPC outlet	psia
Ps30	Total pressure at HPC outlet	psia
P40	Total pressure at burner outlet	psia
P50	Total pressure at LPT outlet	psia

$$s_c = \frac{1}{m_{v*}} \sum_{j=1}^{m_{v*}} exp(\alpha * |y^{(j)} - \hat{y}^{(j)}|) \tag{2}$$

$$\alpha = \begin{cases} \frac{1}{13} & \text{if} \quad y^{(j)} - \hat{y}^{(j)} \leq 0 \\ \frac{1}{10} & \text{if} \quad y^{(j)} - \hat{y}^{(j)} > 0 \end{cases} \tag{3}$$

$$score = 0.5 * RMSE + 0.5 * s_c \tag{4}$$

Table 4. Auxiliary data

Symbol	Description	Units
unit	Unit number	-
cycle	Flight cycle number	-
Fc	Flight class	-
h_s	Health state	-

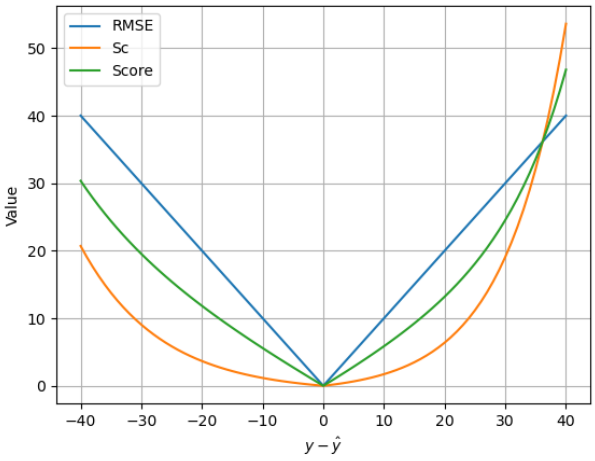

Figure 1. Evaluation scores as a function of the difference between the actual and predicted RUL values.

3. METHODOLOGY

3.1. Data Pre-Processing

In the data pre-processing phase, the scenario descriptors are combined with the measurements. This is done because the scenario descriptors can provide context to the measurement readings. For the time series data of a flight, which contains m values, the scenario descriptors $w \in \mathbb{R}^{m \times 4}$ and measurements $x_s \in \mathbb{R}^{m \times 14}$ are combined to form $x \in \mathbb{R}^{m \times 18}$.

The flight duration is not consistent across each flight, so the length of each x_s and w varies. To account for this, a windowing function is used, which crops the measurement data and scenario descriptors to a specified length. Multiple loca-

tions for this window were considered, including the climb (beginning), cruise (middle), and descent (end) portions of the flight. After applying the window function, the length of the input data is consistent, and the number of model parameters is reduced compared to a model that considers the whole flight. Three window sizes of 30, 60, and 120 samples are considered. In addition to considering a single window of data from each flight, three consecutive windows with 50% overlap were also considered. This triples the amount of training data by creating three training points from each flight, which are each associated with the same RUL. Three windows with 50% overlap were considered because the data within each windowed segment is expected to be similar since the outside segments are adjacent and share half of their data with the middle section. This allows each of the three windowed segments to be treated as separate training samples with the same labels which would not be possible if more dispersed windows were used.

Finally, the input data is standardized so that it has zero mean and unit standard deviation. This is done independently for each of the 18 features. The mean and standard deviation for each feature are calculated from the training data, then applied to both the train, validation, and test data to calculate the standardized input by

$$X_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \tag{5}$$

where μ_j and σ_j are the mean and standard deviation of the j^{th} feature, $x_{i,j}$ is the i^{th} value of the j^{th} feature, and $X_{i,j}$ is the final standardized value input to the model.

3.2. Model Architectures

In this work, multiple deep convolutional neural network (CNN) architectures are investigated for their ability to estimate the RUL of turbofan engines. Three primary architectures that have been shown to be successful in classifying images were evaluated. The first uses blocks of CNN layers similar to the Visual Geometry Group (VGG) architecture (Simonyan & Zisserman, 2014). The second makes use of inception modules introduced in the GoogLeNet architecture (Szegeandy et al., 2015). The final architecture uses residual blocks based on the ResNet architecture (He, Zhang, Ren, & Sun, 2016). All of these models were originally developed for image classification, so the convolutional layers are two dimensional. To adapt these architectures for use on the time series data from the N-CMAPSS data set, one-dimensional convolutional layers are used. The convolutions take place along the time axis.

3.2.1. VGG Architecture

The first architecture tested makes use of blocks of CNN layers similar to the VGG architecture. In the VGG architecture, only small (3x3) convolution filters are used, and each block of convolution layers is followed by a max pooling layer, where the input is down-sampled by taking the maximum over a 2x2 window with a stride of 2 (Simonyan & Zisserman, 2014). An example of an architecture making use of this configuration and adapted for the time series data in the N-CMAPSS data set is shown in Table 5. To adapt for the N-CMAPSS data set, the architecture uses one-dimensional convolutions, as opposed to the original two-dimensional convolutions, so that the input can be a two-dimensional matrix of columns of sensor data instead of an image.

Table 5. Example VGG architecture adapted to time series data. The convolutional layer parameters are denoted as "Conv⟨receptive field size⟩-⟨number of channels⟩".

Input (30x18 raw sensor data)
Conv3-64
Conv3-64
Max pool
Conv3-128
Conv3-128
Max pool
Fully connected-128
Linear activation

3.2.2. Inception Architecture

The next model architecture makes use of inception modules. Inception modules contain parallel convolutional layers of different sizes and a single max pool layer. The outputs of each of these parallel layers are concatenated and sent to the next layer. The original inception network used two dimensional convolutions of sizes 1x1, 3x3, and 5x5 and a 3x3 max pool layer (Szegeandy et al., 2015). The use of variable-sized convolutions allows for variable-sized features to be detected within a shallower neural network. This version of the inception module is shown in Figure 2a. To reduce the dimensionality of the model, convolutions of size 1x1 can be incorporated before each convolution layer and after the max pool layer. These convolutions are performed with a stride of one so that the first and second dimension remain unchanged, but the third dimension is reduced. An inception module that incorporates dimensionality reduction is shown in Figure 2b. An example of a model architecture making use of these inception modules and adapted for the time series data in the N-CMAPSS data set is shown in Figure 3.

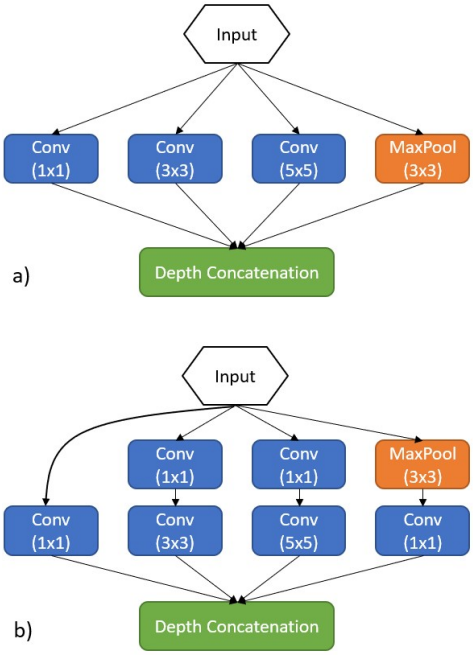

Figure 2. Inception module a) without dimensionality reduction and b) with it. Adapted from (Szegeandy et al., 2015).

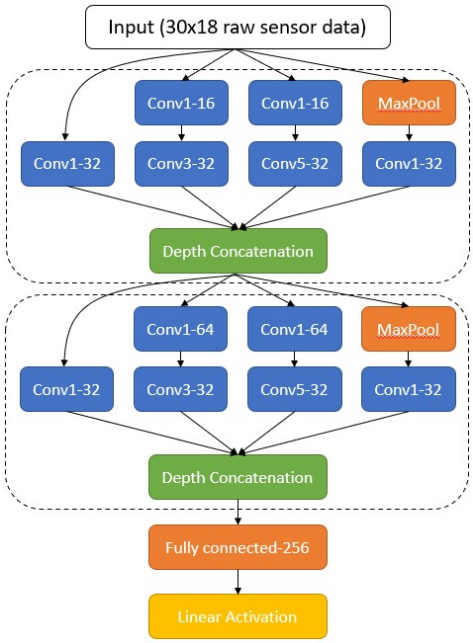

Figure 3. Example inception architecture adapted to time series data. The convolutional layer parameters are denoted as "Conv⟨receptive field size⟩-⟨number of channels⟩".

3.2.3. Residual Network Architecture

The final network architecture explored in this work makes use of residual blocks. The residual blocks use a skip connection, which adds the input of a stack of two convolution layers to the output of those two layers. These shortcut connections, which do not add model parameters, are able to increase the accuracy of deep neural networks (He et al., 2016). An illustration of a residual block is shown in Figure 4.

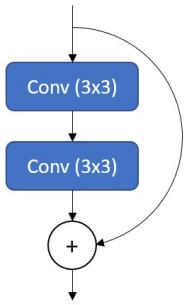

Figure 4. Building block of residual network architecture.

When using the simple implementation shown in Figure 4, errors can occur when the output of the two convolution layers is not the same as the input. To correct for this, the shortcut can be padded with zeros, or a 1x1 convolution layer can be used to match the shortcut to the output of the convolution layers (He et al., 2016). An example of a model architecture making use of residual blocks and adapted for the time series data in the N-CMAPSS data set is shown in Figure 5. This model uses single dimensional convolutions on the shortcut connection to correct for mismatches in the number of filters.

3.3. Model Training

Across all the failure types and tested units, flight data from 6,825 flights were used for training and testing. Within the N-CMAPSS data set, this data is split with into development and test sets with data from 4,089 flights (60%) in the development set and data from 2,736 flights (40%) in the test set. In this work, the development data was further split using 5-fold cross validation. The average validation accuracy across the five folds was used to tune the hyperparameters for each model architecture, and all validation scores reported in this work are the average across the five folds. After hyperparameter tuning, the test data was used on the final trained model from each network architecture. In the instances where three overlapping windows are used, the amount of training data is increased threefold, but the amount of test data remains the same. This is because in training, the three windows are treated independently as three separate samples each having the same RUL label. In validation and testing, the average RUL estimate over the three windows is used as the final prediction for the unit under test.

The models were implemented in Python (version 3.8.12) using the Keras (Chollet et al., 2015) library (version 2.4.3) and Tensorflow (Abadi et al., 2015) backend (version 2.3.0). In training, the accuracy of the model was evaluated by taking the mean squared error (MSE) between the predicted and ac-

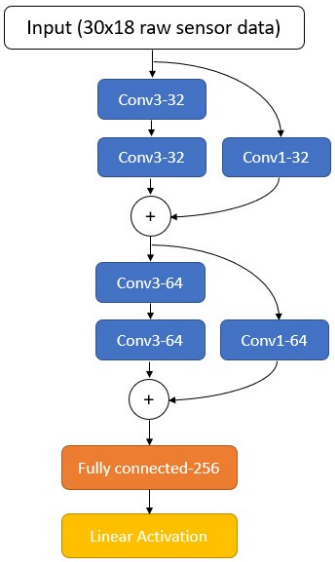

Figure 5. Example residual network architecture adapted to time series data. The convolutional layer parameters are denoted as "Conv⟨receptive field size⟩-⟨number of channels⟩".

tual remaining useful life of all the samples. MSE was used in training for ease of implementation, but score values as described in Eq. (4) were computed on the validation and test data sets. During hyperparameter tuning, training was conducted for 40 epochs, and once the final architecture was set, the final models were trained for 60 epochs. Hyperparameters included the number of layers, number of nodes in each layer, batch size, and drop-out rate were set using a grid-search (L. Yang & Shami, 2020). For the model architectures, the search space started with minimal models with few layers and nodes and the complexity was increased until overfitting was observed. The search space included batch sizes of 32, 64, and 128 and dropout rates of 40%, 50%, and 60%. Following past research, a dropout of 50% was used as the midpoint for the dropout rate search space (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Ultimately, a batch size of 64 and dropout rate of 50% were used for all models. For both training and testing, the Adam optimizer and a learning rate of 0.001 were used. The Adam optimizer was used because of its recent broad adoption in deep learning (Kingma & Ba, 2014; Ruder, 2016). Because the Adam optimizer uses an adaptive learning-rate, the default initial learning rate of 0.001 can be used without tuning (Ruder, 2016).

4. RESULTS

4.1. VGG Architecture

In the development of the VGG-based model, network configurations with different numbers of convolution layers and filters were tested. The details of the best performing architecture, based on the validation score, is given in Table 6. After the last convolution layer, the filters are flattened and fully connected to a hidden layer with 128 cells. Dropout at a rate of 50% is included in this layer to reduce overfitting. In the last layer, a single node with a linear activation outputs the RUL estimation of the flight data under investigation. The model performed best when trained with 3 overlapping segments of 60 sensor readings from the climb stage of the flight. The training and validation scores of this architecture, trained with different input configurations, are compared in Table 7. The flight stages were explored equally, but more results for the climb stage are shown in Table 7 because models trained with data from the climb stage produced lower scores. This trend continued for all model architectures investigated. In total, this model has 286,593 trainable parameters, and the breakdown of these parameters by layer is given in Table 6. A parity plot of the final VGG model retrained with all of the training data is shown in Figure 6.

Table 6. Best performing VGG-based architecture. Where not specified, all layers use the ReLU activation function.

Input (60x18 raw sensor data)	# Params
Conv3-64	3,520
Conv3-64	12,352
Max pool	-
Conv3-64	12,352
Conv3-64	12,352
Max pool	-
Fully connected-128	245,888
Linear activation	129

Table 7. VGG-based architecture scores when trained with different inputs.

Model Input	Flight Stage	Overlap Used	Train Score	Validation Score
60x18	**Climb**	**Yes**	**5.43**	**5.87**
60x18	Climb	No	6.32	6.67
30x18	Climb	Yes	5.89	6.27
120x18	Climb	Yes	5.53	5.98
60x18	Cruise	Yes	7.03	7.64
60x18	Descent	Yes	6.84	7.24

4.2. Inception Architecture

In testing the inception architecture, the number of inception modules and fully connected layers, as well as the numbers of filters and nodes in each were varied. The details of the architecture that yielded the best score on the validation data is shown in Table 8. Three inception modules are used in the network. In the first module, the dimensionality reductions before the convolution layers are not included so that

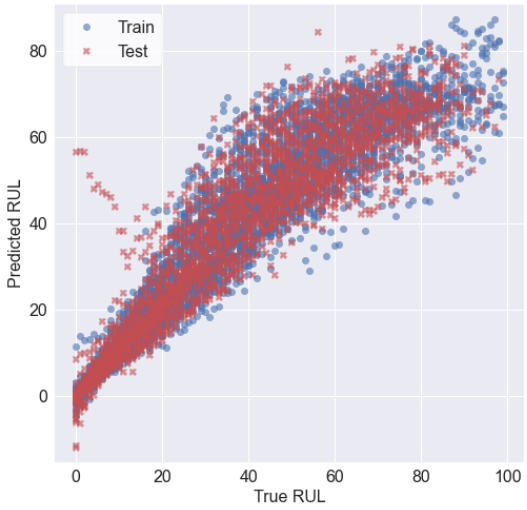

Figure 6. Parity plot for the VGG-based model.

the first convolutions see the raw signal. This is done because the input data is relatively small with 18 sensor readings, so the benefits of dimensionality reduction are minimal. In the second and third modules, the dimensionality reduction is included because the stacked filters from the first inception module expand the dimensionality. After the last inception module, the convolution filters are flattened and fully connected to a hidden layer with 256 cells. Dropout at a rate of 50% is included in this layer to reduce overfitting. Finally, in the last layer, a single node with a linear activation outputs the RUL estimation of the flight data under investigation. The training and validation scores of this architecture, trained with different input configurations, are compared in Table 10. In total, the model has 2.05 million trainable parameters. The breakdown of these parameters by layer is shown in Table 9. A parity plot of the final inception model retrained with all of the training data is shown in Figure 7.

4.3. Residual Network Architecture

To test the performance of residual architecture, models with varying numbers of residual blocks and fully connected layers, and varied numbers of filters and nodes in each were trained. The details of the architecture that yielded the best score on the validation data are shown in Table 11. This model was comprised of two residual blocks after which the convolution filters are flattened and fully connected to a hidden layer with 128 cells. Dropout at a rate of 50% is included in this layer to reduce overfitting. Finally, in the last layer, a single node with a linear activation outputs the RUL estimation of the flight data under investigation. The training and validation scores of this architecture, trained with different input configurations, are compared in Table 12. In total, this model contained 1.03 M trainable parameters when setup for a 60x18 input. Unlike the other architectures tested, for the residual-based architectures the best performance was for the shorter, 30 sample input as opposed to the 60. The improvement, however, was minimal so the 60 sample input was used for easy comparison back to the other models. A parity plot of the final residual-based model retrained with all of the training data is shown in Figure 8.

Table 8. Best performing inception-based architecture. Where not specified, all layers use the ReLU activation function.

Input (60x18 raw sensor data)			
Conv1-18	Conv3-36	Conv5-36	Max Pool Conv1-18
Depth Concatenation			
Conv1-32	Conv1-64 Conv3-32	Conv1-64 Conv5-32	Max Pool Conv1-32
Depth Concatenation			
Conv1-32	Conv1-64 Conv3-32	Conv1-64 Conv5-32	Max Pool Conv1-32
Depth Concatenation			
Fully connected-256			
Linear activation			

Table 9. Number of parameters broken down by layer for the best performing inception-based model.

Layer	# Params
Inception1	5,940
Inception2	37.4 k
Inception2	41.2 k
Fully Connected	1.97 M
Linear	257
Total	2.05 M

4.4. Comparison

The train and test scores for the final model from each category are shown in Table 13. The means from 10-fold cross validation are reported with their standard deviation. The train score is from the final model trained on all training data, and the test score is calculated on the previously unused test set. A one-way ANOVA was performed on the cross-validation results and found that the differences between the models were not significant ($F(2,27)=0.957$ $p=0.40$). The ratio of the standard deviations was 1.4, and the Q-Q plot used to verify normal distributions is shown in Figure 9. It is worth noting that the overlapping training sets of 10-fold cross validation may exhibit increased type I error when used for statistical tests (Dietterich, 1998). In addition to each

Table 10. Inception-based architecture scores when trained with different inputs.

Model Input	Flight Stage	Overlap Used	Train Score	Validation Score
60x18	**Climb**	**Yes**	**5.32**	**5.97**
60x18	Climb	No	5.72	6.40
30x18	Climb	Yes	5.60	6.15
120x18	Climb	Yes	5.19	5.90
60x18	Cruise	Yes	6.82	7.47
60x18	Descent	Yes	6.45	6.95

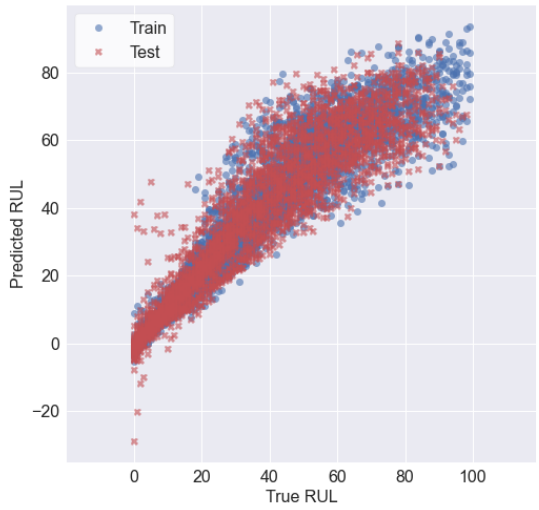

Figure 7. Parity plot for the inception-based model.

Table 11. Best performing residual-based architecture. Where not specified, all layers use the ReLU activation function.

Input (60x18 raw sensor data)	# Params
ResidualModule-64	17,088
ResidualModule-64	24,704
Fully connected-256	983,296
Linear activation	257

model's accuracy, it is important to compare the model size, training time, and inference time. These values are reported in Table 14. The reported inference times are per estimation and are computed by taking the total time to predict all test data points and dividing by the total number of test points. Training was completed on an Intel Core i7 CPU with 32 GB of RAM.

Table 12. Residual-based architecture scores when trained with different inputs.

Model Input	Flight Stage	Overlap Used	Train Score	Validation Score
60x18	Climb	Yes	5.81	6.24
60x18	Climb	No	6.65	7.25
30x18	**Climb**	**Yes**	**5.81**	**6.21**
120x18	Climb	Yes	5.98	6.41
60x18	Cruise	Yes	7.90	8.48
60x18	Descent	Yes	6.96	7.41

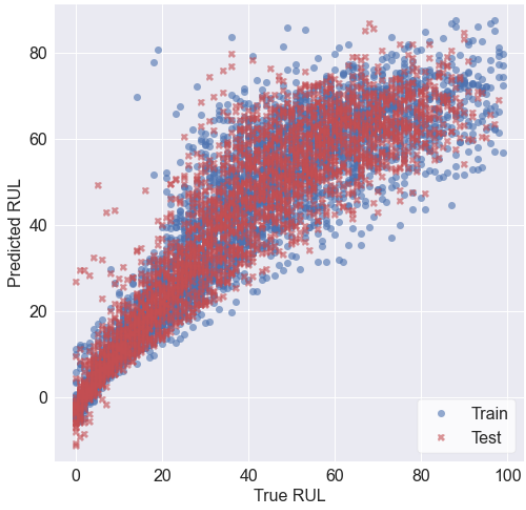

Figure 8. Parity plot for the residual-based model.

5. DISCUSSION

The results of this work presented in Table 13 demonstrate that all of the CNN architectures were able to predict the RUL with similar accuracy. The inception architecture offered the greatest accuracy, but the increase in its performance compared to the simpler VGG was minimal, considering the increase in trainable parameters and training time. This indicates that a very deep model is not required for RUL estimation, so the benefits that the residual- and inception-based architectures offer are not realized. The applicability of CNNs on the N-CMAPSS data set, demonstrated in this work, and their past success on the original CMAPSS data set (Sateesh Babu, Zhao, & Li, 2016; Li et al., 2018; H. Yang et al., 2019), show the broad applicability that they have in RUL estimation.

To determine if the differences in the performance of the models when using the climb portion was significant, a one-way ANOVA was performed on the model scored obtained

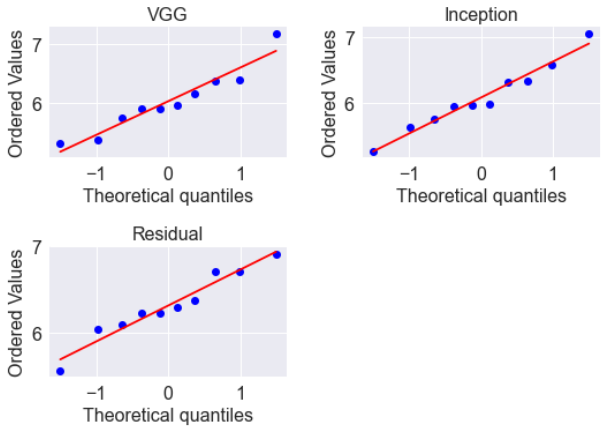

Figure 9. Q-Q plot for comparing the performance of various model architectures.

Table 13. Comparison of the best performing architecture from each category.

Model Architecture	10-Fold CV Score	Train Score	Test Score
VGG	6.03 (SD=0.54)	5.09	5.78
Inception	6.10 (SD=0.51)	4.77	5.64
Residual	7.26 (SD=0.50)	6.40	6.14

Table 14. Comparison number of trainable parameters, training time, and inference time.

Model Architecture	Trainable Parameters	Train Time	Inference Time
VGG	164 k	154 s	7e-5 s
Inception	2.05 M	740 s	2e-4 s
Residual	1.03 M	268 s	1e-4 s

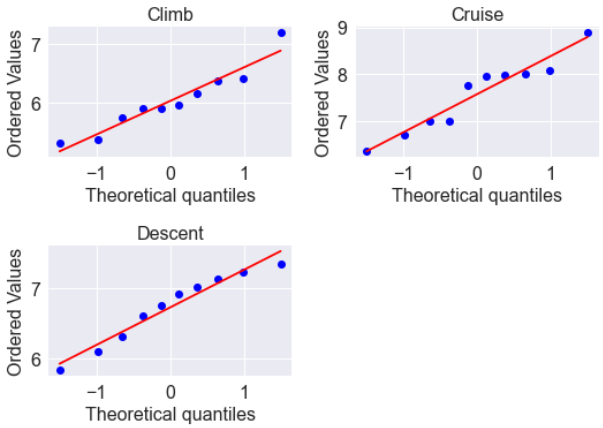

Figure 10. Q-Q plot for comparing the performance of various flight regions when using the VGG model architecture.

in 10-fold cross validation. The means and standard deviations from each case are reported in Table 15. The result was that the flight stage was found to be a significant factor for all model architectures (VGG F(2,27)=15.7 p=3.0e-5, Inception F(2,27)=16.9 p=1.7e-5, and Residual F(2,27)=23.9 p=1.0e-6). The ratio of the standard deviations for the VGG, Inception, and Residual architectures were 1.5, 1.3, and 1.3 respectively. Q-Q plots used to verify normal distributions are shown in Figure 10-12.

Tukey's HSD Test for multiple comparisons found that for the VGG architecture, the model difference was significant when comparing all flight stages. For the Inception and Residual architectures, the climb stage was significantly different from the cruise and descent, but the cruise and descent difference was not statistically significant. The p-values for the Tukey test are shown in Table 16.

From the parity plots in Figures 6 - 8 it is clear that all three models performed better as the RUL of the engine under investigation decreased. To understand this change, the scores of each model for true RUL values below 40 and above 40 cycles were computed separately. The results of the models using data with true RUL under 40 cycles are shown in Table 17 and true RUL above 40 cycles are shown in Table 18. Splitting the 4,089 training data in this manner left 2,214 below an RUL of 40 cycles, and 1,875 above an RUL of 40 cycles. Splitting the 2,736 test data points left 1,476 below an RUL of 40 units and 1,260 data points above an RUL of 40 units. Note that the models were not retrained with this split data, only reevaluated. Tables 17 and 18 highlight the increase in model accuracy as the true RUL decreases. This is not surprising, as one may expect it to become easier to estimate the RUL as the signs of wear become more obvious. With this trend in mind, future modeling efforts may find it worthwhile to focus efforts on making accurate estimation only after an engine has been used for a certain number of cycles. This has the potential to increase the model accuracy as the engine nears the end of its life, where an accurate RUL estimate becomes more valuable.

It is worth noting that the parity plots for each model show negative estimated RUL, which does not make practical sense. This would not occur if the model output used a rectified linear unit (ReLU). A ReLU is linear for inputs greater than zero and zero otherwise (Agarap, 2018). Training the models with a ReLU activation on the output node is not practical, so instead the model is trained with a linear activation function. The parity plots show the negative values to illustrate the raw model outputs.

The accuracy of the CNN developed here demonstrates that the information to make an accurate RUL prediction is contained within a single flight's measurements. This was demonstrated in previous work on the original CMAPSS data set (Sateesh Babu et al., 2016; Li et al., 2018; H. Yang et

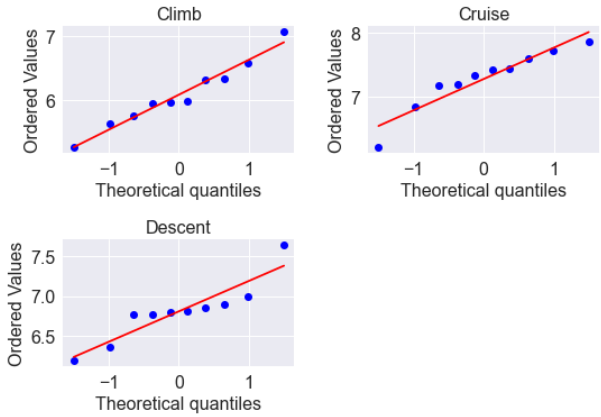

Figure 11. Q-Q plot for comparing the performance of various flight regions when using the Inception model architecture.

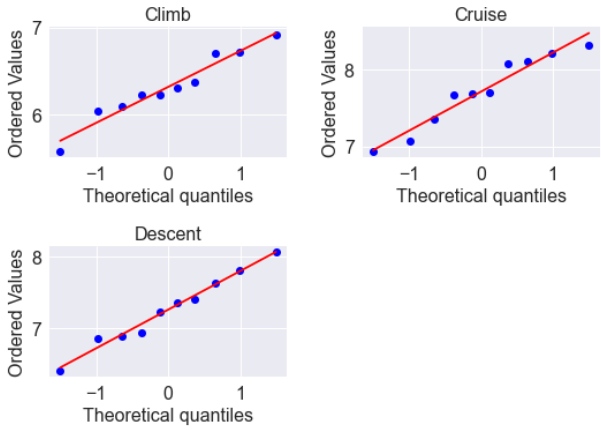

Figure 12. Q-Q plot for comparing the performance of various flight regions when using the Residual model architecture.

al., 2019), and its success here on the N-CMAPSS further validates its applicability. While accurate RUL estimations can be made using data from a single flight, the use of historical data may improve the accuracy of the RUL estimations. Future work should investigate incorporating historical data using recurrent architectures, such as LSTM.

One of the limitations of this work is that RUL estimations are not explainable, or the model is not able to explain why an RUL estimation was made. In future work, this could be improved upon by incorporating the ability of the model to predict what component is causing the RUL to be reduced. Since the N-CMAPSS data set contains multiple failure modes across all components, and the failure mode of each unit is labeled, future work on the N-CMAPSS data set should consider labeling the failing component, in addition to estimating the RUL. Another limitation with this approach is that run-to-failure data from a variety of failure modes is needed. In cases where this is not applicable, methods that make RUL estimations from only early cycle data, such as that demonstrated in (Coble & Hines, 2011), should be considered instead. Finally, the techniques used in this work to improve model accuracy, such as averaging the result of overlapping windows, cannot be guaranteed to extend to other systems. Future work should consider the conditions for applying these techniques to other systems.

Table 15. 10-Fold cross validation results for the influence of the flight stage on the score of different model architectures.

Model Architecture	Climb	Cruise	Descent
VGG	6.03 (SD=0.54)	7.57 (SD=0.77)	6.73 (SD=0.50)
Inception	6.10 (SD=0.51)	7.28 (SD=0.48)	6.81 (SD=0.38)
Residual	6.31 (SD=0.39)	7.71 (SD=0.48)	7.26 (SD=0.50)

Table 16. Tukey's HSD test p-values when comparing the significance of the flight stage on the score of different model architectures. Each column represents a test.

Group1	Group2	VGG	Inception	Residual
Climb	Cruise	0.001	0.001	0.001
Climb	Descent	0.046	0.0046	0.001
Cruise	Descent	0.013	0.074	0.087

6. CONCLUSION

This work demonstrated the applicability of adapting popular image-based, two-dimensional CNN for the use of estimating RUL from one-dimensional time series data. The residual- and inception-based architectures were able to increase the number of trainable parameters without overfitting, but only the inception architecture showed an improvement, although small, over the VGG-based model. Adapting these architectures for time series data allows for RUL estimation to take advantage of the recent advances in CNN architectures that have been driven by image classification. The performance of the residual- and inception-based architectures relative to the VGG-based model indicate the complexity of image classification architectures has surpassed what is necessary for RUL estimation of turbofan engines and thus further advances to image classification architectures may not benefit RUL estimation.

This work also investigated which portion of the flight (climb, cruise, or descent) contained the most information for RUL estimation. For all model architectures investigated, the climb portion was found to be better than the cruise or descent portions. Additionally, the use of overlapping segments was in-

Table 17. Comparison of network architectures when the true RUL of an engine is below 40 cycles.

Model Architecture	10-Fold CV Score	Train Score	Test Score
VGG	4.38 (SD=0.31)	3.91	5.10
Inception	4.61 (SD=0.52)	3.55	4.70
Residual	4.74 (SD=0.50)	5.55	5.62

Table 18. Comparison of network architectures when the true RUL of an engine is above 40 cycles.

Model Architecture	10-Fold CV Score	Train Score	Test Score
VGG	7.47 (SD=0.55)	6.24	6.45
Inception	7.58 (SD=0.62)	5.93	6.59
Residual	7.63 (SD=0.37)	7.25	6.69

vestigated. Using overlapping segments increased the amount of training data available, and the final estimate of an engine's RUL was determined by averaging the model output for each of its inputs. This approach was found to increase the average accuracy for all tested models and presents a potential solution that other researchers can use when working with windowed segments of time series data.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Department of Energy DE-EE0008303 and the U.S. Department of Defense, Office of Local Defense Community Cooperation, Industry Resilience Program, Award #ST1449-21-03 2016-2166.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Agarap, A. F. (2018). *Deep learning using rectified linear units (relu).* arXiv.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, 1–6. doi: 10.1109/ICEngTechnol.2017.8308186

Bolander, N., Qiu, H., Eklund, N., Hindle, E., & Rosenfeld, T. (2009). Physics-based remaining useful life prediction for aircraft engine bearing prognosis. *Annual Conference of the PHM Society*, *1*(1), 1–10.

Chao, M. A., Kulkarni, C., Goebel, K., & Fink, O. (2021). Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics. *NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA*. doi: 10.3390/data6010005

Chollet, F., et al. (2015). *Keras.* `https://keras.io`.

Coble, J., & Hines, J. W. (2011). Applying the general path model to estimation of remaining useful life. *International Journal of Prognostics and Health Management*, *2*(1), 71–82.

da Costa, P. R. d. O., Akeay, A., Zhang, Y., & Kaymak, U. (2019). Attention and Long Short-Term Memory Network for Remaining Useful Lifetime Predictions of Turbofan Engine Degradation. *International Journal of PHM Society*, *10*(4), 1–12. doi: 10.1115/GTINDIA2019-2368

DeVol, N., Saldana, C., Fu, K., & Woodruff, G. W. (2021). Inception Based Deep Convolutional Neural Network for Remaining Useful Life Estimation of Turbofan Engines. , *1*.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, *10*(7), 1895–1923.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr).*

Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*(7), 1483–1510. doi: 10.1016/j.ymssp.2005.09.012

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv. doi: 10.48550/ARXIV.1412.6980

Kong, H. B., Jo, S. H., Jung, J. H., Ha, J. M., Shin, Y. C., Yoon, H., ... Jeon, B. C. (2020). A hybrid approach of data-driven and physics-based methods for estimation and prediction of fatigue crack growth. *International Journal of Prognostics and Health Management*, *11*, 1–12. doi: 10.36001/ijphm.2020.v11i1.2605

Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, *172*, 1–11. doi: 10.1016/j.ress.2017.11.021

Ruder, S. (2016). *An overview of gradient descent optimization algorithms.* arXiv. doi: 10.48550/ARXIV.1609.04747

Sateesh Babu, G., Zhao, P., & Li, X.-L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International conference on database systems for advanced applications* (pp. 214–228).

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *2008 International Conference on Prognostics and Health Management, PHM 2008*. doi: 10.1109/PHM.2008.4711414

Si, X. S., Wang, W., Hu, C. H., & Zhou, D. H. (2011). Remaining useful life estimation - A review on the statistical data driven approaches. *European Journal of Operational Research*, *213*(1), 1–14. doi: 10.1016/j.ejor.2010.11.018

Sikorska, J. Z., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, *25*(5), 1803–1836. doi: 10.1016/j.ymssp.2010.11.018

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Szegeandy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1–9. doi: 10.1109/CVPR.2015.7298594

Wu, J., Hu, K., Cheng, Y., Zhu, H., Shao, X., & Wang, Y. (2020). Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network. *ISA Transactions*, *97*, 241–250. doi: 10.1016/j.isatra.2019.07.004

Yang, H., Zhao, F., Jiang, G., Sun, Z., & Mei, X. (2019). A novel deep learning approach for machinery prognostics based on time windows. *Applied Sciences*, *9*(22), 4813. doi: 10.3390/app9224813

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295-316. doi: https://doi.org/10.1016/j.neucom.2020.07.061

Zhao, C., Huang, X., Li, Y., & Iqbal, M. Y. (2020). A double-channel hybrid deep neural network based on CNN and BiLSTM for remaining useful life prediction. *Sensors (Switzerland)*, *20*(24), 1–15. doi: 10.3390/s20247109

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long Short-Term Memory Network for Remaining Useful Life estimation. *2017 IEEE International Conference on Prognostics and Health Management, ICPHM 2017*, 88–95. doi: 10.1109/ICPHM.2017.7998311