# REMAINING USEFUL LIFE PREDICTIONS WITH DEEP LEARNING METHODS

*(Final report)*


Thomas Guillebot de Nerville*, Paul Strähle*, Anass Akrim†‡ and Rob Vingerhoeds†

*Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31400 Toulouse, FRANCE
Email: {thomas.guillebot-de-nerville,paul.strahle}@student.isae-supaero.fr

†Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31400 Toulouse, FRANCE
Email: {anass.akrim,rob.vingerhoeds}@isae-supaero.fr

‡Institut Clément Ader (UMR CNRS 5312) INSA/UPS/ISAE/Mines Albi, Université de Toulouse, 31400 Toulouse, FRANCE
Email: anass.akrim@univ-tlse3.fr



***Abstract*—This paper aims to compare Deep Learning models to predict the Remaining Useful Life (RUL) of a structure. These models are separated into two families: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Once implemented, their hyperparameters are optimized and the models are compared on the basis of a metric.**


## I. BACKGROUND

### Deep Learning

In recent years Deep Learning, a subbranch of Machine Learning, has shown impressive results, especially in the fields of speech recognition, visual object recognition and object detection [1]. One requirement in using Deep Learning is the presence of sufficient amounts of data [2]. As more data becomes available in the engineering domain there is a recent surge of interest in using Deep Learning in engineering [3].

One of the strengths of Deep Learning approaches is their ability to deal with and detect complex relationships in large datasets [4]. This strength makes their usage also interesting in the Prognostics and Health Management (PHM) domain [5]. The potential of Deep Learning in PHM might not be fully exploited yet [6].

### Prognostics and Health Management

According to Zio [7] PHM is a field of research and application which aims at making use of past, present and future information on the environmental, operational and usage conditions of a piece of equipment in order to detect its degradation, diagnose its faults, predict and proactively manage its failures. In the context of this project only the detection of degradation and prediction of failure are relevant.

PHM models can be divided into single and multi-model approaches. Multi-model approaches are a combination of different single-model approaches. Single-model approaches can be further divided into knowledge-based, data-driven and physics-based models. Within the data-driven models there are statistical, stochastic and Machine Learning models which is the category of Deep Learning models. [4]

## II. INTEREST

Deep Learning has shown some impressive results when applied to RUL prediction as shown in [8]–[13] and other publications (For an overview see Akrim et al. [6]).

Within the available Deep Learning models there are two algorithms which are promising for RUL prediction: RNNs (Little [14] and Hopfield [15]) and CNNs (Lecun et al. [16]) [6].

RNNs are the common Deep Learning approach for time-dependent relationships and have therefore also achieved great interest in the PHM domain [6]. Pioneers models were developed such as Elman Recurrent Networks (ERMs) [17] or Jordan Networks [18], outperforming traditional Machine Learning Programs (MLPs) for sequence-prediction [6]. These algorithms were then widely explored by several researchers. Among them can be cited Yan et al. [19] for their work on material degradation evaluation and life prediction in 2007, and Kramti et al. [20] for having used ERMs for high-speed shaft bearing prognostics based on vibration signals [6].

An emerging type of Deep Learning network for time-dependent relationships are CNNs which might be able to outperform RNNs [21]. In one of the first applications of CNNs to PHM Li et al. suggested that CNNs can be used to obtain RUL prognoses for machinery [9]. The model was applied to the C-MAPSS dataset [22] and outperformed state-of-the-art prognostics approaches including RNN and Long short-term memory (LSTM) models.

In this work a synthetic dataset is used as according to Fink et al. [23] the use of simulation environments and adaption to real-life applications is a promising future research approach as the data will more likely be sufficient in the source domain. The synthetic dataset used for this study aims at using this advantage.

The synthetic dataset matches strain gauge data to the RUL. The use of strain gauges as an input to an PHM model is interesting as they play a vital role in PHM in general [24] and specifically for the aircraft domain [25].

## III. Aim

The goal of this study is to predict the RUL of precracked plates based on strain gauge measurements using Deep Learning. For the application of Deep Learning to crack growth for RUL prediction based on strain gauge measurements no results in the literature could be found. This study attempts to fill this research gap.

In a work done by Akrim [26] the Paris-Erdogan Law [27] was used to create a synthetic dataset of crack growth to train the model. The dataset consists of strain data from virtual strain gauges placed in the area around the crack and the corresponding RUL of the fuselage panels.

The Paris-Erdogan Law is applied to an infinite plate as shown in Figure XXX. The strain gauges are placed in a 45°-Angle at the positions shown in Table XXX relative to the center of the plate. The crack length in the Paris-Erdogan Law is a function of: $k$ number of cylces, $\Delta\sigma$ stress range, $m, C$ material constants, $a_0$ initial crack length. By calculating the crack growth until the crack length $a$ reaches the critical crack length $a_{crit} = (\frac{K_{IC}}{\Delta\sigma\sqrt{\pi}})^2$ the simulation can determine the RUL for each simulated cycle. For the simulations the stress range $\Delta\sigma$ is kept constant while the initial crack length $a_0$ and the material parameters $m$ and $C$ are drawn from a normal distribution. 10000 structures are generated this way to form the training and validation dataset. For the testing dataset 100 structures are generated. The output of theses simulations which form the input for the deep learning models is a table matching the strains from the strain gauges to the RUL calculated by the simulations. To reduce the dataset the results are only saved to the table every 500 cycles. Table I shows the structure of the generated training and validation dataset. The label $ID$ denotes the specific structures.

TABLE I
Training and validation dataset structure

| $t$ | $ID$ | $cycle$ | $\epsilon_1$ | $\epsilon_2$ | $\epsilon_3$ | $RUL_{bin}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | $\epsilon_{1,1}$ | $\epsilon_{2,1}$ | $\epsilon_{3,1}$ | $RUL_{bin,1}$ |
| 2 | 1 | 500 | $\epsilon_{1,2}$ | $\epsilon_{2,2}$ | $\epsilon_{3,2}$ | $RUL_{bin,2}$ |
| 3 | 1 | 1000 | $\epsilon_{1,3}$ | $\epsilon_{2,3}$ | $\epsilon_{3,3}$ | $RUL_{bin,3}$ |
| 4 | 1 | 1500 | $\epsilon_{1,4}$ | $\epsilon_{2,4}$ | $\epsilon_{3,4}$ | $RUL_{bin,4}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n$ | 10000 | $N_{f,10000}$ | $\epsilon_{1,5}$ | $\epsilon_{2,5}$ | $\epsilon_{3,5}$ | $RUL_{bin,5}$ |

The aim of this work is to predict the RUL based on current and past information from the 3 strain gauges using deep learning models. Different Deep Learning model architectures are to be used for that. The developed models must be trained and optimized before a comparison between them is made.

## IV. Methods

**Available Dataset**

The provided datasets were separated in two distinguished parts, namely training and testing dataset. The first consists in 10,000 structures with different initial crack. All the models were trained on this dataset using only a part of it, respectively 100, 500 and 1,000 structures. While training, the last 100 structures of the training dataset were used as a validation dataset.

Then, the 100 structures from the second dataset were used to evaluate the performances of our models.

As it showed promising results in other applications of Deep Learning to PHM ([28], [29]), classification is an interesting alternative to regression for RUL prediction. Instead of trying to predict an exact RUL value the goal is to predict the correct RUL class with a lower and upper bound for the RUL. The generated dataset was composed of 80,000 cycles at most. Therefore the chosen strategy has been to create 20 intervals following a parabolic equation. The last class would be for a RUL over 80,000 cycles away.

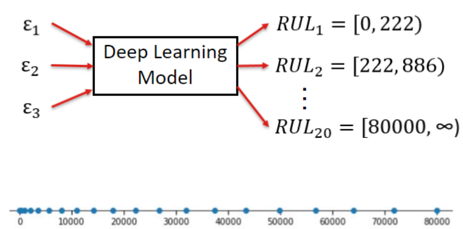



Fig. 1. RUL classification strategy

To prepare the data for the CNN and RNNs models a sliding window approach was used. This approach maps the RUL at time $t_i$ onto the current and past time steps of the input features $[x_{t-N+1}, x_{t-N+2}, ..., x_{t-1}, x_t]$ where $N$ is the length of the sequence. The resulting input matrix therefore has the dimension $k \times N$ where $k$ is the number of features.

As an input for the different models, datasets had to be normalized. While a specific layer was added at the beggining of CNN and Temporal Convolutional Network (TCN) architectures to do so, the preprocessed data had to be normalized before being fed to the RNNs models.

**Recurrent Neural Networks**

The common type of Deep Learning model for time series prediction are RNNs [21]. Due to their ability to store information within the cell, it can better remember information of a time-dependant data.

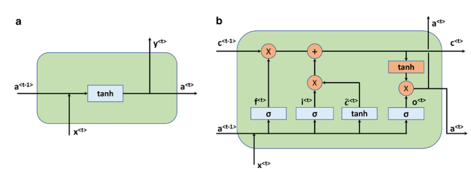



Fig. 2. RNN cell architecture. a. A simple RNN unit; b. an LSTM unit [30]

However, standard RNNs have some major drawbacks, such as the vanishing or exploding gradient problem, which limit their application [31]. LSTM networks (Hochreiter and Schmidhuber [32]) avoid this problem and have established themselves as one of the most used Deep Learning model types, especially for Natural Language Processing (NLP) [33]. For these reasons LSTMs will be one of the investigated RNN approaches in this project. Another investigated RNN approach is the Gated Recurrent Units (GRUs). It is a simplified version of the LSTM. Due to this simplicity it has been gaining in popularity in recent years [34].

The key idea to GRU and LSTM is the memory cell which allows the network to remember information without much loss.

LSTM architecture consists in an input gate, a forget gate and an output gate. The input gate decides to add new information from the present input to the present cell state scaled by how much it wishes to add them. The forget gate allows the cell to know how to partially forget the previous cell state. Then, the output gate decides what to output from the new celle state.

For the GRU architecture, only two gates are implemented. The update gate decides the portion of updated candidate needed to calculate the current cell state, which in turn also decides the portion of the previous cell state retained. The relevance gate calculates how relevant the previous information is, and then is used in the candidate for the updated value.

Then, a simple RNN architecture will be explored to evaluate the possible benefits of more complex structures. An RNN unit only have a tanh layer after combining the cell state and the candidate.

**Convolutional Neural Networks**

Recent results suggest that CNNs can match or even outperform RNNs in time series related tasks [21]. The second major focus of this work are therefore CNN models.

The common CNN models deal with 2-Dimensional data as input such as pictures. The time sequence data used for PHM is in 1-Dimensional format. For this application 1D-CNN have been introduced. The key differences between them and the 2D-CNNs are that their input data is reduced by one dimension and the convolution filter only slides in one dimension [6]. Figure 3 shows an example of a simple CNN architecture with an illustration of the filter sliding over the time series.

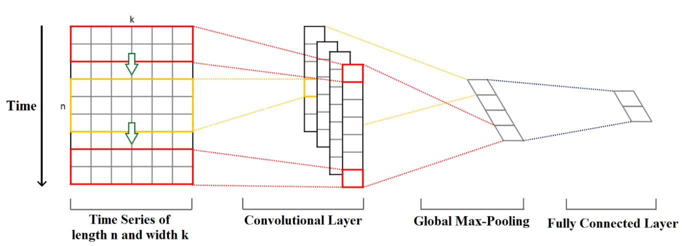



Fig. 3. Example of an 1D-CNN architecture [35]

Besides the general CNN architectures TCNs (Bai et al. [21]) are investigated in this work. A TCN is a specific CNN architecture that tries to replicate some of the best practices for CNN architectures. As depicted in Fig. 4 a TCN is composed of multiple layers which include dilation. The dilation for each layer can be set arbitrary but it is common practice to use multiples of 2 for it. Through dilation TCNs can increase their receptive field and therefore capture relationships over longer time sequences. One TCN layer is composed of a TCN residual block (see Figure 5 (a)). One block consists of two dilated convolutional layers. The activation function for the layers is always the ReLu function. The residual blocks include dropout and normalization layers between the convolutional layers. The blocks also include an optional skip connection with a 1x1 convolutional layer. Figure 5 (b) shows an exemplary TCN block with a kernel size of $k = 3$ and a dilation of $d = 1$. For more details on TCNs see [21].

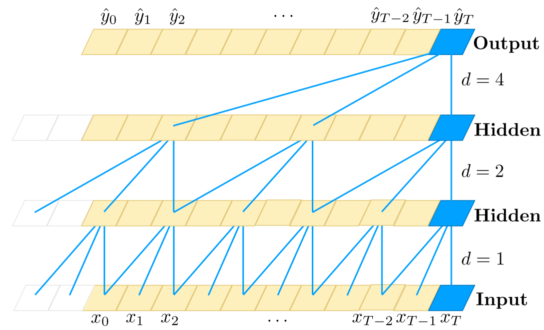
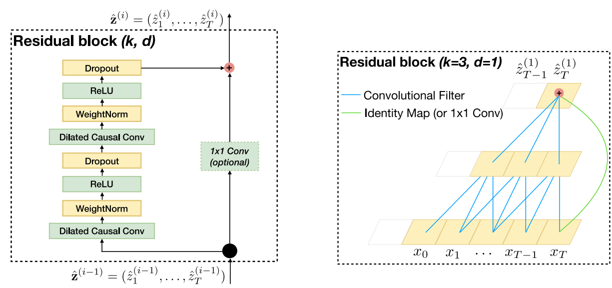



Fig. 4. TCN with dilation factors d = 1; 2; 4 and filter size k = 3 [21]




(a) (b)

Fig. 5. TCN block elements, (a): Generic TCN block, (b): Example for a TCN block [21]

**Metric and loss function**

To evaluate the performance of developed models a metric is needed. As the networks are used to perform classification instead of regression the accuracy metric is used. This metric calculates how many of the predictions made with the dataset are correct (e.g. the right RUL range is predicted)

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}. \tag{1}$$

This metric is applied during training and validation to compare different models against each other. The final assessment of the models is made by applying the accuracy metric to the testing dataset.

The used loss function which is used as the objective function for minimization by the backpropagation algorithm is the categorical crossentropy loss function

$$CE = -log(\frac{e^{s_p}}{\sum_j^C e^{s_j}}) \tag{2}$$

where $s_j$ is the output of the network for Class $j$ with $s_p$ beeing the positive class.

**Hyperparameter Optimization**

To optimize the proposed network architectures a hyperparameter optimization is performed. The hyperparameters include the parameters for the training process (Learning rate and batch size), network parameters (Dropout rate, activation funciton etc.) and the network architecture itself (number of layers, size of the layers etc.).

There are different strategies for hyperparameter optimization which can be divided in Model-Free and Model-Based approaches. For this study only Model-Free approaches are considered as they are more common and easier to apply. The two common approaches are grid search and random search [36].

For grid search the user defines a discrete number of values for each parameter. The algorithm then tries out all possible combinations of these sets of values. This quickly leads to an exesive amount of training runs that need to be done as the number of possible combinations $B$ grows exponentially with the dimensionality $d$ of the search space. If the number of values per parameter is $n = 3$ and the dimensionality is just $d = 5$ the number of combinations already equals $B = n^d = 243$.

Random search works with a fixed number of evaluations where for each evaluation a value randomly is selected from a predefined set of values for each parameter. The set of values can either be continuous with a lower and upper bound for the values or a discrete set of predefined values. Random search works better than grid search when some parameters are more important than others, which is a property that holds true in many cases [36]. Figure 6 illustrates this property for one important and one unimportant parameter. With a fixed amount of $B$ evaluations random search can evaluate up to $B$ different values for each parameter. Whereas grid search is limited to $B^{1/N}$ values per parameter. It is expected that the networks in this study behave like most networks with regard to the varying importance of the hyperparameters and therefore random search is choosen for hyperparameter optimization.

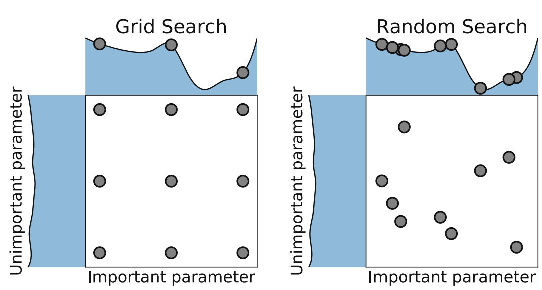



Fig. 6. Comparison of grid search and random search for an important and a unimportant parameter with 9 evaluations [36]

To select the best model after performing the hyperparameter optimization the accuracy of the model on the validation dataset is used.

**Fine Tuning**

After the optimum parameters for a model are found the model can be further improved by running more training epochs. To do so the models are trained with learning rate decay (lrDecay)). The best model from the hyperparameter optimization is taken and trained for a predefined number of epochs on the learning rate which was identified as the best one on the optimization. The learning rate is then lowered before the model is trained again for a fixed number of epochs. This step is repeated until convergence is achieved and no more significant improvements are visible. By using this approach the model in the early stages of training is less likely to get stuck in a local minimum and explores a wider range of possible configurations. As the training comes closer to an optimum the decaying learning rate helps with convergence and avoid oscillations. Besides this common beliefs there are probably more reasons to the effectiveness of lrDecay). According to You et al. [37] is the initially large learning rate preventing the network from memorizing noisy data while the decaying learnig rate helps with learning complex patterns in the dataset.

## V. Results

**Model architectures**

Based on the CNN architecture of Li et al. [9] which was used for aero-engine RUL prediction a vanilla CNN architecture is generated. The first layer of the architecture is for layer normalization. Afterwards a flexible number of 1D-convolutional layers is added which all have the same number of filters and an identical kernel size. The output of the convolutional layers is flattend before the final dense output layer which has 20 nodes to match the number of RUL classes $C = 20$. As an example Figure 7 shows the final architecture after the hyperparameter optimization with 1000 training structures.

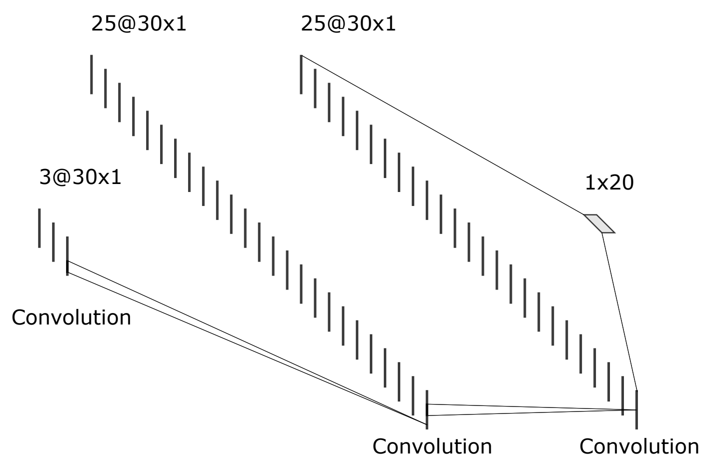



Fig. 7. CNN architecture after hyperparameter optimization with 1000 training structures

A TCN architecture template is generated based on the results of Liu et al. [10] who used the network for RUL prediction of roller bearings. The architecture consists of multiple TCN residual blocks with a dilation increasing with multiples of 2 with the first layer having a dilation of $d = 1$. Each block features the same number of filters with an identical kernel size. The next layer after the TCN blocks is the dense output layer which has 20 nodes to match the number of RUL classes $C = 20$. The skip connection layer is activated for all residual blocks.

All RNNs architectures were based on the same template. The dataset being noramlized before being fed to the models, there were no need for a specific layer of normalization. A flexible number of layers were implemented in the networks, from 1 hidden layer to a maximum of 3. Each layer includes from 32 to 256 units using a scaling by power of 2. A possible dropout and recurrent dropout could also be added for each layer, with a possibility of 0% or 10% for each. At the end of the architecture, a dense layer with 20 units were added in order to match the number of RUL classes $C = 20$.

**Hyperparameter optimisation**

## VI. Conclusions

## References


[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, ISSN: 1476-4687. DOI: 10.1038/nature14539.

[2] J. Z. Sikorska, M. Hodkiewicz, and L. Ma, "Prognostic modelling options for remaining useful life estimation by industry," *Mechanical Systems and Signal Processing*, vol. 25, no. 5, pp. 1803–1836, 2011, ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2010.11.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327010004218.

[3] A. Voulodimos, N. Doulamis, G. Bebis, and T. Stathaki, "Recent developments in deep learning for engineering applications," *Computational Intelligence and Neuroscience*, vol. 2018, p. 8 141 259, 2018, ISSN: 1687-5265. DOI: 10.1155/2018/8141259.

[4] J. J. Montero Jimenez, S. Schwartz, R. Vingerhoeds, B. Grabot, and M. Salaün, "Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics," *Journal of Manufacturing Systems*, vol. 56, pp. 539–557, 2020, ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2020.07.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520301187.

[5] S. Wu, K. L. Tsui, N. Chen, Q. Zhou, Y. Hai, and W. Wang, "Prognostics and health management: A review on data driven approaches," *Mathematical Problems in Engineering*, vol. 2015, p. 793 161, 2015, ISSN: 1024-123X. DOI: 10.1155/2015/793161.

[6] A. Akrim, C. Gogua, R. Vingerhoeds, and M. Salaun, "Review of prognostics approaches with a particularfocus on the current machine learning algorithms," 2021.

[7] E. Zio, "Prognostics and Health Management of Industrial Equipment," in *Diagnostics and Prognostics of Engineering Systems: Methods and Techniques*, S. Kadry, Ed., ISBN13: 9781466620957, IGI Global, Sep. 2012, pp. 333–356. DOI: 10.4018/978-1-4666-2095-7.ch017. [Online]. Available: https://hal-supelec.archives-ouvertes.fr/hal-00778377.

[8] L. Xu, S. F. Yuan, J. Chen, and Q. Bao, "Deep learning based fatigue crack diagnosis of aircraft structures," in *7th Asia-Pacific Workshop on Structural Health Monitoring*, 2018. [Online]. Available: https://www.ndt.net/search/docs.php3?id=24093.

[9] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018, ISSN: 0951-8320. DOI: 10.1016/j.ress.2017.11.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832017307779.

[10] C. Liu, L. Zhang, and C. Wu, "Direct remaining useful life prediction for rolling bearing using temporal convolutional networks," (Dec. 6, 2019), Xiamen, China:

IEEE, Dec. 6, 2019, pp. 2965–2971, ISBN: 978-1-7281-2486-5. DOI: 10.1109/SSCI44817.2019.9003163.

[11] M. Yuan, Y. Wu, and L. Lin, “Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network,” in *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, 2016, pp. 135–140. DOI: 10.1109/AUS.2016.7748035.

[12] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, “Remaining useful life estimation of engineered systems using vanilla lstm neural networks,” *Neurocomputing*, vol. 275, pp. 167–179, 2018, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.05.063. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217309505.

[13] K. Park, Y. Choi, W. Choi, H.-Y. Ryu, and H. Kim, “Lstm-based battery remaining useful life prediction with multi-channel charging profiles,” *IEEE Access*, vol. PP, pp. 1–1, Jan. 2020. DOI: 10.1109/ACCESS.2020.2968939.

[14] W. A. Little, “The existence of persistent states in the brain,” in *From High-Temperature Superconductivity to Microminiature Refrigeration*, B. Cabrera, H. Gutfreund, and V. Kresin, Eds. Boston, MA: Springer US, 1996, pp. 145–164, ISBN: 978-1-4613-0411-1. DOI: 10.1007/978-1-4613-0411-1_12.

[15] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982, ISSN: 0027-8424. DOI: 10.1073/pnas.79.8.2554. eprint: https://www.pnas.org/content/79/8/2554.full.pdf. [Online]. Available: https://www.pnas.org/content/79/8/2554.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[17] J. L. Elman, “Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990, ISSN: 0364-0213. DOI: 10.1016/0364-0213(90)90002-E. [Online]. Available: https://www.sciencedirect.com/science/article/pii/036402139090002E.

[18] M. I. Jordan, “Chapter 25 - serial order: A parallel distributed processing approach,” in *Neural-Network Models of Cognition*, ser. Advances in Psychology, J. W. Donahoe and V. Packard Dorsel, Eds., vol. 121, North-Holland, 1997, pp. 471–495. DOI: 10.1016/S0166-4115(97)80111-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166411597801112.

[19] J. Yan and P. Wang, “Blade material fatigue assessment using elman neural networks,” vol. 2007, Jan. 2007. DOI: 10.1115/IMECE2007-43311.

[20] S. E. Kramti, J. Ben Ali, L. Saidi, M. Sayadi, and E. Bechhoefer, “Direct wind turbine drivetrain prognosis approach using elman neural network,” in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2018, pp. 859–864. DOI: 10.1109/CoDIT.2018.8394926.

[21] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *CoRR*, vol. abs/1803.01271, 2018. arXiv: 1803.01271. [Online]. Available: http://arxiv.org/abs/1803.01271.

[22] A. Saxena, K. Goebel, D. Simon, and N. Eklund, “Damage propagation modeling for aircraft engine run-to-failure simulation,” in *2008 International Conference on Prognostics and Health Management*, 2008, pp. 1–9. DOI: 10.1109/PHM.2008.4711414.

[23] O. Fink, Q. Wang, M. Svensén, P. Dersin, W.-J. Lee, and M. Ducoffe, “Potential, challenges and future directions for deep learning in prognostics and health management applications,” *Engineering Applications of Artificial Intelligence*, vol. 92, p. 103 678, 2020, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2020.103678. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197620301184.

[24] T. Tinga and R. Loendersloot, “Physical model-based prognostics and health monitoring to enable predictive maintenance,” in *Predictive Maintenance in Dynamic Systems: Advanced Methods, Decision Support Tools and Real-World Applications*, E. Lughofer and M. Sayed-Mouchaweh, Eds. Cham: Springer International Publishing, 2019, pp. 313–353, ISBN: 978-3-030-05645-2. DOI: 10.1007/978-3-030-05645-2_11.

[25] F. Timothy, M. Devinder, and H. Iain, “F-35 joint strike fighter structural prognosis and health management an overview,” Jan. 2009. DOI: 10.1007/978-90-481-2746-7_68.

[26] A. Akrim, “Description algorithm.”

[27] P. Paris and F. Erdogan, “A critical analysis of crack propagation laws,” *Journal of Basic Engineering*, vol. 85, no. 4, pp. 528–533, Dec. 1963. DOI: 10.1115/1.3656900.

[28] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, “Time series classification with multivariate convolutional neural network,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788–4797, Dec. 6, 2019. DOI: 10.1109/TIE.2018.2864702.

[29] W. Xiao, “A probabilistic machine learning approach to detect industrial plant faults,” Mar. 2016. arXiv: 1603.05770 [stat.ML].

[30] G. Chen, “Recurrent neural networks (rnns) learn the constitutive law of viscoelasticity,” *Computational Mechanics*, vol. 67, Mar. 2021. DOI: 10.1007/s00466-021-01981-y.

[31] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult.,” eng, *IEEE transactions on neural networks*, vol. 5, pp. 157–66, 2 1994. DOI: 10.1109/72.279181.

[32] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.

1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

[33] Y. Wu, M. Schuster, Z. Chen, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," Sep. 2016. arXiv: 1609.08144 [cs.CL].

[34] R. Rana, "Gated recurrent unit (gru) for emotion classification from noisy speech," Dec. 2016. arXiv: 1612.07778 [cs.HC].

[35] R. A. Sayyad, "How to use convolutional neural networks for time series classification," *Medium*, [Online]. Available: https://medium.com/@Rehan_Sayyad/how-to-use-convolutional-neural-networks-for-time-series-classification-80575131a474.

[36] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning*, Springer, Cham, 2019, pp. 3–33.

[37] K. You, M. Long, J. Wang, and M. I. Jordan, *How does learning rate decay help modern neural networks?* 2019. arXiv: 1908.01878 [cs.LG].