A NOVEL RESOURCE-AWARE TENSOR DECOMPOSITION DESIGN BASED ON REINFORCEMENT LEARNING

*Behnaz Ghoraani**
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University

ABSTRACT

Tensor decomposition is a promising solution for analyzing and classifying multidimensional data streams in long-term monitoring of physical systems. However, the key challenge is to continuously perform the expensive tensor decomposition to ensure the effective classification of the time-evolving data as the system evolves over time. This paper explores a novel resource-aware tensor decomposition framework using reinforcement learning (RL). The proposed framework is developed to update a tensor decomposition-based classifier only using the data that is important to the evolving nature of the system. It learns an RL agent to look-ahead and identifies the data that is important to the classifier update and performs the tensor decomposition on those data streams. The proposed resource-aware framework was evaluated using synthetic data and motion data from patients with Parkinson's disease and indicated a significant performance in both classification accuracy and number of tensor decompositions compared to a continuous update approach.

Index Terms— tensor decomposition, reinforcement learning, time-evolving systems, long-term monitoring

1. INTRODUCTION

Tensor (i.e., multi-way array) decomposition is a promising solution for data-driven analysis of multiset fused data as needed for monitoring of physical systems in a wide range of applications, e.g., environmental, infrastructure, transportation, and health monitoring [1]. Tensor decomposition provides a supervised and unsupervised feature extraction and classification tool that captures multi-linear and multi-aspect structures in large-scale and multidimensional/multiset datasets [2]. Despite the fundamental discoveries in the tensor decomposition field, there are limitations that restrict their application in long-term monitoring of physical systems. Hence, tensor decomposition-based classifiers have been mainly applied for detecting patterns in static datasets and their full potential for evolving data streams of nonstationary, evolving physical systems has yet to be realized [3]. Iterative tensor decompositions have been studied [4, 5], but previous studies focus on minimizing the cost of classifier update for every incoming data and are still based on continuous updates.

In this paper, we investigate the possibility of developing a resource-aware approach that, instead of continuous updates of a classifier, intelligently updates a tensor decomposition-based classifier according to the importance of data to the classifier's performance. The development of such resource-aware tensor updates is extremely important as it significantly advances the analysis of time-evolving data streams and could extend the application of powerful tensor decomposition-based classifications to long-term monitoring of physical systems.

2. MATERIALS AND METHODS

2.1. Tensor Decomposition-based Classifiers

The data stream of K sensors with P channels $x_T \in \mathbb{R}^{K\times P}$ for $T = 1, \dots, n, n+1, \dots$ is represented by a 3-way tensor $\underline{\mathbf{X}}_{K\times N\times P}$ with time dimension N growing as more data is received (see **Figure 1**). Tensor decompositions have been extremely helpful in the classification cases that pre-known, hand-crafted-features are unavailable [10]. They decompose the tensor data into the representing basis vectors, in different data domains, which indicate a set of optimal, data-driven features to represent the data in a classifier $l = \psi_T(\underline{\mathbf{X}})$ to predict the category labels l from data $\underline{\mathbf{X}}$. There are several tensor decomposition models [6, 7]. One well-known approach is parallel factor analysis (PARAFAC) [8], which decomposes a tensor $\underline{\mathbf{X}}_{K\times N\times P}$ to M rank-1 bases according to the following equation:

$$\underline{\mathbf{X}} = \sum_{m=1}^{M} b_m \circledcirc c_m \circledcirc d_m \qquad (1)$$

where $\circledcirc$ denotes the outer product, and the vectors $b_m \in \mathbb{R}^{K\times 1}, c_m \in \mathbb{R}^{N\times 1}, d_m \in \mathbb{R}^{P\times 1}$ are the decomposed rank-1 basis vectors and represent the columns of the basis matrices $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, respectively. The decomposed basis vectors (b_m, d_m) indicate the data-driven features in the classifier ψ_T. The bases are estimated based on a least square cost function in Eq. 2, which is solved using stochastic gradient approaches [9]. For simplicity, the notation of $[\![\mathbf{B}, \mathbf{C}, \mathbf{D}]\!]$ is used instead of Eq. 1.

$$f(\mathbf{B}, \mathbf{C}, \mathbf{D}) = \frac{1}{2}\left\|\underline{\mathbf{X}} - [\![\mathbf{B}, \mathbf{C}, \mathbf{D}]\!]\right\|_{\mathrm{F}}^{2} \qquad (2)$$

* Corresponding author. Email: bghoraani@ieee.org

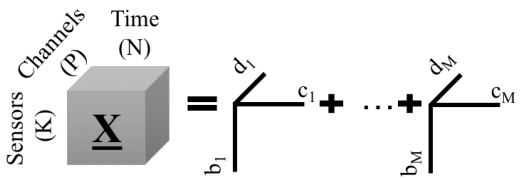

Figure 1. Tensor decomposition based on parallel factor analysis. The 3-way tensor $\underline{\mathbf{X}}_{K\times N\times P}$ is decomposed to M rank-1 basis vectors $b_m \in \mathbb{R}^{K\times 1}, c_m \in \mathbb{R}^{N\times 1}, d_m \in \mathbb{R}^{P\times 1}$.

2.2. Reinforcement Learning

Reinforcement learning (RL) is a biologically inspired machine learning technique [11] where an RL agent interacts with a dynamic and incompletely known environment over time and learns to take optimal actions so as to control the environment (see **Figure 2**). At every step T, an agent observes the environment's current state s_T, takes an action a_T from the set of possible actions according to its policy, $\pi_\theta(a_T|s_T)$, receives a reward r_T from the environment, and then transitions to a state s_{T+1}. This process continues until the agent reaches a terminal state or continues forever in the case of non-episodic problems. The goal of the agent is to maximize the expectation of the long-term accumulated reward, $R_T = \sum_{\tau=0}^{\infty} \lambda^\tau r_{T+\tau}$, where τ refers to future rewards and $\lambda \in (0,1]$ is a discount factor to tradeoffs between the immediate and future rewards. Although RL proved to be successful for controlling human-built systems [12], its ability to learn from real-physical systems is yet to be explored. This is because the core assumption of RL that the agent's actions must influence the environment's current state do not necessarily hold when modeling real-world problems.

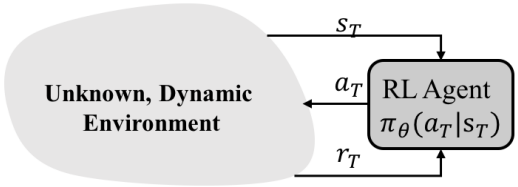

Figure 2. Interactions between reinforcement learning agent and unknown, dynamic environment. At every step T, the RL agent observes the environment's current state s_T, takes an action a_T according to its policy, $\pi_\theta(a_T|s_T)$, receives a reward r_T from the environment, and then transitions to a state s_{T+1}.

2.3. Novel Resource-aware Tensor Decomposition

The developed method is inspired by the ability of tensor decomposition to derive data-driven classifiers ψ_T from multiset fused data and RL to think ahead and predict optimized actions a_T that control an environment toward a desired state (**Figure 3**). Hence, adapting RL modeling should be highly promising for learning the optimal action policy $\pi_\theta(a_T|s_T)$ for when to update the classifier $\psi_T => \psi_{T+1}$. Based on the integration of tensor decomposition and RL, a resource-constraint update for tensor decomposition is proposed in such a way that a classifier update will happen only at predicted times. Specifically, the resource-constraint update predicts when $\underline{\mathbf{X}}_{T+1}$ is important to the system evolution and provides appropriate action $a_{T+1} \in \{0,1\}$, where $a_{T+1} = 1$ indicates a classifier update is recommended. In long-term monitoring of evolutionary systems, $\{\dots, a_T, a_{T+1}\}$ will be very sparse resulting in a significant reduction in the computation cost of by reducing the number of time that a tensor decomposition needs to be performed for a tensor decomposition-based classifier.

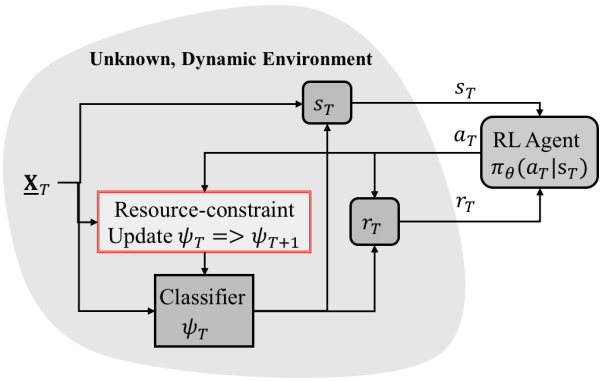

Figure 3. The developed resource-aware tensor decomposition design based on reinforcement learning.

To predict action a_{T+1}, a dynamic environment consisting of both sensors and a classifier is defined (**Figure 3**). An RL agent is designed to control this environment according to policy $\pi_\theta(a_T|s_T)$. In this design, the data $\underline{\mathbf{X}}_T$ and classifier performance $\psi_T(\underline{\mathbf{X}}_T)$ provide the environment's observables s_T. The changes in the classifier performance and resource usage (i.e., a_T being 1 or 0) define reward r_T to an update. This design satisfies the requirement that an RL agent's action $a_T \in \{0,1\}$ (when to update) must influence the subsequent observation (i.e., classifier performance). The observation s_T is defined as follows.

$$s_T = [\psi_{T-T_0:T}\ g(\underline{\mathbf{X}}_T)] \tag{3}$$

where $\psi_{T-T_0:T}$ indicates the classifier's performance over a window of $T_0 + 1$ and function $g(\underline{\mathbf{X}}_T) = (\boldsymbol{e}_T - \mu)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{e}_T - \mu)$ calculates the likelihood of change of the incoming data stream $\underline{\mathbf{X}}_T$ with respect to the last T_0 received data. In this formulation, $\boldsymbol{e}_T \in \mathbb{R}^{1\times T_0}$ is the probability of the last T_0 incoming data previous to $\underline{\mathbf{X}}_T$ belonging to the data pdf $P_T(\underline{\mathbf{X}})$, which is obtained from the raw data stream. Parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of a Normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are estimated using Maximum Likelihood Estimation on the vector series $[\boldsymbol{e}_{T-I+1}, \dots, \boldsymbol{e}_T]$.

The reward function is defined as in Eq. 4 such that $h(a_T)$ penalizes the RL agent for an update and $f(\psi_{T_0:T+1})$ is defined to reward the agent when the action improves the performance and penalizes it otherwise.

$$r_T = \psi_{T+1} - \mu_\psi - \alpha_h a_T \tag{4}$$

where $0 \leq \alpha_h \leq 1$ represents the update-cost weight, $a_T \in \{0,1\}$ is the associated action to whether to update the classifier or not, and μ_ψ is the average performance over $\psi_{T-T_0:T+1}$.

The details of the method to train the described RL agent are outlined in **Algorithm 1**. The algorithm, first, trains an initial classifier ψ_0 by applying PARAFAC on the initial data, and then an RL agent is trained by applying and updating ψ_0 over the evolving data. For every action of the RL agent, the algorithm updates the RL policy $\pi_\theta(a_T|s_T)$ according to the environment's observable s_{T+1} and the received reward r_{T+1}. Given that we defined the environment such that it rewards the RL agent if the selected action increases the classifier performance and minimizes the number of updates, it is expected that the trained RL agent learns to predict the data that is important to the system evolution and update the classifier. Step 8 is important as it defines which RL algorithm will be used to learn the described RL agent. In this work, we will use REINFORCE algorithm [12] which is a policy gradient method that directly optimizes the policy $\pi_\theta(a_T|s_T)$. We use Recurrent Neural Network (RNN) as a linear combination of observations and actions.

Algorithm 1 Train RL Agent for Resource-aware Tensor Decomposition

Input: Training data $\{\underline{\mathbf{X}}^1, \ldots, \underline{\mathbf{X}}^q, \ldots, \underline{\mathbf{X}}^Q\} \in \mathbb{R}^{K \times N(q) \times P}$, T_0 performance window size, α_h update-cost weight, λ discount factor, RL policy $\pi_\theta(a_T|s_T)$ (RNN with # of hidden layers and # of recurrent neurons), episode duration

1. Initialize the weights of RL RNN with random values.
2. **repeat** $\{\underline{\mathbf{X}}^q\}$
3. Train an initial classifier ψ_0 by applying PARAFAC on the initial data: $\underline{\mathbf{X}}_T$ (T_0 length of $\underline{\mathbf{X}}^q$).
4. Pick an action a_1 and a state s_1
5. **repeat** {Initialize T=T_0}
6. **repeat** {For each episode}
7. Follow action a_T, update classifier $\psi_T => \psi_{T+1}$, observe state s_{T+1} (Eq. 3), and receive reward r_{T+1} (Eq. 4).
8. If s_{T+1} is terminal: Update RNN weights and go to next episode
9. **until** the end of episode
10. Pick an action a_{T+1} according to the RL policy $\pi_\theta(a_{T+1}|s_{T+1})$.
11. Increment T
12. **until** T is equal to $N(q)$
13. **for** $q = 1, \ldots, Q$

Output: RNN weights for RL policy $\pi_\theta(a_T|s_T)$

The parameter selection is application dependent and has to be tuned empirically. The parameters λ, T_0, and α_h are critical to ensure the stability and efficiency of the reward function. The literature suggests that a fixed λ near 0.9 usually provides a reasonable prediction [13]. The value of T_0 has to be large enough to preserve memory of the long-term dynamics, but small enough to limit the effect of older behavior on the reward function by emphasizing the most recent changes. The selection of α_h provides a trade-off between classification performance and update cost. Developing a resource-aware tensor decomposition would extend the potential of tensor decomposition beyond static systems and enable non-stationary analysis of multiset fused data that is needed for monitoring of evolving systems.

3. RESULTS

3.1. Experimental Setup

Synthetic data: A synthetic case with a data structure of $K = 10$ and $P = 3$ at every time point was generated where the basis matrices were drawn from a normal distribution $(\mu = 0, \sigma = 0.1)$. A series of evolutionary effect was simulated by changing the mean from 0 to 0.35 with steps of 0.01. The duration of each data was selected from a uniform distribution (2,10) minutes with a sampling frequency of 10 Hz. SimTensor [14] was used for simulating the standard normal tensors that were later manipulated in MATLAB to generate the dataset. With this setup, for every initial data, on average, there is 250 minutes of long-term evolutionary data. We investigated whether the developed resource-aware tensor decomposition design can be trained to predict the data that is important to the evolving nature of the synthetic dataset.

Parkinson's Disease Patient Data: (PD) is one of the most common chronic progressive neurological disorders, affecting over half a million Americans annually and resulting in over $20 billion costs each year [15]. Levodopa is the most common medication used to improve motor impairments in subjects with PD. Unfortunately, prolonged treatment with Levodopa causes troubling motor fluctuations [16] occurring in 50% of patients within 3 to 5 years of diagnosis and 80% of patients after 10 years [17]. These complications result in frequent fluctuations of response to treatment intervention between "ON" state with maximum benefit from Levodopa and "OFF" state with least benefit from Levodopa, and are a major focus of PD management [18]. In this paper, we investigated whether the developed resource-aware tensor decomposition design can be used to train a classifier that accurately detects PD patients' medication ON/OFF states from motion data for the purpose of PD patient monitoring.

For this purpose, an initial classifier (based on subjects S1 to S5) was designed and adapted to a new test subject S6. A KinetiSense motion sensor unit with a triaxial accelerometer and triaxial gyroscope with 128 Hz sampling rate was used to record motion data from the most affected wrist and ankle while the patients were in their ON and OFF medication states and performed a variety of daily living activities (e.g., resting, walking, drinking, dressing, hair brushing, unpacking groceries, and cutting food) [19]. It provided a PD dataset with six PD subjects (age: 65 ± 7 years) and a total OFF time of 90.32 ± 21.78 mins and ON time of 27.53 ± 10.49 mins with $K = 4$ sensors and $P = 3$ (x,y,z)

axes. The study was approved by the institutional review board, and all patients provided written informed consent.

3.2. Classification of Evolutionary Synthetic Data

To examine the performance of resource-aware tensor decomposition framework, an RL agent was learned using all the simulated data, except one. The observation and reward were calculated using T_0=10 samples and an update cost factor of $0 \leq \alpha_h \leq 1$ and $\lambda = 0.9$. The action policy was modeled using a RNN architecture with one hidden layer consisting of 25 recurrent neurons. The output layer consisted of one sigmoid functions to produce the probability of actions between 0 to 1 corresponding to a_T. NN learning rate of 10^{-3} was used to update the policy network after every one second. We ran an episodic RL with maximum 180 epochs (30 mins of data). To test the performance of the learned RL agent, the update policy was used on the remaining data to control the non-uniform classifier updates. The average performance and total number of updates for $0 \leq \alpha_h \leq 1$ are illustrated in **Figure 4**. The results are shown for the continuous update and the designed resource-aware framework. For comparison purposes, the total number of updates and performance were normalized to the corresponding values from the continuous-update results. As shown in this figure, the resource-aware tensor decomposition design was able to significantly reduce the number of the performed classifier updates (i.e., tensor decompositions). Moreover, for $0.2 \leq \alpha_h \leq 0.6$, the performance of the resource-aware framework is comparable to the continuous update, while the number of updates is much less.

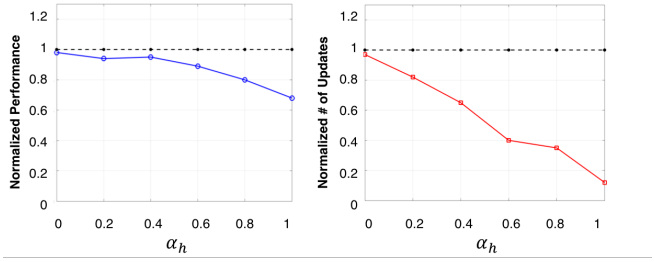

Figure 4. Employing the resource-aware tensor decomposition design significantly reduced the update costs (the plot on the left hand side) for $0.2 \leq \alpha_h \leq 0.6$ with considerably high performance (the plot on the left hand side). The total number of updates and performance were normalized to the corresponding values from the continuous-update results (the dashed line).

3.3 Medication ON/OFF Classification in PD Patients

It was investigated if the resource-aware design of tensor decomposition was able to provide an evolutionary medication ON/OFF classifier for monitoring of PD patients. Training the and RL agent was performed as described in Section 3.2 $\alpha_h = 0.4$. This process repeated for S1 to S5 in a leave-one-out manner. The learned RL policy was used to adapt an initial classifier (based on S1-S5) to the test subject S6 and the result was reported in **Figure 5**. For comparison purposes, a static, generalized classifier (train on S1-S5 and test on S6) was also implemented. In this assessment, the ON/OFF ground truth was provided by a physician. As shown in this figure, the resource-aware design adapted a classifier to a new subject better than the static classifier. Another interesting observation concerns the number of updates over time. Interestingly, as the classifier evolved, fewer classifier updates were activated, with less updates using the more expensive sensor.

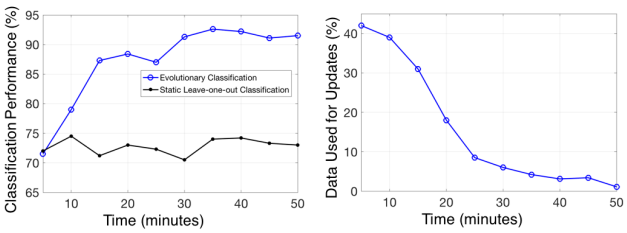

Figure 5. The resource-aware tensor decomposition for evolutionary classification of medication ON/OFF (blue red) was compared to a static leave-one-out classification approach (black line). The analysis showed that the resource-aware tensor decomposition design was able to adapt to the subject's dynamics as indicated by the increase in the performance and decrease in the number of updates.

4. CONCLUSION

In this paper, a novel resource-aware tensor decomposition framework based on reinforcement learning was developed to enable evolutionary analysis of multiset data from a system as it evolves over time. The application of the proposed algorithm on a synthetic dataset and a Parkinson's disease dataset demonstrated its effectiveness for a better analysis of the characteristics that comprise an evolving system in terms of both performance (i.e., accuracy) and efficiency (i.e., number of classifier updates). This framework can be modified to improve the effectiveness by appropriate selection of model parameters using analytical calculations and numerical simulations. Formulating the performance of the non-uniform updates using the resource-aware framework will facilitate efficient selection of the parameters. Furthermore, since this algorithm offers reduced number of classifier updates, its implications would extend beyond reduced computation costs for applications when there are other costs associated with each update, such as transmission cost between sensor and the cloud.

5. REFERENCES

[1] Soh P., Vandenbosch G., Mercuri M. and Schreurs D., "Wearable wireless health monitoring: Current developments, challenges, and future trends," *IEEE Microwave Magazine,* vol. 16, no. 4, pp. 55-70, 2015.

[2] Sidiropoulos N.D., De Lathauwer L., Fu X., Huang K., Papalexakis E.E. and Faloutsos C., "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing,* vol. 65, no. 13, pp. 3551-3582, 2017.

[3] Fanaee-T H. and Gama J., "Multi-aspect-streaming tensor analysis," *Knowledge-Based Systems,* vol. 89, pp. 332-345, 2015.

[4] Su Y., Wang H., Jing P. and Xu C., "A spatial-temporal iterative tensor decomposition technique for action and gesture recognition," *Multimedia Tools and Applications,* vol. 76, no. 8, pp. 10635-10652, 2017.

[5] Nion D. and Sidiropoulos N.D., "Adaptive algorithms to track the parafac decomposition of a third-order tensor," *IEEE Transactions on Signal Processing,* vol. 57, no. 6, pp. 2299-2310, 2009

[6] Ballani J., Grasedyck L. and Kluge M., "Black box approximation of tensors in hierarchical Tucker format," *Linear Algebra and its Applications,* vol. 438, no. 2, pp. 639–657, 2013

[7] Oseledets I., "Tensor-train decomposition," *SIAM J. Scientific Computing,* vol. 33, no. 5, pp. 2295–2317, 2011

[8] Harshman R.A., "Foundations of the PARAFAC procedure: Models and conditions foran "explanatory" multi-modal factor analysis," *In: UCLA working papers in pho-netics,* pp. 1-84, 1970.

[9] Bottou L., "Large-scale machine learning with stochastic gradient descent," *In Proceedings of COMPSTAT'2010. Physica-Verlag HD.,* pp. 177-186, 2010.

[10] Lahat D., Adali T. and Jutten C., "Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects," *Proceedings of the IEEE (2015),* vol. 103, no. 9, pp. 1449-1477, 2015.

[11] Sutton R.S. and Barto A.G., Reinforcement Learning: An Introduction (2nd Edition, in preparation), MIT Press, 2017.

[12] Silver D., Huang A., Maddison C., Guez A., Sifre L., Van Den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M. and Dieleman S., "Mastering the game of go with deep neural networks and tree search.," *Nature,* vol. 529, no. 7587, pp. 484–489, 2016.

[13] Mnih V., Kavukcuoglu K., Silver D., Rusu A., Veness J., Bellemare M., Graves A., Riedmiller M., Fidjeland A., Ostrovski G. and Petersen S., "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[14] Fanaee-T H. and Gama J., "SimTensor: A synthetic tensor data generator," *arXiv preprint arXiv:1406.3496,* 2016.

[15] Gooch C.L., Pracht E. and Borenstein A.R., "The burden of neurological disease in the United States: a summary report and call to action," *Annals of neurology,* vol. 81, no. 4, pp. 479-484, 2017.

[16] Dewey R.B., "Management of motor complications in Parkinson's disease", *Neurology*, vol. 62, pp. S3–S7, 2004.

[17] Davie C., "A review of Parkinson's disease", *British Medical Bulletin,* vol. 86, no. 1, pp. 109–127, 2008.

[18] Jankovic J., "Motor fluctuations and dyskinesias in Parkinson's disease: clinical manifestations, " *Movement Disorders*, vol. 20, pp. S11–S16, 2005.

[19] Pulliam C. L., Heldman D.A., Brokaw E.B., Mera T.O., Mari Z.K., Burack M.A., "Continuous assessment of Levodopa response in Parkinson's disease using wearable motion sensors", IEEE Transactions on Biomedical Engineering, vol. 65, no. 1, pp. 159–164, 2018.