Evolving Deep Recurrent Neural Networks Using A New Variable-Length Genetic Algorithm

Ramya Anasseriyil Viswambaran, Gang Chen, Bing Xue and Mohammad Nekooei
School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600
Wellington 6140, NEW ZEALAND
Emails: {anasseramy, aaron.chen, bing.xue, mohammad.nekooei}@ecs.vuw.ac.nz

***Abstract*—Deep Recurrent Neural Network (DRNN) is an effective deep learning method with a wide variety of applications. Manually designing the architecture of a DRNN for any specific task requires expert knowledge and the optimal DRNN architecture can vary substantially for different tasks. This paper focuses on developing an algorithm to automatically evolve task-specific DRNN architectures by using a Genetic Algorithm (GA). A variable-length encoding strategy is developed to represent DRNNs of different depths because it is not possible to determine the required depth of a DRNN in advance. Activation functions play an important role in the performance of DRNNs and must be carefully used in these networks. Driven by this understanding, knowledge-driven crossover and mutation operators will be proposed to carefully control the use of activation functions in GA in order for the algorithm to evolve best performing DRNNs. Our algorithm focuses particularly on evolving DRNN architectures that use Long Short Term Memory (LSTM) units. As a leading type of DRNN, LSTM-based DRNN can effectively handle long-term dependencies, achieving cutting-edge performance while processing various sequential data. Three different types of publicly available benchmark datasets for both classification and regression tasks have been considered in our experiments. The obtained results show that the proposed variable-length GA can evolve DRNN architectures that significantly outperform many state-of-the-art systems on most of the datasets.**

***Index Terms*—Deep Recurrent Neural Networks, Genetic Algorithm, Long Short Term Memory**

I. Introduction

Deep Learning is a sub-domain of machine learning, which focuses on algorithms influenced by the structure, organization and function of the human brain called artificial neural networks [1]. Deep Recurrent Neural Network (DRNN) is one of the most popular artificial neural networks, enjoying a dominating performance in many application areas [2], [3]. In particular, DRNNs consisting of multiple non-linear layers have demonstrated to learn the feature representations at progressively higher levels of abstraction, leading to eminent performance in many applications, such as polyphonic music prediction [4], character-level and word-level language modelling [2], sequential MNIST classification and action recognition [5]. DRNNs can be clearly distinguished from other common neural network architectures in terms of how they accept and process input data and produce output. Specifically, DRNNs take into account previous output or historical information as inputs along with the input at the current time step. This makes DRNNs ideal to recognize characteristics of sequential data and use sequential patterns to predict the next likely scenario [6].

Designing network architecture has a huge impact on the performance of DRNNs [7]. Manual architecture design is especially labour-intensive as it depends heavily on expert domain knowledge. Meanwhile, the usual trial-and-error design process is highly resource-consuming and error-prone, which affects the successful use of DRNNs on many mission-critical applications [8], [9]. As deep learning has scaled up to more challenging tasks, architecture has become difficult to design manually [10]. Hence effective methods for automatic design of DRNN architectures are considered vital for the success of DRNNs in many domains [6].

Evolutionary Computation (EC) [11] approaches show great potential for optimizing the design of neural network architectures in a fully automated and effective way [12]. Exceptional research works on the application of different EC techniques for designing and training artificial neural networks have already achieved some success [13], [14]. Genetic Algorithms (GAs) is one of the prominent EC techniques and has been widely used in different problem domains for automatic DRNN architecture optimization [15]. By using GA, the hyper-parameters of DRNN can be adjusted automatically while optimizing the DRNN architecture. Despite promising progress [16], existing GA-based architecture design methods face several challenges for effectively designing DRNN architectures.

In fact, existing GA is designed to work on network designs with fixed depth [17]. This limits the discovery of best possible DRNN architectures, especially for complex problems. This is because the number of layers and the number of neurons per layer varies substantially for different tasks. In the absence of tremendous domain knowledge, it is impossible to determine the suitable network architecture depth in advance before using GA. In line with the understanding above a flexible variable-length encoding must be provided for GA to flexibly design DRNN architectures with varying depth. The encoding scheme must be easily processible by GA too.

While variable-length encoding can support DRNNs of any depth during the evolutionary process. It may not be a good idea to create DRNNs with a wide range of depths in the initial population. Note that the computational complexity of training very deep networks are significantly higher than that

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

of shallow networks. Meanwhile, it is easy to extend well-designed shallow networks to create high-performing deep networks. Hence a systematic approach must be proposed to progressively determine the suitable depth of DRNNs by gradually increasing the depths of designed DRNN architectures through GA evolutions. Moreover, the boundaries of the DRNN architecture has to be limited to avoid growing the network indefinitely.

Generally, DRNN architectures randomly created by GA have no restrictions on the use of activation functions. This issue is further amplified by the use of common crossover and mutation operators. However, careful use of activation functions is crucial to ensure good performance of designed DRNNs. It is well-known in the literature that many activation functions suffer from the issue of saturation [18]. Therefore unrestricted use of these activation functions can lead to DRNN architectures that are hard to train [19]. However, few works have ever attempted to identify undesirable patterns over the use of activation functions and explicitly adopt identified patterns to guide the automatic design of DRNN architectures. Our research in this paper is inspired by the idea that activation functions that are either suitable or unsuitable for designing DRNNs for a specific task can be identified by analysing DRNN samples obtained from a pre-sampling phase. Based on the analysed knowledge, restrictions on the use of activation functions can be incorporated into GA in the form of knowledge-driven crossover and mutation operators.

Motivated by the limitations of existing GA approaches, we aim to develop a new algorithm called Knowledge Driven Variable Length GA (*KD-VL-GA*) that can effectively design DRNNs of appropriate depths for a wide range of machine learning tasks. In comparison to baseline GA methods, KD-VL-GA is expected to significantly increase the chance of evolving highly trainable and effective DRNN architectures.

A. Objectives

To fulfil the goal of developing KD-VL-GA, we have three objectives as given below. Each objective is achieved through a separately technical innovation introduced in this paper.

1) Develop a new variable-length encoding scheme to flexibly represent DRNN architectures with any depth: By extending [20], this objective develops a variable-length encoding scheme to represent DRNNs of varying depths during the evolutionary process. The new encoding scheme can comprehensively encode a large collection of hyper-parameters with respect to every DRNN layer. It is also easy to transform or decode the architecture representation into real DRNN implementations based on the deep learning library TensorFlow [21] for training, evaluation and testing.

2) Propose a progressive approach for evolving DRNN architectures with gradually increasing depths: This objective proposes a new approach where the initial DRNN architectures created by GA has a minimal depth level. A progressively increasing strategy is then adopted to gradually increase the maximum allowed depth levels across many GA evolutions. This progressive approach can effectively expand effective shallow network designs into useful designs of deeper networks. It will also significantly reduce the risk of training badly designed deep networks that would consume substantial computation resources.

3) Propose a new knowledge-driven technique for crossover and mutation operators: This objective proposes knowledge-driven crossover and mutation operators to identify and repair DRNN designs affected by inappropriate use of activation functions, thereby significantly reducing the chances for GA to evolve hard-to-train DRNN architectures. We aim to study whether useful knowledge regarding activation function usage can be extracted from randomly sampled DRNN architectures and whether the extracted knowledge can effectively guide the repair of any badly designed DRNN architectures. This knowledge-driven approach is expected to prevent GA from generating a large number of low-fit DRNN designs, contributing significantly to the efficient and effective operation of KD-VL-GA.

II. Background

A. RNN

RNNs are a class of neural networks with a backward feeding link in the network of nodes to learn the dynamics of input sequence data [22]. Unlike feedforward neural networks, RNNs has a feedback loop serves as a form of memory to store historic information for a long context window. This is vital for RNN to effectively process sequence data. On the other hand, the main drawback of standard RNN is the difficulty of accessing distant information in the sequence data. If the input sequence is lengthy, the time gap between where the significant information is available and where it is required may be very large. This is called long term dependency. Long Short Term Memory (LSTM), a special type of RNN, has widely demonstrated the capability of learning long-term dependencies [23].

B. LSTM

LSTM is characterized by its ability to remember information for long periods of time [23]. LSTM extends the idea of memory in RNN by creating a long-term memory component. LSTMs also have the chain-like structure of traditional DRNNs, but the internal structure of individual units is very different. A key feature of LSTMs is its cell state, which stores past information for future references. LSTM eventually adds or removes information to and from the cell state based on its importance.

LSTM can handle continuous values, noise and distributed representations. LSTM can often make much more accurate predictions than RNN on nonlinear time-series data. It has hence been utilized to tackle many difficult problems including speech synthesis [24], audio-video data analysis [4], text generation [25], etc. In this work, GA is adopted to evolve LSTM based DRNNs.

III. Related Works

The drawbacks of GAs with fixed-length chromosome for designing network architecture is explained in [26], which

found that fixed-length chromosome can cause wastage of computational power. A GA with variable-length chromosomes for warship simulation is discussed in [27]. This work demonstrated that the variable-length GA can find better solutions with less computational cost compared with the one-time brute-force approach. Moreover, by changing the chromosome length dynamically, GA is capable of adapting to different situations. This motivates us to design a new algorithm to evolve DRNN using variable-length GA.

Apart from its depth, another very important aspect of DRNN is regarding its use of activation functions. Gulcehre et al. [18] found that several commonly used activation functions can cause training difficulties due to the saturation behaviour of these functions. Quantitative comparisons of four popular activation functions including linear, sigmoid, tanh and Gaussian has been reported in [28]. The results suggest that tanh outperforms the other three activation functions. The key findings of these research works motivated us to design a new knowledge-driven approach to carefully control the use of activation functions in DRNNs.

IV. Proposed Algorithm

This section explains the overall structure of the KD-VL-GA algorithm. It also presents detailed discussion regarding a variable-length encoding for representing evolved DRNN designs, an incremental progressive approach to update the depth of DRNNs, knowledge-driven crossover and mutation operators, population initialization and selection, and the fitness evaluation method.

A. Overview

The basic operation of KD-VL-GA is summarized in Algorithms 1. The step numbers 5 and 6 are for the pre-sampling phase which is explained in subsection *D* to facilitate the development of knowledge-driven crossover and mutation operators. The algorithm starts with an initial population of DRNN designs. Each design follows the depth set in hyper-parameter "MaxDepth", that is every DRNN architecture can have the number of layers between 3 and "MaxDepth". Successive generations will be created by GA. If no progress in the fitness can be witnessed across a number of consecutive generations, the "MaxDepth" value will be incremented by one. If no improvement of fitness can be obtained after multiple increments of "MaxDepth", the algorithm stops updating "MaxDepth" and returns the best performing DRNN design ever discovered. If there are more than one DRNN architectures with the same best fitness value but different depth, all these DRNN designs will be returned together with their respective depth (number of layers) and width (neurons per layer) settings. This progressive approach is designed for GA to efficiently use its computational power by sampling and training shallow networks initially, paving the way to build effective deep networks in later generations. Shallow networks are much more lightweight to process than deep networks.

Algorithm 1 The pseudocode of KD-VL-GA

```
1:  MaxDepth ← 3
2:  Pop ← Create the initial population, the depth of each
    DRNN bounded by MaxDepth
3:  Calculate the fitness of every DRNN in Pop
4:  for Generations 1 to N for every MaxDepth value do
5:      if decision tree (DT) is not available
            Put all DRNNs in the Pop to the sample set
            Pop ← Create next generation using standard
            crosover and mutation operators
            Calculate the fitness of every DRNN in Pop
6:      if sufficient samples have been collected and DT is
        not available
            Prepare learning examples for DT from all
            sampled DRNNs and construct the DT
7:      Pop ← Create next generation by knowledge-driven
        crossover and mutation operators, governed by
        MaxDepth restriction
8:      Calculate the fitness of every DRNN in Pop
9:      if no improvement of fitness is witnessed in past X
        generations
10:         if no improvement of fitness is witnessed in past
            Y updates of MaxDepth
11:             Select and return the fittest individual from
                Pop as the evolved architecture by GA
12:         else
13:             MaxDepth ← MaxDepth + 1
14: end for
```

B. The variable-length encoding scheme

We develop a new variable-length encoding by extending the fixed-length encoding scheme proposed in [20]. The key difference lies in the fact that with the variable-length encoding scheme, the number of design blocks in different GA chromosomes are not the same because each block represents one layer of the network. Since different architecture can have a different number of layers, the length of the encoded DRNN designs varies. We have followed the same method discussed in [20] to represent the hyper-parameters and their types within each design block in a binary format.

Table I lists all the hyper-parameters covered by the encoding scheme. For each hyper-parameter, its corresponding integer representation is written in the bracket followed. Table II further presents the acceptable integer value range for each hyper-parameter, the corresponding parameter types and the number of binary bits utilized to encode their values. Both LSTM and dense layers are having the same set of parameters, therefore the type information in Table II is used to differentiate which layer the parameters belongs to. More explanations and examples of encoding can be found in [20].

C. Progressive increment to DRNN depth during GA evolution

The suitable depth for DRNN design with respect to any machine learning tasks is often unknown in advance. On one hand, shallow networks are not capable of handling compli-

cated tasks. On the other hand, networks with high depth are overkill for simple tasks and they can also cause a high computational burden. KD-VL-GA is designed to gradually increase the depth of the designed DRNNs and hence the length of the corresponding chromosomes. To dynamically adjust the length of the chromosome, either a top-down or bottom-up approach can be used. The top-down approach starts from a very long chromosome and gradually decreases the length and the bottom-up approach starts with short chromosomes and increases the length gradually. We have decided to follow the bottom-up approach because of two main reasons: (1) creating and evaluating shallow networks incur low computation cost, enabling fast design of high-quality shallow networks in early generations of KD-VL-GA; and (2) the designed shallow networks can further facilitate the design of good deep networks, mitigating the risk of spending substantial computation resources to evaluate ill-designed deep networks.

In line with the bottom-up approach, KD-VL-GA creates an initial population with minimal depth for all DRNNs and searches for best-performing DRNN architectures in the simple design space. Trainable shallow networks can be created easily compared to deep networks. Through this incremental approach, linear increment to the maximum depth of evolved DRNNs will be applied from time to time to avoid substantial change to the performance of these DRNNs. In fact, a six-layer network obtained by directly mixing the design of two DRNNs with three layers usually does not perform well in our preliminary experimental studies.

It is important to limit the initial search space of GA and minimize the computational cost required for fitness evaluation, ensuring that good shallow network designs can be quickly identified as the basis for building more effective but more complicated deep networks. Creating many networks and selecting the trainable ones are affordable for shallow networks because they can be evaluated quickly. But deep networks have to be created carefully because its computational complexity is much higher compared to shallow network. Therefore good trainable shallow networks can be used to create deep networks. With the incremental approach, KD-VL-GA is expected to find the best performing DRNN architectures with respect to each depth level in order to identify the most suitable depth setting for any machine learning task.

D. Knowledge-driven crossover and mutation operators

Following the algorithm overview, we introduce a pre-sampling phase to collect a group of sampled DRNN designs in order to analyse the trainability of DRNNs. In the pre-sampling phase, a sufficient number of trainable and untrainable DRNNs must be created. They must also allow us to determine the suitability of using any activation functions. The total number of sampled DRNNs is considered sufficient, subject to two conditions: (1) over 200 trainable and untrainable DRNNs can be found in the sampled collection; and (2) each activation function appears at least once in the input layer, intermediate layer and output layer of any sampled DRNN designs. Based on these conditions, we found that the number of samples required for each benchmark dataset (see Section V. A.) varies from 500 to 600.

TABLE I
SELECTED HYPER-PARAMETERS AND ITS POSSIBLE VALUES CHOSEN FOR GA TO EVOLVE DRNN ARCHITECTURES

Parameter	Possible values
Activation	Softsign(0), Elu(1), Relu(2), Selu(3), Tanh(4), Softplus(5), Sigmoid(6),Hard_Sigmoid(7), Softmax(8), Exponential(9), Linear(10)
Initializer	Glorot_normal(0), Glorot_uniform(1), He_normal(2), He_uniform(3), Lecun_uniform(4), Lecun_normal(5), Normal(6),RandomNormal(7), RandomUniform(8), TruncatedNormal(9), VarianceScaling(10),Uniform(11),
Constraint	MaxNorm(0), UniNorm(1), MinMaxNorm(2)
Dropout	0.1(0), 0.105(1), 0.11(2), 0.115(3), 0.12(4), 0.125(5), 0.13(6), 0.135(7), 0.14(8), 0.145(9), 0.15(10), 0.155(11), 0.16(12), 0.165(13), 0.17(14), 0.175(15)

TABLE II
PARAMETERS OF DRNN SELECTED FOR TUNING AND ITS ASSOCIATED INFORMATION

Layer	Parameter	Range	No of Bits in binary	Type
LSTM	Output space	[0, 511]	9	1
LSTM	Activation	[0, 15]	4	2
LSTM	Initialization	[0, 15]	4	3
LSTM	Constraint	[0, 3]	2	4
LSTM	Dropout	[0, 15]	4	5
Dense	Output space	[0, 511]	9	6
Dense	Activation	[0, 15]	4	7
Dense	Initialization	[0, 15]	4	8
Dense	Constraint	[0, 3]	2	9
Dense	Dropout	[0,15]	4	10

To extract useful knowledge from the sampled DRNNs to guide KD-VL-GA to identify untrainable DRNN designs, we adopt a decision tree (DT) based approach for analyzing these samples [29]. DT is suitable for our algorithm because it presents the precise conditions for a DRNN to be untrainable. These conditions will be explicitly utilized by KD-VL-GA to repair this DRNN to make it trainable. To build the DT, we prepare the learning examples by extracting three types of features from the sampled DRNNs. The first type gives the frequency of using each activation function in the DRNN. The second type captures the location of the activation function in the DRNN, with three possibilities, i.e, located in the input layer, intermediate layer, or output layer of the DRNN. Finally, the third type indicates whether the activation function is used in any LSTM layer. Based on all the learning samples, the DT can be constructed automatically by using a well-established algorithm, such as C4.5 [29]. An example representation of features to create DT is (Elu:1, Elu:Input, Elu:LSTM), that is the activation function Elu is used once in a DRNN design and its position is in the input LSTM layer.

One example DT obtained in our study on UCI-Smartphone dataset has been presented in Fig. 1. As shown in this example, a DRNN is considered untrainable if the activation "Exponential" is used in LSTM layers or more than once. Driven by these conditions defined in the DT, we can perform the repair mechanism on problematic DRNN designs evolved by

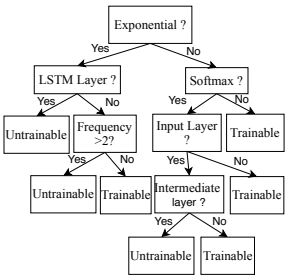

Fig. 1. Sample Decision Tree based on UCI-Smartphone dataset

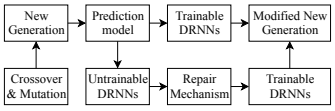

Fig. 2. General concept of predicting trainability of DRNNs and correction mechanism

KD-VL-GA. The repair process is embedded in the crossover and mutation operators. For example, whenever a new DRNN design is generated from the crossover operator, it will be checked against each condition in the DT. If it violates one condition, such as a higher frequency of usage of specific activation function, we will repair the DRNN to meet this condition by lowering the use of the specific activation function. Other conditions in the DT can be checked and handled similarly. To avoid spending too much time on repairing the DRNN, if we have performed up to 3 rounds of repair but still cannot pass all the conditions in the DT, the DRNN will be discarded and the crossover operator will be performed one more time. The general overview of the process is illustrated in Fig. 2. Similarly, the repair process is realized in the mutation operator too and the details will be omitted here.

All sampled networks for knowledge extraction are shallow networks (i.e., the depth of these networks is bounded above by 3). We prefer shallow networks because they are computationally efficient to verify their trainability. However, it is a question whether the knowledge extracted from shallow networks can guide KD-VL-GA to accurately determine the trainability of deep networks. To answer this question, we applied the DT to a group of randomly generated DRNNs with the depth ranging from 3 to 15. We found that DT can give accurate predictions on the trainability of these DRNNs, with the accuracy as high as 95%. Moreover, experimental comparisons have been performed in between two settings of KD-VL-GA, one uses DT in the crossover and mutation operators and one uses the standard version of these operators. Relevant experiment results can be found in Fig. 3 to 6 that confirm the performance advantage of using knowledge-driven crossover and mutation operators.

In order to clarify the feasibility of applying knowledge learned from shallow networks to deep networks, we have compared the performance of improper activations in some randomly generated DRNNs and found that they are making both shallow and deep networks untrainable. For example, we found that the frequent use of Linear activation in the hidden layers of DRNNs of three layers, seven layers and nine layers makes it untrainable for UCI-Smartphone dataset. Experiments conducted with and without using knowledge-driven crossover and mutation operators also shows a difference in the progress of successive generations of GA. Filtering the use of activation functions help to refine the search space of GA and increase the chances of generating good DRNNs. This also helps to reduce the computational cost by bypassing training of unfit DRNNs.

E. Population Initialization and Selection

To generate the initial population, the hyper-parameters with respect to each DRNN design will be determined randomly. We apply the restriction to make sure that these randomly initialized DRNNs must have at least one LSTM layer. Meanwhile, the output layer must be a dense layer.

Every DRNN in the initial population will be ranked according to their performance on the given task. The best performing DRNN will be passed directly to the next generation without any modification through elitism. The remaining DRNNs of the next generation will be created by crossover and mutation based on the selected parents. Parents are selected using the roulette wheel selection algorithm [30]. The probability of selecting any DRNN as a parent is proportional to its fitness. Roulette wheel selection is known to be suitable for automatic network architecture design since it can maintain reasonable diversity in the GA population.

F. Fitness Evaluation

For fitness calculations, the DRNN architecture will be constructed by decoding the whole chromosome into a Tensorflow-based DRNN implementation. For the purpose of training and evaluation, the dataset is split into two parts, a train data and a test data, and the train data is further split into two parts, the training data and fitness evaluation data. Training data is used to train the neural network, and the fitness evaluation data is used to measure the fitness in terms of classification accuracy or prediction error of the trained DRNN. During fitness evaluation, an adaptive technique is used to train a DRNN. In particular, we adopt the early stop mechanism supported by Tensorflow to stop the training of DRNNs based on the progress of fitness on evaluation data.

V. Experiment Design

This section explains the details of datasets of different types, performance metrics used and general settings of the algorithm.

A. Benchmark Datasets

We have used three different types of datasets, human activity recognition (HAR), solar flare prediction, and air quality prediction. Three different types of datasets are selected to verify the general applicability of KD-VL-GA.

1) Human Activity Recognition: HAR is a challenging broad field of study concerned with identifying the specific action or movement of a person. Here we are using three publicly available benchmark datasets on HAR. These three datasets are chosen because it includes a variety of indoor and outdoor activities. They include balanced and unbalanced datasets with attribute range 9 to 561.

a) UCI-Smartphone, Human Activity recognition dataset created using a smartphone. Experiments have been conducted with 30 people for six activities.

b) UCI-FOG, Daphnet Freezing of Gait dataset. This dataset contains information on Parkinson's disease patients experiencing freezing of gait (FOG) during walking. There are two activities, Freeze and Normal.

c) SKODA Mini Checkpoint dataset. This dataset contains 11 activities performed in a car maintenance scenario.

2) Solar Flare Prediction: A Solar flare is the rapid release of a large amount of energy stored in the solar atmosphere. Solar flares are classified as A, B, C, M or X according to the peak flux of 1 to 8 Angstroms. This work is predicting whether an active region would produce a γ class flare in one day.

a) NCEI, National Centers for Environmental Information dataset contains records of flare events occurred from 2010 May to 2018 May. Three γ classes considered for prediction are $\geqslant$ M5.0 class, $\geqslant$ M class, and $\geqslant$ C class.

3) Air Quality Prediction: Air pollution refers to the contamination of the atmosphere by harmful chemicals or biological materials. Three publicly available benchmark datasets are used.

a) UCI Beijing PM2.5. This hourly dataset stores the information on particle matter (PM) 2.5 of the US embassy in Beijing and meteorological data from Beijing Capital International Airport between January 1st, 2010 to December 31st, 2014. The evolved DRNN is predicting the concentration of PM2.5 in the air.

b) UCI Italy. The data was recorded from March 2004 to February 2005 from an Italian city. The evolved DRNN is predicting the concentration of CO, NO2 and NOx.

c) UCI PM2.5 Data of three Chinese Cities. This hourly dataset contains the PM2.5 data of three cities Beijing, Shanghai, and Shenyang. Data recorded during the period January 1st, 2010 to December 31st, 2015. The task is to predict the concentration of PM2.5 in these three cities.

Summary of all the above-mentioned dataset is listed in Table III. For all the dataset, 80% of the data is used for training and 20% is used for testing. 33% of the training data is used for fitness evaluation.

TABLE III
SUMMARY OF DATASETS

Dataset	Samples	Attributes
UCI-Smartphone	10299	561
UCI-FOG	1029470	9
SKODA	22000	390
NCEI-C Class	155739	14
NCEI-M Class	155739	22
NCEI-M5 Class	155739	20
UCI Beijing PM2.5	43799	8
UCI Italy	8991	12
UCI PM2.5 Beijing	59351	10
UCI PM2.5 Shanghai	34774	10
UCI PM2.5 Shenyang	25881	10

B. Performance Metrics

Classification accuracy is used for evaluating the performance of the model on HAR and Solar flare prediction because it is a classification problem. Root Mean Square Error (RMSE) of the model is used for fitness measure on Air quality prediction tasks because it is a regression problem.

C. General Settings

We have set the minimum and maximum values for the depth of the evolved DRNNs as 3 and 15 because we found that GA is terminating with DRNNs of layers less than 15 for different tasks. For every depth, GA evolves a minimum 5 and maximum of 20 generations. That is for every depth value, GA evolves for a minimum of 5 generations and checks the progress. If no progress found for 5 generations, it increments the depth value, otherwise, it continues to evolve to the next generation. We found that GA can not find better solutions in the successive generation if it can not find improvements in the last 5 generations and GA often increments the depth value before reaching 20 generations. GA stops incrementing the depth if no progress in the fitness of DRNNs found for the last 5 increments of depth value. Each generation consists of 60 individuals. The crossover rate used is 0.9 and the mutation rate is 0.3.

Two-point crossover is used and the crossover points are randomly selected. Meanwhile, the new DRNN designs created through crossover cannot violate the maximum and minimum depth restrictions. The mutation is done as a bit flipping on the binary representation. The optimizer used for training evolved DRNNs for fitness evaluation is Adam. The settings used for Adam optimizer is learning rate 0.001, decay 1e-2, clipnorm 1.0 and clipvalue=5.0 and bias regularizer for LSTM is L1L2(l1=0.008, l2=0.0).

VI. Experimental Results

We found that the influence of activation functions on the trainability of DRNNs is based on the specified task. For example, the activation functions Exponential, Linear, Sigmoid, Hard_sigmoid, and Softplus could not be used too often for classification on UCI-Smartphone data, so that their usage must be carefully controlled using the repair mechanism. The activation functions making good trainable DRNNs for UCI-Smartphone data are Elu, Selu, Relu, Tanh and Softsign.

A. HAR

The evolved DRNN architectures of depth 7, 4 and 5 yield the best results for the UCI-Smartphone, UCI-FOG and SKODA dataset respectively. The overall accuracy of all these datasets is given in Table IV.

1) Comparison between General GA and GA with filtering approaches: We compared the performance of KD-VL-GA, general variable-length GA (VL-GA) approach without imposing any restriction on the use of activation functions (i.e. using standard crossover and mutation operators) and non-flexible variable-length GA (NonFlexible-VL-GA) in terms of obtained classification accuracy. The NonFlexible-VL-GA creates DRNNs of any depth between the lower and upper limit

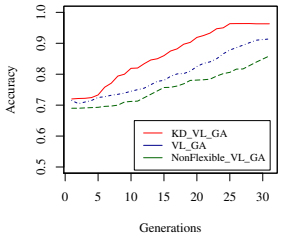

Fig. 3. Progress of fitness of various methods on UCI-Smartphone dataset.

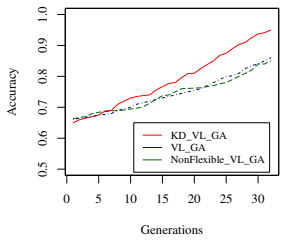

Fig. 4. Progress of fitness of various methods on UCI-FOG dataset

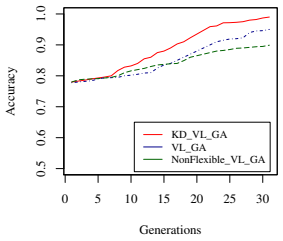

Fig. 5. Progress of fitness of various methods on SKODA dataset.

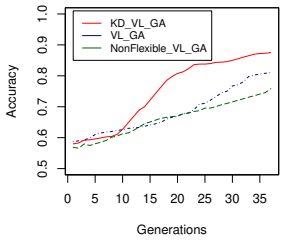

Fig. 6. Progress of fitness of various methods on NCEI-C Class dataset

from the initial population onwards. In this work, the lower and upper limits used are 3 and 15. Graphs showing the progress of fitness over generations based on these three methods on 3 HAR datasets and NCEI-C class dataset on Solar flare are shown in Fig. 3, Fig. 4, Fig. 5 and Fig. 6 respectively. The results show that KD-VL-GA could generate good trainable DRNNs of comparatively high accuracy.

For the UCI-Smartphone dataset, the best accuracy obtained by KD-VL-GA is 96.33%, the best accuracy obtained by VL-GA approach is 91.4% and the best accuracy achieved using NonFlexible-VL-GA is 86%. For the UCI-FOG dataset, the best accuracy obtained by KD-VL-GA is 95%, the best accuracy obtained by VL-GA approach is 86% and the best accuracy achieved using NonFlexible-VL-GA is 85%. For the UCI-Skoda dataset, the best accuracy obtained by KD-VL-GA is 99.6%, the best accuracy obtained by VL-GA approach is 95% and the best accuracy achieved using NonFlexible-VL-GA is 90%.

B. Solar Flare Prediction

Prediction result of the evolved DRNN architectures for the three classes of flares i.e.$\geqslant$ M5.0 class, $\geqslant$ M class and $\geqslant$ C class are shown in Table IV. The best result achieved is 99.6% for the $\geqslant$ M5.0 class flare. Highest accuracy earned for the $\geqslant$ M class flare is 98.4%, and the best accuracy for the $\geqslant$ C class flare is 88.5%. For all the three classes of flares, DRNNs of 4 layers generated the best results.

TABLE IV
COMPARATIVE ACCURACY OF THE KD-VL-GA AGAINST OTHER MODELS

Dataset	Competing Algorithms	KD-VL-GA (Accuracy %)	Time in hours
Smartphone	95.5 [31], 95.0 [32]	**96.00**±0.33	177 ± 05.57
UCI-FOG	93.0 [32], 94.0 [33]	94.00±1.00	523 ± 13.30
SKODA	96.6 [31], 96.0 [32]	**99.00**±0.60	174 ± 05.59
NCEI-C Class	8.03 [34], 82.0 [35]	**87.00**±1.50	432 ± 14.70
NCEI-M Class	90.0 [34], 92.0 [35]	**98.00**±0.40	440 ± 19.10
NCEI-M5 Class	90.0 [34], 88.0 [35]	**99.00**±0.60	437 ± 18.20
Dataset	Competing Algorithms	KD-VL-GA (RMSE)	Time in hours
UCI-Beijing PM2.5	14.50 [36], 15.60 [36]	**11.92**±0.85	192 ± 05.32
UCI-Italy CO	01.96 [37], 01.72 [37]	**00.33**±0.20	102 ± 06.61
UCI-Italy NO2	149.8 [37], 152.8 [37]	**12.25**±3.50	097 ± 04.64
UCI-Italy NOx	94.40 [38], 83.60 [38]	**27.16**±3.00	092 ± 05.70
PM2.5 Beijing	16.55 [39], 56.70 [40]	**12.40**±2.30	119 ± 05.70
PM2.5 Shanghai	08.55 [39], 28.00 [40]	08.00±0.75	124 ± 08.19
PM2.5 Shenyang	20.10 [39], 59.00 [40]	**12.00**±1.50	122 ± 07.92

C. Air Quality Prediction

DRNNs with 4 layers evolved by GA yield the best result for the UCI- Beijing PM2.5 dataset. The evolved DRNNs of 4 layers achieved the best results on all the three UCI-Italy datasets, CO, NO2 and NOx. DRNNs of 5 layers achieved the best results on all the three UCI-PM2.5 datasets, Beijing, Shanghai and Shenyang. RMSE obtained on all the given datasets are listed in Table IV.

D. Comparisons with recently published results

Table VI lists the obtained results of different tasks by KD-VL-GA and the best results published in the latest works on the same dataset [31]–[42]. The obtained results shows that DRNN evolved by KD-VL-GA performs consistently better compared to the other state-of-the-art approaches [31]–[42]. Our KD-VL-GA could increase accuracy and decrease RMSE over a good margin.

The classification accuracy and prediction accuracy on different datasets demonstrate that KD-VL-GA could evolve DRNN architecture which can produce competitive results. We found that the KD-VL-GA approach is performing well on different tasks.

All the experiments are conducted on GPUs with NVIDIA GeForce RTX 2080 cards. Execution time in hours to evolve DRNNs for each dataset by KD-VL-GA is listed in Table IV.

VII. CONCLUSIONS

The proposed KD-VL-GA method features the use of variable-length encoding along with the incremental progressive approach to automatically evolve DRNNs. We successfully developed a GA-based approach with knowledge-driven crossover and mutation operators to create trainable DRNNs to better use GA for evolving architectures of DRNNs.

We found that KD-VL-GA is general enough to evolve DRNNs for different tasks. We have verified its applicability on three different types of datasets for classification and regression tasks. GA is found to be an efficient tool to tune the hyperparameters of DRNNs. The proposed variable-length encoding scheme supports the use of the proposed GA operators and

refines the search space by gradually increasing the depth of DRNNs. Deeper networks could be generated effectively from trainable shallow networks. Computation complexity could be reduced by avoiding untrainable DRNNs through knowledge-driven GA operators.

Future work will focus on automated analysis of DRNNs to find commonalities in those DRNNs which are not trainable. We will also find a method to apply the analysis on the trainability of DRNNs throughout the evolutionary process not only on the initial shallow networks. We will also examine the usefulness of our GA based approach with new operators on other important machine learning tasks.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[3] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review," *Artificial Intelligence Review*, vol. 39, no. 3, pp. 251–260, 2013.

[4] R. M. Cichy and D. Kaiser, "Deep neural networks as scientific models," *Trends in Cognitive Sciences*, vol. 23, no. 4, pp. 305–317, 2019.

[5] S. Li, W. Li, C. Cook, Y. Gao, and C. Zhu, "Deep independently recurrent neural network (indrnn)," *arXiv preprint arXiv:1910.06251*, 2019.

[6] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[7] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[8] M. Schrimpf, S. Merity, J. Bradbury, and R. Socher, "A flexible approach to automated rnn architecture generation," *arXiv preprint arXiv:1712.07316*, 2017.

[9] J. Lee and I. Tashev, "High-level feature representation using recurrent neural network for speech emotion recognition," in *Sixteenth annual conference of the international speech communication association*, 2015.

[10] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

[11] D. B. Fogel, "What is evolutionary computation?" *IEEE spectrum*, vol. 37, no. 2, pp. 26–32, 2000.

[12] R. Akut and S. Kulkarni, "Neuroevolution: Using genetic algorithm for optimal design of deep learning models." in *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, 2019, pp. 1–6.

[13] S. Mirjalili, "Evolutionary algorithms and neural networks," *Studies in Computational Intelligence*, 2019.

[14] S. Chopra, D. Yadav, and A. Chopra, "Artificial neural networks based indian stock market price prediction: Before and after demonetization," *J Swarm Intel Evol Comput*, vol. 8, no. 174, p. 2, 2019.

[15] H. Chiroma, S. Abdulkareem, A. Abubakar, and T. Herawan, "Neural networks optimization through genetic algorithm searches: a review," *Appl. Math*, vol. 11, no. 6, pp. 1543–1564, 2017.

[16] C.-J. Chen and T.-C. Chen, "Design of recurrent neural network power system stabilizer based on genetic algorithm," in *Proceedings of the IASTED International Conference Intelligent System and Control*, 2008, pp. 284–289.

[17] Y. Matsuoka, H. Aizawa, J. Sato, and K. Kato, "Training time reduction for network architecture search using genetic algorithm," in *Fourteenth International Conference on Quality Control by Artificial Vision*, vol. 11172. International Society for Optics and Photonics, 2019, p. 111720O.

[18] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, "Noisy activation functions," in *International conference on machine learning*, 2016, pp. 3059–3068.

[19] A. Marchisio, M. A. Hanif, S. Rehman, M. Martina, and M. Shafique, "A methodology for automatic selection of activation functions to design hybrid deep neural networks," *arXiv preprint arXiv:1811.03980*, 2018.

[20] R. A. Viswambaran, G. Chen, B. Xue, and M. Nekooei, "Evolutionary design of recurrent neural network architecture for human activity recognition," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 554–561.

[21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[22] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[23] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, 1997, pp. 473–479.

[24] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[25] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[26] S. N. Pawar and R. S. Bichkar, "Genetic algorithm with variable length chromosomes for network intrusion detection," *International Journal of Automation and Computing*, vol. 12, no. 3, pp. 337–342, 2015.

[27] K.-m. Park, S.-h. Shin, D. Shin, S.-d. Chi *et al.*, "Design of warship simulation using variable-chromosome genetic algorithm," *Applied Sciences*, vol. 9, no. 19, p. 4131, 2019.

[28] D. L. Tong and R. Mintram, "Genetic algorithm-neural network (gann): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 75–87, 2010.

[29] J. Liu, B. Ning, and D. Shi, "Application of improved decision tree c4. 5 algorithms in the judgment of diabetes diagnostic effectiveness," in *Journal of Physics: Conference Series*, vol. 1237, no. 2. IOP Publishing, 2019, p. 022116.

[30] L. Zhang, H. Chang, and R. Xu, "Equal-width partitioning roulette wheel selection in genetic algorithm," in *2012 Conference on Technologies and Applications of Artificial Intelligence*. IEEE, 2012, pp. 62–67.

[31] X. Zhang, Y. Wong, M. S. Kankanhalli, and W. Geng, "Hierarchical multi-view aggregation network for sensor-based human activity recognition," *PloS one*, vol. 14, no. 9, 2019.

[32] F. M. Rueda and G. A. Fink, "Learning attribute representation for human activity recognition," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 523–528.

[33] A. Murad and J.-Y. Pyun, "Deep recurrent neural networks for human activity recognition," *Sensors*, vol. 17, no. 11, p. 2556, 2017.

[34] H. Liu, C. Liu, J. T. Wang, and H. Wang, "Predicting solar flares using a long short-term memory network," *The Astrophysical Journal*, vol. 877, no. 2, p. 121, 2019.

[35] G. Barnes, K. Leka, C. Schrijver, T. Colak, R. Qahwaji, O. Ashamari, Y. Yuan, J. Zhang, R. McAteer, D. Bloomfield *et al.*, "A comparison of flare forecasting methods. i. results from the "all-clear" workshop," *The Astrophysical Journal*, vol. 829, no. 2, p. 89, 2016.

[36] Q. Tao, F. Liu, Y. Li, and D. Sidorov, "Air pollution forecasting using a deep learning model based on 1d convnets and bidirectional gru," *IEEE Access*, vol. 7, pp. 76 690–76 698, 2019.

[37] S. R. Patra, "Time series forecasting of air pollutant concentration levels using machine learning."

[38] H. Liu, Q. Li, D. Yu, and Y. Gu, "Air quality index and air pollutant concentration prediction based on machine learning algorithms," *Applied Sciences*, vol. 9, no. 19, p. 4069, 2019.

[39] D. Liu and K. Sun, "Short-term pm2. 5 forecasting based on ceemd-rf in five cities of china," *Environmental Science and Pollution Research*, vol. 26, no. 32, pp. 32 790–32 803, 2019.

[40] X. Feng, Q. Li, Y. Zhu, J. Hou, L. Jin, and J. Wang, "Artificial neural networks forecasting of pm2. 5 pollution using air mass trajectory based geographic model and wavelet transformation," *Atmospheric Environment*, vol. 107, pp. 118–128, 2015.

[41] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, "A deep learning approach to on-node sensor data analytics for mobile or wearable devices," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 56–64, 2016.

[42] ——, "A deep learning approach to on-node sensor data analytics for mobile or wearable devices," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 56–64, 2017.