Received 20 June 2022, accepted 16 August 2022, date of publication 26 August 2022, date of current version 6 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3201882

RESEARCH ARTICLE

A Machine-Learning-Based Labelling Diversity Model for Predictive Analysis: Using 16QAM as a Case Study

SHAHEEN SOLWA[1], MOHAMED K. ELMEZUGHI[1], (Student Member, IEEE), OMRAN SALIH[2], ALI M. ALMAKTOOF[3], AND M. T. E. KAHN[3]
[1]Discipline of Electrical, Electronic and Computer Engineering, University of KwaZulu-Natal, Durban 4041, South Africa
[2]Department of Mathematics and Computer Science, Alzaiem Alazhari University, Khartoum 12217, Sudan
[3]Department of Electrical, Electronic and Computer Engineering, Cape Peninsula University of Technology, Cape Town 7535, South Africa
Corresponding author: Shaheen Solwa (shaheensolwa786@gmail.com)

ABSTRACT The recent advancement and enhancement that optimized uncoded space-time labelling diversity (USTLD) have provided significant diversity gains. By adopting the use of evolutionary algorithms, labelling diversity (LD) mapper designs produced are near-optimal in quality. The only disadvantage to the use of evolutionary algorithms is that the produced solution is not always optimal. To ease the calculation of how much a mapper design had achieved LD, this paper proposes a machine learning-based analysis to predict the amount of LD achieved by a mapper. In this paper, only the 16QAM constellation is studied as a simple case. Six machine learning-based algorithms were proposed in this paper, namely multi-linear regression (MLR), support vector regression (SVR), decision trees (DT), random forest (RF), K-nearest neighbours (KNN) and a simple artificial neural network (ANN). From the results obtained from the experiments, it can be seen that the MLR algorithm is the least time complex while the ANN is the most time complex. It is also important to note that the DT and KNN algorithms take a comparatively short amount of time to execute. When compared in terms of machine learning metrics, it was shown that the ANN algorithm performed the best with the least amount of error while the MLR algorithm performed the worst with the highest amount of error. Thus, it could be seen that the results from this paper provide a positive outlook on applying machine learning algorithms to the LD problem.

INDEX TERMS Labelling diversity, machine learning, mean square error, neural networks, predictions.

I. INTRODUCTION

The quadratic assignment problem (QAP) is a complex combinatorial problem that aims to assign “a” number of points to “b” number of locations while minimizing the total computational cost. The QAP is considered a NP-hard problem [1]. The QAP can be used to solve a variety of problems such as wiring problems in electronics, scheduling, transportation and routing and sports. In the real world, different variations of the QAP have been introduced to solve problems such as the biquadratic assignment problem (BiQAP) and the multi-objective quadratic assignment problem (mQAP) [1]. Recently, an application of the QAP was applied to wireless communications particularly in the field of constellation design for different communication systems.

Uncoded space-time labelling diversity (USTLD) is a recent space-time bloc code (STBC) scheme that achieves improved diversity gains over existing STBC systems such as the Alamouti STBC [2]. The major improvement in performance is due to USTLD systems exploiting space diversity (SD) and labelling diversity (LD) by using mapper designs of different arrangements. The objective of LD mapper design is to place adjacent points on a second constellation as far apart as possible than in its base constellation [3]. Hence, his can be seen as an instance of the QAP [4].

The amount of LD achieved by a mapper design depends on the algorithm or heuristic used to design the LD mappers. Prior to the proposal of USTLD, Samra *et al.* [5] had proposed

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

a brute-force algorithm that solved the QAP – subsequently the LD problem – for the 16QAM and 16PSK constellations. However, [5] reported that the computational complexity of the algorithm was too high to be applied to higher order modulation techniques. Seddik *et al.* [6] had also proposed an algorithmic approach to designing LD mappers for the 16QAM constellation, similar to the algorithm found in [5].

When the USTLD system was proposed, the authors had also provided a heuristic mapper design to produce LD mappers [3]. The approach could only be used for symmetrical constellations [4]. However, this heuristic proved to be optimal for the 16QAM, 32QAM and 64QAM constellations. Coupled with the system model and mapper design, [3] had also provided a design metric to evaluate LD mapper designs. Other heuristic works include bit-flipping of constellation points and constellation transforms [7] which were shown to have equal performance to the heuristic in [3]. Hence, the use of heuristic algorithms have the following limitations; i) symmetry-based heuristics cannot be applied to asymmetric constellations, ii) heuristic algorithms may not be able to produce good or optimal DL mapper designs and iii) algorithmic approaches are too complex in terms of the computation required.

More recently, the authors of [4] and [8] have proposed meta-heuristic algorithms in the form of a genetic algorithm (GA). GAs imitate the process of evolution and survival of the fittest in nature [4]. The power of GAs are in its approach to produce solutions quick and efficiently by using naturally-inspired techniques such as ‘mating’. This is performed over a number of iterations until a good, stable solution is found. A GA can produce a solution of either a local optimal or global optimal, which depends on the ‘mating’ technique used [8], [10]. In the case of the GA found in [4], LD mapper designs have illustrated significant diversity gains from approximately 3 dB upto 17 dB. The GA found in [8] was able to improve upon the LD values achieved by the GA in [4], but small performance gains of approximately 0.5dB up to 4 dB were achieved. Leveraging on the performance of the aforementioned evolutionary algorithms, another subset of AI, machine learning, can be applied to the LD problem.

Machine learning (ML) is a subset of artificial intelligence (AI) that allows computers to automate data-driven model programming and produce models that can detect patterns and sequences in pre-processed data [2]. The first practical use of ML was created by Ross who made an attempt to mimic a living being’s behaviour [9]. In 1959, Samuel [10] refined the definition of ML as “a field of study that gives a computer the ability to learn without being explicitly programmed”. ML has many applications in the real world, such as image recognition, stock prediction and business modelling. There are five main branches of ML, namely supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning and inductive learning. Supervised and unsupervised learning are polar opposites of each other. Supervised learning is used when input data and target values are known while unsupervised learning has input data without target values [2]. Unsupervised learning looks for “closeness” within data, for example a similar sequence with a similar difference between each term. Semi-supervised learning combines the qualities of supervised and unsupervised learning, hence being able to learn from both data that is classified as well as unclassified [2]. Reinforcement learning is a reward-punishment type of learning algorithm that learns through the interaction within an environment [3]. Just as a feedback loop, reinforcement learning gets feedback about the accuracy obtained by a response [3]. The last category, inductive learning, is a ML technique that learns based on previous knowledge with their own inductive bias [11]. Inductive learning is a recent addition to the ML subset.

A. MOTIVATIONS AND CONTRIBUTIONS

The application of meta-heuristic algorithms to the LD problem had produced significant advantages such as reduced complexity [8] and improved error performance [8], [9]. Hence, in this paper the authors’ apply another subset of artificial intelligence called machine learning (ML) which is applied to the LD problem to predict the amount of LD achieved. This is a new, novel approach for the LD problem to this best of the authors’ knowledge. When compared to designing LD mappers using meta-heuristics, predicting the amount of LD achieved by a mapper reduces the time needed to determine how a LD mapper design will perform and how does it compare with other LD mappers of the same modulation. Hence, the novel contributions of this paper include:

- A supervised machine learning-based approach that predicts the amount of LD achieved is proposed for the 16QAM constellation dataset.
- An artificial neural network-based approach that recognizes the patterns within the data for the 16QAM constellation dataset for prediction analysis is proposed.
- Machine learning techniques and the artificial neural network are compared in terms of five different statistical metrics.
- The graphs of accuracy, error and predictions are presented.
- Evaluate the feasibility of using ML-based algorithms for predicting LD mapper designs.

The applications of this paper can be summarized as follows:

- A quick, feasible solution can be determined by predicting the amount of LD achieved.
- A graphical user interface (GUI) can be designed to easily predict the amount of LD achieved.
- This novel approach can be used as a stepping stone for using ML and DL methods for optimization.

B. STRUCTURES AND NOTATIONS

The structure of the paper is as follows: section 2 briefly outlines the USTLD system model together with its theoretical performance analysis and mapper design metric. Section 3 describes how the raw data is collected, processed and setup for the machine learning application. This section

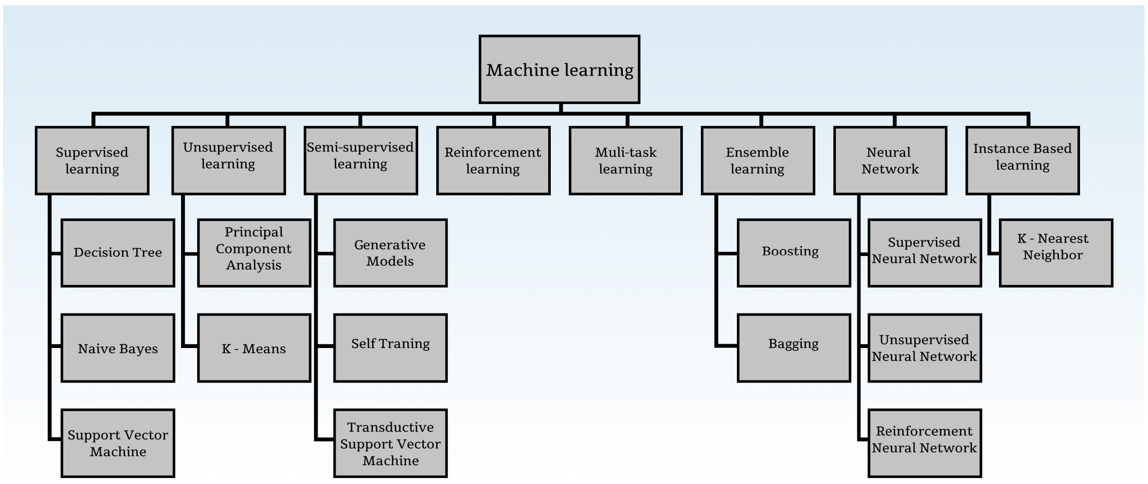

FIGURE 1. **Summarized diagram of available machine learning techniques and algorithms.**

also includes the five ML-based metrics used for statistical comparison and the hyper-parameter tuning for each ML algorithm. Section 4 outlines the ML models used in this paper, as well as the hyper-parameters set for each algorithm. Section 5 describes the results and discusses the implications of using ML-based models. Finally, the concluding thoughts are mentioned in section 6.

II. THE USTLD SYSTEM MODEL

The conventional USTLD system considers $N_T = 2$ and N_R number of receiver antenna in a MIMO configuration. Two information bitstreams $m_1 = [m_{1,1}, m_{1,2}, \ldots, m_{1,r}]$ and $m_2 = [m_{2,1}, m_{2,2}, \ldots, m_{2,r}]$, each of length $r = log_2 M$ are mapped using two different binary mappings Ω_1 and Ω_2 which produces the symbols $\Omega_1(m_1)$ and $\Omega_2(m_2)$. The two symbols that are produced after mapping are transmitted over two time slots. Hence, the resulting $N_R \times 1$ transmission vector is given as:

$$\boldsymbol{y}_t = \frac{\rho}{2}[\boldsymbol{h}_{t,1}\Omega_t(L_{(1)}) + \boldsymbol{h}_{t,2}\Omega_t(L_{(2)})] + \boldsymbol{n}_t \quad (1)$$

where $\frac{\rho}{2}$ is the mean SNR at each transmitting antenna, $\boldsymbol{h}_{t,u}$, $t, u \in [1:2]$, is the multipath fading that a transmitted symbol experiences when being transmitted over a communications channel and $\boldsymbol{n}_t$ is the AWGN vector with each entry that follows a complex normal distribution with a mean of zero and variance $\frac{E_s}{2}$ per dimension, where E_s is the energy of the symbol transmitted. In this paper, the fading channel considered is the frequency flat fast fading channel.

At the receiver end, the maximum likelihood detection algorithm is used to estimate the transmitted codeword. Assuming perfect channel state information at the receiver, the estimated symbols can be given as:

$$\widehat{L}_{(1)}, \widehat{L}_{(2)} = arg \min_{L_{(1)} and L_{(2)}} \|\boldsymbol{y}_t - \sum_{u=1}^{2} \frac{\rho}{2}\boldsymbol{h}_{t,u}\Omega_t(L_{(u)})\|^2 \quad (2)$$

where $t \in [1:2]$

In equation (2), the estimated symbols are represented as $\widehat{L_{(1)}}$, $\widehat{L_{(2)}}$ while the encoded symbols are presented as $\Omega_t(L_{(u)})$,$t, u \in [1:2]$ respectively.

A. THEORETICAL ERROR PERFORMANCE AND MAPPER DESIGN OF USTLD SYSTEMS

From the derivation in [3], it was shown that – using the union bound approach – that the average bit error probability for USTLD systems with $N_T = 2$ is given as:

$$ABEP(\rho) \leq \frac{1}{Ms} \sum_{L=0}^{M-1} \sum_{\widehat{L}=0, \widehat{L}\neq L}^{M-1} \delta(L, \widehat{L})P(L \rightarrow \widehat{L}) \quad (3)$$

In equation (3), M is the order of modulation, $s = log_2 M$ is the number of bits per symbol, $\delta(L, L)$ is the number of bit errors of the transmitted and received symbols and $L \rightarrow> \widehat{L})$ is the pairwise error probability that one transmitted symbol is estimated perfectly while the other is estimated with error. Hence, the PEP of USTLD systems with $N_T = 2$ is given as:

$$P(L \rightarrow \widehat{L}) = \frac{1}{4n} \prod_{a=1}^{2} (1 + \frac{\rho d_a^2}{8})^{-N_R} + \frac{1}{2n} \sum_{k=1}^{n} \prod_{a=1}^{2} (1 + \frac{\rho d_a^2}{8sin^2(\frac{k\pi}{2n})})^{-N_R} \quad (4)$$

where $d_a = |\Omega_t(L) - \Omega_t(\widehat{L})|$, $a \in [1:2]$ is the Euclidean distance between symbols on the constellation and n is a comparatively large integer such that $n > 10$. At high SNR, the authors of [3] have shown that $\frac{\rho d_a^2}{8} \gg 1$, which reduces the PEP in equation (4) to the following:

$$P(L \rightarrow \widehat{L}) = \frac{1}{4n} \prod_{a=1}^{2} (\frac{\rho d_a^2}{8})^{-N_R} + \frac{1}{2n} \sum_{a=1}^{2} \prod_{a=1}^{2} (\frac{\rho d_a^2}{8sin^2(\frac{k\pi}{2n})})^{-N_R} \quad (5)$$

The PEP in equation (5) was shows that the product distance term dominates the PEP. Hence, this gives rise to a design metric that could be used to evaluate how much of LD a mapper design has achieved. This is presented as:

$$\phi(\Omega_1, \Omega_2) = \min_{L,\widehat{L}\,\epsilon\,[0:M-1],L\neq\widehat{L}} \prod_{t=1}^{2} |\Omega_t(L) - \Omega_t(\widehat{L})| \quad (6)$$

In equation (6), Ω_1 and Ω_2 denote the pseudo gray and LD mapper respectively, ϕ is the fitness or the amount of LD achieved by Ω_2 with respect to Ω_1 and $L, \widehat{L}$ are the mapped symbols from mappers Ω_1 and Ω_2 respectively. The higher the value of ϕ, the more LD has been achieved. Hence, the end goal of the LD mapper design problem is the maximise the minimum product Euclidean distance between symbols on a constellation [7], [8].

III. DATA PREPARATION AND SETUP

In this section, the authors present the raw data preparation for the case of applying machine-learning algorithms to the LD problem. A further optimization technique using hyper-parameter tuning that selects the best parameters to use for good results is also presented. The models' stability of prediction is outlined, and the metrics used to measure and compare ML-based models are discussed.

A. DATA PREPARATION

Data preparation is a key step in defining how well an ML algorithm performs. Data that is passed through any ML algorithm needs to be of high quality to reach the desired results. For example, when training a machine learning to recognize and classify images, it is empirical that the nature of the images be of the best resolution and format. These images can then be processed and converted into the ML algorithm's data types (tensors). Thus, in this paper, the authors use a simple function that is able to produce randomly generated LD mappers for the 16QAM constellation, and its corresponding fitness values are calculated using equation (6). Additionally, the genetic algorithm (GA) found in [8] and [9] is used to produce data on the optimal LD mapper designs available for the 16QAM constellation. The data is then parsed into an excel spreadsheet format for easy use and access. The algorithm used to produce the mapper design data is outlined as follows:

Algorithm 1 Algorithm Used for LD Mapper Design Production

```
Input: x (number of mapper designs required), $\Omega_1$
       (pseudo gray mapper)
Output: $\Omega_2$ (LD mapper)
  0: for $iteration = 1, 2, \ldots, x$ do
  0:     $\Omega_2 \leftarrow$ Randomly shuffle $\Omega_1$
  0:     Calculate fitness of $\Omega_2$, $\phi(\Omega_1, \Omega_2)$
  0:     Append $\Omega_2$, $\phi(\Omega_1, \Omega_2)$ to an array, $G$
  0: end for
  0: Convert $G$ to excel format.
  1: return 0 = 0
```

In algorithm 1, x is the number of LD mappers required for the function to generate, Ω_1 is the pseudo gray mapper, Ω_2 is the LD mapper produced and $\phi(\Omega_1, \Omega_2)$ is the fitness or the amount of LD achieved by Ω_2 with respect to Ω_1 respectively. For this paper, the authors had chosen the value of $x = 10000$, and another 100 optimal mapper designs produced by the GA in [8] and [9] had also been added to the excel file. This ensures that good predictions can be made with the least amount of error.

B. MACHINE LEARNING MODEL STABILITY

A stability analysis is adopted to allow users to determine the impact of input variations on the output of a system [2]. In terms of machine learning, the stability analysis is used to verify the ML model's ability to keep rather constant accuracy metrics with respect to a changing dataset. In this paper, the K -folds cross validation technique is used to assess the stability of the ML algorithms. The general approach of the K-folds validation technique comprises of 4 steps and 4 sub-steps as outlined [11]:

- Randomly shuffle the dataset
- Split the dataset K number of times
- Loop through each split in the dataset (groups)
  - Use the selected group as a test dataset
  - Use the other groups as the training set
  - Train the model by passing through the training set and evaluate the model's performance on the test dataset
  - Keep the accuracy score and drop the current model
- Summarize the model evaluation metrics

It is reported in [2] that the error estimation is taken as the mean over all K number of splits, hence providing the total effectiveness of the ML model. By training the model on each split on the dataset exactly once, it reduces the bias within the model as most – if not, all – of the data from the dataset is used to fit the model. Elmezughi *et al.* [2] also reports that the K -folds technique adds a good amount of stability to the overall effectiveness of the ML model since the measured data is interchanged between training and testing sets.

C. MACHINE LEARNING MODEL EVALUATION METRICS

Model evaluation metrics are mathematical equations that are used to measure the performance of the given ML-based algorithms. The metrics can be used as a benchmark to compare with other algorithms within the same class, namely machine learning algorithms. The five metrics used in the application to this area are:

- The R-Squared (R^2) metric – a statistical measurement of the closeness of data to the regression line.
- Root mean squared error (RMSE) metric – a measure of the difference between the actual target value and predicted values.
- Mean absolute percentage error (MAPE) – a measure of how accurate the prediction system is said to be.

- Mean square error (MSE) – an estimated measure of error within a statistical model.
- Pearson's Correlation (Corr) – the measure of linear correlation between datasets.

These metrics were adopted as they are the most widely used within the research community and they are a good method for comparing algorithms. Each of the five performance metrics can be stated mathematically as:

$$R^2 = 1 - \frac{\sum_{i=1}^{Q}(\phi_i - \widehat{\phi}_i)^2}{\sum_{i=1}^{Q}(\phi_i - \bar{\phi}_i)^2} \tag{7}$$

$$RMSE = \sqrt{\frac{1}{Q}\sum_{i=1}^{Q}(\phi_i - \widehat{\phi}_i)^2} \tag{8}$$

$$MAPE = \frac{1}{Q}\sum_{i=1}^{Q}\left|\frac{\phi_i - \widehat{\phi}_i}{\phi_i}\right| \tag{9}$$

$$MSE = \frac{1}{Q}\sum_{i=1}^{Q}(\phi_i - \widehat{\phi}_i)^2 \tag{10}$$

$$Corr = \frac{\sum_{i=1}^{Q}(\phi_i - \bar{\phi})(\widehat{\phi}_i - \bar{\widehat{\phi}})}{\sqrt{\sum_{i=1}^{Q}(\phi_i - \bar{\phi})^2}\sqrt{(\widehat{\phi}_i - \bar{\widehat{\phi}})^2}} \tag{11}$$

In equations (7) – (11), Q is the number of samples used to determine the values of the metrics, ϕ_i is the true fitness value of the mapper design, $\widehat{\phi_l}$ is the predicted fitness value of the mapper design and $\bar{\phi}$, $\bar{\widehat{\phi}}$ are the average values of ϕ_i and $\bar{\phi}$ respectively.

D. HYPERPARAMETER TUNING ANALYSIS

Hyperparameters are parameters that define the model architecture. Naturally, in order to get the best results out of any algorithm or model, the most optimized hyperparameters are required. Hence, hyperparameter tuning plays a vital role in transversing the hyperparameter search space in order to select the best possible parameters within a range of values [2]. ML algorithms function on complex hyperparameters, and finding the optimal values for the hyperparameters can lead to an optimization challenge [2]. Trying all possible combinations and permutations of these settings can be very time consuming. However, with the technology of today, many hyperparameter tuning solutions are available that have proven to select the best hyperparameters for a given application. Examples of these solutions include: i) Bayesian Optimization Automate Hyperparameter Tuning (Hyperpot) [12], Spearmint Bayesian Optimization [13], Sequential Model-based Optimization (SMAC) [14], Autotune: a derivative-free optimization [15] and Optuna [16]. The outlined approaches aim to maximise the accuracy of the model while minimizing on the error losses [2]. This is achieved by running multiple trials on the ML algorithms with different hyperparameter values, thereafter the best possible set of hyperparameters are selected.

For the application in this paper, the Optuna framework was selected to determine the best set of hyperparameters for all ML-based models. Optuna is a process framework that automates the tuning and selection of the best hyperparameters. Optuna is said to be efficient at transversing large search spaces and minimizes the number of trials needed by pruning [16]. Furthermore, to get the bet out of ML models, the proper parameters need to be identified and tuned. This is one of the main attributes of this paper that allows the ML model to achieve a high accuracy score [16].

IV. MACHINE LEARNING MODELS

In this subsection, the ML-based algorithms used to predict the amount of LD achieved by a LD mapper are discussed. In this paper, the authors use six ML models, namely Multi-Linear Regression (MLR), Support Vector Regression (SVR), Decision Trees (DT), Random Forrests (RF), K-Nearest Neighbours (KNN) and Artificial Neural Networks (ANNs).

A. MULTI-LINEAR REGRESSION

Multi-linear regression (MLR) is a statistical model based on the linear regression model that predicts the result of a dependant variable based on multiple independent variables [2]. MLR algorithms identifies the interrelationship between input features (the independent variables) and the outcome variable (the dependant variable) to find the best fit for the data. The inputs of the MLR algorithm are represented as $X_1, X_2, X_3, \ldots, X_n$ where n is the number of input variables. Hence, the MLR algorithm can be expressed mathematically as [2]:

$$Y = f(X_i) = \beta_0 + \sum_{i=1}^{n}\beta_i X_i + \epsilon \tag{12}$$

In equation (12), β_i are the model coefficients of each regression line created to fit the data and represents the error margin of the best line fit. In order to calculate the coefficients of the MLR model, an estimation algorithm called the Least Squares Regression (LSR) is adopted to pick the best coefficients of β_i such that the mean squared error (MSE) is minimized [2]. In the application of MLR, the MSE is represented as [2]:

$$MSE = \sum_{j=1}^{K}(Y_j - \beta_0 - f(X_i))^2 \tag{13}$$

B. SUPPORT VECTOR REGRESSION MODEL

Support vector machines (SVMs) are a type of machine learning algorithm that attempts to find a line of best fit on a hyperplane (in multidimensional space) that splits data classes [2]. Initially, SVMs were proposed as a solution by [2] for binary classification problems. After research had expanded, SVMs had gained the ability to produce models for both multi-classification (SVC) and regression analysis (SVR). The SVR algorithm is similar to the SVM and SVC algorithms, with a few changes. SVR uses a selected boundary between the values $-\epsilon : \epsilon$ from the hyperplane which predicts the real valued target values [2]. The boundary is said to be a tolerance margin that only considers data points with the boundary. Hence, the main aim of SVR algorithms is to minimize the prediction error and maximise the boundary by individualizing the hyperplane [2]. It was reported in [2] that the SVR uses linear regression functions as an alternative to the hyperplane. This is expressed in equation (14). A threshold error ϵ can be selected that will minimize the equation (15), which is termed as the ϵ-sensitivity loss error function [2]. Hence, the process of SVR – as stated previously – aims to minimize ϵ in equation (15) and $\|W\|^2$ in equation (16). The equations (15) – (16) is given as:

$$Y = W^T X + b \tag{14}$$

In equation (14), W are the SVR co-efficients, X is the input features and Y is the target variable. Thus, we define the error and R equations as:

$$|\ddot{Y} - Y|_\epsilon = \begin{cases} 0 & |\ddot{Y} - Y| \leq \epsilon \\ |\ddot{Y} - Y| - \epsilon & otherwise \end{cases} \tag{15}$$

$$R = \frac{1}{\epsilon}\|W\|^2 + C(\sum_{i=1}^{N} |\ddot{Y} - Y|_\epsilon) \tag{16}$$

In equations (15) and (16), $\ddot{Y}$ is the predicted target value, R is the SVM objective function and C is the tolerence variable. In order to account for the error and training losses, tolerance variables are introduced. This limits the value to the regression target [2]. The tolerance variables are defined as ϵ and ω respectively. Hence, taking into account the error and training losses, equation (16) and (14) can be presented as:

$$R = \frac{1}{\epsilon}\|W\|^2 + C\sum_{i=1}^{N} |\omega_i - \ddot{\omega}_i^*|_\epsilon \tag{17}$$

$$(W^T X + b) - Y_i \leq \epsilon - \omega_i \tag{18}$$

$$Y_i - (W^T X + b) \leq \epsilon + \omega_i^* \tag{19}$$

In this study, the standard kernel functions are considered such as the linear, radial, and polynomial functions. The kernel functions denoted by $K(X_i, X_j)$ are given as:

$$K(X_i, X_j) = X_i^T X_j \tag{20}$$

$$K(X_i, X_j) = e^{-\lambda\|X_i - X_j\|} \tag{21}$$

where $\lambda > 0$

$$K(X_i, X_j) = (X_i^T X_j + 1)^d \tag{22}$$

where d is the radial basis degree function and i, j corresponds to the i-th and j-th input features respectively.

As noted in [2], it is clearly noted that the kernel function used directly affects the results obtained from the SVR algorithm. Hence, the hyperparameters C, λ and d must be optimized in order to obtain the best results. By using the Optuna technique described in section III.D, the best results were produced with the following hyperparameters: i) the kernel that best suited the problem was the radial basis function (RBF), ii) the RBF coefficient produced was 0.01 and iii) the C value was $C = 4.7445$. The SVR process is illustrated in Fig. 2 respectively.

C. DECISION TREE REGRESSION MODEL

Decision Tree (DT) algorithms are amongst the most commonly used ML-based algorithms [2]. A decision tree is a graph that represents both choices as well as implications to those choices. Each node on a graph is the event that will take place, while the edges represent each choice and the implications that come with it [2]. Hence, in a tree-based graph, each node represents data to be classified and each branch represents a value that the node can be set to. Examples of real-world decision tree-based classifiers include ID3Q [17], C4.5Q [18] and CART [19].

In this paper, the decision tree-based model creates a regression model in the form of a tree structure. The input data is broken into smaller datasets and the decision tree begins to develop from each of these smaller datasets. In decision trees, the highest node is known as the root node [2]. The DT algorithm discovers a method to split the input data such that it is repeated several times until the best results are produced [2]. From [2], it can be noted that best results are obtained by using a variance reduction as a measure of impurity. The results of the DT algorithm are used to calculate the variance reduction for each output [2]. Thus, a higher variance tends to a higher impurity and vice versa. For this paper, the DT model hyperparameters are set as follows: i) the variance calculation function is the MSE and ii) the number of nodes on the tree are 57. Other hyperparameters are set to: i) maximum features = log2, ii) minimum samples per leaf = 2 and iii) data splitting, splitter = best.

D. RANDOM FOREST REGRESSION MODEL

The random forest (RF) algorithm is a supervised learning algorithm that uses a tree-based learning technique [2]. RF algorithms use multiple tree-based models and combines them to achieve a more accurate and stable prediction. RF was introduced as an improved regression model as a single regressor is not adequate for producing a good fit of the data due to its inability to distinguish between patterns and noise [2]. Each tree in the RF algorithm contains a root node, leaf nodes and internal nodes [2]. As in DT algorithms, the root node contains the training data while the leaf nodes contain the output of the algorithm. All internals nodes are broken up by features, which use the MSE as a criterion to break up the nodes [2]. As in [2], each tree learns by using four randomly selected features. For the application of the RF algorithm in this paper, the total number of tree nodes chosen after hyperparameter optimization using Optuna, was set to n = 1610, number of internal nodes is set to 39. Other hyperparameters are set to: i) maximum features = log2, ii) minimum samples per leaf = 1 and iii) minimum number of splits = 2.

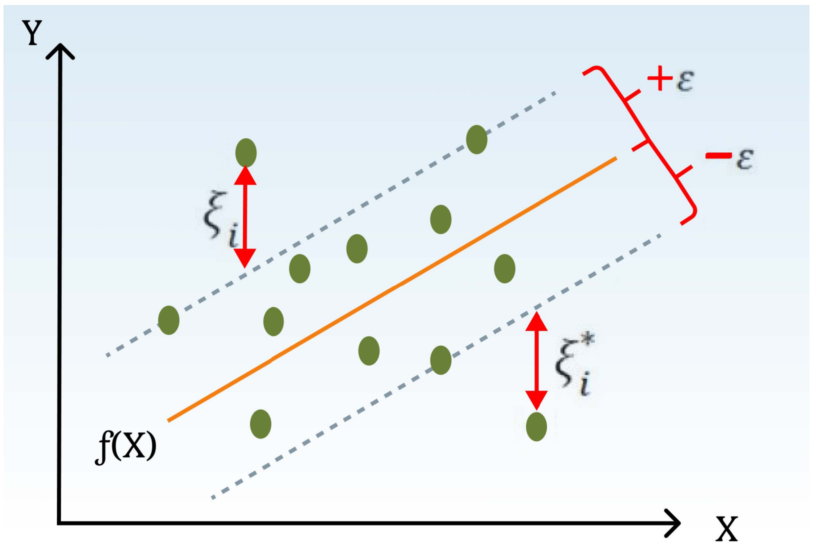

FIGURE 2. Illustration of the mechanics of the SVR algorithm.

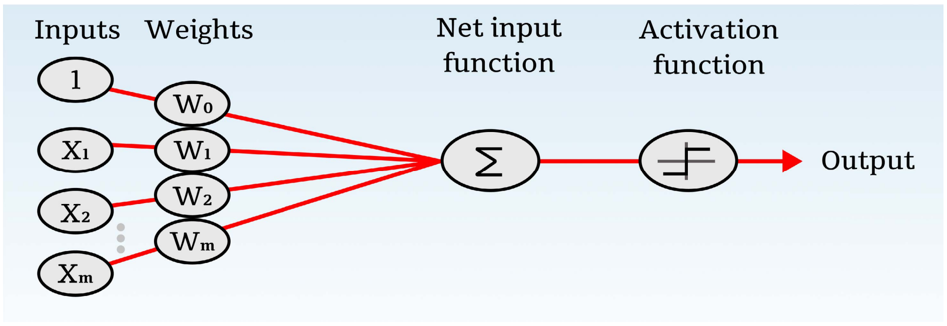

FIGURE 3. Block diagram of the artificial neural network (ANN) algorithm.

TABLE 1. Run-time analysis of each machine learning algorithm.

Machine Learning Model	Run Time (seconds)
Multi-Linear Regression	1.256
Support Vector Regression	6.374
Decision Trees	2.686
Random Forest	25.156
K-Nearest Neighbours	1.456
Artificial Neural Network	256.231

E. K-NEAREST NEIGHBOURS REGRESSION MODEL

The k-nearest neighbours (KNN) algorithm is a ML-based algorithm that can be used for both classification as well as regression [2]. It is classified under the supervised machine learning category. A significant limitation of the KNN algorithm is that as the size of input data increases, the algorithm becomes slow [2]. The KNN algorithm categorizes data points with similar target values, hence the name. The process of the KNN algorithm works by finding the distances between a single data point and all other data within the dataset. It then choses a number of neighbours (K) that has the closest target value to the data point in question. Finally, in the case of classification the algorithm votes for the most frequent target value or in the case of regression, it averages over the target values [2]. By selecting an optimal value for K, the best fit is obtained for both the classification and regress models produced. In the case of this paper, the hyperparameter tuning algorithm, Optuna, had determined that the optimal value for K was K = 25 and the optimal algorithm for this application was the kd tree algorithm. In general, the KNN uses an average of the target values of the neighbours by applying any of the distance functions given as:

$$Euclidean : D = \sqrt{\sum_{i=1}^{K}(X_i - Y_i)^2} \tag{23}$$

$$Manhattan : D = \sum_{i=1}^{K}|X_i - Y_i| \tag{24}$$

$$Minkowski : D = \left(\sqrt{\sum_{i=1}^{K}(X_i - Y_i)}\right)^{\frac{1}{q}} \tag{25}$$

TABLE 2. Performance metrics' values of all the ML-based models selected.

Models	R^2			RMSE			MAPE			MSE			Corr		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
MLR	0.0075	0.0746	0.1068	0.3251	0.4	0.5168	0.0332	0.0355	0.0382	0.1057	0.1633	0.4323	0.21287	0.2946	0.3762
SVR	0.7133	0.8541	0.9483	0.1223	0.1496	0.1748	0.0159	0.0169	0.018	0.0149	0.0228	0.0305	0.8516	0.9258	0.9743
DT	0.6634	0.8437	0.9196	0.1273	0.1562	0.1962	0.0022	0.0031	0.0046	0.0162	0.0251	0.0385	0.8262	0.9212	0.9595
RF	0.7856	0.9123	0.9721	0.0912	0.1137	0.1511	0.0044	0.0048	0.0059	0.0083	0.0133	0.0228	0.8867	0.9550	0.9863
KNN	0.3574	0.626	0.8441	0.2157	0.2432	0.2616	0.0063	0.0081	0.0105	0.0465	0.0594	0.0685	0.7176	0.8359	0.928
ANN	0.8809	0.9362	0.9738	0.0193	0.0243	0.0353	0.9897	1.3199	2.7470	0.0004	0.0006	0.0012	0.9397	0.9700	0.9888

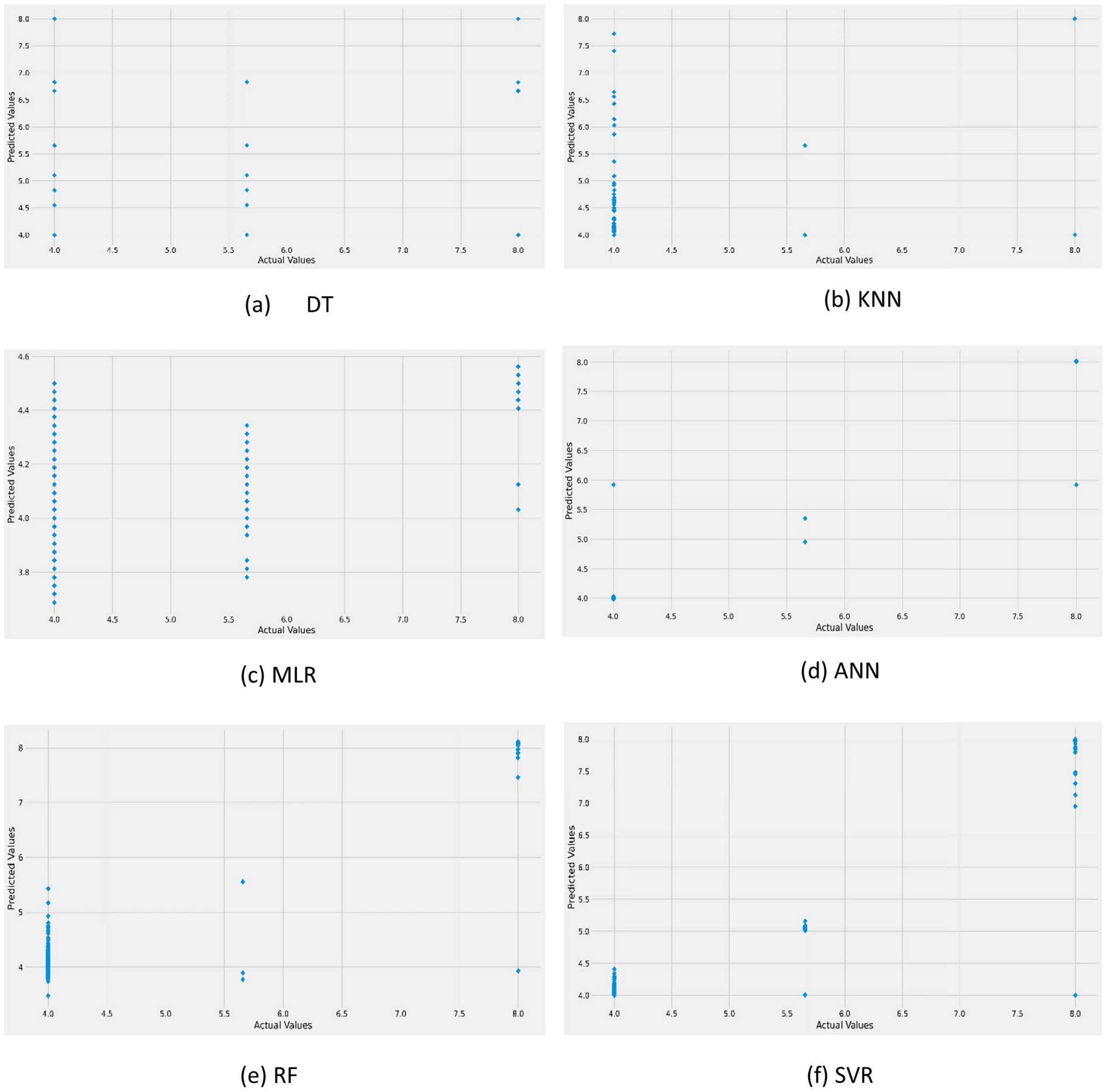

FIGURE 4. Predicted vs actual fitness values of each ML-based algorithm for the training dataset.

F. ARTIFICIAL NEURAL NETWORKS

Artificial neural networks or ANN for short is an algorithm that mimics the process of the human brain by finding and understanding patterns within the input data [2]. Just like the human brain, ANNs contain neurons which can either be organic or artificial in nature [2]. The ANN algorithm

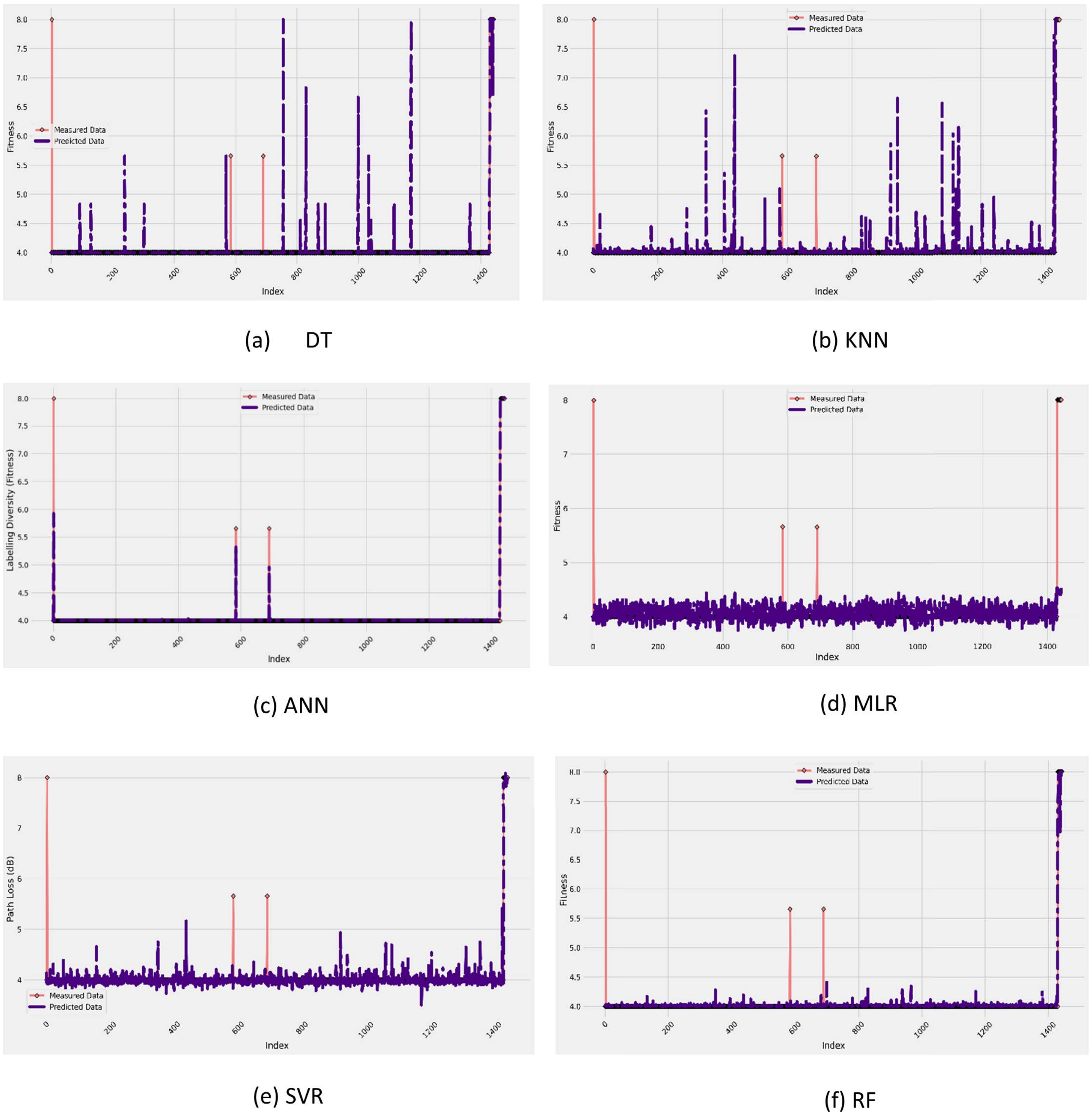

FIGURE 5. Fitness values vs the index for each ML-based algorithm.

also has the ability to adapt to a changing environment, hence the need to redesign the architecture of the ANN is not needed [2]. A simple ANN architecture is illustrated in Fig. 3. The architecture of an ANN consists of inputs, hidden layers and outputs. In the case of this paper, the input to the ANN is the LD mapper design dataset and the output is the predicted value of the amount of LD achieved. In this paper, the ANN comprised of 5 hidden layers, and the hyperparameters were set by the Optuna algorithm as follows: i) the input layer had 52 neurons with a shape of 16, ii) the second layer contained 48 neurons with activation function ReLu, iii) the third and forth layers had 78 neurons each with activation function sigmoid and iv) the fifth and final layer had 1 neuron with the activation function linear. The input of the ANN accepts the 16 features from the pre-processed dataset. The output of the ANN algorithm is the value of LD achieved by the mapper design. Other hyperparameters also optimized by Optuna were: i) learning rate = 0.001 and ii) the loss function was optimized to be mean squared error.

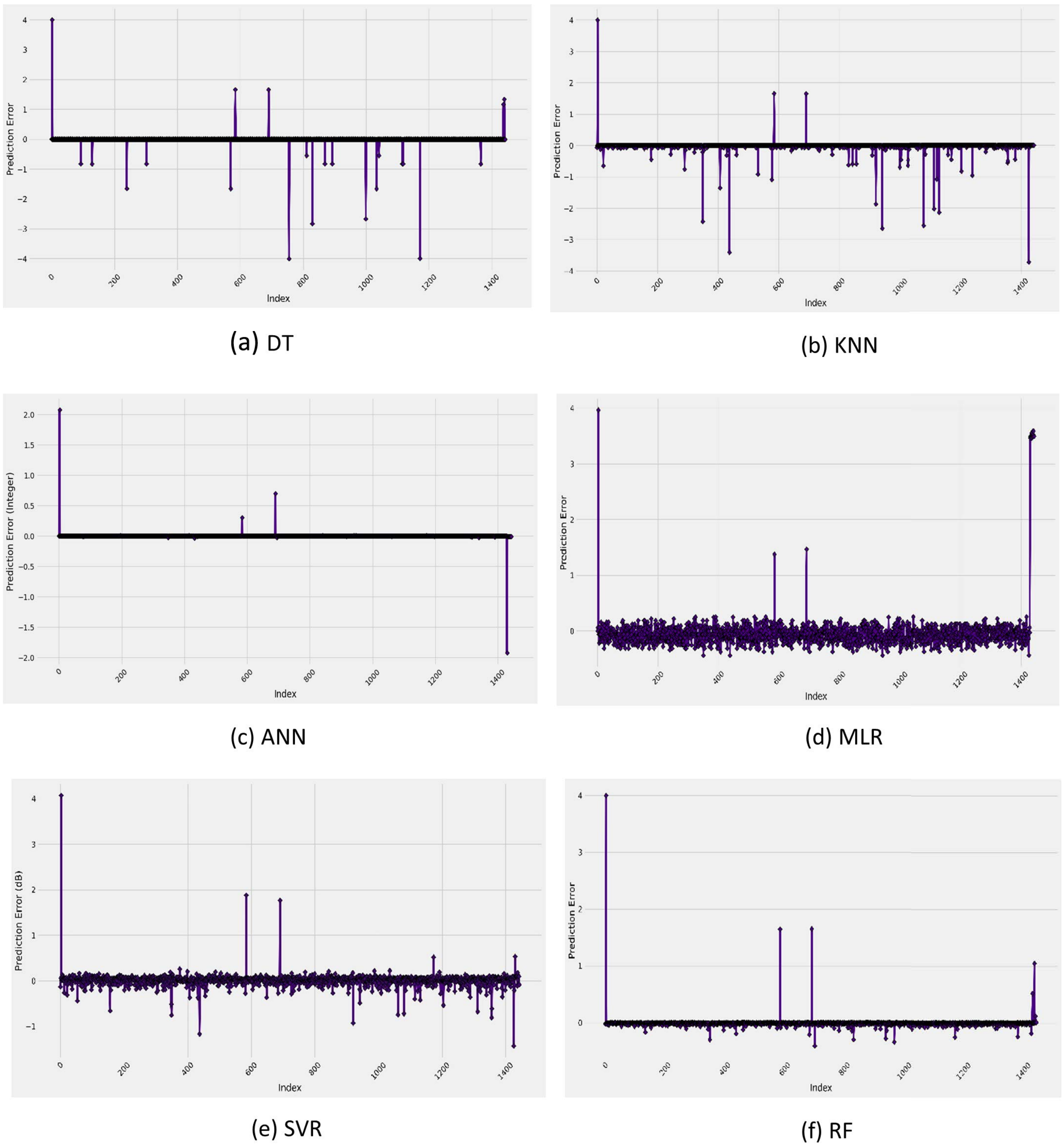

FIGURE 6. Prediction error vs the index for each ML-based algorithm.

V. RESULTS AND DISCUSSIONS

In this section, the results of the ML and ANN algorithms are presented and compared in terms of their respective algorithm metrics outlined in (7) – (11). The experiments performed for this paper was on a PC with an i7 Intel Quad-core processor (6-th Gen) 2.56GHz 64-bit operating system running Windows 10. The hard drive on this PC had 1TB of storage and 8GB of RAM. The software solution used for the implementation of the experiments was Python with packages Tensorflow v1.1.0 and TFlearn 0.3. Additionally, the online python-friendly environment, Google Colab was also used in the experiments. Google Colab comes standard with approximately 100GB of storage space for datasets and approximately 13GB of total RAM. The runtime for each algorithm is also presented as part of the comparison process.

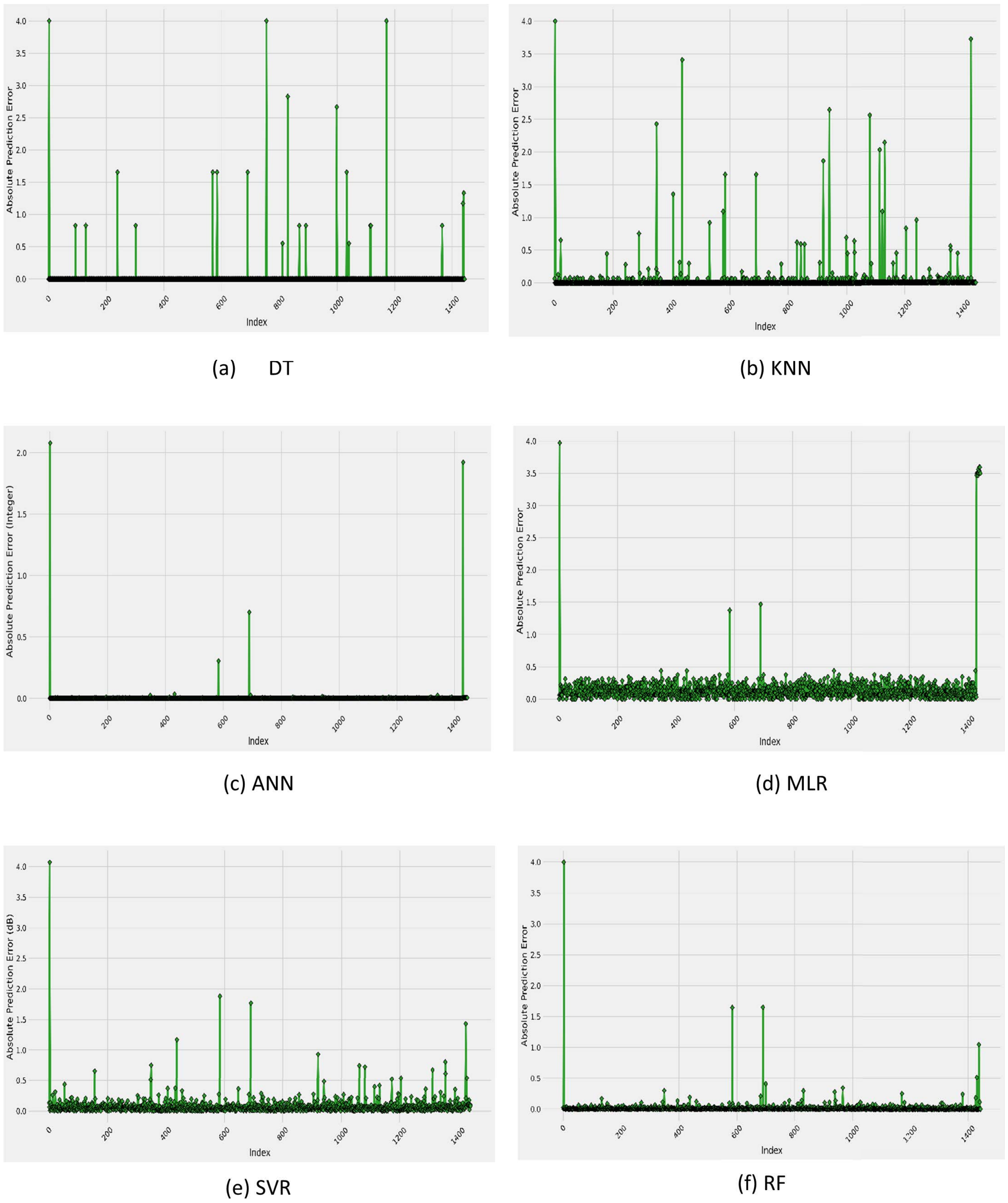

FIGURE 7. Absolute prediction error vs index for the ML-based algorithms.

A. RUN-TIME OF EACH ALGORITHM

The most fundamental way of determining and understanding the complexity of a system is to conduct a runtime analysis. Runtime analyses are important for determining which algorithm is more suitable for a given application [2]. Hence, it forms part of good coding practices. The runtime

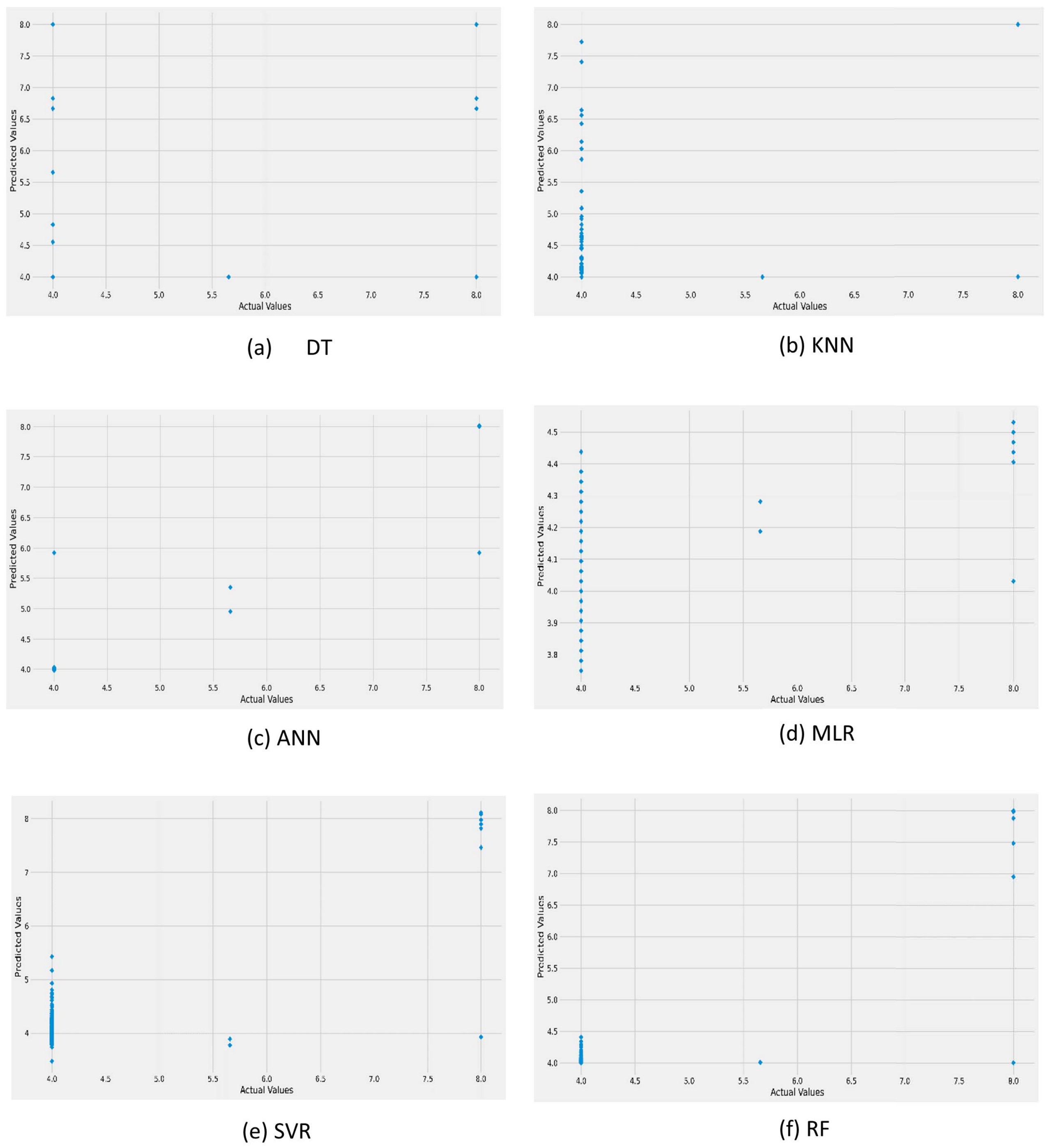

FIGURE 8. **Actual vs predicted fitness values for each ML-based algorithm for the test dataset.**

analyses can be spoken of in two ways, namely i) time complexity and ii) computational complexity. In this paper, the authors perform a time complexity analysis on the ML-based algorithms with respect to the given problem. Table 1 presents the runtimes of each machine learning model used in this paper. As seen in Table 1, the ML algorithm with the lowest runtime was MLR with a runtime of 1.256 seconds, while the longest runtime was the ANN with a runtime of 256.231 seconds. From the runtime results, it can be seen that the MLR algorithm is the least time complex, while the ANN is the most complex. Other notable mentions are the KNN algorithm with a runtime of 1.456 seconds and the DT algorithm with a runtime of 2.686 seconds. The notable algorithms are significantly less complex than the ANN algorithm. Since the

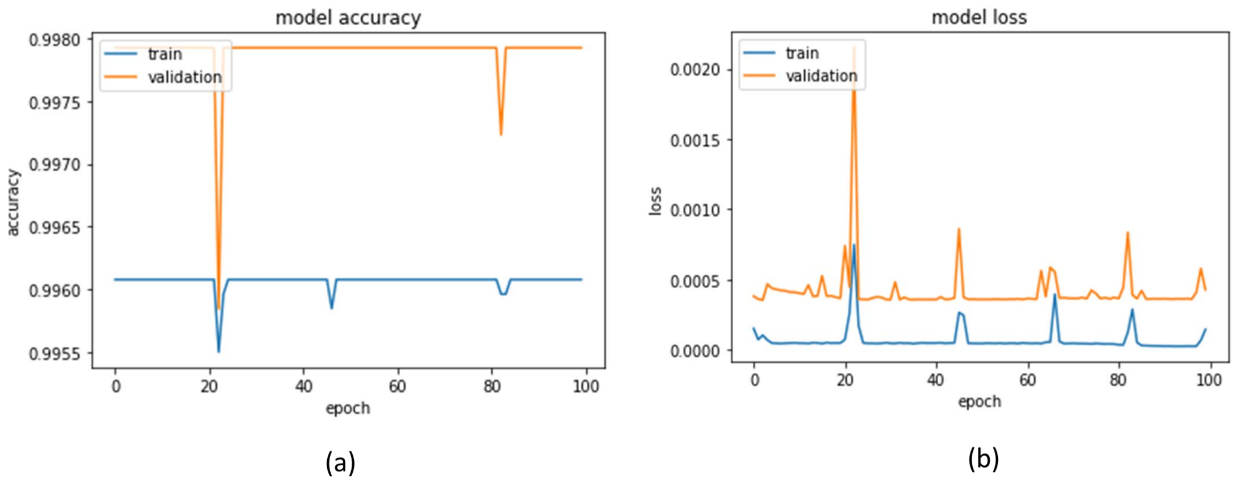

FIGURE 9. Training and testing datasets of (a) accuracy vs epochs and (b) loss vs epochs.

runtime of each algorithm is relatively short, the ML-based models used in this paper can produce models based on the LD problem with a low time complexity.

B. MACHINE LEARNING ALGORITHM COMPARISON

In this subsection, the ML algorithms that are presented in this paper are compared in terms of five accuracy metrics, namely R^2, $RMSE$, $MAPE$, MSE and Pearson's correlation ($Corr$). The results presented are for approximately 25% of the total dataset. This is a good measure to evaluate the performance of each algorithm. As stated, the input features of the ML-based algorithms are each point on the 16QAM constellation with different labels.

Fig. 4 illustrates the predicted values against the actual values for 25% of the dataset. It is observed from the results that the MLR algorithm performed the worst when tested on the LD problem ($R^2 = 0.1068$), while the RF and ANN algorithms have approximately shown the best performance with $R^2 = 0.9721$ and $R^2 = 0.9738$ respectively. From Fig. 6, for all ML-based models - except the MLR algorithm – have exhibited a good match between the predicted and actual target values. Additionally, the RMSE values were $0.0353 \leq RMSE \leq 0.1962$ for the best performing models while the MLR algorithm exhibited a RMSE value of $RMSE = 0.5168$. Hence, the MLR algorithm had once again performed the worst.

Furthermore, the MAPE ranges between $0.0382\% \leq MAPE \leq 2.75\%$. From the defined ranges of MAPE values that are classified, a MAPE value of less than 10% is an excellent measure and values greater than 50% are bad. Hence, the MAPE range for this application exhibits an excellent performance. Moreover, the Pearson's correlation (Corr.) show that the best performing models have a correlation of $93\% \leq Corr \leq 99\%$ while the MLR algorithm have a correlation of $Corr = 37.62\%$. For the Pearson's correlation, values that are acceptable are between the range of -1 and 1, where -1 exhibits a perfect negative correlation and 1 exhibits a perfect positive correlation. Thus, for this application, the ideal scenario is to produce models with a correlation of $Corr \geq 80\%$. Hence, using this constraint, the approximately best performing ML-based algorithm was the RF and ANN algorithms respectively. These results are summarized in Table 2. In the case of this paper, the authors note a few points regarding the accuracy of the models: i) the dataset used to train the models were sufficient, ii) the input features to the ML-based algorithms encompassed all the essential factors and iii) the use of a hyperparameter tuning model led to a smaller error being produced. Another important factor that influenced the performance of the ANN algorithm was the pre-processing and normalization of the dataset. Pre-processing of the data refers to the transformation of raw data into a specific format that the ANN could understand while normalization refers to the process of transforming the pre-processed data onto a unit sphere, essentially transforming all data points between the range of the users choice, typically between $[-1, 1]$.

Fig. 5 illustrates the prediction error of the dataset for the ML-based models in this paper. Additionally, the figure also shows the differences between the predicted values per index respectively. The figure also illustrates that the average error value was approximately equal to 6, which was shown to be produced by the MLR algorithm.

Figs. 6. and 7. illustrate the prediction error against the index and the absolute percentage error against the index respectively. From the results obtained, it can be seen that apart from the MLR algorithm, the prediction error of the other studied ML-based algorithms have shown values between $0.01 \leq PredictionError \leq 4$ with an average prediction error of 2. Hence, the results show that the ML-based algorithms that are trained appropriately and with sufficient data can reduce the prediction error to a minimum.

Finally, Fig. 8 depicts the predicted values against the actual values of the test set, i.e. 25% of the entire dataset as mentioned. It can be observed that the MLR algorithm produced the most incorrect predictions which can be inferred from the metric values, while the ANN and RF algorithms produced the best predictions. Therefore, from the entire analysis, the authors have concluded that the best ML-based algorithms that can be used for the application of the LD problem are the ANN and RF algorithms respectively.

In addition to the metrics used to evaluate and compare ML-based algorithms, another important factor for neural networks is to compare its training and testing accuracies and losses. This provides an insight to how well a neural network model has performed and where inconsistencies show up. Fig. 9 presents two graphs, Fig 9(a). illustrates the model accuracy for the training and testing datasets, while Fig. 9(b) illustrates the model loss for the training and testing datasets respectively. As seen in the figure, the model accuracy maintains a constant linear line, while at some epochs there is a drop in accuracy. This is due to the prediction error by the ANN algorithm. The same could be mentioned about the model losses. The model loss on the training and testing set maintains a relatively straight line, while at some epochs fluctuations occur due to the prediction error. The findings of this research are critical in developing efficient wireless communication systems that can meet future needs. Therefore, the primary motivation for increasing research into accurate and high-speed communication networks is to achieve future goals and demands [20], [21], [22], [23].

VI. CONCLUSION

Artificial intelligence (AI) had recently been applied to the wireless communications domain, especially to the STBC realm. One particular instance of AI applied to STBC systems was evolutionary algorithms that was able to produce near optimal LD mapper designs. However, this process takes time to produce a mapper design of said quality. Hence, motivated by the enhancements, this paper proposes a machine learning-based (ML-based) model that predicts the amount of LD achieved by a mapper design. The ML-based models considered in this paper were the MLR, SVR, DT, RF, KNN and ANN algorithms respectively. Input data for the 16QAM constellation was collected using a simple randomized algorithm and an evolutionary algorithm which provided the optimal mapper designs for the ML-based algorithms to learn patterns from. Each input feature was a single point on the 16QAM constellation located at different points. To ensure stability and reliable results, a cross validation technique was used to split the data and train the algorithms multiple times. This affords the algorithm to learn and recognize patterns within the data. In order to get the best possible results from the ML algorithms, a hyperparameter tuning method was also used to find the best suited parameters for the problem. Additionally, five commonly used ML metrics were used to evaluate each algorithms performance, namely R^2, $RMSE$, MSE, $MAPE$ and the correlation coefficient. From the results obtained, the ANN algorithm was able to produce the best results, with a R^2 value of 0.9738 and an $RMSE$ value of 0.0353. These values were the best results obtained when compared to other algorithms such as the DT, MLR, RF, KNN and SVR algorithms respectively. The results in the paper have shown a positive response and can have many applications to the LD problem.

Future works in this area include: i) a comparison of deep learning algorithms for the LD problem such as long-short-term memory (LSTMs) and transformers, ii) research on combining the properties of evolutionary algorithms with the power of machine learning algorithms to produce improved models and iii) extend the research of this paper to higher order modulations and other constellations.

REFERENCES

[1] S. S. Syed-Abdullah, S. Abdul-Rahman, A. M. Benjamin, A. Wibowo, and K.-R. Ku-Mahamud, "Solving quadratic assignment problem with fixed assignment (QAPFA) using branch and bound approach," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 300, Jan. 2018, Art. no. 012002, doi: 10.1088/1757-899X/300/1/012002.

[2] M. K. Elmezughi, O. Salih, T. J. Afullo, and K. J. Duffy, "Comparative analysis of major machine-learning-based path loss models for enclosed indoor channels," *Sensors*, vol. 22, no. 13, p. 4967, Jun. 2022, doi: 10.3390/s22134967.

[3] B. Mahesh, "Machine learning algorithms—A review," *Int. J. Sci. Res.*, vol. 9, pp. 381–386, Jan. 2020, doi: 10.21275/ART20203995.

[4] S. S. Patel, T. Quazi, and H. Xu, "A genetic algorithm for designing uncoded space-time labelling diversity mappers," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2018, pp. 1–6.

[5] H. Samra, Z. Ding, and P. M. Hahn, "Symbol mapping diversity design for multiple packet transmissions," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 810–817, May 2005, doi: 10.1109/TCOMM.2005.847132.

[6] K. G. Seddik, A. S. Ibrahim, and K. J. R. Liu, "Trans-modulation in wireless relay networks," *IEEE Commun. Lett.*, vol. 12, no. 3, pp. 170–172, Mar. 2008, doi: 10.1109/LCOMM.2008.071734.

[7] D. Ayanda, H. Xu, and N. Pillay, "Uncoded M-ary quadrature amplitude modulation space-time labeling diversity with three transmit antennas," *Int. J. Commun. Syst.*, vol. 31, no. 18, p. e3818, Oct. 2018, doi: 10.1002/dac.3818.

[8] S. Solwa, "An ordered crossover based approach to designing labeling diversity mappers," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 4, Dec. 2021, Art. no. e4431.

[9] T. Ross, "The synthesis of intelligence—Its implications," *Psychol. Rev.*, vol. 45, no. 2, pp. 185–189, Mar. 1938.

[10] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Develop.*, vol. 3, no. 3, pp. 210–229, Jul. 1959.

[11] K. Battula, "Research of machine learning algorithms using K-fold cross validation," *Int. J. Eng. Adv. Tech.*, vol. 8, no. 6, pp. 215–218, Aug. 2019.

[12] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Comput. Sci. Discovery*, vol. 8, no. 1, Jul. 2015, Art. no. 014008.

[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.

[14] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proc. Int. Conf. Learn. Intell. Optim.*, Jan. 2011, pp. 507–523.

[15] P. Koch, O. Golovidov, S. Gardner, B. Wujek, J. Griffin, and Y. Xu, "Autotune: A derivative-free optimization framework for hyperparameter tuning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 443–452.

[16] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.

[17] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[18] J. R. Quinlan, "*Programs for Machine Learning*. Amsterdam, The Netherlands: Elsevier, Jun. 2014.

[19] W.-Y. Loh, "Classification and regression trees," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 1, no. 1, pp. 14–23, Jan. 2011.

[20] T. S. Rappaport, G. R. Maccartney, M. K. Samimi, and S. Sun, "Wideband millimeter-wave propagation measurements and channel models for future wireless communication system design," *IEEE Trans. Commun.*, vol. 63, no. 9, pp. 3029–3056, Sep. 2015.

[21] M. K. Elmezughi, T. J. Afullo, and N. O. Oyie, "Performance study of path loss models at 14, 18, and 22 GHz in an indoor corridor environment for wireless communications," *SAIEE Afr. Res. J.*, vol. 112, no. 1, pp. 32–45, Mar. 2021.

[22] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, 2013.

[23] M. K. Elmezughi and T. J. Afullo, "An efficient approach of improving path loss models for future mobile networks in enclosed indoor environments," *IEEE Access*, vol. 9, pp. 110332–110345, 2021.

[24] Z. Gong, P. Zhong, and W. Hu, "Diversity in machine learning," *IEEE Access*, vol. 7, pp. 64323–64350, 2019.

[25] B. S. Adejumobi and T. Shongwe, "Labeling diversity for media-based space-time block coded spatial modulation," *IEEE Access*, vol. 8, pp. 99870–99879, 2020.

[26] S. S. Roy, R. Chopra, K. C. Lee, C. Spampinato, and B. Mohammadi-Ivatlood, "Random forest, gradient boosted machines and deep neural network for stock price forecasting: A comparative analysis on South Korean companies," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 33, no. 1, pp. 62–71, Jan. 2020.

[27] P. Samui, S. S. Roy, and V. E. Balas, *Handbook of Neural Computation*. New York, NY, USA: Academic, Jul. 2017.

SHAHEEN SOLWA was born in Durban, South Africa, in 1997. He received the B.Sc. degree in electronic engineering and the M.Sc. degree in electronic engineering in the fields of artificial intelligence (AI) and wireless communications from the University of KwaZulu-Natal (UKZN), Durban, in 2019 and 2021, respectively. He is currently a researching methodologies into applying other AI techniques to the wireless communications domain, especially in the fields of Space-Time Block Coding (STBC). His research interests include machine learning, evolutionary algorithms, local search, space-time block coding, labelling diversity, and deep learning (e-mail: shaheensolwa786@gmail.com).

MOHAMED K. ELMEZUGHI (Student Member, IEEE) was born in Tripoli, Libya, in 1995. He received the bachelor's degree (Hons.) in electrical engineering from the University of Tripoli, Libya, in 2017, and the master's degree *(Cum Laude)* in electronic engineering from the University of KwaZulu-Natal (UKZN), Durban, South Africa, in 2020, where he is currently pursuing the Ph.D. degree, under the supervision of Prof. Thomas J. Afullo. The focus of his Ph.D. research is on millimeter-wave channel modeling for 5G mobile communication systems and beyond. His main research interests include artificial intelligence, millimeter-wave systems, wireless channel modeling, signal detection, antennas design, radio propagation, and channel parameters estimation (e-mail: m.k.elmezughi@gmail.com).

OMRAN SALIH received the B.Sc. degree in mathematics and computer science from the International University of Africa, Sudan, in 2010, the P.G.D. degree in mathematics from the University of Cape Town (AIMS Program), South Africa, in 2012, and the M.Sc. degree in applied mathematics and the Ph.D. degree in computer science from the University of KwaZulu-Natal, South Africa, in 2014 and 2020, respectively. He has published his research works in some accredited *Computer Vision and Image Processing* journal, and international and national conferences proceedings. He serves as a reviewer for several accredited journals. His research interests include machine learning, computer vision, image processing, and medical image analysis (e-mail: omran@aims.ac.za).

ALI M. ALMAKTOOF was born in Libya, in 1978. He received the bachelor's degree from Al-Tahady University, Libya, in 2001, the M.Sc. degree from the University of Tripoli, Libya, in 2008, and the Ph.D. degree from the Cape Peninsula University of Technology (CPUT), Cape Town, South Africa, in 2015, all in electrical engineering. He was an Innovation Postdoctoral Research Fellow with the Centre for Distributed Power and Electronics Systems (CDPES), CPUT, from November 2015 to October 2016, where he is currently a Lecturer and a Researcher in electrical engineering with the Department of Electrical, Electronic and Computer Engineering. He is a member with the Institution of Engineering and Technology (MIET), the South African Institute of Electrical Engineers (SAIEE), and the World Society of Sustainable Energy Technology (WSSET). He is also an Active Member with the Energy Institute and the Centre for Distributed Power and Electronics Systems (CDPES), CPUT. His research interests include control of power converters, especially model predictive control, power electronics and drives, energy efficiency, wireless communications, renewable energy, real-time simulators, and desalination technologies (e-mail: almaktoofa@cput.ac.za).

M. T. E. KAHN received the bachelor's, master's, and Ph.D. degrees in engineering. He is currently a Professor of electrical engineering, a Research Chair in energy, and the Director of the Energy Institute, Cape Peninsula University of Technology. He is also a Supervisor of over 130 bachelor's, 40 master's, and 27 Ph.D. students, in electrical and energy engineering. He is the author of several books and chapters, including coauthor of *Concise Higher Electrical Engineering*. He has published over 200 scientific journals and conferences proceedings papers. He chairs the organizing committee/review board of some key international conferences, including convening the Energy and Human Habitat and the Industrial and Commercial Use of Energy Conferences (e-mail: khant@cput.ac.za).