Received 20 April 2024, accepted 15 May 2024, date of publication 20 May 2024, date of current version 29 May 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3403420*

RESEARCH ARTICLE

# Harnessing Learn Rate Schedule for Adaptive Deep Learning in LoRaWAN-IoT Localization

**R. SWATHIKA**[1], **S. M. DILIP KUMAR**[1], **N. N. SRINIDHI**[2], **AND B. R. HARSHITHA**[1]
[1]Department of Computer Science and Engineering, University of Visvesvaraya College of Engineering (UVCE), Bengaluru, Karnataka 560001, India
[2]Department of Computer Science and Engineering, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, Bengaluru, Karnataka 576104, India

Corresponding author: N. N. Srinidhi (srinidhi.nn@manipal.edu)

**ABSTRACT** The learning rate is one of the most crucial hyper-parameters to regulate during the training of the Deep Learning (DL) models and optimizers. Adaptive learning rate algorithms try to automate the time-consuming process of manually setting a suitable learning rate, which is still exhausting. This research uses the learn rate schedule mechanism for training DL models. The learn rate schedule mechanism updates the learning rate for each step or iteration in DL models and optimizers for problem-solving. This paper implements a learn rate schedule mechanism and hybrid learn rate schedule mechanism like piecewise, exponential decay, polynomial time, reciprocal time and cosine annealing decay as adaptive learning rate mechanisms for DL models and optimizers like Adadelta, Adam, RMSprop and Stochastic Gradient Descent with Momentum (SGDM) to improve the accuracy of Received Signal Strength Indicator (RSSI)-based localization in LoRaWAN (Long Range Wide Area Networks) based Internet of Things (IoT) networks. These techniques aim to automate the process of determining suitable learning rates that dynamically update the learning rate for each step or iteration for optimizers and deep learning models. This technique improves the model's performance by introducing adaptability into the learning process and departing from conventional set learning rates. The mathematical model of the learning rate schedule is derived and formulated with adaptive deep learning rate models to map with the LoRaWAN RSSI-based localization datasets for accessing the performance parameters. The learn rate schedule for different types of localization datasets is also analyzed. The results were compared for all the learning rate schedule mechanisms with the default parameter settings of DL models, and it gives a better accuracy of 98.98%, which is higher than the existing models.

**INDEX TERMS** Adaptive learning rate, cosine annealing decay, deep learning, exponential decay, IoT, learn rate schedule, LoRaWAN, optimizers, piecewise, polynomial time, reciprocal time, RSSI.

## I. INTRODUCTION

DL has been one of computer vision's most crucial Machine Learning (ML) tools throughout the past decade [1]. The vast amount of data generated by the rapidly expanding IoT and its ability to connect billions of devices encourages innovation in the industrial 4.0 physical network system's monitoring process [2]. With this growing volume of data, DL's benefits for handling large-scale data are brought to light [3]. LoRaWAN in IoT has recently recorded a transmission range of 830 miles and has grabbed the attention of the researchers towards it [4]. Locating the devices in IoT or LoRaWAN is a tedious process [5], and hence, the DL models and optimizers can be used to improve the location accuracy [6] and reduce error distance of the actual and predicted locations [7].

The order of learning rates significantly impacts training effectiveness and generalization performance. An excessive learning rate will cause instability in the training process. Under such circumstances, the estimate may not converge or overshoot the intended minimum to the point that they merely osculate about it. There are further undesirable effects if learning rates are too low, ineffective training is one of them. Low learning rates cause the training process to

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

update the estimates slowly, which adds needless time to the process [8].

We need an automated method for determining appropriate learning rates to effectively train a DL model and optimize it with strong generalization performance. Driven by the significance of learning rate, this research aims to create an algorithm for an adaptable learning rate that would significantly enhance the models' accuracy. It can be achieved using the learn rate schedule mechanisms like multi-step, exponential decay, power, reciprocal time and cosine annealing decay.

The DL models used here are Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Multilayer Perceptron (MLP), Radial Basis Functions (RBF), Self-Organizing Maps (SOMs) and Generative Adversarial Network (GAN). The DL optimizers used here are Adadelta, Adam, RMSprop and SGDM. All these DL models are evaluated and compared regarding accuracy and error recorded between the estimated and predicted location in terms of meter (m). Three types of Received Signal Strength (RSS) considered here are linear, exponential, and powered RSS. To our knowledge, this is the first work where learning rate schedule mechanisms are adapted as adaptive learning rate factors to be experimented with, along with the DL models and optimizers for the RSSI-based localization in LoRaWAN-IoT networks.

## A. PAPER ORGANIZATION

The remaining part of this paper is organized as follows: The remaining Section I provides the background details of this paper and related works and also includes contributions of this paper. Section II consists of this paper's problem statement and objectives. Section III demonstrates the learn rate schedule mechanisms as adaptive learning rate in DL models for RSSI-based localization. Section IV presents numerical results as graphs and tables related to DL models, optimizers, types of RSS models and learn rate schedule mechanisms. Section V concludes with a significance of research findings and future work.

## B. BACKGROUND

This section discusses RSS types, and metrics-related formulas for RSS localization.

### 1) RECEIVED SIGNAL STRENGTH (RSS)

RSS is the strength of the received signal. When the End Devices (EDs) send the information to the Gateway (GW), it measures the RSS, represented as a dBm unit. It is mainly to find the signal quality, evaluate and assess the strength of the signal, or find how strong the signal is received. There are 3 types of RSS models. They are Linear RSS, Exponential RSS, and Powered RSS. The Linear RSS can be formulated as below:

$$L = L_0 - 10 \times \alpha \times log_{10}(m/m_0) \tag{1}$$

where, $L$ is the RSS at distance $m$, $L_0$ is RSS at reference distance $m_0$, $\alpha$ is the path loss linear to the slope, $m$ is the present direction and $m_0$ is the reference distance with respect to $L_0$. For example, RSS of -50 dBm, $m_0$ is 1 meter, $\alpha$ is 3 and estimate $d$ at 10 meters. By using the formula, $L$ value will $-80$ dBm.

The Exponential RSS can be formulated as below:

$$E = E_0 - 10 \times \beta \times log_{10}(m/m_0) \tag{2}$$

where, $E$ is the RSS at distance $m$, $E_0$ is RSS at reference distance $m_0$, $\beta$ is the decay rate of signal strength, $m$ is the present direction and $m_0$ is the reference distance with respect to $E_0$. For example, RSS of $-50$ dBm, $m_0$ is 1 meter, $\beta$ is 2.5 and estimate $d$ at 10 meters. By using the formula, $E$ value will $-75$ dBm.

The powered RSS can be formulated as below:

$$P = P_0 - 10 \times \theta \times log_{10}(m/m_0) \tag{3}$$

where, $P$ is the RSS at distance $m$, $P_0$ is RSS at reference distance $m_0$, $\theta$ is the path loss exponent, $m$ is the present direction and $m_0$ is the reference distance with respect to $P_0$. For example, RSS of $-50$ dBm, $m_0$ is 1 meter, $\theta$ is 2 and estimate d at 10 meters. By using the formula, P value will $-70$ dBm.

RSS localization is estimating the device locations by using the RSS measurements. These RSS measurements are collected from the GW distributed across the LoRaWAN network, either collected from the intersection of 3 GWs, called the trilateration technique, or more than 3 GWs, called the multilateration method. Various approaches can also be used for RSS localization in LoRaWAN, which includes fingerprinting techniques, ML algorithms, DL algorithms, optimization algorithms, and many more.

### 2) METRIC FORMULAS

In this paper, we have considered metrics like accuracy, error, R-squared, and evaluation time. Accuracy is the quantity of several correct predicted locations by the total number of locations. It is given by:

$$Accuracy = \frac{CL}{L} \times 100\% \tag{4}$$

where $CL$ is the number of correct predicted locations and $L$ is the total number of locations. Error in terms of meters (m) is calculated by the remaining percent of accuracy as follows:

$$error(m) = (100 - Accuracy) \times 100 \tag{5}$$

or

$$error(m) = abs(exact\_location - predicted\_location) \tag{6}$$

And it is the difference between the exact location and predicted location.

### C. RELATED WORKS

Kwasme et al. [9] assessed accuracy and error regarding evaluation time and distance using 10 distinct ML algorithms. It has attained the 340 m location estimation inaccuracy using the forest regression algorithm. The accuracy achieved by $k$-means was not the same as that of the random forest regression point; its performance is different. The software-defined radios were utilized by Aarif et al. [10] to determine the RSSI-based localization's performance parameters. It has been discovered that the vast variance in RSSI would impact localization performance. In the future, many strategies could be created to solve the aforementioned issue. To find the RSSI-based localization, Yoshitome et al. [11] employed 2 various indoor and outdoor environments and a few ML approaches such as decision trees and support vector machines. It compares LoRaWAN to WiFi, Bluetooth, ZigBee, and other wireless technologies using the trilateration method to see the findings experimentally. Environmental conditions significantly impact the signal quality. Thus, the solution can be practically provided by DL models. Using Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS), Ingabire et al. [12] attempted to measure an RSSI in an indoor short-distance setting. Unlike NLOS, the signal strength seen in LoS situations has less loss. Future research can be conducted in long-range interior and outdoor settings and contrasted with related technologies such as WiFi. When the real localization strategy isn't functioning in some circumstances, Sadawski et al. [13] employed a Global Positioning System (GPS) to localize the device placements. However, always using a GPS technique will be more expensive and energy-intensive. At that point, nodes without GPS capability can offer the localization estimate. Using the RNN technique, Maduranga et al. [14] created a less power localization system for indoor situations. Examining the LoS and NLoS scenarios demonstrates that the localization error in these scenarios is lower than in other related methodologies. Priroddi et al. [15] examined the energy consumption and localization accuracy of various wireless technologies, including WiFi, Bluetooth Low Energy (BLE), ZigBee, and LoRaWAN. It localizes using the RSSI data and the trilateration approach. Perkovic et al. [16] found the localization and the precise RSSI values published to the Message Queuing Telemetry Transport (MQTT) server of the remote location using an ML method. We assess and explain the localization inaccuracy between the predicted and actual. The goal of Ali et al. [17] is localization in extremely crowded urban areas. It estimates the mean localization accuracy and energy usage by fingerprinting RSSI-based localization using a ML technique. Hoang et al. [18] use ML with neural network architecture to achieve excellent RSSI localization accuracy. Up to 98% of the device's accuracy can be estimated. It is possible to utilize the ML technique in the microcontroller and compare it with other ML approaches. Neuron architecture is used by Li and Yanget al. [19] to determine localization accuracy. The accuracy between projected and accurate positions is assessed to adapt to network changes. The suggested method adapts well to the shifting environmental circumstances.

### D. CONTRIBUTIONS

This paper makes the following important contributions:

(i) Formulating the learn rate schedule mechanisms and hybrid learn rate schedule for DL models and optimizers.
(ii) Designing an algorithm for RSSI-based localization using learn rate schedule as adaptive learning rate for DL models.
(iii) Simulate and compare the learn rate schedule strategies to find this dataset's suitable learn rate schedule mechanism.
(iv) Analysis of learn rate schedules for different localization datasets in LoRaWAN is also illustrated.

## II. PROBLEM STATEMENT AND OBJECTIVES

In this paper, the problem is to apply adaptive learning rate through learn rate schedule factors on DL models and optimizers to improve accuracy in RSSI-based localization for LoRaWAN-IoT networks. The objectives of this paper are:

(i) To derive a mathematical model for multi-step, exponential decay, power, reciprocal time, and cosine annealing decay learn rate schedule.
(ii) To formulate a learn rate schedule and hybrid learn rate schedule as an adaptive learning rate for the DL models optimizer.
(iii) To apply a formulated adaptive learn rate schedule for the DL model optimizers to improve the RSSI-based localization in IoT networks.
(iv) Comparison of learn rate schedule mechanisms for DL model optimizers with accuracy and error.

## III. MATERIALS AND METHODS

This section formulates the five different types of learning rate schedules: multi-step, exponential decay, power, reciprocal time, and cosine annealing decay learn rate schedules into DL models and optimizers as adaptive learning rate schedules to map for improving the accuracy for RSSI-based localization. This section also covers the hybrid learn rate schedule with transitions. An adaptive learning rate schedule is suggested for ensemble learning [20], allowing the model to converge initially and then navigate away from local optimal solutions [21].

### A. MULTI-STEP LEARN RATE SCHEDULE

The multi-step, step-wise, or piece-wise learning rate schedule adjusts the learning rate at specified epochs or steps by a particular factor. The general formula for a step-wise learning rate schedule is as follows:

$$lr_{\mathrm{t}} = \begin{cases} lr_0, & \text{if } t < s1 \\ lr_0 \times fac1, & \text{if } s1 \leq t < s2 \\ lr_0 \times fac2, & \text{if } s2 \leq t < s3 \\ \vdots & \\ lr_0 \times facN, & \text{if } t \geq sN \end{cases} \tag{7}$$

where $lr_{\mathrm{t}}$ is the learning rate at time step $t$, $lr_0$ is the initial learning rate, $s1, s2, \ldots, sN$ is the epoch or step value and $fac1, fac2, \ldots, fac2$ is the scaling factors for each epoch of step for the learning rate adjustments. Eq. (7) continues to adjust the learning rate for each epoch or step. The learning rate is updated by using the following formula:

$$lr_{\mathrm{t+1}} = lr_{\mathrm{t}} \times DropFactor \tag{8}$$

where $lr_{\mathrm{t+1}}$ is the new learning rate, $lr_{\mathrm{t}}$ is the old learning rate, and $DropFactor$ is to represent the learning rate drop by a specific factor for each epoch or steps. The drop factor decides how much the learning rate decreases in each step, and the value is between 0 and 1. The value 1 means no drop in the learning rate, and the value less than 1 represents a reduction in the learning rate.

## B. EXPONENTIAL DECAY LEARN RATE SCHEDULE

The exponential decay learn rate schedule follows an exponential decay over time. The learning rate can be initialized and it can be updated at each time step or epoch as follows:

$$lr_{\mathrm{t+1}} = lr_{\mathrm{t}} \times e^{-\alpha t} \tag{9}$$

where $\alpha$ is the decay rate and depends on the various factors. But it is often determined empirically. For complex models, $\alpha$ should be small for which the learning rate will decrease over time and is used for stable convergence. For large $\alpha$, the learning rate will drop rapidly, aiming for faster convergence. So, it is always a trade-off between the stability and the convergence speed.

Now, the step-wise piece-wise schedule for exponential decay can be written as,

$$lr_{\mathrm{t}} = \begin{cases} lr_0, & \text{if } t < s1 \\ lr_0 \times e^{-\alpha(t-s1)}, & \text{if } s1 \leq t < s2 \\ lr_0 \times e^{-\alpha(t-s2)}, & \text{if } s2 \leq t < s3 \\ \vdots & \\ lr_0 \times e^{-\alpha(t-sN)}, & \text{if } t \geq sN \end{cases} \tag{10}$$

The above modified expression shows exponential decay with different $\alpha$ for each step or epoch.

## C. POWER LEARN RATE SCHEDULE

The polynomial or power learn rate schedule follows polynomial decay over time. It is to adjust the learning rate at each iteration according to the polynomial function. The learning rate can be initialized and it can be updated at each time step or epoch as follows:

$$lr_{\mathrm{t+1}} = lr_{\mathrm{t}} \times \left(1 - \frac{t}{max\_steps}\right)^{power} \tag{11}$$

where $max\_steps$ is the maximum number of epochs or steps and $power$ is the degree of polynomial. It represents the polynomial decay function where the learning rate decreases as the time $t$ increases. The larger $power$ means sharper and more aggressive decay. When the learning rate reaches towards 0, as it approaches $max\_steps$. Tuning the parameters like $power$ and $max\_steps$ will enhance the model's performance. Now, the step-wise schedule for power learn rate can be written as,

$$lr_{\mathrm{t}} = \begin{cases} lr_0, & \text{if } < s1 \\ lr_0 \times \\ \left(1 - \frac{t - s1}{max\_steps - s1}\right)^{p1}, & \text{if } s1 \leq t < s2 \\ lr_0 \times \\ \left(1 - \frac{t - s2}{max\_steps - s2}\right)^{p2}, & \text{if } s2 \leq t < s3 \\ \vdots & \\ lr_0 \times \\ \left(1 - \frac{t - sN}{max\_steps - sN}\right)^{pN}, & \text{if } t \geq sN \end{cases} \tag{12}$$

The above modified expression shows the polynomial or power learn rate with different $\alpha$ for each step or epoch, and p is the power or polynomial of the degree.

## D. RECIPROCAL TIME DECAY LEARN RATE SCHEDULE

The inverse time decay or reciprocal time decay schedule decreases the learning rate over time concerning the reciprocal of step or epoch during the process. The learning rate can be initialized and it can be updated at each time step or epoch as follows:

$$lr_{\mathrm{t+1}} = \left(\frac{lr_0}{1 + \alpha \times t}\right) \tag{13}$$

where $\alpha$ is the decay rate parameter. The learning rate decrease over time $t$ is controlled by the $\alpha$. For larger $t$, the denominator gets larger, resulting in a small learning rate. Smaller $\alpha$ slower the learning rate and leads to slow convergence. Larger $\alpha$ decreases the learning rate rapidly, hence faster convergence. Now, the step-wise schedule for inverse decay rate can be written as,

$$lr_{t} = \begin{cases} lr_0, & \text{if } < s1 \\ \left(\dfrac{lr_0}{1+\alpha \times (t-s1)}\right), & \text{if } s1 \le t < s2 \\ \left(\dfrac{lr_0}{1+\alpha \times (t-s2)}\right), & \text{if } s2 \le t < s3 \\ \vdots \\ \left(\dfrac{lr_0}{1+\alpha \times (t-sN)}\right), & \text{if } t \ge sN \end{cases} \quad (14)$$

The above modified expression shows reciprocal or inverse time learn rate with different $\alpha$ for each step or epoch.

## E. COSINE ANNEALING DECAY LEARN RATE SCHEDULE

The smooth decay or cosine annealing learning rate schedule gradually increases and decreases the learning rate in a periodic smooth manner. It is to introduce cyclic behaviour during the training process. The general formula for smooth decay is as follows:

$$lr_{t} = min\_lr + \frac{1}{2}(max\_lr - min\_lr) \left(1 + cos\left(\frac{t - start\_epoch}{cycle_{l}ength}\pi\right)\right) \quad (15)$$

where $min\_lr$ is the minimum learning rate, $max\_lr$ is the maximum learning rate, $start\_epoch$ is at the epoch at which the cycle begins, $cycle\_length$ is the length of the cosine annealing cycle and $\pi$ is the mathematical constant. The cosine function varies between $-1$ and 1. Larger values for $min\_lr$, $max\_lr$ and $cycle\_length$ leads to the faster convergence. Smaller values for $min\_lr$, $max\_lr$ and $cycle\_length$ lead to exploring a broader learning rate and a stable learning rate schedule.

Now, the step-wise schedule for smooth annealing decay is as follows:

$$lr_{t} = \begin{cases} lr_0, & \text{if } < s1 \\ lr_0 + \frac{1}{2}(max\_lr - min\_lr)\left(1 + cos\left(\dfrac{t-s1}{cycle\_length}\pi\right)\right), & \text{if } s1 \le t < s2 \\ lr_0 + \frac{1}{2}(max\_lr - min\_lr)\left(1 + cos\left(\dfrac{t-s2}{cycle\_length}\pi\right)\right), & \text{if } s2 \le t < s3 \\ \vdots \\ lr_0 + \frac{1}{2}(max\_lr - min\_lr)\left(1 + cos\left(\dfrac{t-sN}{cycle\_length}\pi\right)\right), & \text{if } t \ge sN \end{cases} \quad (16)$$

The above modified expression shows smooth decay with possible $max\_lr$, $min\_lr$ and $cycle\_length$ for each step $t$ or epoch. The cosine function is guaranteed to complete one full period over the designated cycle length when $\pi$ is used.

## F. HYBRID LEARNING RATE SCHEDULE WITH TRANSITIONS

Hybrid learning rate schedules change the learning rate during training by combining different decay function types or adding extra components. It incorporates different decay functions or components to improve convergence, stability, and generalization throughout the optimization process [22]. Creating hybrid learning rate schedules that include several decay functions or extra elements like temperature or momentum might result in more flexible and efficient optimization during training. Formulating mathematical functions that describe the transition behaviour based on the training epoch progress is essential to derive a hybrid learning rate schedule that considers additional parameters such as temperature and seamless transitions between different types of decay [23]. For multi-step decay, use a step function that decreases the learning rate at specific epochs:

$$\eta_{\text{step}}(t) = \begin{cases} \eta_0 & \text{if } t < t_1 \\ \eta_0 \times \gamma & \text{if } t_1 \le t < t_2 \\ \vdots \\ \eta_0 \times \gamma^n & \text{if } t \ge t_n \end{cases} \quad (17)$$

where $\eta_{\text{step}}(t)$ is the represents the learning rate at epoch $t$ using the multi-step decay function, $\eta_0$ is the initial learning rate, $\gamma$ is the decay factor and $t_1, t_2, \ldots, t_n$. The exponential decay of the learning rate over epochs is given by:

$$\eta_{\exp}(t) = 2\eta_0 \times \gamma^{\frac{t}{T_{\max}}} \quad (18)$$

where $T_{\max}$ is the total number of epochs and $t$ is the current epoch number. The power decay of the learning rate over epochs is given by:

$$\eta_{\text{pow}}(t) = 2\eta_0 \times \left(1 + \frac{t}{T_{\max}}\right)^{-\alpha} \quad (19)$$

where $\alpha$ is the power parameter. The reciprocal time decay of the learning rate over epochs is given by:

$$\eta_{\text{rec}}(t) = 2\eta_0 \times \left(1 + \frac{t}{T_{\max}}\right)^{-1} \quad (20)$$

The cosine annealing decay of the learning rate over epochs is given by:

$$\eta_{\cos}(t) = 2\eta_0 \left(1 + \cos\left(\frac{\pi t}{T_{\max}}\right)\right) \quad (21)$$

where $\pi$ is used to convert the epoch number $t$ into radians. Several additional transition functions are available to easily convert between various learning rate schedules, like linear transition, sigmoidal transition, exponential transition, and power transition. The learning rate transitions linearly from one schedule to another over a defined transition period. Linear transition is derived as follows:

$$\eta_{\text{trans}}(t, \tau, T) = \eta_{\text{x}}(t) \cdot (1 - \frac{T}{\tau}) + \eta_{\text{y}}(t) \cdot \frac{T}{\tau} \tag{22}$$

where $\eta_{\text{trans}}(t, \tau, T)$ represents the transition learning rate at epoch $t$ with parameters $\tau$ and $T$, $\eta_{\text{x}}(t)$ and $\eta_{\text{y}}(t)$ represents the decay functions ($\eta_{\text{step}}(t)$, $\eta_{\text{exp}}(t)$, $\eta_{\text{pow}}(t)$, $\eta_{\text{rec}}(t)$, $\eta_{\text{cos}}(t)$) which needs to be transited. $\tau$ is a parameter representing the duration of the transition and $T$ is the current epoch number. The transition is modeled using a sigmoidal function, allowing for a smooth and gradual change in the learning rate and Sigmoidal transition is derived as follows:

$$\begin{aligned}\eta_{\text{trans}}(t, \tau, T) = &\frac{1}{2}\left(\eta_{\text{x}}(t) + \eta_{\text{y}}(t)\right)\\ &+ \frac{1}{2}\left(\eta_{\text{y}}(t) - \eta_{\text{x}}(t)\right) \cdot \frac{1}{1 + e^{-k(t-T/2)}}\end{aligned} \tag{23}$$

where $k$ is a parameter controlling the steepness of the transition. The transition follows an exponential curve, gradually shifting the learning rate from one schedule to another and the exponential transition is given by:

$$\eta_{\text{trans}}(t, \tau, T) = \eta_{\text{x}}(t) \cdot e^{-\alpha\tau} + \eta_{\text{y}}(t) \cdot (1 - e^{-\alpha\tau}) \tag{24}$$

where $\alpha$ controls the rate of decay. The transition is modeled using a power function, allowing for customized shaping of the transition curve and the power decay is given by:

$$\eta_{\text{trans}}(t, \tau, T) = \eta_{\text{x}}(t) \cdot \left(1 - \frac{T}{\tau}\right)^{\beta} + \eta_{\text{yp}}(t) \cdot \left(\frac{T}{\tau}\right)^{\beta} \tag{25}$$

where $\beta$ determines the shape of the power function. Introduce a temperature parameter $\theta$ that modulates the transition between the decay functions and is given by:

$$\eta_{\text{hybrid}}(t, \theta) = \eta_{\text{trans}}(t, \tau, T) \cdot \theta \tag{26}$$

where $\theta$ adjusts the rate of transition between the decay functions. Several additional parameters or factors can be added or changed to the temperature parameter to improve the hybrid learning rate schedule's behaviour. These settings impact the speed and smoothness of the learning rate transition between various decay functions. Other parameters that can be utilized are as follows:

(i) **Transition Rate Parameter ($\lambda$):** The rate at which a transition rate parameter could determine the learning rate switches between the decay functions. A transition would happen more quickly with a higher value and more slowly with a lower number.

(ii) **Transition Start Epoch ($t_{start}$):** To include a parameter that establishes the epoch at which the transition starts instead of a set transition period. This increases the controllability of the learning rate's transition between decay functions.

(iii) **Transition End Epoch ($t_{end}$ ):** The epoch at which the transition finishes may be determined using the transition end epoch option. This parameter can be used to create a stopping point based on predetermined criteria or to specify a specified length for the transitions

(iv) **Transition Shape Parameter ($\beta$):** To increase the flexibility of the learning rate transition between decay functions, introduce a shape parameter that alters the transition function's shape. Depending on the shape parameter value, other transitions could be sigmoidal, exponential, or linear.

(v) **Transition Amplitude Parameter ($\alpha$):** It could regulate the size of the learning rate change during the transition rather than directly scaling the rate of learning. This parameter can be used to adjust how much of the transition's impact the learning rate receives.

(vi) **Adaptation Parameter ($\gamma$):** During training, include an adaptation parameter that dynamically modifies the transition behaviour in response to specific circumstances or feedback signals. This parameter might allow the learning rate schedule to instantly adjust to variations in the training dynamics or model performance.

The components of the hybrid learning rate schedule with transitions are decay functions, transition mechanism and additional factors like momentum, temperature, or adaptive adjustments based on gradient variance or model performance. Here, the transition functions are given in Eq. (17) to (21), and the transition mechanism is provided in Eq. (22) to (25). The introduction of additional parameters is given in Eq. (26). The designing of a hybrid learning schedule with transitions is as follows: Decay function selection is based on the optimization issue, model architecture, and dataset properties [24]. Different characteristics of the learning rate behaviour, including quick decay, gradual annealing, or adaptive modifications, should be captured by each decay function [25]. Combining decay functions to create a seamless transition between decay functions, combine them using transition functions or interpolation. Create transition mechanisms that adaptively modify each decay function's contribution in response to training progress or other factors. Transition mechanism incorporation is to put the exact transition mechanism in place during specified transition periods, progressively flipping between decay functions [26]. Use transition functions like exponential decay or sigmoidal curves for a smooth transition in the learning rate schedule. To further adjust the optimization process, incorporate other parameters, such as temperature or momentum, into the learning rate schedule. For training purposes, use adaptive algorithms or reinforcement learning approaches to find the best possible combination of decay functions and other parameters.

### G. DERIVATION OF ADAPTIVE DEEP LEARNING SCHEDULE MODELS

The adaptive learning rate for DL models can be represented by $\eta_t$. The multi-step learn rate can be denoted as *piecewise*_$\eta_t$. Eq. (7) can be incorporated into the modified update rule of the optimizer as follows:

$$piecewise\_\eta_t = \begin{cases} \alpha_1 & \text{if } t \in [t_0, t_1) \\ \alpha_2 & \text{if } t \in [t_1, t_2) \\ \vdots & \\ \alpha_k & \text{if } t \in [t_{k-1}, t_k) \end{cases} \tag{27}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_k$ represents the learning rate at each step-time $t$ and $t \in [t_0, t_1, t_2, \ldots t_{k-1}, t_k]$.

The exponential decay learn rate can be denoted as $exp\_\eta_t$. Eq. (9) can be incorporated into the modified update rule of the optimizer as follows:

$$exp\_\eta_t = \alpha \times e^{-\gamma t} \tag{28}$$

where, $\alpha$ is the learning rate and $\gamma$ is the decay rate at time $t$.

The power learn rate can be denoted as $power\_\eta_t$. Eq. (11) can be incorporated into the modified update rule of the optimizer as follows:

$$power\_\eta_t = \eta_{\max} \times \left(1 - \frac{T}{t}\right)^{\alpha} \tag{29}$$

where $\eta_{\max}$ is the maximum learning rate at iteration $t$, $T$ is the period or number of iterations of the polynomial function, and $\alpha$ is the exponent of the polynomial.

The inverse time learn rate can be denoted as $inv\_\eta_t$. Eq. (13) can be incorporated into the modified update rule of the optimizer as follows:

$$inv\_\eta_t = \frac{\eta_{\max}}{1 + \beta t} \tag{30}$$

where, $\eta_{\max}$ is the maximum learning rate at iteration $t$ and $\beta$ is the coefficient.

The smooth decay learn rate can be denoted as $smooth\_\eta_t$. Eq. (15) can be incorporated into the modified update rule of the optimizer as follows:

$$smooth\_\eta_t = \frac{\eta_{\max}}{2}\left(1 + \cos\left(\frac{T}{t}\pi\right)\right) \tag{31}$$

where, $T$ is the total iteration of cosine function and the $cos$ is the cosine function.

### 1) FORMULATION OF ADAPTIVE LEARNING RATE IN ADADELTA OPTIMIZER

The Adadelta technique keeps a running average of the second moments of the gradients to solve the disappearing and bursting gradient problems [27]. Initialize the accumulators $E[g^2]_0$ and $\Delta x_0$ to 0. Set the decay rate, maximum learning rate, total number of iterations, polynomial exponent and the coefficient for required and respective learn rate schedules. Adadelta maintains two exponentially decaying moving averages: one for the gradient's squared values $E[g^2]_0$ and another for the parameter updates' squared values $E[\Delta x^2]_0$ These averages are updated at each time step using the following equations:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2 \tag{32}$$

$$E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho)x_t^2 \tag{33}$$

where, $g_t$ is the gradient at time step t. Compute the update as follows:

$$\Delta x_t = -\frac{\sqrt{\Delta x_{t-1} + \epsilon}}{\sqrt{E[g2]_t + \epsilon}} g_t \tag{34}$$

Update the parameters as follows:

$$x_{t+1} = x_t + \Delta x_t \tag{35}$$

To incorporate the learning rate $\eta_t$, into the Adadelta parameter update equation, we simply multiply it with the update step $\Delta x_t$. Here's how we can modify the parameter update equation to include $\eta_t$:

$$x_{t+1} = x_t + \eta_t \cdot \Delta x_t \tag{36}$$

$\eta_t$ can be substituted with $piecewise\_\eta_t$, $exp\_\eta_t$, $power\_\eta_t$, $inv\_\eta_t$ and $smooth\_\eta_t$ for the respective learn rate schedules. Algorithm 1 represents the Adadelta optimizer with an adaptive learn rate schedule for RSSI localization in LoRaWAN.

**Algorithm 1** Adadelta Optimizer With Adaptive Learn Rate Schedule for RSSI Localization in LoRaWAN.

```
1:  Input: $X, Y$                          ▷ RSSI values, device locations
2:  Output: $Acc, E(m)$                    ▷ Accuracy, Error Distance
3:  Normalize $X$, Split into $X$ train and test data.
4:  Select suitable DL model.
5:  Initialize the Adadelta optimizer model parameters:
6:      $\theta_0$                          ▷ Initial estimated location
7:      $E[g^2]_0=0$                        ▷ Initial running average of squared parameter updates
8:      $E[\Delta x]_0$                     ▷ Initial running average of squared gradients
9:      $\epsilon$                          ▷ Small constant to prevent divide by zero error
10:     $\eta_t$                            ▷ Adaptive learning rate schedule factor
11:     $t$                                 ▷ Time step
12: for $i$ in $\{1, 2, \ldots, \text{max\_iterations}\}$ do ▷ Train the model.
13:     if not converged then
14:         Compute $\Delta E(\theta)$      ▷ Gradient of the localization error.
15:         Update $E[g^2]_t$ and $E[\Delta x^2]_t$ ▷ Squared gradients average
16:         Update $\Delta x_t$             ▷ Average of squared gradients
17:         Compute $\eta_t$                ▷ Adaptive learning rate
18:         Update $x_{t+1}$                ▷ Estimated location
19:     end if
20: end for
21: Predict the location using trained model.
22: Denormalize $X$ and evaluate output.
```

### 2) FORMULATION OF ADAPTIVE LEARNING RATE IN ADAM OPTIMIZER

During training, it dynamically modifies each parameter's learning rate according to its previous gradients. This modification aids in increasing performance and convergence on various optimization issues [28]. A moving average of earlier gradients with exponential decay is denoted by $m_t$. It is an approximation of the gradient's mean. A moving average of previously squared gradients with exponential decay is denoted by $v_t$. It is an approximation of the gradient's uncentered variance. The following updating rules are used to determine these moments:

$$m_t = \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t \tag{37}$$

$$v_t = \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2 \tag{38}$$

Here, the gradient of the objective function concerning the parameters at time step $t$ is denoted by $g_t$, and the exponential decay rate are represented by $\beta_1$ and $\beta_2$ (usually set to 0.9, 0.99 and 0.999). Particularly in early stages of training, there is a bias towards zero in the estimates $m_t$ and $v_t$. In order to rectify this prejudice, bias-corrected estimates of $\bar{m}_t$ and $\bar{v}_t$ as follows:

$$\bar{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{39}$$

$$\bar{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{40}$$

The ratio of the bias-corrected mean to the square root of the bias-corrected variance at time step t determines the adaptive learning rate for each parameter as follows:

$$\eta_t = \frac{\alpha}{\sqrt{\bar{v_t}} + \epsilon} \tag{41}$$

where $\alpha$ is the initial learning rate and $\epsilon$ is the small constant. The parameter $\theta$ is updated using the computed adaptive learning rate. The ratio of the bias-corrected mean to the square root of the bias-corrected variance at time step t determines the adaptive learning rate for each parameter as follows:

$$\theta_{t+1} = \theta_t - \eta_t \times \bar{m_t} \tag{42}$$

From the above update rule, the gradients' mean $\bar{m}_t$ is scaled by the adaptive learning rate $\eta_t$, which will produce a more effective and adaptive learning rate for every parameter. Can be written as follows:

$$\theta_{t+1} = \theta_t - \eta_t \times \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon} \tag{43}$$

$\eta_t$ can be substituted with *piecewise*_$\eta_t$, *exp*_$\eta_t$, *power*_$\eta_t$, *inv*_$\eta_t$ and *smooth*_$\eta_t$ for the respective learn rate schedules. Algorithm 2 represents the Adam optimizer with an adaptive learn rate schedule for RSSI localization in LoRaWAN.

### 3) FORMULATION OF ADAPTIVE LEARNING RATE IN RMSprop OPTIMIZER

RMSprop modifies the learning rates for every parameter during training. The issue of different gradient scales across various parameters is intended to be addressed by the adaptive learning rate schedule in RMSprop [29]. A moving average of squared slopes is maintained, and the learning rates for each parameter are scaled up using this moving average.

**Algorithm 2** Adam Optimizer With Adaptive Learn Rate Schedule for RSSI Localization in LoRaWAN.

```
 1: Input: $X$, $Y$                      ▷ RSSI values, device locations
 2: Output: $Acc$, $E(m)$                ▷ Accuracy, Error Distance
 3: Normalize $X$, Split into $X$ train and test data.
 4: Select suitable DL model.
 5: Initialize the Adam optimizer model parameters:
 6:     $\theta_0$                        ▷ Initial estimated location
 7:     $m_0$=0                           ▷ Initial value for the first moment
 8:     $v_0$=0                           ▷ Initial value for the second moment
 9:     $\alpha$                          ▷ Initial learning rate
10:     $\beta_1$                         ▷ First Momentum coefficient
11:     $\beta_2$                         ▷ Second Momentum coefficient
12:     $\eta_t$                          ▷ Adaptive learning rate schedule factor
13:     $\epsilon$                        ▷ Small constant to prevent divide by zero error
14:     $t$                               ▷ Time Step
15: for $i$ in $\{1, 2, \ldots, \text{max\_iterations}\}$ do ▷ Train the model.
16:     if not converged then
17:         Compute $\Delta E(\theta)$    ▷ Gradient of the localization error
18:         Update $m_t$                  ▷ First moment estimate
19:         Update $v_t$                  ▷ Second moment estimate
20:         Calculate $\bar{m_t}$ and $\bar{v_t}$   ▷ Bias corrected estimates
21:         Compute $\eta_t$              ▷ Adaptive learning rate
22:         Update $\theta_{t+1}$         ▷ Estimated location
23:     end if
24: end for
25: Predict the location using trained model.
26: Denormalize $X$ and evaluate output.
```

At every time step $t$, the moving average of squared gradients is updated according to the following formula:

$$v_t = \beta \times v_{t-1} + (1 - \beta) \times g_t^2 \tag{44}$$

where $g_t$ is the gradient of the objective function concerning the parameters at time $t$, and $\beta$ is the decay rate. To derive the expression for $g_t$ in the RMSprop optimization algorithm, we first need to understand that $g_t$ represents the gradient of the loss function with respect to the parameters at time step $t$. This gradient is obtained through back propagation during the training process. Given the loss function $L(\theta)$ and the parameters $\theta$, the gradient can be calculated as:

$$g_t = \frac{\partial \theta}{\partial L(\theta)} \tag{45}$$

The adaptive learning rate is as follows:

$$\eta_t = \frac{\alpha}{\sqrt{\bar{v_t}} + \epsilon} \tag{46}$$

where $\alpha$ is the initial learning rate, and $\epsilon$ is the small constant. The parameters are then updated using the adaptive learning rate as follows:

$$\theta_{t+1} = \theta_t - \eta_t \times g_t \tag{47}$$

The concept underlying this adaptive learning rate schedule is that the effective learning rate of parameters associated with large gradients will be smaller, and the practical learning rate of the parameters related to small gradients will be more significant. As a result, RMSprop can handle the difficulties caused by different gradient scales more effectively, increasing its resilience and adaptability across training. The learning rates for each parameter are adequately scaled by using the square root of the moving average of squared gradients in the denominator.

**Algorithm 3** RMSprop Optimizer With Adaptive Learn Rate Schedule for RSSI Localization in LoRaWAN.

```
1:  Input: $X, Y$                    ▷ RSSI values, device locations
2:  Output: $Acc$, $E(m)$            ▷ Accuracy, Error Distance
3:  Normalize $X$, Split into $X$ train and test data.
4:  Select suitable DL model.
5:  Initialize the RMSprop optimizer model parameters:
6:      $\theta_0$                   ▷ Initial estimated location
7:      $E_0$=0                      ▷ Initial value for the square of the gradient
8:      $\alpha$                     ▷ Initial learning rate
9:      $\beta$                      ▷ Decay rate for the running average of squared gradients
10:     $\eta_t$                     ▷ Adaptive learning rate schedule factor
11:     $t$                          ▷ Time Step
12: for $i$ in $\{1, 2, \dots, \text{max\_iterations}\}$ do ▷ Train the model.
13:     if not converged then
14:         Compute $\Delta E(\theta)$   ▷ Gradient of the localization error
15:         Update $v_t$                 ▷ Average of squared gradients
16:         Compute $\eta_t$             ▷ Adaptive learning rate
17:         Update $\theta_{t+1}$        ▷ Estimated location
18:     end if
19: end for
20: Predict the location using trained model.
21: Denormalize $X$ and evaluate output.
```

$\eta_t$ can be substituted with *piecewise*_$\eta_t$, *exp*_$\eta_t$, *power*_$\eta_t$, *inv*_$\eta_t$ and *smooth*_$\eta_t$ for the respective learn rate schedules. Algorithm 3 represents the RMSprop optimizer with an adaptive learn rate schedule for RSSI localization in LoRaWAN.

### 4) FORMULATION OF ADAPTIVE LEARNING RATE IN SGDM OPTIMIZER

The traditional SGDM with momentum updates the parameters $\theta$ based on the gradient of the loss function $L(\theta)$ with a momentum term:

$$v_{t+1} = \gamma \cdot v_t + \eta \cdot \nabla L(\theta_t) \tag{48}$$

where $\gamma$ is the momentum coefficient, $\eta$ is the learning rate, $L(\theta_t)$ is the gradient of the loss function with respect to the parameters at iteration $t$, and $v_t$ is the momentum term at time step $t$. It is possible to insert the adaptive factor directly into the momentum term update of the SGDM update rule to include an adaptive learning rate schedule factor about the momentum term [30]. Update the momentum term by incorporating the adaptive learning rate schedule factor $\eta_t$:

$$v_t = \beta \times v_{t-1} + (1-\beta) \times \eta_t \cdot g_t \tag{49}$$

where $g_t$ is the gradient of the objective function concerning the parameters at time $t$, and $\beta$ is the decay rate. As discussed in earlier sections, the adaptive learning rate factor $\eta_t$ can be updated with any chosen adaptive learning rate schedule. The parameter is then updated using the adaptive learning rate in $v_t$ as follows:

$$\theta_{t+1} = \theta_t - \alpha \times v_t \tag{50}$$

The SGDM update rules directly incorporate the adaptive learning rate schedule with this formulation. The adaptive learning rate schedule factor modifies the learning rate according to this moment's time step. The learning rate schedule gradually adjusts the step size while the momentum term smoothes out the updates and improves convergence behaviour. Based on the features and convergence behaviour of your optimization problem, modify the parameters such as $\alpha$, $\beta$, and the schedule criterion.

**Algorithm 4** SGDM Optimizer With Adaptive Learn Rate Schedule for RSSI Localization in LoRaWAN.

```
1:  Input: $X, Y$                    ▷ RSSI values, device locations
2:  Output: $Acc$, $E(m)$            ▷ Accuracy, Error Distance
3:  Normalize $X$, Split into $X$ train and test data.
4:  Select suitable DL model.
5:  Initialize the SGDM optimizer model parameters:
6:      $\theta_0$                   ▷ Initial estimated location
7:      $v_0$=0                      ▷ Initial value for the momentum
8:      $\alpha$                     ▷ Initial learning rate
9:      $\beta$                      ▷ Momentum coefficient
10:     $\eta_t$                     ▷ Adaptive learning rate schedule factor
11:     $t$                          ▷ Time Step
12: for $i$ in $\{1, 2, \dots, \text{max\_iterations}\}$ do ▷ Train the model.
13:     if not converged then
14:         Compute $\Delta E(\theta)$   ▷ Negative Gradient of the localization error
15:         Update $v_t$)                ▷ Moment update rule
16:         Update $\theta_{t+1}$        ▷ Estimated location
17:     end if
18: end for
19: Predict the location using trained model.
20: Denormalize $X$ and evaluate output.
```

$\eta_t$ can be substituted with *piecewise*_$\eta_t$, *exp*_$\eta_t$, *power*_$\eta_t$, *inv*_$\eta_t$ and *smooth*_$\eta_t$ for the respective learn rate schedules. Algorithm 4 represents the SGDM optimizer with an adaptive learn rate schedule for RSSI localization in LoRaWAN.

## IV. SIMULATION RESULTS AND DISCUSSION

In the simulation environment, MATLAB R2020a, all of the 7 DL models with four optimizer models are investigated in this research and have built-in functionalities [31]. The dataset was acquired in Antwerp, Belgium, which

**TABLE 1.** DL models parameter setup.

| Models | Parameters | Values |
|---|---|---|
| **LSTM** | Hidden Units | 100 |
| | Fully Connected Layer | 6 Classes |
| | Max Epochs | 100 |
| | Mini Batch Size | 27 |
| | Activation Layer | Softmax |
| | Classification Output | Crossentropyex |
| | Output Mode | Last |
| | Sequence Input | 1 dimensions |
| **CNN** | No. of Input Channels | 1 |
| | No. of Layers | 7 Classes |
| | Layers Cell | 1 * 7 cell |
| | Loss Function | cros |
| | Learning Rate | 0.01 |
| | Batch Size | 12 |
| | Iterations | 100 |
| | Layer C Activation Function | ReLu |
| | Layer P Activation Function | tanh |
| | Layer F Activation Function | sigma |
| **MLP** | Input Layer Neurons | 6 |
| | Middle Layer Neurons | 20 |
| | Output Layer Neurons | 1 |
| | Learning Rate | 0.5 |
| **RBF** | Batch Size | 3 |
| | MaxEpochs | 20 |
| | Target Vectors | [2.0, 4.1, 5.9] |
| | Mean Square Error | 2.54 |
| **SOMs** | Dimension sizes | [2, 2] row vector |
| | Initial Cover steps | 100 |
| | Initial Neighborhood size | 3 |
| | Topological Function | hextop |
| | Distance Function | linkdist |
| **GAN** | Latent Dimensions | 100 |
| | Learning Rate Discriminator | 0.0002 |
| | Learning Rate Generator | 0.0002 |
| | Batch Size | 32 |
| | Beta1 | 0.5 |
| | Beta2 | 0.999 |
| | Maxepochs | 50 |
| | Sigma | 0.05 |

**TABLE 2.** DL optimizer models parameter setup.

| Models | Parameters | Values |
|---|---|---|
| **Adadelta** | Gradient Decay Factor | 0.9 |
| | Squared Gradient Decay Factor | 0.999 |
| | Initial Learn rate | 0.0095 |
| | Learning Rate Drop Period | 5 |
| | Gradient Threshold Method | 12norm |
| | Gradient Threshold | 1 |
| | Epsilon | 1e−8 |
| **Adam** | Mini Batch Size | 64 |
| | Batch Size | 3 |
| | Learning Rate Drop Factor | 0.2 |
| | Max Epochs | 10 |
| | Learning Rate | 0.001 |
| | Beta1 | 0.9 |
| | Beta2 | 0.999 |
| | Loss Function | mse |
| | Epsilon | 1e−8 |
| **RMSprop** | Squared Gradient Decay Factor | 0.999 |
| | Gradient Threshold | Inf |
| | Learning Rate Drop Factor | 0.2 |
| | Initial Learn rate | 0.0095 |
| | Minibatch Size | 64 |
| | Epsilon | 1−8 |
| **SGDM** | Mini Batch Size | 64 |
| | Max Epochs | 20 |
| | Learning Rate Drop Factor | 0.2 |
| | Learning Rate Drop Period | 5 |
| | Momentum | 0.9 |
| | Initial Learn rate | 0.0095 |

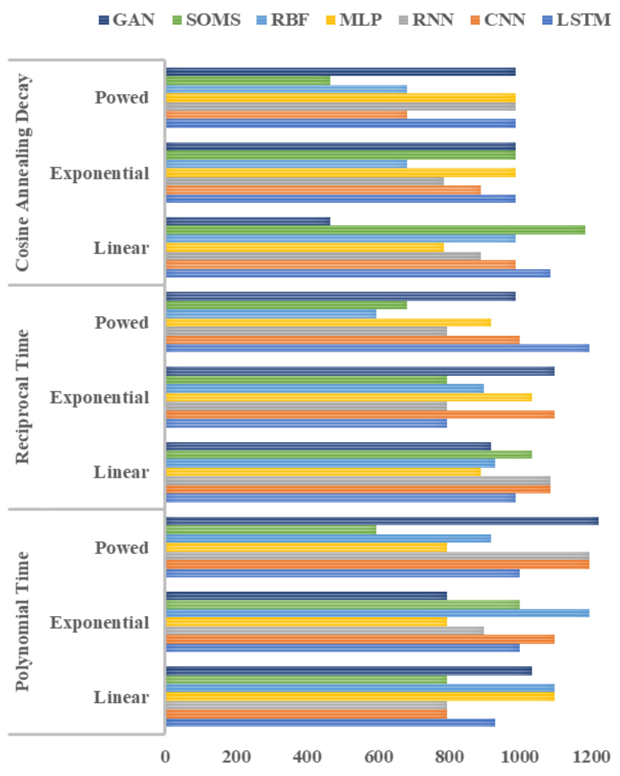

**FIGURE 1.** Polynomial time, reciprocal time and cosine annealing decay learn rate schedule simulation results comparison for Adadelta.

encompasses 72 features and 1, 30, 426 LoRaWAN message samples. Each row in a dataset represents a LoRaWAN message and incorporates information about receiving Base Stations (BS), the time the message was received (Rx Time), the LoRa spreading factor, and the transmitter's latitude and longitude at the time of transmission [32]. The parameter settings for each of the 7 DL models and the optimizers are shown in TABLE 1 and 2, respectively. The remainder of the section discusses using optimizers to analyze simulation results for piecewise, exponential decay, polynomial time, reciprocal time, and cosine annealing decay learn rate schedule. TABLE 3 presents the DL model simulation results with no learning rate schedule and can be taken as the default initial setting simulation results of DL models.

## *A. ADAPTIVE LEARN RATE SCHEDULES IN ADADELTA*

TABLE 4 presents the RSSI localization accuracy and error (m) for linear, exponential, and powered RSS using piecewise and exponential decay learn rate schedule Adadelta DL model. The highest accuracy for piecewise learn rate schedule is 92.222 and error (m) of 777.78 for RNN, RBF and GAN in the linear RSS model and CNN in the power RSS model. The highest accuracy for the exponential learning rate schedule is 93.258 and error (m) of 674.16 for GAN in

**TABLE 3. DL models simulation results with no learn rate schedule.**

| RSS Type | Parameters | Learn Rate Schedule | LSTM | CNN | RNN | MLP | RBF | SOMs | GAN |
|---|---|---|---|---|---|---|---|---|---|
| Linear | Accuracy | None/ Default | 90.909 | 89.655 | 90.11 | 89.655 | 92.045 | 88.764 | 88.764 |
| | Error (m) | | 909.1 | 1034.5 | 989 | 1034.5 | 795.5 | 1123.6 | 1123.6 |
| Exponential | Accuracy | | 89.888 | 87.64 | 93.182 | 88.636 | 89.011 | 87.778 | 84.946 |
| | Error (m) | | 1011.2 | 1236 | 681.8 | 1136.4 | 1098.9 | 1222.2 | 1505.4 |
| Powered | Accuracy | | 89.888 | 90.698 | 89.13 | 86.667 | 88.043 | 87.778 | 87.778 |
| | Error (m) | | 1011.2 | 930.2 | 1087 | 1333.3 | 1195.7 | 1222.2 | 1222.2 |

**TABLE 4. Piecewise and exponential decay learn rate schedule simulation results for Adadelta.**

| RSS Type | Parameters | Learn Rate Schedule | LSTM | CNN | RNN | MLP | RBF | SOMs | GAN |
|---|---|---|---|---|---|---|---|---|---|
| Linear | Accuracy | Piecewise | 88.2979 | 87.3684 | 92.2222 | 88.2979 | 92.2222 | 90.2174 | 92.2222 |
| | Error (m) | | 1170.21 | 1263.16 | 777.78 | 1170.21 | 777.78 | 978.26 | 777.78 |
| Exponential | Accuracy | | 89.2473 | 90.2174 | 90.2174 | 89.2473 | 90.2174 | 90.2174 | 90.2174 |
| | Error (m) | | 1075.27 | 978.26 | 978.26 | 1075.27 | 978.26 | 978.26 | 978.26 |
| Powered | Accuracy | | 88.2979 | 92.2222 | 88.2979 | 91.2088 | 89.2473 | 90.2174 | 87.3684 |
| | Error (m) | | 1170.21 | 777.78 | 1170.21 | 879.12 | 1075.27 | 978.26 | 1263.16 |
| Linear | Accuracy | Exponential Decay | 91.209 | 91.209 | 92.222 | 91.209 | 92.222 | 91.209 | 90.217 |
| | Error (m) | | 879.1 | 879.1 | 777.8 | 879.1 | 777.8 | 879.1 | 978.3 |
| Exponential | Accuracy | | 92.222 | 92.222 | 92.222 | 88.298 | 88.298 | 91.209 | 93.258 |
| | Error (m) | | 777.78 | 777.78 | 777.78 | 1170.2 | 1170.2 | 879.12 | 674.16 |
| Powered | Accuracy | | 91.209 | 89.247 | 92.222 | 93.258 | 93.258 | 90.217 | 90.217 |
| | Error (m) | | 879.1 | 1075.3 | 777.8 | 674.2 | 674.2 | 978.3 | 978.3 |

the exponential RSS model, MLP and RBF in the powered RSS model. Figure 1 presents the RSSI localization error (m) for linear, exponential, and powered RSS using polynomial time, reciprocal time, and cosine annealing decay learn rate schedule Adadelta DL model. The highest accuracy for the polynomial learn rate schedule is 94.048 and error (m) of 595.2 for SOMs in the power RSS model. The highest accuracy for the reciprocal learning rate schedule is 94.048 and error (m) of 595.2 for RBF in the power RSS model. The highest accuracy for cosine annealing decay learns rate schedule is 95.349 and error (m) of 465.1 for GAN in the linear RSS model and SOMs in the power RSS model.

## B. ADAPTIVE LEARN RATE SCHEDULES IN ADAM

TABLE 5 presents the RSSI localization accuracy and error (m) for linear, exponential, and powered RSS using piecewise and exponential decay learn rate schedule Adam DL model. The highest accuracy for piecewise learn rate schedule is 96.516 and error (m) of 348.84 for RNN in the exponential RSS model. The highest accuracy for the exponential learning rate schedule is 96.518, and the error (m) of 348.8 for GAN in the power RSS model. Figure 2 presents the RSSI localization error (m) for linear, exponential, and powered RSS using polynomial time, reciprocal time, and cosine annealing decay learn rate schedule Adam DL model. The highest accuracy for the polynomial learn rate schedule is 95.349, and the error (m) of 465.1 for MLP in the linear RSS model. The highest accuracy for the reciprocal learning rate schedule is 94.048 and error (m) of 595.2 for SOMs in the power RSS model. The highest accuracy for cosine annealing decay learn rate schedule is 93.182 and error (m) of 681.8 for GAN in the power RSS model.

## C. ADAPTIVE LEARN RATE SCHEDULES IN RMSprop

TABLE 6 presents the RSSI localization accuracy and error (m) for linear, exponential, and powered RSS using piecewise

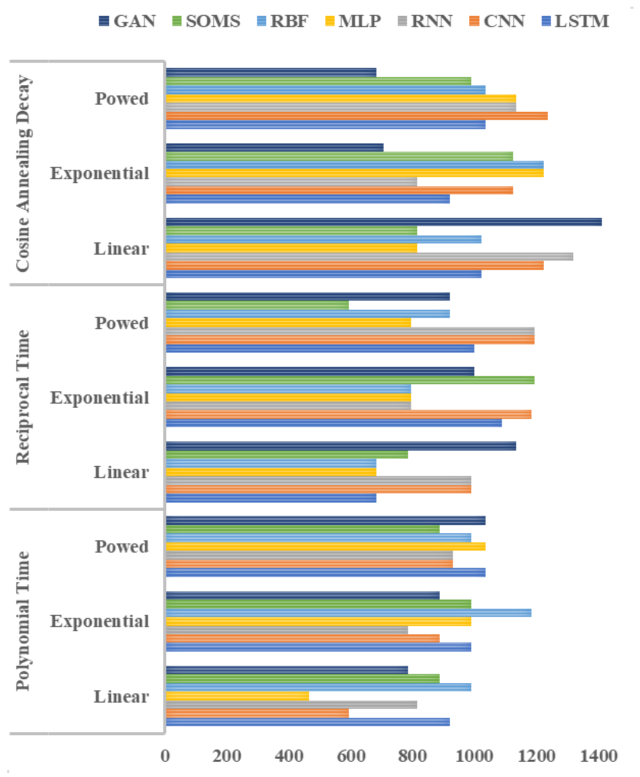

**FIGURE 2. Polynomial time, reciprocal time and cosine annealing decay learn rate schedule simulation results comparison for Adam.**

and exponential decay learn rate schedule RMSprop DL model. The highest accuracy for piecewise learn rate schedule is 93.2584 and error (m) of 674.16 for SOMs in the linear RSS model and MLP in the power RSS model. The highest accuracy for the exponential learning rate schedule is 96.512, and the error (m) of 348.84 for MLP in the linear RSS model. Figure 3 presents the RSSI localization error (m)

**TABLE 5.** Piecewise and exponential decay learn rate schedule simulation results for Adam.

| RSS Type | Parameters | Learn Rate Schedule | LSTM | CNN | RNN | MLP | RBF | SOMs | GAN |
|---|---|---|---|---|---|---|---|---|---|
| Linear | Accuracy | Piecewise | 92.2222 | 90.2174 | 90.2174 | 90.2174 | 90.2174 | 92.2222 | 89.2473 |
| | Error (m) | | 777.78 | 978.26 | 978.26 | 978.26 | 978.26 | 777.78 | 1075.27 |
| Exponential | Accuracy | | 91.2088 | 90.2174 | 96.5116 | 89.2473 | 93.2584 | 91.2088 | 87.3684 |
| | Error (m) | | 879.12 | 978.26 | 348.84 | 1075.27 | 674.16 | 879.12 | 1263.16 |
| Powered | Accuracy | | 90.2174 | 90.2174 | 88.2979 | 93.2584 | 88.2979 | 92.2222 | 89.2473 |
| | Error (m) | | 978.26 | 978.26 | 1170.21 | 674.16 | 1170.21 | 777.78 | 1075.27 |
| Linear | Accuracy | Exponential Decay | 95.402 | 91.209 | 92.222 | 91.209 | 93.258 | 90.217 | 92.222 |
| | Error (m) | | 459.8 | 879.1 | 777.8 | 879.1 | 674.2 | 978.3 | 777.8 |
| Exponential | Accuracy | | 91.209 | 90.217 | 93.258 | 93.258 | 93.258 | 93.258 | 91.209 |
| | Error (m) | | 879.1 | 978.3 | 674.2 | 674.2 | 674.2 | 674.2 | 879.1 |
| Powered | Accuracy | | 90.217 | 94.318 | 93.258 | 92.222 | 90.217 | 90.217 | 96.512 |
| | Error (m) | | 978.3 | 568.2 | 674.2 | 777.8 | 978.3 | 978.3 | 348.8 |

**TABLE 6.** Piecewise and exponential decay learn rate schedule simulation results for RSMprop.

| RSS Type | Parameters | Learn Rate Schedule | LSTM | CNN | RNN | MLP | RBF | SOMs | GAN |
|---|---|---|---|---|---|---|---|---|---|
| Linear | Accuracy | Piecewise | 89.2473 | 90.2174 | 91.2088 | 90.2174 | 91.2088 | 93.2584 | 91.2088 |
| | Error (m) | | 1075.27 | 978.26 | 879.12 | 978.26 | 879.12 | 674.16 | 879.12 |
| Exponential | Accuracy | | 90.2174 | 91.2088 | 90.2174 | 91.2088 | 92.2222 | 88.2979 | 91.2088 |
| | Error (m) | | 978.26 | 879.12 | 978.26 | 879.12 | 777.78 | 1170.21 | 879.12 |
| Powered | Accuracy | | 90.2174 | 90.2174 | 88.2979 | 93.2584 | 88.2979 | 92.2222 | 89.2473 |
| | Error (m) | | 978.26 | 978.26 | 1170.21 | 674.16 | 1170.21 | 777.78 | 1075.27 |
| Linear | Accuracy | Exponential Decay | 91.209 | 91.209 | 90.217 | 96.512 | 92.222 | 90.217 | 92.222 |
| | Error (m) | | 879.12 | 879.12 | 978.26 | 348.84 | 777.78 | 978.26 | 777.78 |
| Exponential | Accuracy | | 92.222 | 93.258 | 92.222 | 91.209 | 90.217 | 92.222 | 90.217 |
| | Error (m) | | 777.78 | 674.16 | 777.78 | 879.12 | 978.26 | 777.78 | 978.26 |
| Powered | Accuracy | | 88.298 | 91.209 | 92.222 | 91.209 | 92.222 | 91.209 | 93.258 |
| | Error (m) | | 1170.2 | 879.1 | 777.8 | 879.1 | 777.8 | 879.1 | 674.2 |

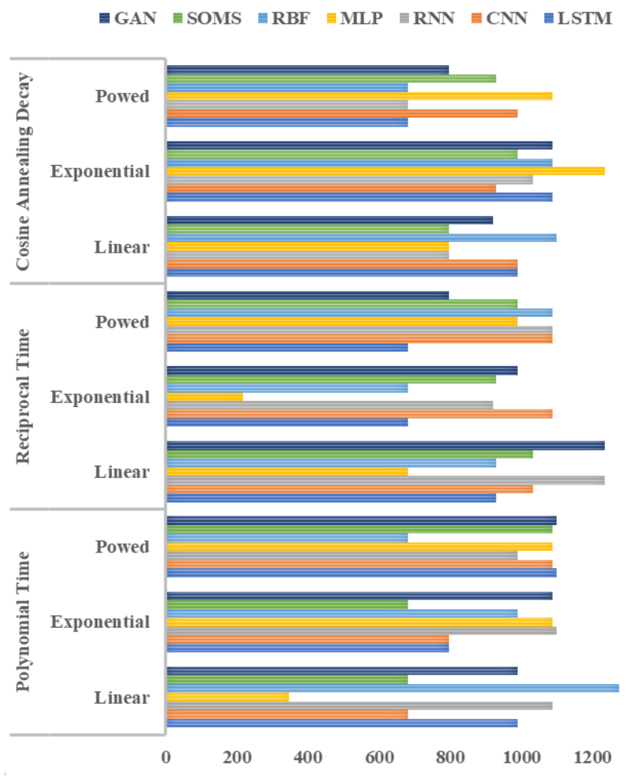

**FIGURE 3.** Polynomial time, reciprocal time and cosine annealing decay learn rate schedule simulation results comparison for RMSprop.

for linear, exponential, and powered RSS using polynomial time, reciprocal time, and cosine annealing decay learn rate schedule RMSprop DL model. The highest accuracy for the polynomial learn rate schedule is 96.54, and the error (m) of 346 for MLP in the linear RSS model. The highest accuracy for the reciprocal learning rate schedule is 97.82, and the error (m) of 218 for MLP in the exponential RSS model. The highest accuracy for cosine annealing decay learn rate schedule is 93.182 and error (m) of 681.8 for LSTM, RNN and RBF in the power RSS model.

## D. ADAPTIVE LEARN RATE SCHEDULES IN SGDM

TABLE 7 presents the RSSI localization accuracy and error (m) for linear, exponential, and powered RSS using piecewise and exponential decay learn rate schedule SGDM DL model. The highest accuracy for piecewise learn rate schedule is 93.2584 and error (m) of 674.16 for RNN, RBF and GAN in the linear RSS model and CNN in the power RSS model. The highest accuracy for the exponential learn rate schedule is 94.318 and error (m) of 568.2 for MLP in the powered RSS model. Figure 4 presents the RSSI localization error (m) for linear, exponential, and powered RSS using polynomial time, reciprocal time, and cosine annealing decay learn rate schedule SGDM DL model. The highest accuracy for the polynomial learn rate schedule is 96.54 and error (m) of 346 for RNN in the linear RSS model and RBF in the power RSS model. The highest accuracy for the reciprocal learning rate schedule is 93.182 and error (m) of 681.8 for MLP in linear and exponential RSS models. The highest accuracy for cosine annealing decay learn rate schedule is 98.98 and error (m) of 102 for MLP in the exponential RSS model.

## E. ANALYSIS OF RESULTS

From the above simulation results, it is observed that the LSTM model's highest accuracy of 95.402 is achieved (linear

**TABLE 7.** Piecewise and exponential decay learn rate schedule simulation results for SGDM.

| RSS Type | Parameters | Learn Rate Schedule | LSTM | CNN | RNN | MLP | RBF | SOMs | GAN |
|---|---|---|---|---|---|---|---|---|---|
| Linear | Accuracy | Piecewise | 90.2174 | 93.2584 | 90.2174 | 87.3684 | 88.2979 | 92.2222 | 85.567 |
|  | Error (m) |  | 978.26 | 674.16 | 978.26 | 1263.16 | 1170.21 | 777.78 | 1443.3 |
| Exponential | Accuracy |  | 90.2174 | 91.2088 | 90.2174 | 91.2088 | 92.2222 | 88.2979 | 91.2088 |
|  | Error (m) |  | 978.26 | 879.12 | 978.26 | 879.12 | 777.78 | 1170.21 | 879.12 |
| Powered | Accuracy |  | 90.2174 | 90.2174 | 88.2979 | 93.2584 | 88.2979 | 92.2222 | 89.2473 |
|  | Error (m) |  | 978.26 | 978.26 | 1170.21 | 674.16 | 1170.21 | 777.78 | 1075.27 |
| Linear | Accuracy | Exponential Decay | 94.318 | 90.217 | 92.222 | 91.209 | 92.222 | 89.247 | 93.258 |
|  | Error (m) |  | 568.2 | 978.3 | 777.8 | 879.1 | 777.8 | 1075.3 | 674.2 |
| Exponential | Accuracy |  | 93.258 | 90.217 | 86.458 | 91.209 | 90.217 | 91.209 | 92.222 |
|  | Error (m) |  | 674.16 | 978.26 | 1354.2 | 879.12 | 978.26 | 879.12 | 777.78 |
| Powered | Accuracy |  | 91.209 | 92.222 | 92.222 | 94.318 | 91.209 | 90.217 | 91.209 |
|  | Error (m) |  | 879.1 | 777.8 | 777.8 | 568.2 | 879.1 | 978.3 | 879.1 |

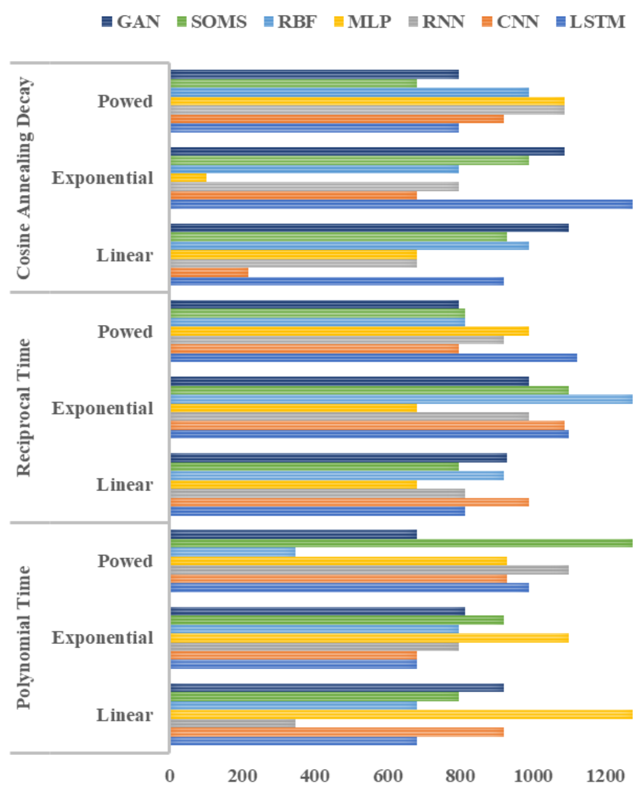

**FIGURE 4.** Polynomial time, reciprocal time and cosine annealing decay learn rate schedule simulation results comparison for SGDM.

RSS model, Adam optimizer, and exponential decay learn rate schedule). For the CNN model, the highest accuracy of 97.82 is achieved (linear RSS model, SGDM optimizer, and cosine annealing learn rate schedule). For the RNN model, the highest accuracy of 96.54 is achieved (linear RSS model, SGDM optimizer, and polynomial learn rate schedule). For the MLP model, the highest accuracy of 98.82 is achieved (exponential RSS model, SGDM optimizer, and cosine annealing learn rate schedule). The RBF model achieves the highest accuracy of 96.54 (powered RSS model, SGDM optimizer, and reciprocal time learn rate schedule). For the SOMs model, the highest accuracy of 95.349 is achieved (powered RSS model, Adadelta optimizer, and cosine annealing learn rate schedule). For the Adadelta optimizer, the highest accuracy of 95.349 is achieved (powered RSS model in SOMs and linear RSS in GAN, Cosine annealing learn rate schedule). Adam optimizer achieves the highest accuracy of 96.518 (powered RSS model in GAN and exponential decay learn rate schedule). For RMSprop, the highest accuracy of 97.82 is achieved (exponential RSS model in MLP and exponential decay learn rate schedule). For SGDM, the highest accuracy of 98.82 is achieved (exponential RSS model in MLP and cosine annealing learn rate schedule). For the linear RSS model, the highest accuracy of 97.82 is achieved by using CNN, SGDM, and cosine annealing learn rate schedule; for the exponential RSS model, the highest accuracy of 98.98 is achieved by using MLP, SGDM and cosine annealing learn rate schedule, for powered RSS model, the highest accuracy of 96.54 is achieved by using RBF, SGDM and reciprocal learn rate schedule. It is observed that the most suitable DL model is MLP, the optimizer is SGDM, and the learning rate schedule is cosine annealing decay with the maximum accuracy of 98.98, which is witnessed in many of the comparison results for this considered dataset.

### F. ANALYSIS OF LEARN RATE SCHEDULES FOR DIFFERENT LOCALIZATION DATASETS

The datasets used in LoRaWAN localization usually contain information gathered from LoRaWAN devices with location-tracking capabilities. These datasets can be divided into groups according to several criteria, including the kind of localization method applied, the data setting, and the particular application area. Typical dataset types that are accessible in LoRaWAN localization include the following:

(i) **Signal Strength Datasets:** RSSI and Signal Noise-to-Ratio (SNR) measurements obtained from LoRaWAN gateways are commonly included in signal strength databases. Data points may include timestamps, gateway IDs, device IDs, and signal strength values. Signal strength datasets are often used for fingerprinting-based localization and trilateration techniques. Since multi-step decay and exponential decay can adjust the learning rate in response to variations in signal strength over time, they might be appropriate in this situation. It would be ideal to utilize multi-step or exponential decay to lower the learning rate after processing every interval's data in a signal strength dataset and aid in the model's adaptation to variations in signal strength over various time intervals.

(ii) **Time Difference of Arrival (TDoA) Datasets:** Time stamps captured by several LoRaWAN gateways are included in TDoA datasets. These datasets enable multilateration-based localization techniques by measuring the differences in arrival times of signals at different gateways. Data points typically include timestamps, gateway IDs, signal strength or other signal characteristics. Power decay or reciprocal time decay may be appropriate because they can modify the learning rate depending upon the difficulty of the multilateration computations, which is relevant as TDoA data frequently deals with multilateration-based localization. Power decay can adjust the learning rate for TDoA datasets according to the number of gateways used in the localization process.

(iii) **Received Signal Strength Indicator (RSSI) Fingerprints:** Gathering RSSI readings from several reference points in a localization area creates RSSI fingerprint databases. A fingerprint database links each reference point to a specific location. A device's location is estimated during localization by comparing its RSSI measurements with the fingerprint database. Cosine annealing decay could be appropriate since RSSI fingerprints are frequently utilized for fingerprinting-based localization. Because it can assist the model in exploring various regions of the fingerprint database, when training a localisation model with RSSI fingerprints, use cosine annealing decay to alter the learning rate cyclically. It would enable the model to explore various reference locations inside the fingerprint database.

(iv) **Anchor Node Datasets:** The fixed reference nodes, or anchors, placed throughout the localization region are the source of these databases. Usually, anchor nodes communicate with the LoRaWAN device that is being localized and have known positions. For localization algorithms, data gathered from anchor nodes may include timestamps, anchor IDs, and signal measurements. Here, exponential decay or reciprocal time decay might work well since they can progressively change the learning rate in response to anchor node observations that show stability. Using exponential decay will lower the learning rate in an anchor node dataset as the model converges based on the stability of anchor node measurements.

(v) **Crowdsourced Localization Datasets:** Crowdsourced datasets involve collecting localization data from many LoRaWAN devices deployed and operated by end-users or volunteers. These datasets help assess how well localization algorithms work in practical situations with various environmental factors and device locations. In this case, multi-step decay or cosine annealing decay would be appropriate since they can modify the learning rate in response to the density and diversity of the crowdsourced data. Multi-step decay can alter the learning rate of a localization model trained on crowdsourced data according to the density of data gathered from various locations.

(vi) **Indoor vs. Outdoor Datasets:** Datasets can be classed depending on whether the localization occurs indoors or outdoors. Outdoor datasets might offer more apparent LoS circumstances, but indoor datasets frequently feature more complicated propagation environments with obstructions and multipath effects. Data from indoor and outdoor environments may differ regarding environmental conditions and signal transmission. Both indoor and outdoor datasets may benefit from using exponential decay or reciprocal time decay since they can modify the learning rate in response to the intricacy of the localization environment. Reciprocal time decay can be used to adjust the learning rate according to the temporal dynamics of the localization environment in both indoor and outdoor datasets.

(vii) **Publicly Available Datasets:** Publicly accessible datasets for LoRaWAN localization research and development are available from a few organizations and research institutions. These datasets are useful for benchmarking and comparing various localization techniques because they might contain ground truth location data. Given that they can modify the learning rate in response to the quantity and complexity of the dataset, exponential decay or cosine annealing decay may be appropriate in this situation. Use exponential decay to progressively lower the learning rate as the model converges when conducting localization research utilizing a publically available dataset.

### 1) ILLUSTRATIVE EXAMPLE FOR MULTI-STEP DECAY

A dataset was gathered from LoRaWAN gateways to train a neural network for signal strength prediction. We employ multi-step decay to adjust the learning rate in response to signal strength variations over time. Also, muti-step can be preferred to adapt the learning rate based on the diversity and density of the crowdsourced data during training. Assume the decay factor is set to 0.1 and the initial learning rate to 0.01. Every ten epochs, you choose to reduce the learning rate. The learning rate stays at 0.01 at epoch 0. The learning rate increases to $0.01 \times 0.1 = 0.001$ at epoch 10. The learning rate stays at 0.001 at epoch 20. And so on. It may enhance convergence and performance by enabling the model to modify its learning rate in response to the observed variations in signal strength over time. It also allows the model to adjust its learning rate based on the diversity and density of the crowdsourced data, potentially improving convergence and performance during training.

### 2) ILLUSTRATIVE EXAMPLE FOR EXPONENTIAL DECAY

For forecasting the signal strength, choose exponential decay. Anchor node datasets can opt for Exponential decay as they can gradually adjust the learning rate based on the stability of the anchor node measurements. Data collected from indoor and outdoor environments and publicly available datasets can also use exponential decay to adjust the learning rate based on the size and complexity of the localization environment during training. We have set the decay rate to 0.001 and the initial learning rate to 0.01. The learning rate stays at 0.01 at epoch 0. The learning rate changes to $0.01 \times e^{-0.001\times 10} \approx 0.00951$ at epoch 10. The learning rate changes to $0.01 \times e^{-0.001\times 20} \approx 0.00903$ at epoch 20. And so on. Analogously, exponential decay, whose decay rate is determined by the decay rate selected, enables the model to modify its learning rate in response to variations in signal strength over time. This model also allows it to adjust its learning rate gradually based on the stability of the anchor node measurements, with faster decay for more unstable conditions. This model also allows it to gradually adjust its learning rate based on the complexity of the indoor and outdoor environments, with a faster decay for more complex scenarios. This model also allows it to gradually adjust its learning rate based on the size and complexity of the publicly available dataset, with a faster decay for larger and more complex datasets.

### 3) ILLUSTRATIVE EXAMPLE FOR POWER DECAY

Using a TDoA dataset gathered from LoRaWAN gateways; we are building a localization model for multilateration-based localization. The choice of how to modify the learning rate by using power decay is influenced by the intricacy of the multilateration computations. Set the power factor to 2, decay rate to 0.001, and initial learning rate to 0.01. The learning rate stays at 0.01 at epoch 0. The learning rate changes to $\frac{(1+0.001\times 10)^2}{0.01} \approx 0.00673$ at epoch 10. The learning rate changes to $\frac{(1+0.001\times 20)^2}{0.01} \approx 0.00452$ at epoch 20. And so on. The model can modify its learning rate according to the multilateration's complexity through power decay.

### 4) ILLUSTRATIVE EXAMPLE FOR RECIPROCAL TIME DECAY:

Consider reciprocal time decay for the TDoA, anchor node, and indoor & outdoor datasets. Set the initial learning rate to 0.01 and the decay rate to 0.001. The learning rate stays at 0.01 at epoch 0. The learning rate becomes $\frac{1+0.001\times 10}{0.01} \approx 0.00990$ at epoch 10. The learning rate becomes $\frac{1+0.001\times 20}{0.01} \approx 0.00980$ at epoch 20. And so on. Reciprocal time decay enables the model to gradually respond to variations in the complexity of multilateration computations by adjusting the learning rate in response to the training duration. It also can adjust the learning rate based on the training time, allowing the model to adapt to changes in the stability of the anchor node measurements over time. It also can adjust the learning rate based on the training time, allowing the model to adapt to the complexity of the indoor and outdoor environments over time.

### 5) ILLUSTRATIVE EXAMPLE FOR COSINE ANNEALING DECAY

An RSSI fingerprint dataset from several reference points trains a localization model. We employ cosine annealing decay to aid the model in exploring various regions of the fingerprint database throughout training. Also, we decided to use cosine annealing decay to adapt the learning rate based on the diversity and density of the crowdsourced data and publicly available datasets during training. Set the minimum learning rate to 0.001 and the maximum learning rate to 0.01. the total number of epochs is set to 100. At epoch 0, the learning rate becomes:

$\mathrm{lr} = 0.001 + \frac{2}{1}(0.01 - 0.001) \times \left(1 + \cos\left(\frac{100}{\pi} \times 0\right)\right) = 0.001 + \frac{2}{0.009} \times (1+1) = 0.001 + 0.0045 = 0.0055$

At epoch 50 (halfway through training), the learning rate becomes:

$\mathrm{lr} = 0.001 + \frac{2}{1}(0.01 - 0.001) \times \left(1 + \cos\left(\frac{100}{\pi} \times 50\right)\right) = 0.001 + \frac{2}{0.009} \times (1+0) = 0.001 + 0.0045 = 0.0055$

At epoch 100 (end of training), the learning rate becomes:

$\mathrm{lr} = 0.001 + \frac{2}{1}(0.01 - 0.001) \times \left(1 + \cos\left(\frac{100}{\pi} \times 100\right)\right) = 0.001 + \frac{2}{0.009} \times (1-1) = 0.001$

During training, cosine annealing decay helps the model adjust to the quantity and complexity of the publicly available dataset by allowing the learning rate to vary cyclically between the minimum and maximum values throughout epochs. This helps the model adapt to the diversity and density of the crowdsourced data during training and also to the size and complexity of the publicly available datasets.

## V. CONCLUSION AND FUTURE WORK

This paper proposes a DL model and optimizers with a learning rate schedule mechanism to improve the accuracy in RSSI- localization for LoRaWAN-IoT-based networks. The different learn rate schedule formulation and its adaptation to DL models to improve the accuracy of RSSI-based localization have been simulated, and the results are compared, tabulated, and analyzed. The results show that the highest accuracy of 98.98 and error distance of 102 meters is observed using the MLP model, SGDM optimizer, exponential RSS model, and cosine annealing decay learn rate schedule method. Experiments show that the training efficiency and accuracy are improved compared to the existing default parameter setting DL models. The study demonstrates how learning rate schedules enhance the precision of RSSI-based localization in LoRaWAN IoT networks. The models can perform better by more effectively adapting to the features of the localization datasets by dynamically varying the learning rate during training with hybrid learn rate schedules. Analysis of different learning rate schedules for various localization datasets with illustrated examples presents how one can choose the proper learning rate schedule for other LoRaWAN parameter settings and the datasets. However, further study is required when applying the DL model with an adaptive learning rate to more real-world practical problems.

## REFERENCES

[1] S. M. D. Kumar, "Multi-objective stochastic gradient based ADR mechanism for throughput and latency optimization in LoRaWAN," *Int. J. Sensors, Wireless Commun. Control*, vol. 13, no. 6, pp. 403–417, Nov. 2023.

[2] R. Swathika and S. M. D. Kumar, "Optimizing throughput using effective contention aware adaptive data rate in LoRaWAN," in *Proc. Int. Conf. Frontiers Comput. Syst.*, vol. 690, 2022, pp. 291–301.

[3] R. Swathika and S. M. D. Kumar, "Analysis of BER performance over AWGN and Rayleigh channels using FSK and PSK modulation schemes in LoRa based IoT networks," in *Proc. Int. Conf. Intell. Innov. Technol. Comput., Electr. Electron. (IITCEE)*, Jan. 2023, pp. 351–357.

[4] *LoRaWAN New Transmission Record*. Accessed: May 2023. [Online]. Available: https://www.hackster.io/news/another-record-breaking-transmission-for-lorawan-0cca5f6cd032

[5] T. Janssen, R. Berkvens, and M. Weyn, "Comparing ML algorithms for RSS-based localization in LPWAN," in *Proc. Int. Conf. P2P, Parallel, Grid, Cloud*, 2020, pp. 1–10.

[6] E. Svertoka, A. Rusu-Casandra, R. Burget, I. Marghescu, J. Hosek, and A. Ometov, "LoRaWAN: Lost for localization?" *IEEE Sensors J.*, vol. 22, no. 23, pp. 23307–23319, Dec. 2022.

[7] C. Gu, L. Jiang, and R. Tan, "LoRa-based localization: Opportunities and challenges," in *Proc. Int. Conf. Embedded Wireless Syst. Netw.*, 2019, pp. 413–418.

[8] J. Liang, Y. Xu, C. Bao, Y. Quan, and H. Ji, "Barzilai–borwein-based adaptive learning rate for deep learning," *Pattern Recognit. Lett.*, vol. 128, pp. 197–203, Dec. 2019.

[9] H. Kwasme and S. Ekin, "RSSI-based localization using LoRaWAN technology," *IEEE Access*, vol. 7, pp. 99856–99866, 2019.

[10] M. Anjum, M. A. Khan, S. A. Hassan, A. Mahmood, H. K. Qureshi, and M. Gidlund, "RSSI fingerprinting-based localization using machine learning in LoRA networks," *IEEE Internet Things Mag.*, vol. 3, no. 4, pp. 53–59, Dec. 2020.

[11] L. Aarif, M. Tabaa, and H. Hachimi, "Experimental test and performance of RSSI-based indoor localization in LoRA networks," *Proc. Comput. Sci.*, vol. 203, pp. 420–425, Sep. 2022.

[12] E. H. Yoshitome, J. V. R. da Cruz, M. E. P. Monteiro, and J. L. Rebelatto, "LoRa-aided outdoor localization system: RSSI or TDoA?" *Internet Technol. Lett.*, vol. 5, no. 2, pp. e319, Mar. 2022.

[13] W. Ingabire, H. Larijani, R. M. Gibson, and A.-U.-H. Qureshi, "LoRaWAN based indoor localization using random neural networks," *Information*, vol. 13, no. 6, p. 303, Jun. 2022.

[14] S. Sadowski and P. Spachos, "RSSI-based indoor localization with the Internet of Things," *IEEE Access*, vol. 6, pp. 30149–30161, 2018.

[15] M. W. P. Maduranga and R. Abeysekara, "Supervised machine learning for RSSI based indoor localization in IoT applications," *Int. J. Comput. Appl.*, vol. 183, no. 3, pp. 26–32, May 2021.

[16] A. Piroddi and M. Torregiani, "Machine learning applied to LoRaWAN network for improving fingerprint localization accuracy in dense urban areas," *Network*, vol. 3, no. 1, pp. 199–217, Feb. 2023.

[17] T. Perković, L. Dujić Rodić, J. Šabić, and P. Šolić, "Machine learning approach towards LoRaWAN indoor localization," *Electronics*, vol. 12, no. 2, p. 457, Jan. 2023.

[18] I. T. Ali, A. Muis, and R. F. Sari, "A deep learning model implementation based on rssi fingerprinting for LoRA-based indoor localization," *EUREKA, Phys. Eng.*, vol. 1, pp. 40–59, Jan. 2021.

[19] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Recurrent neural networks for accurate RSSI indoor localization," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10639–10651, Dec. 2019.

[20] J. Li and X. Yang, "A cyclical learning rate method in deep learning training," in *Proc. Int. Conf. Comput. Inf. Telecommun. Syst. (CITS)*, Oct. 2020, pp. 1–5.

[21] J. Yang and F. Wang, "Auto-ensemble: An adaptive learning rate scheduling based deep learning model ensembling," *IEEE Access*, vol. 8, pp. 217499–217509, 2020.

[22] M. Mori and M. Nakano, "Efficient cyclic learning rate schedules and their evaluations for neural network ensemble," in *Proc. IEEE 28th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2018, pp. 1–6.

[23] J. Konar, P. Khandelwal, and R. Tripathi, "Comparison of various learning rate scheduling techniques on convolutional neural network," in *Proc. IEEE Int. Students' Conf. Elect., Electron. Comput. Sci.*, Feb. 2020, pp. 1–5.

[24] M. Xue, J. Li, and Q. Luo, "Toward optimal learning rate schedule in scene classification network," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.

[25] S. Noppitak and O. Surinta, "DropCyclic: Snapshot ensemble convolutional neural network based on a new learning rate schedule for land use classification," *IEEE Access*, vol. 10, pp. 60725–60737, 2022.

[26] A. Fajar, R. Sarno, C. Fatichah, R. Indarto Susilo, and G. Pangestu, "Cyclical learning rate optimization on deep learning model for brain tumor segmentation," *IEEE Access*, vol. 11, pp. 119802–119810, 2023.

[27] A. Mustapha, L. Mohamed, and K. Ali, "Comparative study of optimization techniques in deep learning: Application in the ophthalmology field comparative study of optimization techniques in deep learning: Application in the ophthalmology field," *J. Phys. Conf. Series.*, vol. 1743, 1743, Art. no. 012002.

[28] R. Zaheer and H. Shaziya, "A study of the optimization algorithms in deep learning," in *Proc. 3rd Int. Conf. Inventive Syst. Control (ICISC)*, Jan. 2019, pp. 536–539.

[29] M. Chandra Mukkamala and M. Hein, "Variants of RMSProp and adagrad with logarithmic regret bounds," 2017, *arXiv:1706.05507*.

[30] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.

[31] *MATLAB Software*. Accessed: May 2023. [Online]. Available: https://in.mathworks.com/products/new_products/release2020a.html

[32] *Kaggle LoRaWAN Localization Dataset*. Accessed: May 2023. [Online]. Available: https://www.kaggle.com/datasets/thedevastator/sigfox-and-lorawan-localizationtool?select=lorawan_antwerp_2019_dataset.csv

**R. SWATHIKA** is currently a Research Scholar with the Department of Computer Science and Engineering, University of Visvesvaraya College of Engineering (UVCE), Bengaluru, and a Faculty Member with the Department of Computer Science and Engineering, MVJ College of Engineering, Bengaluru. She has been teaching computer science for the past eight years and pursuing research for three years. She has published one Indian Patent and more than 15 research articles in international journals and conferences, including IEEE Xplore, Springer, and Bentham Science. Her research interests include the Internet of Things, networks, machines, and deep learning. She received one best paper award in international conferences. She served as a TPC reviewer for an international conference.

**S. M. DILIP KUMAR** is currently a Professor with the Department of Computer Science and Engineering, University of Visvesvaraya College of Engineering (UVCE), Bengaluru, an autonomous public university on the IIT model as per the UVCE Act 2021. He is also heading the Training and Placement Office at UVCE. He served as the Head of the Department of Computer Science and Engineering, UVCE, from 2020 to 2022. He is involved in research and teaching B.Tech. and M.Tech. students of computer science and engineering and has 26 years of teaching experience. He has guided eight Ph.D. candidates, and six are pursuing the Ph.D. in computer science and engineering under his guidance at Bangalore University. He has published 135 articles in international journals, including IEEE, Elsevier, Springer, Inderscience, and Conference papers. The number of Google Scholar citations is 1026. He has published two Indian Patents. He was a Principal Investigator for a research project in the area of grid computing sponsored by the Science and Engineering Research Board, Department of Science and Technology (SERB-DST), Government of India. Another research project funded by the Science and Engineering Research Board, Department of Science and Technology (SERB-DST), Government of India, in the area of Internet of Things (IoT) is completed. He has completed two consultancy projects in the areas of mobile governance and e-FMS sponsored by the Government of Karnataka.

Further, two research projects in the area of IoT, each sponsored by Bangalore University, are ongoing. He has served in over 100 technical committees of the State Government, Universities, and Colleges. His current research interests include the Internet of Things and fog computing. He has received six best paper awards at International Conferences. 33 papers have appeared in the best quartile SCI Rank Journals, out of which seven papers have appeared in Q1 Journals. He was a recipient of the Exemplary Leadership Accolade Award in recognition of Outstanding Leadership, Vision, and Dedication to Shaping the Future at the Panchajanya Corporate-Academia-Media Ceremony, in August 2023. He has received the Excellence in Academics Award 2024 from the Alma mater.

**N. N. SRINIDHI** received the Ph.D. degree in computer science and engineering. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Manipal Institute of Technology Bengaluru, MAHE Manipal. He was a Project Fellow for a SERB-DST-sponsored research project in the area of IoT worth 40 lakhs. He reviewed four different projects. He has published more than 40 research articles in international journals, including Elsevier, Inderscience, Springer, Taylor & Francis, and International Conferences. He has been involved in research and teaching for six years and has three years of industrial experience. His current research interests include sensor networks, cloud computing, fog computing, edge computing, and the IoT. He served as a reviewer for various reputed journals, including Springer, IEEE, and Elsevier, and delivered expert talks on WSN, the IoT, and Robotics in various colleges, including IIT, NIT, DIAT, and other premier institutions. He served as the guest editor, an editorial member, the session chair, and a TPC member, for various journals and conferences. He is an international reviewer for research projects from Sultan Qaboos University, Oman.

**B. R. HARSHITHA** received the B.E. degree from the University of Visvesvaraya College of Engineering (UVCE), Bengaluru, in 2021. She is currently pursuing the master's degree in computer science with the University of Southern California (USC), Los Angeles, CA, USA. She was a Developer Associate at SAP Labs, India, following which she worked at Indian Institute of Science (IISc), Bengaluru, as a Research Assistant on HCI and computer vision applications. Her research interests include machine learning, deep learning, computer vision, and the IoT.