A REVIEW OF OPTIMIZATION ALGORITHMS IN DEEP LEARNING MODELS FOR IMPROVING THE FORECASTING ACCURACY IN SEQUENTIAL DATASETS WITH APPLICATION IN THE SOUTH AFRICANSTOCK MARKET INDEX

Sanele Makamo

Benguela Global Fund Managers

ABSTRACT

In this paper we review different popular optimization algorithms for machine learning models, we then evaluate the model performance and convergence rates for each optimizer using a multilayer fully connected neural networks. Using sequential dataset of index returns (time-series data) spanning over of 20-years, we demonstrate Adam and RMSprop optimizers can efficiently solve practical deep learning problems dealing with sequential datasets. We use the same parameter initialization when comparing different optimization algorithms. The hyper-parameters, such as learning rate and momentum, are searched over a dense grid and the results are reported using the best hyper-parameter setting.

KEYWORDS

Machine learning, deep learning, neural networks, optimization algorithms, loss function.

1. INTRODUCTION

Most machine learning problems once identified and formulated can be solved as optimization problems which makes optimization an integral component of machine learning (Sun & et. al). Optimization algorithms form the basis on which machine learning models can learn through experience and memory. The goal of machine learning models is to make the model's predictions as close as possible to the actual target values, this is known as minimization of the loss function. The optimization algorithms aim to minimize the loss function and find the optimal values for the parameters that result in the best performance of the model (Kabaar, 2024).

The rapid growth in the number of datasets and increase in model complexity has led to challenges in determining the most optimal algorithm to use since each optimizer has its own characteristics, advantages, and limitations, and their performance can differ depending on the on the problem at hand, datasets, and the network architecture. Normally experimentation and tuning are typically used to determine which optimal algorithm is best applicable to a particular dataset (Kabaar, 2024).

In this paper, we review the characteristics, advantages and limitations of optimization algorithms and we evaluate the model performance for each optimization algorithm using a 20-year time-

David C. Wyld et al. (Eds): AdNLP, CDKP, NCO, SAI, MLT, Signal, SOFT, ICAITA, CMC - 2024
pp. 173-181, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141015

series of returns from the JSE all share index, we only focus on two deep learning models, the recurrent neural network (RNN) and the long-short memory loss model (LSTM) with fully connected hidden layers and dense units with activations and dropout rates to compare the different optimization algorithms.

2. BACKGROUND

The exponential growth in the amount of data has led to the rapid growth in the use of machine learning as a tool to provide solutions to real world problems, these includes solutions in the fields of image recognitions used in self-driving cars, generative artificial intelligence such as ChatGPT's, speech recognition, machine translation, recommendation systems, predictive analytics etc.
Neural networks (NNs) have their origin in the study field of human nervous system, known as neurology, where researchers were curious on how the human brain and its network of interconnected neurons work together. NNs are designed to produce computational representations of biological neural network behaviour (Kabaar, 2024).

The architecture of the NNs work in two ways. Firstly, the neuron receives inputs from the previous layer or directly from the input data. Each input is multiplied by a weight value, which represents the strength or importance of that connection. The weighted inputs are then summed together.

Secondly, after the weighted sum, an activation function is applied to introduce nonlinearity into the output of the neuron. The activation function determines the neuron's output value based on the summed inputs.

In the training process, the NNs adjusts the weights of its connections to improve its performance. An optimization algorithm is required in this process where it computes the gradient of the loss function with respect to the network's weights, allowing the weights to be updated in a manner that minimizes the loss function.

NNs are capable to learn and generalize from data and with the development of deep learning, due to latest developments in deep learning, NNs have been developed with multiple hidden layers making them more suitable for complex artificial intelligent tasks such as image recognition, natural language processing (NLP), machine translation etc.

Deep learning models like the recurrent neural networks (RNNs) and convolution neural networks (CNNs) are the two popular neural networks (NNs) which play a vital role in machine learning. The CNNs are feedforward neural networks with convolution calculation. CNNs have been successfully implemented in many fields such as image processing video processing and natural language processing (NLP).

RNNs are a kind of sequentialNNs model and very active in natural language processing (NLP). RNNs have proven to produce good results in the field of constrained optimization, in such cases the parameters of the weights in the RNNs can be learned by optimization algorithms which can find optimal solutions according to the trajectory of the state solution(Sun & et. al).

LSTM is a special type of Recurrent Neural Networks (RNN) with a broad range of applications including time series analysis, document classification, speech, and voice recognition.the LSTM can solve the long-term dependence problem of the data which can be a limitation for the RNNs.

3. OPTIMIZATION ALGORITHMS

Gradient descent

Gradient descent is the most frequently used optimization algorithm used to minimize the loss function of a model. It serves as the fundamental optimization for first-order optimizations methods used in machine learning. In simpler words, gradient refers to a slope of a surface, to get the lowest point in the surface one must continually tilt or descend the slope.

In deep learning models like neural networks, the gradient descent optimization works by updating the network's weights and biases in the direction opposite to the gradient of the loss function with respect to the parameters from the objective function (Kabaar, 2024).

The advantage of this method is that the solution is global when the objective function is convex. The disadvantage is that in each parameter update, gradients of total samples need to be calculated, so the calculation cost is high (Sun & et. al).

Stochastic Gradient Descent

Stochastic gradient descent (SGD) is an iterative optimization algorithm normallyimplemented for training machine learning models, it is a variant of gradient descent that randomly selects a single training example or a mini batch of examples to compute the gradient and update the parameters. It provides a computationally efficient way and introduces noise in the training process, which can reduce local minima and find better global optima (due to its stochastic nature) (Kabaar, 2024).

The advantage of this algorithm is that the calculation time for each update does not rely on the total number of training samples, and results in saving thecomputation cost. It can improve generalization by exposing the model to different training examples in each iteration, thereby reducing overfitting (Kabaar, 2024).

The disadvantage is that it is difficult to choose an appropriate learning rate, and using the same learning rate for all parameters is not effective. The learning rate will be oscillating in the later training stage of some adaptive methods which may lead to the problem of non-converging (Sun & et. al). This challenge of SGD is normally solved by introducing momentum: a method that helps accelerate SGD in the relevant direction and dampens oscillations (Ruder, 2017).

Adaptative Gradient Descent (AdaGrad)

Adagrad is a gradient-based optimization that works by adapting the learning rate to the parameters, performing larger updates for less frequent and smaller updates for frequent parameters. Dean et al. (2012), results showed that the Adagrad vastly improved the robustness of SGD and it well suitable for training large-scale neural networks at Google and performed good image recognition on YouTube.

The advantage of the Adagrad's optimizer is that it removes the need to manually tune the learning rate whereas most optimizers use a default value of 0.01. The disadvantage of the Adagrad optimizer is its accumulation of the squared gradients in the denominator, that is, since every added term is positive, the accumulated sum keeps growing during training. This then leads to the learning rate diminishing and consequently becoming infinitesimally small, at which point the algorithm is can no longer absorb additional information (Ruder, 2017).

Root Mean Squared Propagation (RMSProp)/AdaDelta

RMSprop and Adadelta have both been developed independently around the same time stemming from the need to resolve Adagrad's radically diminishing learning rates (Ruder, 2017). The RMSprop works by adjusting the learning rate for each parameter based on the mean of the recent squared gradients. It calculates an exponentially weighted moving average of the squared gradients over time (Ruder, 2017).

The advantage of the RMSprop is that it improves the ineffective learning rate problem in late-stage gradient descent, the disadvantage being that in the late training stage the updateprocess may be repeated around thelocal minimum a challenge which can be eliminated by introducing momentum technique.

Adaptive moment estimation (Adam)

This is an optimization method for efficient stochastic optimization that only requires first-order gradients with less memory requirement. The algorithm calculates individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name Adam is derived from adaptive moment estimation (Kingman & Lei Ba, 2015).

This optimization method was designed to combine two popular optimization methods, namely, the Adagrad, which functions well with sparse gradients, and the RMSProp method which works well in on-line and non-stationary settings(Kingman & Lei Ba, 2015). Also, the Adam optimizer adds a bias-correction and momentum to RMSprop which allows it to perform slightly better than the RMSprop at the later stage of training (Ruder, 2017).

Adam's advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its step sizes are approximately bounded by the step size hyperparameter, it does not require a stationary objective and it normally performs a form of step size adaptation (Kingman & Lei Ba, 2015).The method was designed for machine learning problems with large datasets and/or high-dimensional parameter spaces.

4. Dataset and Experimentation

The data considered for this is the time series data of index returns over the past 20-years from the JSE All Share Index (. JALSH) prices obtained from the eikon refinitiv platform.

We perform a 90/20 train-test-split where 90% of the data i.e., 2004/07/31 to 2021/12/31 is reserved for training the model so that it understands the neural network representation to predict the future values and the 10% of the data i.e., 2022/01/31 to 2023/12/31, is reserved for testing the model's performance on the data it has never seen before.

In simple terms in this study, we use a deep learning model to forecast T+1 return, R_{t+1}, of the JSE all share index in a monthly time frame. This is our independent variable. We do this by using the last 240 monthly returns of the index. This being our independent variable. In machine learning we divide four datasets called the train-test split.

Firstly, we have the x-test dataset, which is the in-sample set of features (i.e., independent variables) that explain the variations of the variable that you want to forecast. They are the predictors. Secondly, we have the y-train dataset, the in-sample set of dependent variables (i.e., the right answers) that you want the model to calibrate its forecasting function on.

Thirdly, we have the x-test dataset, the out-of-sample set of features that will be used as a test of the model to see how it performs on this never-before-seen data. Lastly, we have the y-test dataset which contains the real or actual values that the model must approach. In other words, these are the right answers that will be compared with the model's forecasts.

Let $R_{t+1} \in \mathbb{R}^{T\times 1}$ be a logarithm return on a JSE all share market index, $X_t \in \mathbb{R}^{T\times p}$ be a high dimensional set of predictor variables. In this case X_t is formed from the last 20-year monthly returns series and follows a vector autoregressive model VAR(1). The predictors are extracted from the index return series set with the dual goal of good out-of-sampleprediction and in-sample model fit, the mean squared error (MSE) is the model performance metric used for the out-of-sample predictor performance. In this study the model follows an autoregressive formulation:

$$R_{t+1} = c + \beta_1 X_t + \ \beta_2 X_t + \cdots + \beta_n X_k + \varepsilon_{t+1} \ldots (1)$$
$$X_t = c + \beta_1 X_{t-1} + \ \beta_2 X_{t-2} + \cdots + \beta_n X_{t-k} + \varepsilon_t$$

Extending this to deep learning, which is a data reduction method that uses L layers of hidden predictors which can be highly non-linear, we get the model form:In this case$R_{t+1}$is a linear additive combination of input variables X_t and latent factors F_t,

$$X_t = \gamma + \gamma_1 X_{t-1} + \ \beta_f F_t + \varepsilon_t \ldots (2)$$
$$F_t = F^{W,b}(X_t)$$
$$F^{W,b} = f_1^{W,b} + f_2^{W,b} \circ \ldots \circ f_L^{W_L,b_L}$$
$$f^{W_L,b_L}(Z) = f_l(W_l Z + b_l), \qquad \forall \leq l \leq L$$

Where$F: \mathbb{R}^{T\times p} \to \mathbb{R}^{T\times 1}$is a multivariate data reduction map represented as a deep learner.Deep learning will estimate coefficient $\gamma$and latent factors, F_t, jointly.Where(W, b) are weights and offsets to be trained. Here $\varepsilon_t$are the usual idiosyncraticpricing errors.$F_t$is constructed as a composition of univariate semi-affine functions and our choice for activation function is the rectified linear unit$ReLU(x)$. This led to Deep ReLU networks which are mostly used in the application of image processing and gaming.

To train the model, we need a loss function to minimize the prediction error, the commonly used loss function is the mean squared error (MSE) of the in-sample fit of$\hat{X}_{t+1}$

$$L = \frac{1}{T}\sum_{t=1}^{T}\left(X_{t+1} - \hat{X}_{t+1}\right)^T\left(X_{t+1} - \hat{X}_{t+1}\right) + \alpha\gamma(\beta, W, b) \ldots (3)$$

Where $\alpha\gamma(\beta, W, b)$ is the regularization penalty to induce the predictor selection and avoid model overfitting and α controls the amount of regulation.

In our experiments, we made model choices that are consistent with previous publications in the area; a Simple RNN and LSTM neural networks model with two fully connected hidden layers with hidden units each and ReLU, tanh and sigmoid activation are used for this experiment with random choices of minibatch sizes. We compare the effectiveness of optimization algorithm of the first order methods on multi-layer neural networks trained with dropout noise.

5. RESULTS

Model evaluation metrics:

Model evaluation deals with the algorithm’s performance in its forecasts, we evaluate the model using the loss function, which is simply a mathematical calculation that measures the difference between the predictions and the real (test) values. We measure this using the accuracy metrics as defined below:

Accuracy also known as the hit ratio in finance, measures the percentage number of correct predictions relative to the total number of predictions.

Directional accuracy refers to how accurately model predict the future movement direction of index returns. It is measured using the confusion matrix (Tsai & Wang, 2009). A confusion matrix shows the number of Type I and Type II errors, together with the number of correct forecasts. In this case, a Type I error is called as a downward movement in price when an increase is forecast (i.e. a false positive), and a Type II error is called as an upward price movement when a decrease is forecast (i.e. a false negative). Then correct forecasts are also split into true positives and true negatives as illustrated by the table below:

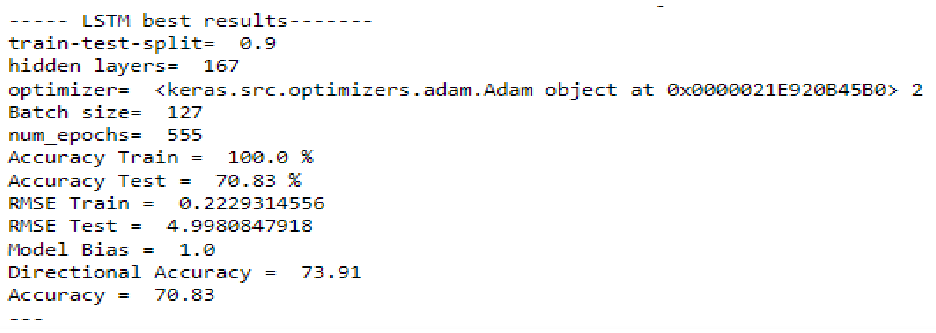
Table 1: Confusion matrix

		Actual Direction	
		Up	Dow
Predicted Direction	Up	True Positive	False Positive (Type I error)
	Down	False Negative (Type II Error)	True Negative

Root mean squared error (RMSE): which is the squared root of the average of the squared differences between the predicted and actual values (MSE).

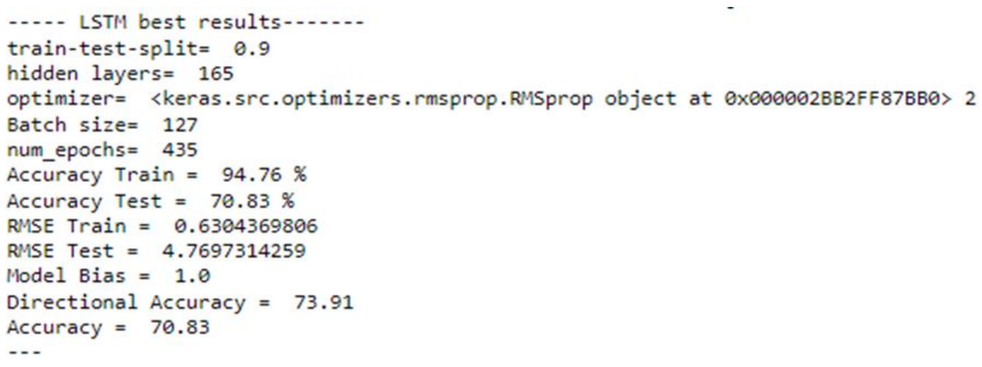
Model bias refers to the directional tendency of algorithms, often caused by structural or external factors, where one trading direction is favored over the other, quantified by the ratio of bullish signals to bearish signals; an ideal bias of 1.00 indicates a balanced trading system:

$$model\ bias = \frac{number\ of\ bullish\ signals}{number\ of\ bearish\ signals}$$



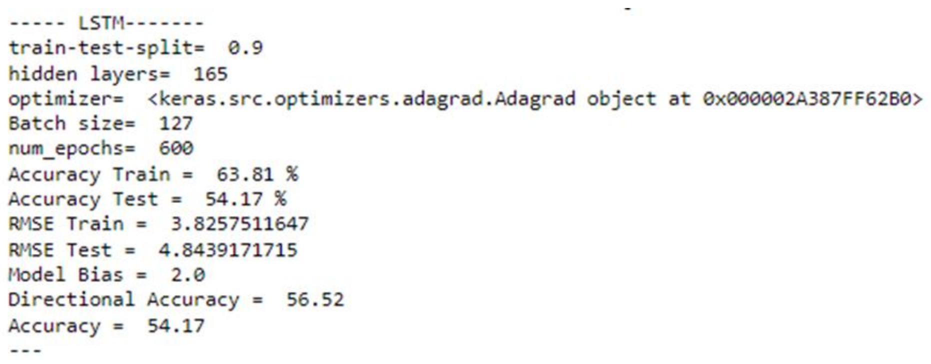
```
----- LSTM best results-------
train-test-split=  0.9
hidden layers=  167
optimizer=  <keras.src.optimizers.adam.Adam object at 0x0000021E920B45B0> 2
Batch size=  127
num_epochs=  555
Accuracy Train =  100.0 %
Accuracy Test =  70.83 %
RMSE Train =  0.2229314556
RMSE Test =  4.9980847918
Model Bias =  1.0
Directional Accuracy =  73.91
Accuracy =  70.83
---
```

Figure 1: Model performance results for Adam optimizer



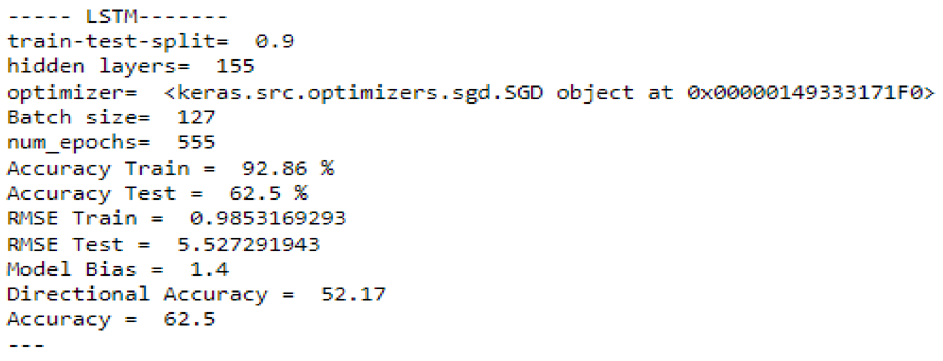
```
----- LSTM best results-------
train-test-split=  0.9
hidden layers=  165
optimizer=  <keras.src.optimizers.rmsprop.RMSprop object at 0x000002BB2FF87BB0> 2
Batch size=  127
num_epochs=  435
Accuracy Train =  94.76 %
Accuracy Test =  70.83 %
RMSE Train =  0.6304369806
RMSE Test =  4.7697314259
Model Bias =  1.0
Directional Accuracy =  73.91
Accuracy =  70.83
---
```

Figure 2: Model performance results for rsmpop optimizer



```
----- LSTM-------
train-test-split=  0.9
hidden layers=  165
optimizer=  <keras.src.optimizers.adagrad.Adagrad object at 0x000002A387FF62B0>
Batch size=  127
num_epochs=  600
Accuracy Train =  63.81 %
Accuracy Test =  54.17 %
RMSE Train =  3.8257511647
RMSE Test =  4.8439171715
Model Bias =  2.0
Directional Accuracy =  56.52
Accuracy =  54.17
---
```

Figure 3: Model performance results for Adagrad optimizer



```
----- LSTM-------
train-test-split=  0.9
hidden layers=  155
optimizer=  <keras.src.optimizers.sgd.SGD object at 0x00000149333171F0>
Batch size=  127
num_epochs=  555
Accuracy Train =  92.86 %
Accuracy Test =  62.5 %
RMSE Train =  0.9853169293
RMSE Test =  5.527291943
Model Bias =  1.4
Directional Accuracy =  52.17
Accuracy =  62.5
---
```

Figure 4: Model performance results for Sgd optimizer

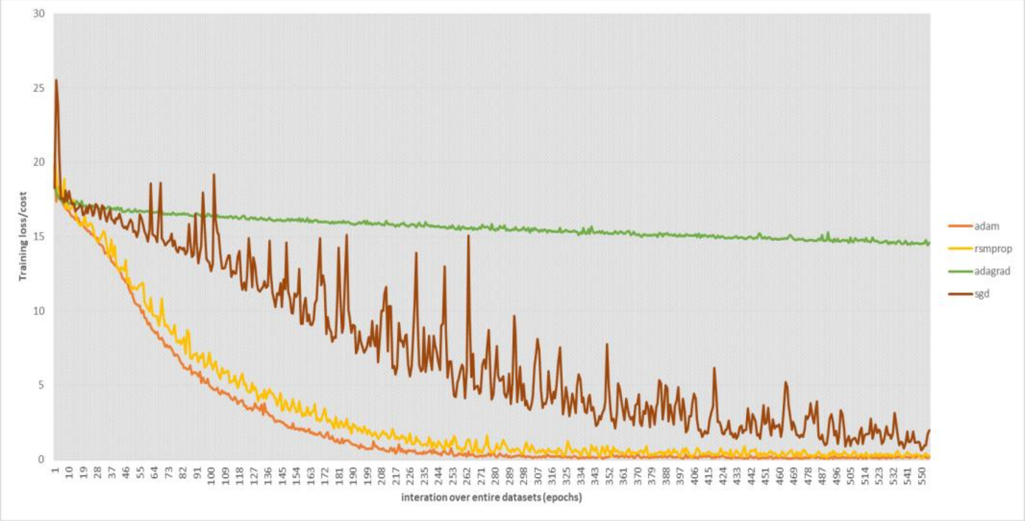


Figure 5: Training loss/cost of the optimizers.

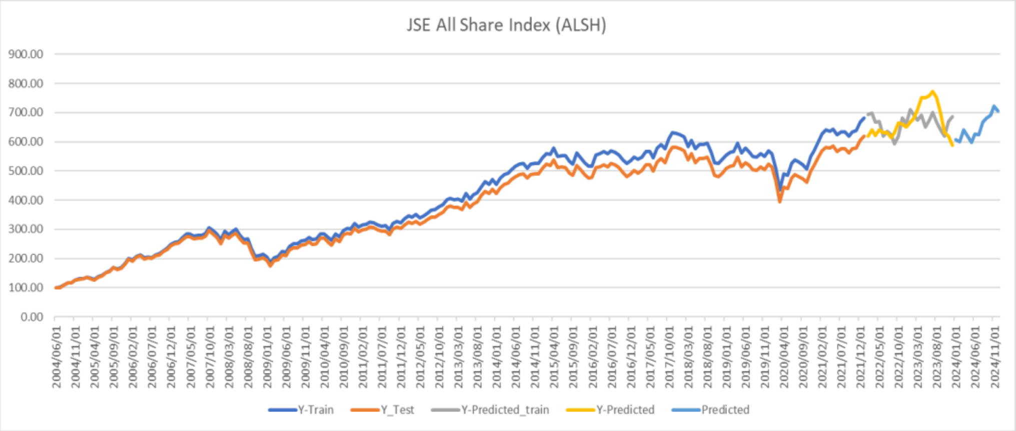


Figure 6: forecast result of the JSE All Share Index using the LSTM with adam optimizer.
Intepretation of results:

The LSTM model (figure 1) achieved its best results with a train-test-split of 90%, 164 hidden layers, Adam optimizer, batch size of 127, and 555 epochs. The model showed high accuracy on the training data (100%) and reasonable accuracy on the test data (70.83%), with low root mean square error (RMSE) on both sets. The model bias of 1.0 indicates a balanced trading system, while the directional accuracy of 73.91% suggests a slightly favorable direction in predictions. The results were similar with the RMSprop (figure 2) optimizer with a difference in hyper-parameter setting.

The Adagrad and SGD (figure 3 & 4) optimizers did not do quite well when dealing with the index returns as the dataset. In figure 5, the adam converges at a faster rate and has a better performance, much better than the adagrad and sgd by wide margin.

In summary, Adam performed equal or slightly better than RMSProp, regardless of hyper-parameter setting and better than the adagrad and sgd by a wide margin. The RMSprop and Adam are similar algorithms that do well in handling sequential datasets. As Kingman & Lei Ba (2019) showed that Adam slightly outperforms the RMSprop towards the end of theoptimization as gradients become sparser due to its bias-correction technique.

The model using adagrad and sgd might require further tuning of hyper-parameters to achieve good model performance. In figure 6, shows the forecast result of the JSE All Share Index showing the in and out sample predictions and a 12-month ahead forecast from the multilayer NN model.

6. CONCLUSION

We have compared and reviewed the characteristics, advantages, and limitations of optimization algorithms. The results showed that the Adam optimizer together with the RMSprop are well suited for handling sequential data like the time-series of index returns with minimum tuning and hyper-parameter as both optimizations minimize the training cost. In ending, Adam is likely the best overall choice when dealing with sequential datasets.

REFERENCES

[1] Dean J, Corrado G.S, Monga R., Chen K.,Le D.M, Mao M.Z., Ranzato M.A., Senior A., Tucker P., Yang K., and NgA.Y., (2012) Large Scale Distributed Deep Networks. NIPS 2012: Neural Information Processing Systems, pages1–11.

[2] Kingma D.P., and Lei Ba J., Adam, (2015) A Method for Stochastic Optimization. International Conference on Learning Representations, pages 1–13.

[3] Kaabar S., (2024) Deep Learning for finance:Creating Machine and Deep Learning Models for Trading in Python. O'Reilly Media, Inc.

[4] Ruder S., (2017) An overview of gradient descent optimization algorithms. Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin.

[5] Shaziya H., Zaheer R., (2019)A Study of the Optimization Algorithms in Deep Learning. International Conference on Inventive Systems and Control (ICISC 2019).

[6] Sun S, Cao Z., Zhu H., and Zhao J., (2019) A Survey of Optimization Methods from a Machine Learning Perspective. https://doi.org/10.48550/arXiv.1906.06821.

[7] Tsai, C., and Wang S., (2009). Stock price forecasting by hybrid machine learning techniques. Proceedings of the International Multiconference of Engineers and Computer Scientists. Vol. 1. 755, p. 60.

AUTHORS

Sanele Makamo (Quantitative Risk Analyst) is a quant risk analyst with over 7 years' experience in the South African and Global financial markets. He holds a BSc Mathematics and Statistics degree; he is currently pursuing his postgraduate studies in financial mathematics and intends to complete his master's in risk analytics by year 2025. His research interests focusis in using machine learning models for solving problems in risk analytics. He currently has more than 15 published research papers.