A CNN based model to forecast the South African Unemployment Rate

Babalwa Nhose
School of Computational and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
1837887@students.wits.ac.za

Rudzani Mulaudzi
School of Computational and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
rudzani@mulaudzi.co.za

Ritesh Ajoodha
School of Computational and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
ritesh.ajoodha@wits.ac.za

***Abstract*—In the second quarter of 2021, the number of unemployed people in South Africa increased by 54 000 to 14.9 million. A key way in which governments address such a problem is by using policy, with forecasted employment rates being a key input. However, accurately forecasted unemployment rates are difficult to obtain and many have started looking to deep learning to improve the accuracy. In South Africa, long-short term memory (LSTM) neural networks and gated recurrent units (GRUs) have been employed in the same regard. These approaches were able to produce lower error rates when compared to traditional forecasting approaches. This paper extends previous research on using deep learning models for unemployment rate forecasting in South Africa by investigating whether convolutional neural networks (CNNs) can offer comparable performance to LSTMs, which produced more accurate forecasts compared to other deep learning approaches. We find that, on average, CNNs are 27.7% less accurate than LSTMs: LSTMs achieve a MAE of 0.2193, whilst CNNs achieve 0.4962. Given that CNNs are generally used for image classification, this result shows that CNNs should not be considered as an effective way to forecast the South African unemployment rate.**

***Index Terms*—Convolutional Neural Networks, Deep Learning, LSTM, MAE, Unemployment Rate**

I. Introduction

The Covid-19 pandemic has negatively impacted the unemployment rate, not only in South Africa, but all over the world. In South Africa, there was a 2.2% increase in the unemployment rate as a result of government imposed national lockdowns. South Africa was also hit by a wave of riots in July 2021, which resulted in a significant disruption of businesses which further increased the unemployment rate. According to Statistics South Africa, 54 000 more people became unemployed between the first and second quarter of 2021 [1].

The unemployment rate is an important indicator because it is used to decide the health of the economy when setting government policies [2]. These policies are developed by policy markers who rely on accurate forecasts. Investors also use the unemployment rate to indicate the sectors that are laying off jobs rapidly, then regulate which mutual funds to exchange [3].

The unemployment rate data, in South Africa and around the world, consists of many micro-economic features which we might not understand. Deep learning architectures solve problems such as data manipulation, and do not require prior knowledge of any of the features mentioned. As a result, the application of deep learning has increased in unemployment rate forecasting all over the world.

The use of deep learning to forecast the unemployment rate is still in its nascent stages in South Africa, the available research shows that long short term memory (LSTM) neural networks followed by gated recurrent units (GRUs) are the two deep learning methods which produce the most accurate results when compared to traditional forecasting approaches [4] [5].

The results found in the available research to forecast the South African unemployment rate, using machine learning, are consistent with similar unemployment rate forecasting papers conducted around the world showing that LSTMs generally produce lower errors, proving LSTMs have syntactic recency (an ability to extract information in a meaningful sense) [6] [7]. For example, in the United States of America (USA), it was shown that when supplied with diverse and complex data, such as the unemployment rate, machine learning approaches (including the LSTM) outperform simpler time-series approaches [6].

Our paper extends the work done in using deep learning for unemployment rate forecasting in South Africa by exploring the application of Convolutional Neural Networks (CNNs). CNNs have not been applied to forecast the South African unemployment rate thus far, but they have been applied to forecast the unemployment rate in countries such as the USA; they found that CNNs (fully connected) and LSTM neural networks perform similarly [6]. In stock price (time series) prediction, one dimensional (1D) CNNs have also increasingly become popular as they provide very low error rates by extracting useful features from the time series data [8] [9].

We will demonstrate that CNN models currently do not provide reliable results when applied to the South African unemployment rate, benchmarked against the LSTM neural network: the most accurate model used for South African unemployment rate forecasting thus far.

The remainder of the paper follows the following structure: Section 2 discusses the background: it discusses the building blocks of the benchmark, the LSTM neural network, and our base CNN. Section 3 discusses the related work. It focuses on how other authors applied these models for time series forecasting. Section 4 goes over the results we achieved when applying the neural networks. Section 5 is the concluding section, which discusses the contribution of this study.

II. Background

The background section goes into detail about the building blocks of the LSTM neural network and CNN, highlighting their special characteristics and explaining why these deep learning methods outperform traditional forecasting methods.

A. Long-short Term Memory (LSTM) Neural Network

A Feed Forward Neural Network (FFNN) is an artificial neural network wherein the connections between the nodes only move in one direction. Recurrent Neural Networks (RNNs) are FFNNs with feedback loops from prior inputs to influence the current input [7].
The illustration can be seen in figure 1 below. Weights in RNNs are assigned at the start of the neural network with random values, and from there the network tries to fit the best combination of weights and biases.

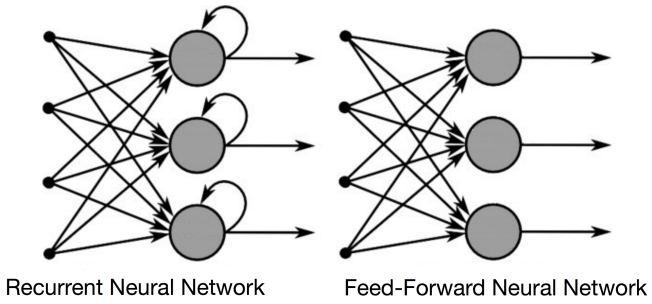

Fig. 1. Comparison of the RNN and FFNN
Adapted from: John R (2019) Data Science Methodology

1) How LSTM solves the problems associated with RNN: After the RNN outputs the prediction vector in the last layer, a prediction error is computed and backpropagation through the time algorithm is used to calculate the gradient, which is then used to update the neural network weights. The problem arises when it comes to updating the weights. The further we go through the network, the lower the gradients become, and the harder it is to train the weights.
Meaning, when we are dealing with a large number of time steps, the model signal is lost, causing a vanishing gradient problem [10]. Concluding that RNNs are better for sequential recency (handling arbitrary input lengths) than syntactic recency (extracting information in a meaningful sense).
The LSTM neural network is a widely successful variant of RNN. The LSTM neural network solves the vanishing gradient problem by adding gated functions. Gated functions give the network more control over the gradients, because they can take on different weights for each layer at each time step.
The learned functions of the current input and hidden state influence the continuously adjusted weights [10]. Hence, they are capable of learning long term dependencies. An explanatory illustration of the LSTM cell is provided in figure 2.

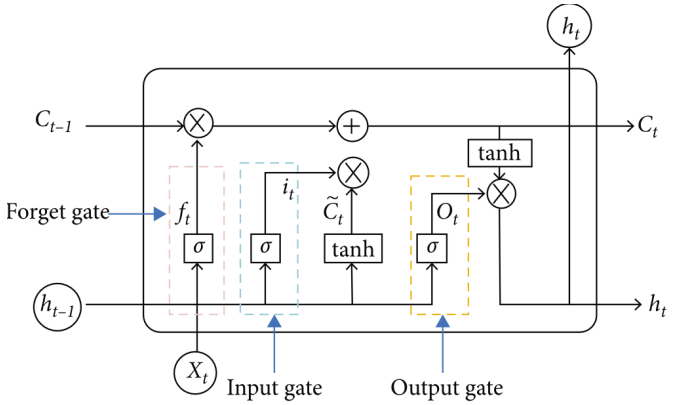

Fig. 2. State of the LSTM, showing the cell state and the forget, input and output gates.
Adapted from: Wenjie L, Jiazheng L, Yifan L, Aijun S, Jingyang W (2020) A CNN-LSTM-Based Model to Forecast Stock Prices.

2) Cell State: The main concept behind the LSTM is the cell state. In figure 2, labeled $C_{t\ 1}$ and C_t, for previous and current timestamps respectively, it is the horizontal line that passes right through the cell. It is normally visualised as a conveyor belt, since it takes information from point A to point B, adding or removing information in the process. The gates below it regulate what information will be added or removed.

3) Forget Gate: Answers: "What can we safely forget from the previous timestamp?"
Labeled f_t in figure 2, it examines the previous hidden state $h_{t\ 1}$ as well as the current input x_t. The weights W_f are multiplied over $h_{t\ 1}$ and x_t and an output is produced. As shown in equation 1, a sigmoid function is applied to the output giving f_t a number of 1 or 0.
If f_t is 0, the network will forget everything, whereas if f_t is 1, the network will forget not forget anything.

$$f_t = \sigma\left(W_f \quad [h_{t\ 1}, x_t] + b_f\right) \tag{1}$$

4) Input Gate : Answers: "What new information are we going to store in the cell state?"
Labeled i_t in figure 2. There are two important parts composing the input gate. The part first is the sigmoid layer, called the input gate layer. It is used to determine the significance of the new data carried by the input x_t. Then we have $tanh$ layer which creates a vector of all new candidate values.
The new data that must be sent to the cell state is now a function of the previous hidden state $h_{t\ 1}$ and input x_t, because of the tanh function, it will produce a value between -1 and 1.

$$i_t = \sigma\left(W_i \quad [h_t \quad _1, x_t] + b_i\right)$$
$$\bar{C}_t = \tanh\left(W_c \quad [h_t \quad _1, x_t] + b_c\right) \tag{2}$$

Where W_i is the weight of the input gate, and W_c is the weight of the candidate input gate.
$\bar{C}_t$, as seen in figure 2, will not be directly added to the cell state. The information is removed from the cell state if the value of $\bar{C}_t$ is negative, and added to the cell state if the value is positive, at the current timestamp. Here's the revised equation of the cell state.

$$C_t = f_t \quad C_t \quad _1 + i_t \quad \widetilde{C_t} \text{ (updating the cell state)} \tag{3}$$

5) Output Gate : Answers: "What are we going to output?" Labeled O_t in figure 2, it produces an output which will depended on the cell state, we only want to output the filtered version of the cell state. First, the sigmoid layer is run which decides which part of the previous information we will output. The output o_t of the output gate is obtained as follows:

$$o_t = \sigma\left(W_0\left[h_t \quad _1, x_t\right] + b_o\right) \tag{4}$$

Then it puts the updated cell state C_t through a $tanh$ function, so it can output values between $(\quad 1, 1)$ then the two values (from the sigmoid layer and the updated cell state) are multiplied.

B. Convolutional Neural Network (CNN)

CNNs are FFNNs that are used mostly for image classification. Recently, they are increasingly being applied for time series prediction. 1D convolutions are applied to extract information throughout the time dimension.
These networks employ a mathematical procedure called convolution followed by a down sampling layer to reduce the feature dimension. The convolution layer is only connected to a few computational nodes (unlike the fully-connected setting) [7].
Figure 3 demonstrates how 1D Convolutional model works and a detailed explanation follows.

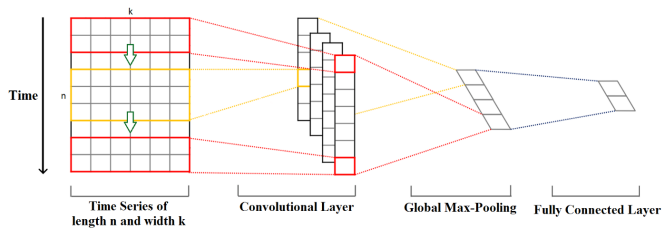

Fig. 3. CNN model adapted for time series prediction
Adapted from: Margarita G (2019) How to Use Convolutional Neural Networks for Time Series Classification

1) CNN 1D model (for time series): For demonstration purposes, assume we have a time series with time steps n and k features. A set of features are extracted in each convolution layer, which contains a multitude of convolution kernels that exam over each time step. The convolutional kernel actions down from the top of a time series towards its bottom.
Equation 5 below shows the corresponding elements of the time series at a specific point when the elements of the kernel get multiplied.

$$\begin{cases} 0 & \text{for } x_t \quad w_t + b_t < 0 \\ x_t \quad w_t + b_t & \text{for } x_t \quad w_t + b_t \quad 0 \end{cases} \tag{5}$$

Where ReLu is the activation function, x_t is the input vector, w_t is the convolution kernel weights, and b_t is the convolution kernel bias.

The results of the multiplication are summed collectively and the value becomes an element of a new "filtered" univariate time series. The kernel then moves on along the time series to produce the next value.
As seen, the number of convolution kernels is equal to the number of new "filtered" time series where different features of the initial time series are captured in each of the new "filtered" series [11]. More convolutional layers may be added if need be, there is no limit to how many layers one may add, as long as there is computational power available.
Once the feature maps are available, a down sampling step is performed, usually max pulling, as denoted in Fig 3. Max pulling is applied to each of the filtered time series vectors, where the largest value is taken from each vector.
A final feature vector is fashioned from these values, essentially, each feature is down sampled to a lower feature sets. The vector of maximums (from global max-pooling) is used as an input to the final FFNN layer that helps congruent all of the features so an output (the predicted time series variables) can be given.

III. Related Work

This section demonstrates how deep learning models have been applied to forecast time series data in South Africa and around the world, focusing on LSTMs and CNNs.

A. Application in South Africa

Nine distinct machine learning models, including four deep learning models, were run in a study that used deep learning to predict the South African unemployment rate (with a multivariate approach). The results showed that the LSTM (with four layers, dropout and early stopping) outperformed all the other machine learning models. The LSTM had a mean average square error (MASE) of 0.914.
It was closely followed by the GRU (a computationally simpler version of the LSTM) with a MASE of 0.915 and the FFNN with a MASE of 1.582 [4]. CNNs were not applied in this investigation. Other papers set out to predict the unemployment rate in South Africa did not apply deep learning models and

mainly focused on traditional statistical and univariate models [5].

B. Application Around the World

Generally, the data used in the investigations in other countries was attained through the country's national research websites, such as FRED (Federal Reserve Economic Data) and the Bureau of Economic Research (BER). In the USA, CNNs were used to predict the unemployment rate and it was found that they slightly lag behind (but still remain competitive with) the other deep learning models applied.

At a 1/4 prediction time frame, fully connected convolutional neural network prediction averages continued to outperform the benchmark used, Survey of Professional Forecasters (SPF) [7]. The Mean Average Error (MAE) was used as a performance measure in many of the papers found for unemployment rate prediction [2] [7].

In China, a CNN-LSTM hybrid model was successfully used to predict stock prices. The CNN was tweaked to extract the features from the input data, and the LSTM was used to learn the feature data and forecast the stock's closing price for the next day.

In comparison to the FFNN, CNN, RNN, LSTM, and CNN-RNN approaches that were also used, the hybrid technique had the highest predicting accuracy and best performance. The following performance indicators were used in the study: the MAE, Root Mean Square Error (RMSE), and R^2 [8].

The majority of the papers found around the world for unemployment rate forecasting using deep learning used FFNN with a range of different layers and time steps. It was elusive to find reports that investigated the LSTM neural network, more so the CNN [2][12][13]. For stock price prediction, applying deep learning models, the LSTM neural network and the CNN were deemed popular and they had the lowest errors. [8][9].

IV. Methodology

In this section, we take an investigative approach in evaluating the CNN, guided by current literature for unemployment rate forecasting in South Africa and around the world. We benchmarked the CNN with the LSTM neural network and run the GRU for comparative reasons.

A. Data Preparation

The unemployment rate data is retrieved from the national Reserve Bank of South Africa (SARB). The SARB data set consists of 147 micro-economic variables [14]. The data set contains all the important sectors and industries of the South African economy. However, due the records having different frequencies, including monthly and quarterly, the combined data set (which combines features and the target) contains a large amount of missing data.

We first removed rows that had a missing target variable (the unemployment rate), then we used sampling techniques to up sample all the monthly data to be quarterly so that all the data is in the same frequency. From 794 observations, we were left with 201, now dating from January 1970 to December 2019. Since the focus of the paper is applying deep learning models, no feature engineering was implemented.

Previous research applying deep learning for unemployment rate forecasting in South Africa showed that models which fit data with high correlated features removed had a lower error rate [4]. For comparative reasons, a new data set is derived from the original set where features with a correlation higher than 0.95 are removed; with this parameter in place, the number of micro-economic variables decreases to 78 (from 147), this implies that each neural network is run twice on different data sets (no feature selection, removed correlation). The data is then split into the training and testing sets, with the first 70% of the data used for training and the rest of the 20% used for testing. On the testing data set, a validation split of 20% was later applied.

The reason for the large spilt for the training set is to ensure that the neural networks have enough data to train and validate on. The models will use the same training and testing data set to ensure that the neural networks are exposed to the same data sets for comparison.

In order to compensate for the high variance in the macroeconomic variables and to improve the learning speed, the train and the test data are individually normalised ranging from 0.001 to 1. They are individually normalised to ensure that the test data is seen as new data when it is applied to the trained model, preventing data leakage. A similar data normalisation technique was applied to the unemployment rate data in Japan [13]. The equation used to normalise the data is shown below in equation 6.

$$\text{normalized data}=\left[\frac{\text{data value} \quad \text{min value}}{\text{max value - min value}}\right] \quad (1 \quad 0.001) + 0.0001 \tag{6}$$

B. Experimentation

During the experimentation, different (yearly) time steps are applied to the data. Remembering that the unemployment rate is forecasted quarterly, meaning that one year equates to four rows of input. The range of time step intervals ranges from 1 year to 20 years [12].

The GRU and LSTM model consists of 4 layers and the CNN consists of 5 layers, not including a dropout layer of 0.2. A drop out layer is added to regularise the algorithms in order to avoid over/under fitting. For the GRU and LSTM, we define two layers with 74 and 42 units respectively; they are followed by a dropout layer of 0.2 and two dense layers with feature maps of 16 and 1 to interpret the input feature. A output layer then is specified that predicts the unemployment rate.

On average, the total number of usable parameters for the LSTM neural network is: 78 609. For the GRU, on average, the total number of usable parameters applied is 65 301.

For the CNN model, we define three convolutional layers with 246, 128 and 64 filters respectively and a kernel size of 1. A kernel size of 1 is used to slide all over the data set in the hope

of extracting as many features from the data as possible. Each convolutional layer is followed by a max pooling layer (with the same number of filters as their respective convolutional layer) as seen in section II figure 3.
Later, a dropout layer of 0.2 and two dense layers follow with feature maps of 16 and 1 respectively to interpret the input feature. A output layer is also specified that predicts the unemployment rate (the target). On average, the total number of usable parameters for the CNN is: 61 6633.
Early stopping, with a patience of 20, and call back is applied to all the neural networks implemented, early stopping is a form of regularization used to avoid overfitting when training (on average, the number of epochs in each run ranged between 24 and 129). A patience is the number of epochs to run before early stopping, if there is no progress on the validation set, a patience of 20 seemed to work well, taking note of the average number of epochs in each run.

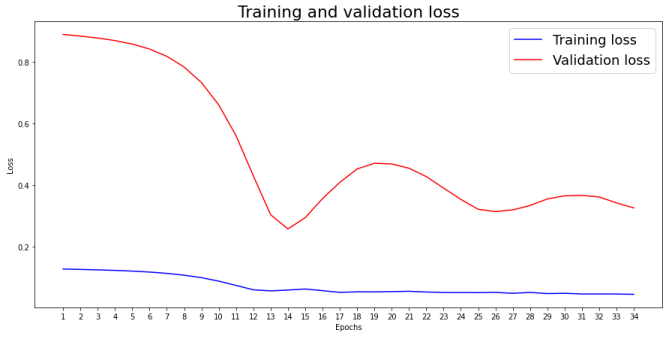

Fig. 4. An example of CNN Training and validation loss plot with early stopping.

All the models are fit using the efficient Adam version of stochastic gradient descent (by default it uses a learning rate of 0.001) and optimized using the mean squared error (MSE) loss function and MAE metrics.
The selection of the number of units and layers in each neural network is based on input data, we want the networks to be proportional to the input data (not to make them too complicated) so no over fitting takes place. We also look at the total number of parameters each model uses, since this is a comparative investigation, we want to ensure that each model is exposed to the same number of parameters.

The following hyperparameters were used for both models.

Batch Size: 128
Epochs: 1000 (with early stopping)
Validation Split: 0.2
Optimiser: Adam version

It has been observed that when employing a larger batch, the model's quality, as evaluated by its ability to generalize, suffers significantly. Hence we limit the batch size to 128.
On average, for each time step, the each model is run four times and the smallest error is noted, meaning on average each model was run 48 times. The results obtained can be seen in tables I and II.

C. Performance Measure

The performance measure used in this investigation is the Mean Average Error (MAE). The MAE is a traditional measure used for time series forecasting. In the USA, it was used to calculate the error for machine learning-based unemployment rate forecasting in a number of investigations [2] [7] [12].
It simply calculates the absolute average difference between the actual and the predicted value. The equation, equation 7, is provided below.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} jy_i \quad \hat{y}_i j \tag{7}$$

Where n is the number of observations, y_i is the actual value (target) and $\hat{y}_i$ is the predicted value.

D. Results

When we apply the data to the neural networks, the results show that on average, the GRU and the LSTM MAEs are very close together. On the other hand, there is a clear distinction in the difference between the average MAE of the LSTM and the CNN. The results found from the GRU and the LSTM error rates are consistent with the results found in existing research [4]. In both data sets (no feature selection, removed high correlation) the LSTM and GRU provided a lower average MAE.
In both cases, the MAE is lower when the neural networks are applied to the data where no feature selection is performed. Emphasising that neural networks may provide accurate results without data manipulation. The results opposes what was mentioned in another paper that showed that neural networks, applied to the South African unemployment rate, perform better with high-correlated features removed [4].
Also, the error rate is lower for smaller time steps in both cases, implying that the neural networks (GRU, LSTM, CNN) are optimised when they are given smaller time steps. We find the same result in majority of the investigations conducted in other countries where the employment rate is forecasted using machine learning [2] [7] [12].
We expected better results for the 1D CNN, since it performed well in the papers that applied it to stock price (time-series) prediction [8] [9]. The reason for the difference in accuracy might be that stock price prediction data is noted in much smaller time intervals (it might be every 15 minutes, hourly, daily ect) explaining the difference in accuracy rates.
Given that CNNs are generally used for image classification, other models that performed similar to the LSTM in previous research done in the South African context, including the GRU and FFNN, should be considered first when exploring deep learning methods to forecast the South African unemployment rate. Even if the error of the CNN is acceptable, it performs, on average, 27.7% worse than the LSTM and might not be worth exploring further.

TABLE I
MAE OF GRU, LSTM AND CNN MODELS (REMOVED CORRELATION)

Time Steps (in years)	GRU	*LSTM*	*CNN*
1	0.1662	*0.1656*	*0.2350*
2	0.1396	*0.2414*	*0.3580*
5	0.1313	*0.2224*	*0.5948*
10	0.3890	*0.1903*	*0.7653*
15	0.2673	*0.2515*	*0.6232*
20	0.5057	*0.2803*	*0.5374*
Average	0.2665	*0.2251*	*0.5190*

Performance of the GRU, LSTM and CNN models, using the South African unemployment test data with features with correlation below 0.95

TABLE II
MAE OF GRU, LSTM AND CNN MODELS (NO FEATURE SELECTION)

Time Steps (in years)	GRU	*LSTM*	*CNN*
1	0.1355	*0.1516*	*0.2118*
2	0.1283	*0.1996*	*0.3090*
5	0.1313	*0.2386*	*0.5038*
10	0.3114	*0.1965*	*0.6783*
15	0.2389	*0.2512*	*0.6364*
20	0.2845	*0.2780*	*0.6377*
Average	0.2050	*0.2193*	*0.4962*

Performance of the GRU, LSTM and CNN models, using the South African unemployment test data with no feature selection.

V. CONCLUSION

This paper seeks to determine whether the convolutional neural network (CNN) can outperform the long-short term memory (LSTM) neural network; when they are both applied to the South African unemployment rate data. From the results, CNNs do not provide accurate prediction results when applied to the South African unemployment rate since the average MAE for the CNN model is much higher than the average MAE of the LSTM model (it is approximately 27.7% higher). In comparison, the GRU has an average MAE that is 1.43% higher than the LSTM average MAE.

In both the neural networks (LSTM, CNN) the MAE is lower when they are applied to the data where no feature selection is conducted. Emphasising that neural networks may provide accurate and reliable results without data manipulation, meaning no prior knowledge of features is required, especially when dealing with a large feature set such as the unemployment rate.

Also, it is noted that the MAE is smaller for smaller data time steps, implying that the neural networks (LSTM, CNN) are optimised when they are given smaller time steps of the unemployment rate data.

The difference in the results from the implementation of the neural networks in South Africa and other more developed countries might be caused by the amount of available unemployment rate data within the countries; with more data, the neural networks in the South African context might have performed better than what is noted.

In future work, the data set used in this experiment does not include data collected during the COVID 19 pandemic (the used data set ends in December 2019). Since the data is now available, one may investigate the difference in the errors when applying the deep learning models to the South African unemployment rate data that includes data from 2020.

REFERENCES

[1] SSA. "Quarterly Labour Force Survey (QLFS) – Q2:2021". In: *Computational Economics* (June 2021).

[2] Aiken Milam. "A neural network to predict civilian unemployment rates". In: *Journal of International Information Management* 5.1 (1996), p. 3.

[3] Rita Diana, I Nyoman Budiantara, Satwiko Darmesto, et al. "Statistical modeling for unemployment rate using smoothing spline in semiparametric multivariable regression model with Bayesian approach". In: *Model Assisted Statistics and Applications* 9.2 (2014), pp. 159–166.

[4] R. Mulaudzi and R. Ajoodha. "Application of Deep Learning to Forecast the South African Unemployment Rate: A Multivariate Approach". In: *Seventh IEEE Computer Science and Data Engineering Conference* (2020).

[5] Rudzani Mulaudzi and Ritesh Ajoodha. "An Exploration of Machine Learning Models to Forecast the Unemployment Rate of South Africa: A Univariate Approach". In: *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)* (2020).

[6] Aaron Smalter Hall and Thomas R Cook. "Macroeconomic indicator forecasting with deep neural networks". In: *Federal Reserve Bank of Kansas City Working Paper* 17-11 (2017).

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] Wenjie Lu et al. "A CNN-LSTM-based model to forecast stock prices". In: *Complexity* (2020).

[9] Xuebo Jin et al. "Prediction for Time Series with CNN and LSTM". In: *Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019)*. Springer. 2020, pp. 631–641.

[10] Noah Weber. *LSTM-Vanishing-Gradients*. 2017. URL: https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html.

[11] G Margarita. *How to Use Convolutional NeuralNetworks for Time Series Classification*. 2019. URL: https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classificationnn-56b1b0a07a57.

[12] A.S Hall. “Machine Learning Approaches to Macroeconomic Forecasting”. In: *The Federal Reserve Bank of Kansas City Economic Review* (2018).

[13] Kanlapat Mahipan, Nipaporn Chutiman, and Bungon Kumphon. “A Forecasting Model for Thailand’s Unemployment Rate”. In: *Modern Applied Science* 7 (2013), p. 10.

[14] South African Reserve Bank. *Economic and financial data for South Africa*. URL: https://wwwrs.resbank.co.za/webindicators/EconFinDataForSA.aspx..