UNIVERSITY OF CAPE TOWN

HONOURS PROJECT

Artificial Neural Networks in Precipitation-Stream Flow Modeling

Authors:
Matthew DE VRIES
Romelon CHETTY

Supervisor:
Dr. Birgit ERNI

A project submitted in fulfillment of the requirements for the degree of Honours in Statistics

in the

Department of Statistics

September 27, 2022

Declaration of Authorship

I, Matthew DE VRIES
Romelon CHETTY, declare that this thesis titled, "Artificial Neural Networks in Precipitation-Stream Flow Modeling" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITY OF CAPE TOWN

Abstract

Faculty of Science
Department of Statistics

Honours in Statistics

Artificial Neural Networks in Precipitation-Stream Flow Modeling

by Matthew DE VRIES
Romelon CHETTY

We investigate the utility of Artificial Neural Networks (ANNs) for short term forecasting of stream flow in the Jonkershoek catchment area. We look at the use of Recurrent Neural Networks, in particular, the Long Short-Term Memory (LSTM) Network in order to forecast stream flow from a number of input variables related to precipitation. The LSTM is optimised to predict the stream flow as good as possible, and has to learn physical principles and laws during the calibration process purely from the data. The relationship between immediate rainfall and stream flow is expected to be shown from this model as well as the long term storage of the catchment area. We ultimately look at describing the effects of a long term drought and the time after which the stream flow is expected to recover. The work considers historical data of precipitation measured in millimetres and stream flow measured in cubic metres from the Jonkershoek catchment area from the year 1940 until 2018.

Keywords : Artificial Neural Networks, Long Short-Term Memory, stream flow, rainfall, drought, Jonkershoek

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	**A**rtificial **N**eural **N**etworks
RNN	**R**ecurrent **N**eural **N**etworks
LSTM	**L**ong **S**hort **T**erm **M**odel

Chapter 1

Introduction

The drought in the Western Cape began in 2015 and has resulted in a severe water shortage in the region, most notably affecting the city of Cape Town. With dam levels predicted to decline to critically low levels, the city announced plans for "Day Zero", when if a particular lower limit of water storage was reached, the municipal water supply would largely be shut off, potentially making Cape Town the first major city to run out of water (Cassim, 2018). Implementation of water conservation efforts have taken place across the province in order to avoid "Day Zero". Understanding how stream flow is likely to be affected by projected changes in precipitation seasonality is crucial in determining the effect of a large drought on dam levels and how long an area may take to recover. Water is of vital importance to food security as it directly affects agricultural production. Its scarcity can cause famine in the region, having massive consequences on health, hunger, education, and poverty of a country and when in excess, produces life-threatening floods. Water is important in attaining, not only the hopes for food harvests, but also a sustainable livelihood, in which agriculture plays a significant role. Proper management of rainwater can lead to plentiful harvests, but its misuse can be responsible for many epidemics. The first step in its management is to quantify runoff produced in the area due to rainfall (Sinha, 2011). Thus, estimation or prediction of stream flow is essential for flood and drought protection works, water harvesting, power generation and risk control.

Precipitation-stream flow modeling has a long history within the field of hydrology with the first attempt to predict discharge from rainfall dating back 170 years (Mulvaney, 1850). Since then, further development of modeling concepts has taken place by progressively incorporating physically based process understanding and concepts into the model formulations. These developments are largely driven by the advancements on computer technology and the availability of data in high spatial and temporal resolution (Kratzert et al., 2018). However, the development

of physically based and spatially explicit representations of the hydrological process at the catchment scale has come at the price of high computational costs and a need for specific, and often difficult to obtain, meteorological input data. For this reason, physically based models are rarely used in operational precipitation-stream flow forecasting. The high computational costs further limit the application of these models, especially if uncertainty estimations exists and multiple model runs within an ensemble forecasting framework are required. Thus, simpler versions of these physically based or conceptual models are often applied for operational purposes. Furthermore, fully data-driven approaches such as regression, fuzzy based or artificial neural networks (ANNs) have been explored in this context.

ANNs are known for their use in nonlinear and complex systems. Due to the fact that the transformation from rainfall to runoff is believed to be highly nonlinear, spatially distributed, and time variant, ANN rainfall-runoff prediction models have been used since the early 1990's (Daniell, 1991; A. H. Halff and Azmoodeh, 1993). Since then, several studies on their applicability to the hydrological process have been published with new ideas constantly arising. There has been concern about the application of the feed forward neural network (FFNN), which have previously been the focused architecture, for time series analysis as any information about the sequential order of the inputs is lost. Recurrent neural networks (RNNs) are a different kind of neural network architecture, which have been designed to understand the sequential nature of the input data. Even though, in some cases, FFNNs have been shown to perform equally well as RNNs (Carriere, Mohaghegh, and Gaskar, 1996), it has been found that the number of delayed inputs, which are provided as driving inputs to the ANN, are a critical hyperparameter (Hsu, Gupta, and Sorooshian, 1995). However, due to the architecture of the RNN, these models make this step in creating and searching for the correct number of lags, obsolete. Furthermore, RNNs have been seen to outperform FFNNs in stream flow prediction in most cases. One downfall of the simple RNN is its apparent inability to learn long-term dependencies. A special type of RNN, proposed by Hochreiter and Schmidhuber (1997), known as the "Long Short Term Memory" (LSTM) model, is currently recognized as the "state of the art" (Donger, 2018) architecture for problems in which the sequential nature of the data matters. This model has been specifically designed to overcome the long-term dependency problem associated with the simple RNN, which may play an important role in catchments with sufficient storage effects eg.

snow driven catchments.

This project is part of a larger project on seasonality which was collaborated with the South African Environmental Observation Network (SAEON). The larger project aims at developing a model linking rainfall to stream flow as well as determining the relative influence of precipitation versus groundwater release and evapotranspiration on stream flow. Such a model could be used for predicting how stream flow is likely to be affected by projected changes in precipitation seasonality. The aim of this project is to investigate the utility of ANNs in predicting stream flow from precipitation. Additionally, we want to compare the different ANN architectures, namely the FFNN, RNN, and the LSTM based on their predictive capabilities. We test our models on the data obtained from SAEON on the Jonkershoek catchment area, including only rainfall and stream flow as meteorological variables.

This paper is structured in the following way: in Chapter 2, we discuss the different types of models used in the prediction of stream flow as well as the important contributions made by researchers for stream flow prediction involving regression and neural network techniques. This is followed by some theoretical considerations of the project which describes the theory of ANNs and the different topologies used, in Chapter 3. We will also give a high level translation of the models in the hydrological sense. Chapter 4 describes the data used and any preprocessing techniques undertaken before model formulation could take place. In Chapter 5, we report and discuss the results of the different models, finally concluding and exploring scenarios in Chapter 6.

Chapter 2

Review of Literature

This chapter deals with the important and relevant contributions made by researchers for stream flow prediction involving both linear regression and artificial neural networks. We will briefly introduce the scope of ANNs, then describe some common rainfall-runoff models used in literature and their characteristics, touching on regression models and onto those ANN models. There is no shortage of studies done on machine learning techniques and their applications are so wide spread. The review of literature has been grouped under five sub-sections including The Scope of ANN Modeling, Common Precipitation-Stream Flow Models, Regression Models on Hydrological Responses and ANN-based Stream Flow Estimation Models.

2.1 Scope of ANN in Hydrology

Forecasting stream flow is of high importance in operational hydrology for reservoir operations and risk control. Stream flow is critical to many activities including the protection of agricultural land, water storage and release, and the designing of flood and drought protection works (Sinha, 2011). The volume of stream flow is dependent on several factors, with precipitation being one of great influence. The other factors include soil moisture, temperature, surrounding flora, and topography (Grist and Nicholson, 2001).

Early hydrologists calculated surface runoff with limited data and simple computational techniques. The first widely used runoff method was the Rational Method published by Thomas Mulvaney in 1851, which used rainfall intensity, drainage area, and a runoff coefficient to determine the peak discharge in a drainage basin (Sitterson et al., 2017). In more recent studies, the unit hydrograph was used to conceptualize catchment response to heavy rainfall based on a superposition principle.

The unit hydrograph is a direct hydrograph resulting from one unit of constant intense uniform rainfall occurring over the entire watershed. Now, with more data available for the verification of these models, they have been found to produce results with large errors (Kothyari, 1995).

The first computational model for neural networks based on mathematics and algorithms called threshold logic was created by McCulloch and Pitts (1943). This model then allowed neural network research to split into two approaches. One focused on the biological process of the brain, while the other on its application to artificial intelligence. A renewed interest in neural networks was then triggered due to Werbos's (1975) backpropagation algorithms which proved to accelerate the training of multi-layer networks.

Since the early nineties, artificial neural networks (ANNs) have been successfully used in hydrology-related areas such as rainfall-runoff modeling, stream flow forecasting, ground-water modeling, water quality, water management policy, precipitation forecasting, hydrologic time series, and reservoir operations (Govindaraju, 2000).

ASCE (2000) described the role of ANNs in hydrology as well as some guidelines of their uses. ANNs were compared with other modelling philosophies in hydrology, presenting the pros and cons. It was found that ANNs are tools for modelling many of the nonlinear hydrologic processes such as rainfall-runoff, stream flow, ground water management, water quality simulation and precipitation. A good physical understanding of the hydrologic process being modelled can help in selecting the input vectors and designing a more efficient neural network.

ANNs are, though, very data intensive and there appears to be little established methodology on their design and successful implementation. Most ANN studies use training and testing and a process of trial and error before obtaining a good model. Rainfall-runoff studies using ANNs provide sound evidence that these models are capable of learning and extracting the behaviour of the system, when sufficient data is available. The performance of ANN hydrological models have been frequently compared to other empirical, physical and statistical models. ANNs have been reported to be superior in performance.

2.2 Common Stream Flow Prediction Models

A runoff model is a mathematical model which intends to describe the rainfall-runoff relationship of a rainfall catchment area. Measuring rainfall and runoff for short time periods is relatively simple and inexpensive. The main question arises when determining the amount of information which could be gained about a system by just studying rainfall and runoff data from a series of rainfall events. If a time series of a sufficient amount of time exists, one can calculate yearly runoff coefficients describing the general reaction of the catchment to rainfall. It is also possible to get a first impression of the rainfall runoff process by determining the different parameters describing the hydrograph and its relationship to input rainfall (eg. peak flow rates, lag times, response times) (Beven, 2012). The calculation of runoff of single events adds information on catchment response. The changes from event to event or from season to season are then of special interest which will serve to give information of hydrological functioning of a catchment area under different events or seasons (Blume, Zehe, and Bronstert, 2010).

Hydrology can be defined as the scientific studies of waters on earth, especially related to the effect of precipitation to the occurrence and characteristics of water in streams or the stream flow (Marshall, 2013). Most hydrological systems are believed to be highly nonlinear. Even if they are assumed to be linear, this assumption is restricted to within some specified range of conditions only (Zbigniew and Napoirkowski, 2009). These models are further complicated by an uncertainty in parameter estimates, as well as a high degree of spatial and temporal variability.

Models which are currently being used by hydrologists can be grouped into three separate categories: conceptual-based models, physicality-based models, and empirical models. The empirical models or the data-driven models use nonlinear statistical relationships between inputs and outputs. For simple rainfall-runoff regression models, inputs are rainfall as well as the historical runoff, with outputs of runoff at a specific location. This simple model can be shown in the function:

$$Q = f(\boldsymbol{X}, \boldsymbol{Y}) \tag{2.1}$$

Where Q is the runoff, $\boldsymbol{X}, \boldsymbol{Y}$ are the input datasets of rainfall and historical runoff.

Empirical models treat hydrological systems as a "black-box", meaning that not much is known about the internal processes controlling the determination of runoff

results. As in machine learning, the function which transforms rainfall to runoff is usually unknown (Sitterson et al., 2017). Empirical runoff models are best used when other outputs are not needed; for example: the distribution of runoff values between upstream and downstream areas cannot be calculated with this model type (Sitterson et al., 2017).

In these data-driven models, very few parameters are needed, making these models easy and efficient to use. They lack parameters of any physical significance due to the fact that there are no real watershed components within the model. One downfall of the empirical process is that it may lead to different conclusions than accepted theoretical analysis would suggest (Beven, 2012). However, this could simply mean that there are multiple ways of finding an answer and not simply that the model is incorrect. Empirical models have benefits of simplicity of implementation, faster computational times, and cost effectiveness. Machine learning techniques fall under the empirical models category which use data-driven approaches such as artificial neural networks which train themselves to learn behaviours of the rainfall-runoff relationship. Machine learning used in ANNs calculates output predictions based on statistics learned from historical data in the training period. Machine learning algorithms can be over trained on specific inputs which cause the model to lose its ability to discern one catchment from another or to perform well on unseen data (Dawson and Wilby, 2001).

Conceptual models look at interpreting rainfall-runoff processes by connecting simplified components in the overall hydrological processes. These models are based on catchment storages and simplified equations of the physical hydrological process, which provide a conceptual idea of the behaviors in a catchment area (Devia, Pa, and Sa, 2015). They represent the water balance equation with the conversion from precipitation to runoff, groundwater and evaportranspiration, as shown in Figure 2.1 below. Every component in this water balance equation is estimated by precipitation input data through mathematical equations. The general governing equations for conceptual models are versions of the water balance equation as shown below:

$$\frac{\partial S}{\partial t} = P + ET - Q_s \pm GW, \tag{2.2}$$

where $\frac{\partial S}{\partial t}$ is the change in reservoir storage, P is precipitation, ET is evaportranspiration, Q_s is surface runoff, and GW is groundwater. Figure 2.1 represents an

example of a conceptual hydrological model viz. the Hydrological Simulation Program—FORTRAN (HSPF) model.

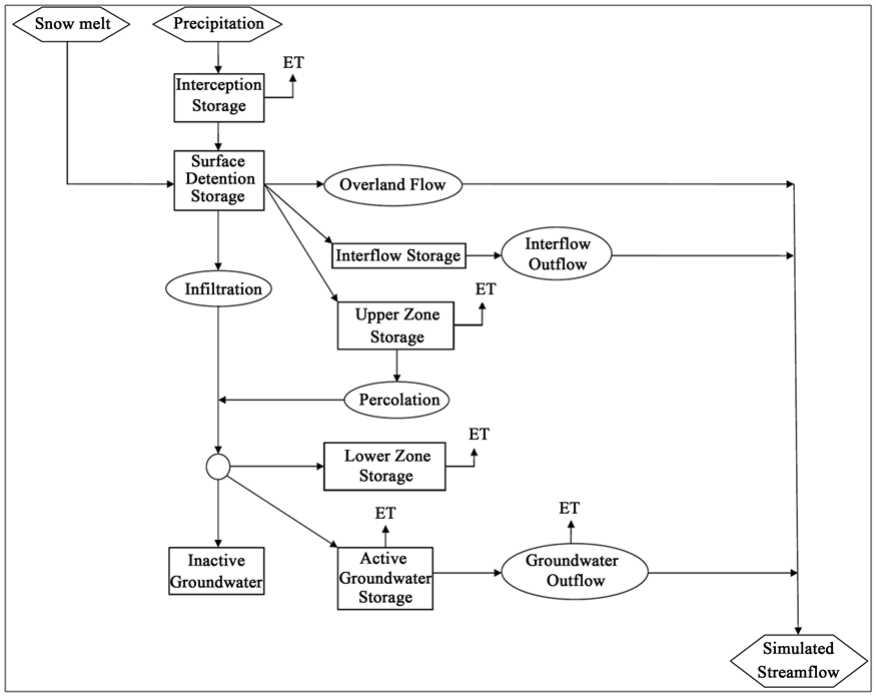

FIGURE 2.1: HSPF conceptual hydrological model [Source: Johnson et al. (2003)]

Conceptual models tend to vary in complexity depending on how sophisticated the balancing equations used to represent the hydrological are. Conceptual models have become more popular in the hydrological modeling community due to the fact that they are easy to use and calibrate. With some, there is a likelihood that a previously calibrated model can be used for different catchment areas (Vaze et al., 2011).

Physical models, also known as mechanistic or process-based models, are based on the general understanding of physics related to the hydrological process. These general physical laws used include the conservation of mass and energy, momentum, kinematics, and water balance equations. An example of a physical based model is shown below in Figure 2.2, which shows the physical movement of water through each Triangular Irregular Network via overland flow, evapotranspiration, infiltration, recharge, and groundwater.

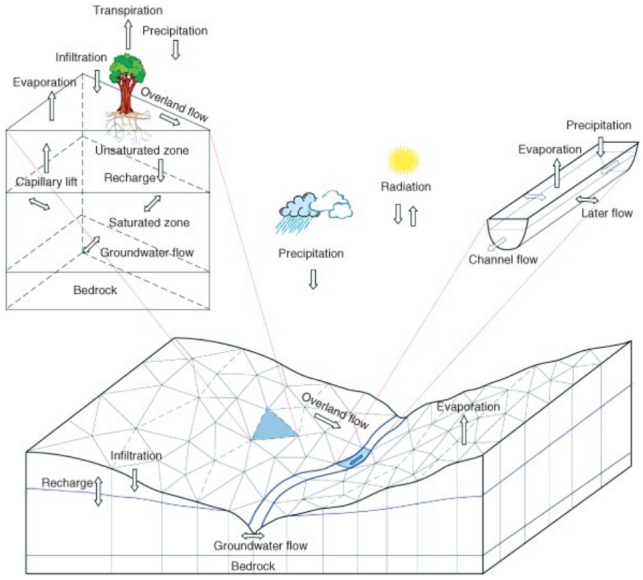

FIGURE 2.2: The model structure of the physically-based Penn State Integrated Hydrologic Modeling (PIHM) system. [Source: Qu (2004)]

What separates a physical model from other models is that it has a logical structure which is similar to the real world system of a hydrological process. This is the greatest strength of a physical model as it connects model parameters and physical catchment characteristics which make it more realistic. This leads to a large number of physical characteristics needed to calibrate the model. A downfall to physical models is that they are site, or catchment specific and may not be used on different catchments as their physicality may be drastically different. Most physical models give a three dimensional view of the hydrological process. The different characteristics of the three models are summarised in Table 2.1.

Empirical Models	**Conceptual Models**	**Physical Models**
Data based/metric or black box model	Parametric or grey box model	Mechanistic or white box model
Involve mathematical equations, derive value from available time series	Based on modeling of reservoirs and Include semi empirical equations with a physical basis.	Based on spatial distribution, Evaluation of parameters describing physical characteristics
Little consideration of features and processes of system	Parameters are derived from field data and calibration.	Require data about initial state of model and morphology of catchment
High predictive power, low explanatory depth	Simple and can be easily implemented in computer code.	Complex model. Require human expertise and computation capability.
Can be generated to other catchments	Require large hydrological and meteorological data	Suffer from scale related problems
ANN, unit hydrograph, other statistical models	HBV model, TOPMODEL	SHE or MIKESHE model, SWAT
Valid within the boundary of given domain	Calibration involves curve fitting make difficult physical interpretation	Valid for wide range of situations.

TABLE 2.1: Table showing characteristics of the three types of hydrological models. [Adapted from Devia, Pa, and Sa (2015)]

Although there seemed to be a trend of empirical models evolving to conceptual models which eventually evolved to physical models, there has been a renewed interest in the empirical models due to the increase in computational power, data availability and new techniques used (Kratzert et al., 2018; Asati and Rathore, 2012; Pan et al., 2008). The benefits of these empirical models is their re-usability across catchments and simplicity of inputs, with most requiring only rainfall data and historical runoff.

Rockwood (1958) developed a Stream-flow Synthesis and Reservoir Regulation (SSARR) model, which was used for the operation of water control projects in the Columbia River basin. Runoff was estimated as a percentage of rainfall which is based on soil moisture status and rain intensity.

Linsley and Crawford (1960) developed a general purpose model which aimed at simulating daily stream flow from rainfall using a unit hydrograph and recession functions. The improved version of this model included parameters of soil moisture, flow routing techniques and evaportranspiration.

Boughton (1968) developed a model which aimed at estimating the runoff in dry regions of a catchment by using daily rainfall and evaporation data. Runoff was produced when moisture supply was in excess of the three soil moisture storages which included upper soil, drainage storage, and subsoil storage levels. The model was modified for British catchments and was represented by 14 parameters.

Gorgens (1983) used a seven-year data set derived from three well-instrumented semi-acid research catchments in South Africa, to test the performance of four conceptual rainfall-runoff models of differing degrees of complexity. To evaluate the adequacy of the models for use in water resources studies where storage-yield determinations are based on monthly data, a comprehensive set of statistical tests that measured the ability of the models to reproduce the observed monthly flow series, were executed. The results indicate that the relatively simple model based on daily input performed as satisfactory, and in some ways even better than the two complex models which operate on hourly input.

Wang and Yung (1986) developed a model using the Box-Jenkins approach. Their model used six parameters to simulate runoff from small catchment areas. Different methods were used to estimate model parameters and the results were compared

with the original hydrograph. It was found that quadratic programming and least square techniques give better results than the other methods observed.

Jakeman, Littlewood, and Whitehead (1993) developed a method for rainfall-runoff modeling by assessing response characteristics of stream flow. Unit hydrograph techniques were used to model rainfall-runoff on a daily basis. By using daily data, this model was able to separate the fast and slow flow components over an annual period.

Agarwal (2002) utilized genetic algorithms to develop a nonlinear dynamic model for predicting runoff and sediment yield from Vamsadhara catchment in India. The catchment had a strong memory for both runoff and sediment yield and maximum three previous days' successive events influenced the current output of the catchment. The non-linear rainfall-runoff sediment yield prediction model performed better than the linear model.

Schulze and Pike (2004) developed and evaluated the Agricultural Catchment Research Unit (ACRU) agrohydrological modelling system in South African catchments. They recommended the development of a more comprehensive, user-friendly and seamlessly operating Hydrological Decision Support Framework (HDSF) in which models are linked with a common, flexible and extensible database. They suggested that these models should be fully integrated with a graphical information system (GIS) to ensure maximum flexibility in model configurations and model interrogations by water resource managers for planning and scenarios analyses.

There are many rainfall-runoff processes which are still not well understood and are thus difficult to incorporate into a modeling scheme. One example is channel transmission losses, a critical runoff component in semiarid flow regimes. Another, in the context of Southern Africa, which is dominated by fractured-rock aquifers, is the nature of surface-groundwater interactions (Hughes, 2004).

Most of the models developed in South Africa have leaned towards a higher degree of complexity, including a large number of parameters (Kapangaziwiri, 2010; Mahe et al., 2008; Sawunyama and Hughes, 2010; Gorgens, 1983). This is primarily due to the tradition of attempting to represent the process of runoff generation within models, rather than opting for a more simple empirical model with fewer parameters to be used across catchments. It is thus important to give a different perspective on hydrological modeling in South Africa, from a data-driven background.

2.3 Regression Models on Hydrological Response

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships among variables. The focus of regression analysis is on the relationship between the dependent (Y) variable and the independent (X) variables. In general, regression analysis estimates the conditional expectation of the dependent variable given the independent variables. Regression models are frequently used in hydrology, and their correct application as well as the optimal use of the information they yield have been an area of much discussion in the field (Valdes, Vicens, and Rodriguez-Iturbe, 1979). As a common example, the mean and variance of annual flows is related to the physiographic and meteorological characteristics of their respective basins by means of a regression equation. This equation is then used in a number of ways by the hydrologist, such as a source of regional information. However, in most applications the commonly made hypothesis of a scalar covariance matrix of the disturbances is not valid, and heterocedastic and correlated disturbances have been frequently observed. Stream flow, as a natural phenomenon, is continuous in time and so are the meteorological variables which influence its variability. In practice, it can be of interest to forecast the whole flow curve as well as points (hourly, daily or weekly) (Masselota et al., 2016).

The most commonly used conventional data driven models refer to multiple linear regression (MLR), the autoregressive integrated moving average (ARIMA), and autoregressive-moving average with exogenous terms (ARMAX) model. In the statistical analysis of time series, ARIMA models tend to provide a description of a stationary stochastic process in terms of two polynomials, one for autoregression and the other for moving average. These models have been used since 1970 (Zhang, Zhang, and Singh, 2018). Typical explanatory variables are precipitations and temperatures. However, since these models cannot handle nonlinear issues, their forecasting accuracy and performance cannot be ensured. Therefore, MLR, ARIMA and ARMAX models have been used for comparing the modeling performance of alternative models, giving a lower bound on performances of other nonlinear models.

Lane, Diskin, and Renard (1971) studied transmission losses and routing in Walnut Gulch using regression of input hydrograph parameters and output hydrograph parameters. It was found that volume and peak of outflow are strongly correlated with input volume and no significant correlation to antecedent wetness index. They

used their predictions of time of travel, hydrograph peak and volume to fit three-parameter Gamma distributions to the downstream hydrograph.

Bonne (1971) developed a model for simulation of monthly stream flow series, which included precipitation and stream flow. The independent variables in the regression function represented the previous month's flow, current precipitation, antecedent month's precipitation and the accumulated precipitation, with a response of current stream flow. Three watersheds with different geographic characteristics were selected and stream flow for each basin was simulated and compared with the measured records. The results were also compared with those obtained by the simple regression Markovian model. In most cases, a definite improvement was achieved over the simple regression model.

Jarboe and Haan (1974) used a MLR model to relate four parameters of Haan (1972) model and measurable catchment characteristics. The four parameters included maximum possible infiltration rate, the maximum possible seepage rate, the maximum capacity of that part of the soil's moisture-holding capacity, which is less readily available for evapotranspiration, and the constant defining the fraction of seepage that becomes runoff.

Magette, Shanholtz, and Carr (1976) used Jones, 1976 procedure to fit a subset of six parameters of the Kentucky watershed model, and were able to obtain an acceptable MLR equation using indices of 15 watershed characteristics.

Krstanovic and Singh (1991) first used ARIMA in univariate long-term stream flow forecasting. They found that forecasts by ARIMA and univariate models were comparable for periodic stream flow, but for forecasting of highly variable stream flows, the univariate model was superior.

Xu, Seibert, and Halldin (1996) developed relationships between parameters of a monthly water balance model and the land-use data for Swedish catchments. They used regression equations to calculate model parameters from catchment characteristics.

Torfs and Warmerdam (2001) presented a black box approach for rainfall-runoff modeling at a daily scale. They highlighted the following aspects: incorporation of memory, periodicity and non-linearity to the catchment of the Beerze in the Netherlands. They found it feasible to model catchment runoff with black box models, using rainfall data only.

Somvanshi et al. (2006) investigated modeling tools for predicting the behavioural

pattern of rainfall. Two different approaches such as a statistical based ARIMA and ANNs were used. In evaluating the prediction accuracy, they made use of mean annual rainfall data from 1901 to 2003. The models were trained on 93 years of data and tested on the remaining 10, showing the ANN model to outperform the ARIMA model in predicting rainfall.

Asati and Rathore (2012) developed an autoregressive model, as well as a multiple linear regression model and ANN for a complex non-linear relationship between rainfall as input data and output as runoff, without considering other elements of the process and compared the performance of each model.

2.4 Application Horizon of ANN

ANN modeling has been a useful technique in many problems, particularly in those with which the characteristics of the processes are tedious to describe by physical equations. Many researchers have proposed the applicability of ANNs to varying problems in science, engineering, industry, business, economics and environmental fields. The application horizon of ANN extends from problems of image detection using convolutional neural networks to remote sensing and seawater pollution classification. A few interesting applications will be presented, followed by those pertaining topics in ecology.

Zazo et al. (2016) investigated language identification in short utterances using Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN). Results showed that a LSTM RNN without offset modeling is able to detect these languages and generalizes robustly to unseen out of set languages.

Suwajanakorn, Seitz, and Kemelmacher-Shlizerman (2017) used recurrent neural networks that used audio and synthesized it with lip motion of a face in a video to reenact politicians. The Google Sunroof Project uses aerial photos from Google Earth to create a 3-D model of one's roof. The project uses Deep Learning neural networks to separate the roof from surrounding trees and shadows. It then uses the sun's trajectory and weather patterns to predict how much energy can be produced by installing solar panels on your roof. Furthermore, the Apple AI, Siri as well as Amazons Alexa use recurrent neural networks, in particular, the LSTM model (Smith, 2016; Vogels, 2016).

ANNs have also been used in river flow prediction (Imrie, Durucan, and Korre, 2000), stream flow (Zealand, Burn, and Simonovic, 1999) and land drainage (Yang, Prasher, and Lacroix, 1996).

Han and Felkar (1997) estimated daily soil water evaporation using radial basis function ANNs. These models were used to evaluate daily water evaporation from average relative humidity, air temperature, wind speed and soil water content in a cactus field study. The results of the ANN proved to be better than a multiple linear regression analysis.

Kumar et al. (2002) used ANNs to estimate daily grass reference crop evapotranspiration and compared the performance of these ANNs with traditional conventional methods. Issues associated with the use of ANNs were examined, including learning methods, the number of hidden nodes as well as the number of hidden layers.

Karmakar, Kowar, and Guhathakurta (2009) used ANNs to model the spatial interpolation of rainfall variables. They applied their methods to 102 rain gauge stations. A three layer perceptron FFNN proved to achieve good results.

The use of ANN in ecological problems has been widely studied, to give promising results more often than not. It turns out that their use in stream flow prediction has been a growing interest in the past two decades.

2.5 ANN Stream Flow Prediction Models

Traditional stream flow estimation techniques require both reliable rainfall-runoff records and a procedure for updating model parameters from time to time. It is not feasible to set up and maintain rain gauge stations over many locations for a long period of time. Although many large catchments are monitored, there exist a lot of smaller catchments which are not, which may play an important role in many area's water supply. Thus, it may be beneficial to compile a general purpose model for all catchments of similar characteristics. It was seen (from literature above) that runoff values have been estimated by many researchers using climatic variables and models which require data that is not easily available. Simpler data driven methods such as fitting a linear relationship between the variables, with runoff as the response, have proven to fall short in prediction accuracy as it is evident that the rainfall-runoff process is highly nonlinear (Krstanovic and Singh, 1991; Somvanshi

et al., 2006). There is a stressed need for accurate estimates of runoff using models which consider the inherent nonlinearity of the process and which may use inputs easily available. It has been reported that ANNs may offer an alternative to conventional data driven methods (ASCE, 2000) for predicting runoff. Black-box models, such as ANNs, describe the relationship between input (rainfall) and output (runoff) without describing the inherent physical process. The ANN is hypothesized to learn these processes in the calibration phase. These ANN models seek to establish a statistical relationship between the input and the output and have proven successful within a range of data available from a catchment (Dawson and Wilby, 2001; Harun, Nor, and Kassim, 2002; Carcano et al., 2005). It is proposed that the mathematical structure of the ANN carries with it an implicit representation of the underlying physical process. The biggest advantage of the ANN approach in predicting runoff is that it requires only rainfall data. First studies using ANNs for rainfall-runoff prediction date back to the early 1990s (A. H. Halff and Azmoodeh, 1993; Daniell, 1991), which find the method useful for forecasting processes in the hydrological application.

Karunanithi et al. (1994) made use of a FFNN model to forecast stream flow. The accuracy of this model was compared with that of a nonlinear power model and it was found that the ANN outperformed the nonlinear power model in predictive accuracy.

Lorrai and Sechi (1995) used ANN to model the rainfall-runoff transformation. Two hidden layers were used with a sigmoid activation function. Using back propagation, a learning rule was developed. Monthly data of rainfall and temperature were used to predict monthly runoff. The data was divided into three 10 year periods in order to calibrate, verify and test the model. It was found that ANNs provided higher efficiency during model development and were superior to multivariate auto-regressive models. There was a downfall in the verification of the results of the developed model.

Hsu, Gupta, and Sorooshian (1995) showed the application of an ANN approach in describing the rainfall-runoff process. The ANN approach was used to provide a better representation of the rainfall-runoff process than the linear ARMAX time series approach and the conceptual Sacramento Soil Moisture Accounting (SAC-SMA) model. A total of six years of data were used for the development and testing of the model. This was split into development and testing data on a ratio of 5:1.

Carriere, Mohaghegh, and Gaskar (1996) conducted the first study using recurrent neural networks (RNNs) for rainfall-runoff modeling. They tested the use of RNNs within laboratory conditions and demonstrated their potential use for event-based applications.

Hsu, Gupta, and Sorooshian (1997) used two different ANN model structures to model the rainfall-runoff process for the Leaf River Basin in Mississippi. Their results showed that both structures, the popular Three Layer Feedforward Neural Network (TLFNN) and the RNN, perform well. However, the TLFNN requires trial-and-error testing to identify the appropriate number of time-delayed input variables to the model. Furthermore, it is not suitable for distributed watershed modeling; i.e., when distributed precipitation information (multiple gages or radar images) is available. The RNN structure provides a representation of the dynamic internal feedbacks loops in the system, thereby eliminating the need for lagged inputs and resulting in a reduction in the number of network weights (and hence training time). The suitability of RNN's for distributed watershed modeling was discussed in their paper.

Shamseldin (1997) applied a multi-layer FFNN to rainfall-runoff modeling on the data of six different catchments. It was tested against three selected rainfall-runoff models, namely, simple linear model (SLM), seasonality based linear perturbation model (LPM) and the nearest neighbour linear perturbation model (NNLPM). In verification, one or the other form of neural network was better than the three models in case of four out of the six catchments. Possible reasons for better results were not specified.

Zealand, Burn, and Simonovic (1999) investigated the use of ANNs for short term forecasting of stream flows. This approach was applied to a portion of the Winnipeg River system in Canada. A close fit was obtained during the calibration phase and the ANNs outperformed a conventional model during the verification phase.

Tokar and Johnson (1999) studied the application of an ANN methodology to predict daily runoff as a function of daily temperature, precipitation and snow melt in the Little Patuxent River catchment in Maryland. It was found that the content and length of training data had a large impact on prediction accuracy. The ANN model outperformed regression and conceptual models. This ANN model provided a more systematic approach, reducing the length of calibration data.

Tokar and Markus (2000) used ANN models to compare their predictive accuracy of runoff as a function of rainfall and temperature against other traditional conceptual models. The ANN model was applied to model monthly stream flow, comparing to a conceptual water balance model. The model was also used to describe the daily rainfall-runoff process comparing this with the Sacremento Soil Moisture Accounting (SAC-SMA) model and the Simple Conceptual Rainfall-Runoff (SCRR) model. In all scenarios, the ANN model showed higher prediction accuracy.

Dawson and Wilby (2001) assessed two neural networks viz. the radial basis function network (RBF), and the multilayer perceptron (MLP), using rainfall-runoff data for the River Yangtze in China from 1991 to 1993. It was seen that both network types could simulate stream flow beyond the range of the training set. Comparisons were made between neural networks and conventional statistical techniques such as stepwise multiple linear regression, and ARIMA models showing that the ANN outperformed the other models.

Rajurkar, Kothyari, and Chaube (2002) developed both linear and non-linear models as well as ANNs for transforming daily rainfall into runoff in the Narmada catchment area. They found that the ANN model produced a higher prediction accuracy than both the linear and non-linear models.

Harun, Nor, and Kassim (2002) designed an ANN model to predict daily runoff using rainfall as input nodes. The Sungai Lui catchment in Selangor Malaysia was used as the study area. Results of the ANN model were compared to the Hydrologic Modeling System (HEC-HMS) model, ultimately finding that the ANN models showed a better generalization of the rainfall-runoff relationship.

Wilby, Abrahart, and Dawson (2003) examined the internal behaviour of an ANN rainfall-runoff model and showed that specific architectural features could be interpreted with respect to the quasi-physical dynamics of a parsimonious water balance model. Neural networks were developed for daily discharge simulated by a conceptual rainfall-runoff model for the Test River basin in southern England. Neural outputs associated with each hidden node, produced from the output node after all other hidden nodes had been deleted, were then compared with state variables and internal fluxes of the conceptual model (including soil moisture, percolation, groundwater recharge and baseflow). Correlation analysis suggested that hidden

nodes in the neural network correspond to dominant processes within the conceptual model. In particular, different hidden nodes are associated with distinct "quickflow" and "baseflow" components, as well as a threshold state in the soil moisture accounting. The results also demonstrated that, for this river basin, a neural network with seven inputs and three hidden nodes could emulate the gross behaviour of the conceptual model.

Nagesh Kumar, Srinivasa Raju, and Sathish (2004) investigated two different networks, namely the feed forward network and the recurrent neural network on modeling rainfall-runoff. The feed forward network was trained using the conventional back propagation algorithm with many improvements and the recurrent neural network was trained using the method of ordered partial derivatives. The selected ANN models were used to train and forecast the monthly flows of a river in India, with a catchment area of 5189 km^2 up to the gauging site. The trained networks were used for both single step ahead and multiple step ahead forecasting. A comparative study of both networks indicated that the recurrent neural networks performed better than the feed forward networks. In addition, the size of the architecture and the training time required were less for the recurrent neural networks. The recurrent neural network gave better results for both single step ahead and multiple step ahead forecasting. Hence recurrent neural networks are recommended as a tool for river flow forecasting.

Riad et al. (2004) developed a rainfall-runoff ANN model and compared this to a MLR model showing the effectiveness of the ANN model. The results and comparative study indicate that the artificial neural network method was more suitable to predict river runoff than a classical regression model.

De Vos and Rientjes (2005) reported on the application of multi-layer feed-forward ANNs for rainfall-runoff modeling of the Geer catchment (Belgium) using both daily and hourly data. The daily forecast results indicate that ANNs can be considered good alternatives for traditional rainfall-runoff modeling approaches, but the simulations based on hourly data reveal timing errors as a result of a dominating autoregressive component. This component was introduced in model simulations by using previously observed runoff values as ANN model input, which is a popular method for indirectly representing the hydrological state of a catchment. They commented on the use of lagged variables as input to the models. They concluded that complementary conceptual models can be valuable additions to ANN model approaches.

Secondly, they found that not all differences between modeled and observed hydrograph characteristics can be adequately expressed by a single performance measure such as the MSE. Using more than one performance measure for the evaluation of ANN models during training might therefore improve the quality of these models.

(Carcano et al., 2005) simulated potential scenarios in rainfall-runoff transformation at the daily scale. This was done for the control and management of water resources. FFNNs and RNNs were investigated to show that the RNN outperformed the FFNN especially when rainfall memory effect was employed as an additional input.

Pan et al. (2008) investigated the application of recurrent neural networks to the rainfall-runoff process for the Keelung River, Taiwan. They compared the performances of recurrent neural network models with general feed forward neural networks. The Deterministic Linearized Recurrent Neural Network (DLRNN) was the main focus of the study and it was found that this model far outperformed the feed forward neural network. The results showed that the performance was satisfactory and DLRNN is competent to simulate dynamic systems, like rainfall-runoff processes.

Sedki, Ouazar, and Mazoudi (2009) investigated the effectiveness of the genetic algorithm (GA) evolved neural network for rainfall-runoff forecasting and its application to predict the runoff in a catchment located in a semi-arid climate in Morocco. To evaluate the performance of the genetic algorithm-based neural network, back propagation neural networks were also involved for a comparison purpose. The results showed that the GA-based neural network model gives superior predictions. The well-trained neural network can be used as a useful tool for runoff forecasting.

Machado et al. (2011) developed three different monthly rainfall-runoff models using ANNs. These were compared with a conceptual model at the monthly time scale to prove the ability of the ANN model to accurately predict the observed flows. This study evaluated the capacity of ANNs to model with accuracy the monthly rainfall-runoff process. The case study was performed in the Jangada River basin, Brazil. The results of the three ANNs that produced the best results were compared to those of a conceptual model at a monthly time scale. The ANNs presented the best results with the highest correlation coefficients and Nash-Sutcliffe statistics and the smallest difference of volume.

Sinha, 2011 developed Back Propagation Artificial Neural Network (BPANN)

and radial basis function ANN (RBFANN) models to assess their runoff simulation applicability for the Upper Kharun catchment. These models were evaluated in comparison with multiple linear regression (MLR) runoff simulation models. Comparisons of the MLR, BPANN and RBFANN models showed that RBFANN models performs better than BPANN and MLR models. It was found that training and testing results revealed that the models simulated the daily, weekly and monthly runoff yield satisfactorily. Therefore, these ANN models, based on simple inputs can be used for estimation of runoff, missing data and testing the accuracy of other models.

Chen, Wang, and Tsou (2013) developed a model for estimating runoff by using rainfall data from a river basin and employed a neural network technique to recover missing data. Hourly rainfall and flow data from Nanhe, Taiwu, and Laii rainfall stations and Sinpi flow station in the Linbien basin were used. The data records were of 27 typhoons between the years 2005 and 2009. The feed forward back propagation network (FFBP) and conventional regression analysis (CRA) were employed to study their performances. From the statistical evaluation, it was found that the performance of FFBP exceeded that of regression analysis.

Gowda and Mayya (2014) used ANN for forecasting stream flow in natural rivers using rainfall data from several different rain gauge stations.

Kratzert et al. (2018) investigated the use of recurrent neural networks, in particular, the long short-term memory (LSTM) networks to model the rainfall-runoff process on a dataset of 671 catchments. It was found that the LSTM model performed very well on the testing data and it was suggested that this model be used over traditional feed-forward neural networks due to its ability to memorize previous output. They concluded with a very interesting result that the LSTM unintentionally generated observable snow dynamics within a cell state, suggesting that there is more to find behind the scenes. In simpler terms, they found that the value of the cell state increases as soon as the temperatures drop below zero and a fast depletion is evident as soon as the temperatures increases above the freezing point. The application of LSTMs and its further development therefore have a high potential to extend data-based mechanistic modeling approaches in the field of hydrology, e.g. rainfall-runoff modeling.

Other soft computing techniques, besides ANNs, have been used for developing rainfall-runoff models. These techniques include Genetic Algorithms and Fuzzy Logic (Chandwani et al., 2015).

The applications of ANN and RNN, in particular, to hydrological modeling has been rapidly growing in recent years. Between 2000 and 2018 over 20 papers were published, and reviewed in terms of the modeling process, in which RNNs have been used for simulation or forecasting of water resources variables. Due to the rapid increase in journals, it is unlikely that complete coverage has been achieved. The LSTM is proposed to be the "state-of-the-art" architecture where the sequential nature of the data matters. Due to this, FFNN and RNN will be the focus of this paper, with the LSTM model being of interest.

Chapter 3

Theoretical Considerations

Transformation of rainfall into runoff - a hydrological process, is believed to be highly nonlinear, spatially distributed, time variant and difficult to describe by simple models. This chapter deals with theoretical considerations of artificial neural networks, recurrent neural networks and the long short-term model.

3.1 ANN Modeling

Artificial neural networks (ANNs) are one of the most powerful tools used in machine learning and artificial intelligence (AI). Inspired by the biological neural networks of the brain, they can tackle a wide range of problems that defeat traditional AI, including computer vision, speech recognition, natural language processing, and robotics (Martin and Schuermann, 2016). A neural network is a system of interconnected neurons which send signals to one another Tchircoff (2017). The strengths of these connections between the neurons, also known as weights, determine the network's behaviour. More importantly, these weights can be systematically tuned to make the neural network behave in a highly specific, desirable way. The ANN occurs in organised layers. Some of the layers are termed hidden layers. The number of hidden layers is controlled by the researcher and subject to the complexity required. The complexity of the model increases as the number of hidden nodes (neurons) increases. The training is done in a way that neurons' connections are optimized until the error in predictions is minimized. Once the training is done, the ANN can be tested with a new set of information. ANN require no knowledge of the data source or its underlying distribution but require large training data sets (Agatonovic-Kustrin and Beresford, 1999).

3.2 Basic concepts of ANN

The framework as well as the optimisation scheme is the basis of ANNs. The structure of the ANN includes many single units called nodes or neurons, which are grouped into layers, viz. input, hidden, and output layers. The input layers consist of neurons that receive input from the external environment. The output layer consists of neurons that communicate the output of the system to the user or external environment. There are generally a number of hidden layers between input and output layers. The network receives inputs to be modeled in the input layer and are multiplied by some weight value and added together to pass through an activation function. This activation function determines if the neuron should react, or be "activated". Each activated neuron in the input layer sends a signal to all the neurons in the next layer, which is the hidden layer. The strength of the signal is dependent on both how much the neuron is activated and on the weight of the connection between the two neurons. Each neuron in the hidden layer then adds up all of the signals it received from neurons in the input layer. The reaction signals then passes through a transfer function which decides the strength of the output signal. Finally, the output signal is sent through all the output connections to other nodes or neurons. The activation function is usually simple and easily differentiable, such as sigmoidal functions.

Network weights are randomly initialized when the network is created and these are allowed to hold any real number value (Ellacott and D, 1996). The number of hidden nodes or neurons can be found through trial and error. By increasing the number of hidden neurons beyond a certain limit, over fitting can occur, consequently, the training data set will be memorized, hence, making the network useless for new data sets (Sinha, 2011).

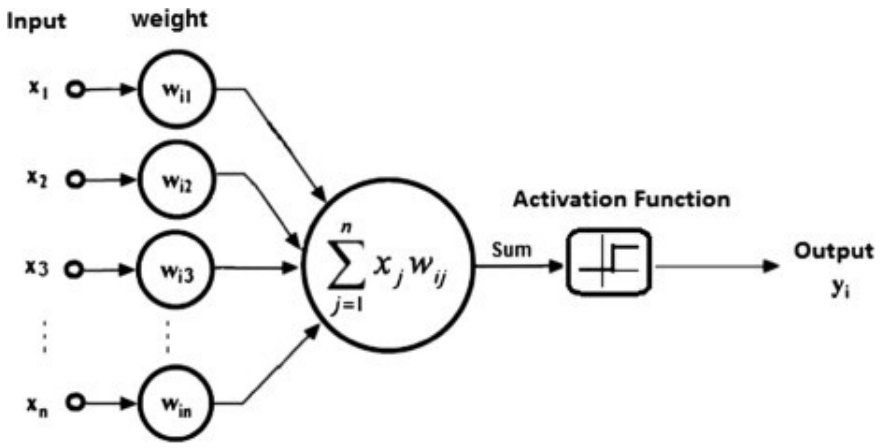

FIGURE 3.1: Visual Structure of neuron in ANN [Source: Saxena (2017)]

Figure 3.1 shows the basic structure of an artificial neuron. It receives input data which are represented by x_1, x_2,...,x_n which are weighted by w_1, w_2,..,w_n respectively and summed. This weighted sum is passed through an activation function (Section 3.3) or transfer function to map the weighted sum in between some range. The result is then used to achieve the output. This output can be used as an input in a feedback neural network or it can be recorded as an output of a FFNN. Bias neurons allow the network output to be translated to the desired output range. A bias neuron is simply a neuron with an activation set to the maximum possible value. This value, along with those of all the other neurons in the same layer, is passed along to all neurons in the next layer. The function of the bias neuron is to shift the input sums in the next layer by a value that is independent of the input to the neural network. This allows the network to represent functions which are affine in nature. Essentially, its effect is the same as shifting the activation functions of the neurons in the next layer horizontally. This allows us to translate the output to the desired output range.

Hornik, Stinchcombe, and White (1999) concluded that an ANN with a single hidden layer, having a sufficient number of neurons can approximate any complex nonlinear function to an acceptable degree. Although several authors have given certain empirical equations to approximately estimate the number of neurons in the hidden layer (Hunter et al., 2012; Molga, 2003; Pendharkar and Rodger, 2003; Tamura and Tateishi, 1997), it is preferred to adopt trial and error approach for deciding the optimal number of neurons in the hidden layer.

3.3 Activation Functions

Activation functions are an integral part of neural networks. They decide whether the information of a specific neuron is important. If the activation function were not present then we would have a linear regression model that may not be able to handle highly nonlinear problems such as those presented in this paper. There exist a wide range of activation functions, each with their benefits, however we will present the few which have been used in this project. The choice of type of activation function effects the evaluation of the model, however this choice is mathematically arbitrary. It is true, though, that the behaviour of the activation functions in hidden layers affects how the model learns in practice. This is due to the fact that the mechanism by which the parameters of the network are tweaked during learning operates via the evaluation of the activation function.

3.3.1 Sigmoid function

The Sigmoid activation function is one of the most frequently used in most applications (Walia, 2017). It is a nonlinear function denoted as the following:

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \qquad \text{for } -\infty \leq x \leq \infty.$$

Other forms of the function include:

$$f(x) = \frac{1}{1+e^{-\alpha x}} \quad \text{and} \quad f(x) = \frac{1}{1+e^{-g(x-b)}}, \tag{3.1}$$

where α, g, and b are the learning rate, gain and bias respectively.

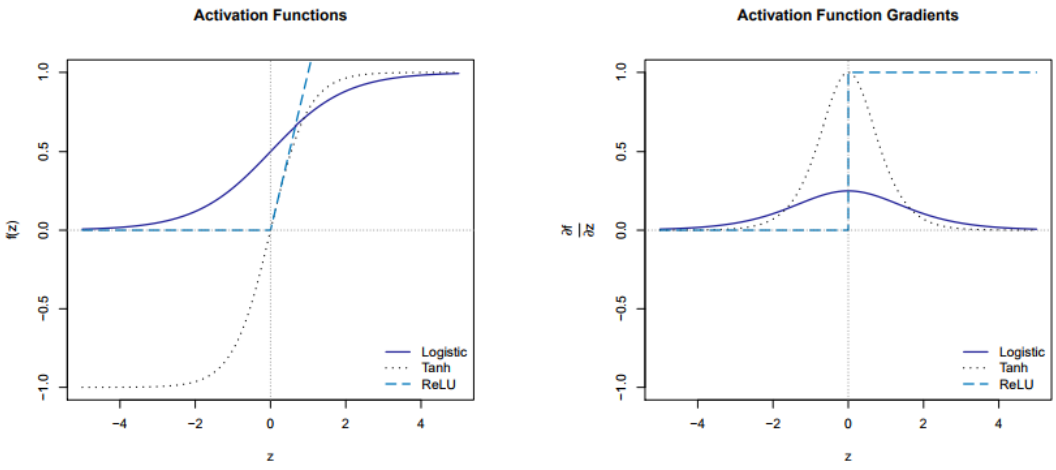

FIGURE 3.2: Activation Functions [Source: Pienaar (2018)]

Figure 3.2 above graphically represents the sigmoid activation function and its derivative. One of the issues regarding the sigmoid functions is that the gradient approaches zero as X becomes more extreme which means at this point the network does not do much learning. The Range of the sigmoid function is between $(0, 1)$ which prevents activations from blowing up but it returns all positive values. This issue can be addressed with the hyperbolic tangent function (3.3.2).

3.3.2 Hyperbolic Tangent function

Though the logistic sigmoid has a pleasing biological interpretation (*The Study of Population Growth in Organisms Grouped by Stages*), it turns out that the logistic sigmoid can cause a neural network to get "stuck" during training (Stansbury, 2018). An alternative to the logistic sigmoid is the hyperbolic tangent, or *tanh* function:

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{3.2}$$

Like the logistic sigmoid, the *tanh* function (Figure 3.2) is also sigmoidal ("s"-shaped), but instead outputs values that range $(-1, 1)$. An important property of this function is that only zero-valued inputs are mapped to near zero outputs.

3.3.3 Rectified Linear Units

Rectified linear units (ReLU) are simply defined as the positive part of its argument:

$$f(x) = x^+ = max(0, x). \tag{3.3}$$

When looking at the gradient functions for each activation in figure 3.2 it is seen that for sufficiently large inputs there may be very little change in the value of the activation function. The affects of this will become clear when one implements back-propagation.

3.4 Network Topologies

ANNs have become one of the most prominent concepts in the field of artificial intelligence. ANNs have already been applied in the thousands of applications across all fields of study. The key issue of ANN modeling is that in almost all situations the performance depends largely on the architecture of the model (Hochreiter and Schmidhuber, 1997). As a result, designing a proper ANN is always a vital issue in the field of neural networks. The determination of an appropriate ANN architecture is always a challenging task for the ANN designers (Islam et al., 2007). The network topology refers to the arrangement of the ANN's nodes and connections. There are different kinds of topology such as feed forward networks which is a non-recurrent network meaning that the output produced by the network does not get feed back into to the network as a variable. There are two types of feed forward networks, single layer and multilayer layer feed forward networks. Another network type is that of feedback networks, in which the signal can flow in both directions via loops. In deciding on a network typology, the researcher should look specifically at the nature of the problem as well as endure a period of trial and error. Here, we explain the different topologies used in this paper explained by their updating equations.

3.4.1 Feed-Forward ANN

One way of forecasting stream flow from rainfall would be to ignore the sequential nature and build a per time interval regression that considers each time step in isolation. Also known as a multilayer perceptron, Feed-Forward Neural Networks (FFNN) are the commonest types of neural networks (NN) in practical application. This topology of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has connections directed to the neurons of the next layer. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (there may be many) and to the output nodes. There are no cycles or loops in the network. Multilayer networks use a variety of learning techniques, the most popular being back-propagation.

Consider modeling a dataset consisting observations of p inputs $\boldsymbol{x}_i^T = [x_{i1}, x_{i2}, \ldots, x_{ip}]$ and q outputs $\boldsymbol{y}_i^T = [y_{i1}, y_{i2}, \ldots, y_{iq}]$ for $i = 1, 2, \ldots, N$ (a total of N training examples). Let $\boldsymbol{a}_j^l$ denote the j^{th} node on the l^{th} layer of a standard feed-forward neural network, then the network structure can be written as an updating equation:

$$a_j^l = \sigma_l\Big(\sum_{k=1}^{d_l-1} a_k^{l-1} w_{kj}^l + b_j^l\Big) \quad l = 1, \ldots, J; \quad j = 1, \ldots, d_l, J-1. \tag{3.4}$$

Where:

- $\sigma_l(.)$ denotes an activation function on layer l. Addressed in Section 3.3.
- d_{l-1} denotes the number of nodes in layer $l-1$.
- w_{kj}^l denotes the kj^{th} weight parameter linking the k^{th} node in layer $l-1$ and j^{th} node in layer l.
- b_j^l denotes the j^{th} bias in layer l.
- and the equation is evaluated subject to the initial conditions $a_j^{(0)} = x_{ij}$ for all j at the i^{th} training example.

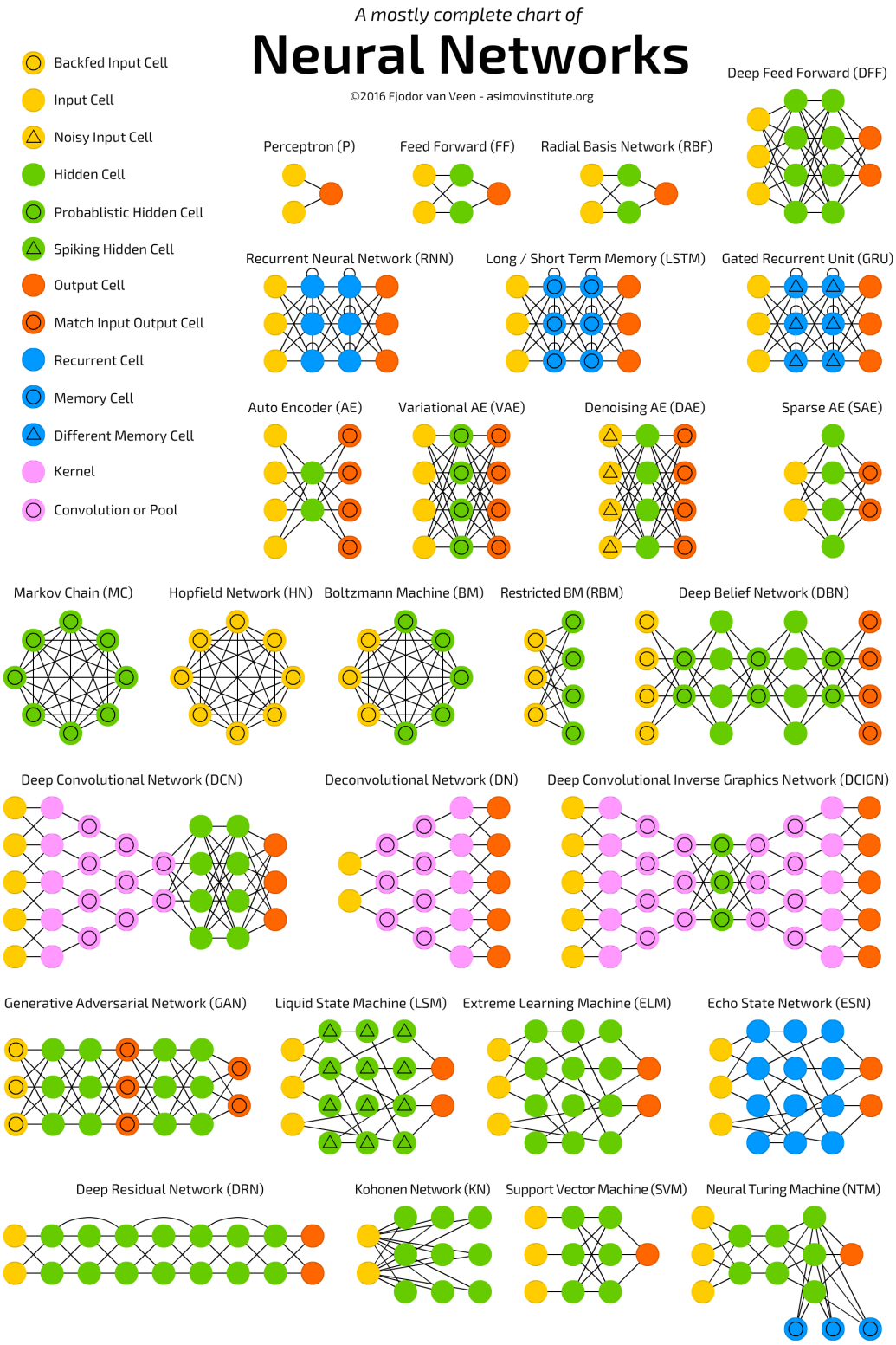

FIGURE 3.3: A mostly complete chart of neural networks [Source: Tchircoff (2017)]

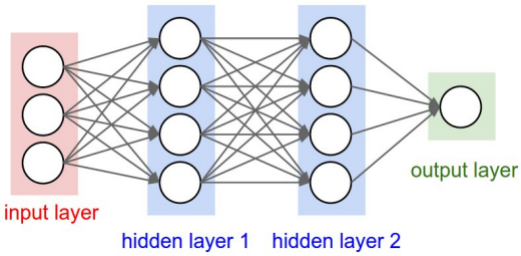

FIGURE 3.4: Architecture of a feed-forward neural network [Source: McGonagle (2018)]

With respect to the problem at hand, the simplest FFNN would consist of a single input (rainfall) and a single output (stream flow). We do, however, know that the stream flow would be a function of both current rainfall as well as historical rainfall and historical stream flow. It may, therefore, be beneficial to include lag variables as input data as well as current rainfall. These lags may be of more importance than current rainfall itself, considering the memory of catchments including groundwater, snow or even glacier storages, with lag times between precipitation and discharge up to several years.

3.4.2 Recurrent Neural Network

Ignoring the sequential aspect of the data could prove problematic. If after a drought, it rained substantially, stream flow may not respond as it would had it been in a wetter period. One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task. RNNs, such as LSTM (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU), have shown to achieve the state-of-the-art results in many applications with time series or sequential data, including machine translation (Sutskever, Vinyals, and Le, 2014) and speech recognition (Hinton et al., 2012).

Unlike the FFNN, RNNs contain hidden states which are distributed across time. This allows them to efficiently store a lot of information about the past. As with a regular FFNN, the nonlinear dynamics introduced by the nodes allows them to capture complicated time series dynamics (Lewis, 2017). Neural networks can be classified into dynamic and static. Static neural networks calculate output directly from the input through feed-forward connections described above. In a basic feed-forward neural network, the information flows in a single direction from input to output. These networks have no feedback properties. In a dynamic neural network,

the output depends on the current input to the network, and the previous inputs, outputs, and/or hidden states of the network. Recurrent neural networks are examples of dynamic neural networks.

The core idea of recurrent neural networks is that the connections allow memory of previous inputs to persist in the network's internal state, and thereby influence the network's output. The steps in the process of the recurrent neural network are similar to that of the feed-forward neural network, however, the value held in the delay unit is fed back to the hidden units as additional input. The delay units allow the network to have short term memory. The RNN has a "memory" which captures information about what has been calculated by the hidden units at an earlier time step. Time-series data contain patterns ordered by time. Information about the underlying data generating mechanism is contained in these patterns. RNNs take advantage of this ordering because the delay units exhibit persistence. It is this "short term" memory feature that allows an RNN to learn and generalize across sequences of inputs (Lewis, 2017).

Furthermore, RNNs are constructed of multiple copies of the same network, each passing a signal to a successor. The model therefore shares the same parameters across all time steps as it perform the same task at each step, only with different inputs. This allows a reduced total number of parameter required to learn relative to a traditional deep neural network which uses a different set of weights and biases for each layer. This is easily seen in the network architecture is displayed below:

The output (in our case discharge) for a specific time step is predicted from the input $\boldsymbol{x} = [x_{-n}, ..., x_0]$ consisting of the last n consecutive time steps of independent variables (in our case daily or weekly precipitation) and is processed sequentially. In each time step $t(-n \leq t \leq 0)$, the current input $\boldsymbol{x_t}$ is processed in the recurrent cells of each layer in the network.

When trying to predict stream flow from rainfall, it is entirely possible for the gap between relevant information and the point where it is needed to become very large. In theory, RNNs are absolutely capable of handling such long term dependencies. However, in practice, this is not the case (Kratzert et al., 2018). The problem was explored in depth by Hochreiter and Schmidhuber (1997) and Bengio, Simard, and Frasconi (1994), who found some pretty fundamental reasons why it might be difficult. The Long Short-Term Model (LSTM) does not have this problem.

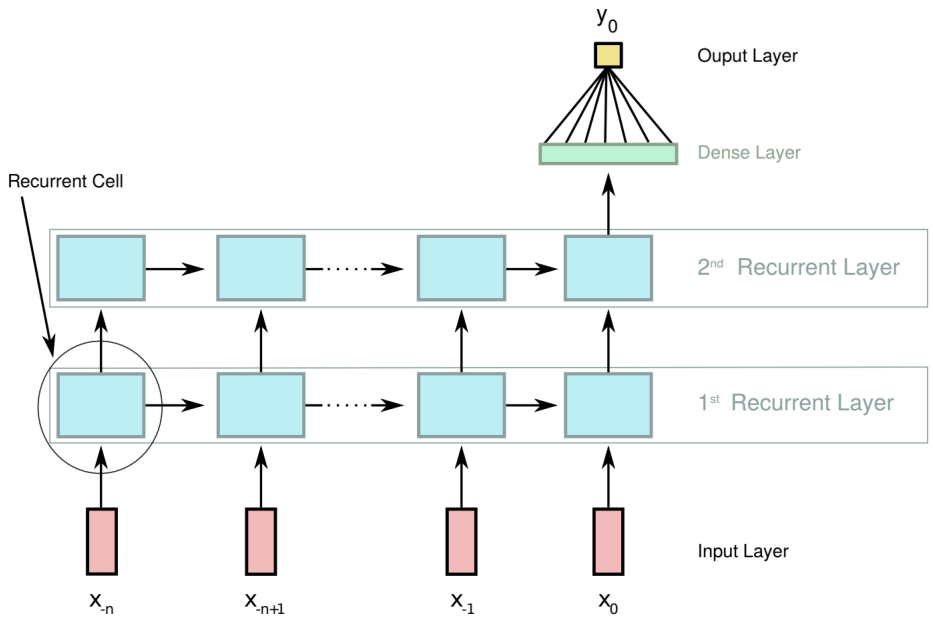

FIGURE 3.5: RNN architecture [Source: Kratzert et al. (2018)]

3.4.3 Long Short-Term Model

In this section, we introduce the LSTM architecture in more detail. We will describe the technical aspects of the network internals as well as the hydrological interpretation of the LSTM in order to bridge the differences between the hydrology and deep learning research communities[1]. For problems, for which the sequential order of the inputs matters, the current state-of-the-art network architecture is the so-called LSTM, which in its initial form was introduced by Hochreiter and Schmidhuber (1997).

The LSTM topology is a special kind of RNN, designed to overcome the weakness of the traditional RNN to learn long-term dependencies (Kratzert et al., 2018). The traditional RNN has been shown to hardly remember sequences with a length of over 10 (Bengio, Simard, and Frasconi, 1994). For modeling daily runoff, this would imply that we could only use the previous 10 days of hydrological data as input to predict the runoff of the next day. This period is too short considering the memory of catchments including groundwater, snow or even glacier storages, with lag times between precipitation and discharge up to several years (Kratzert et al., 2018).

All RNNs have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single *tanh* layer.

[1]This section was largely from Kratzert et al. (2018)

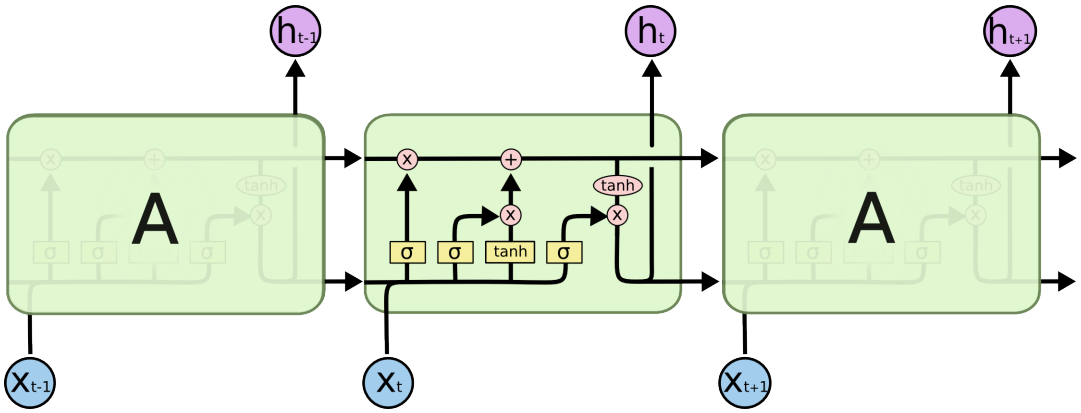

FIGURE 3.7: Recurrent cell differences of RNN and LSTM [Source: Olah (2015)]

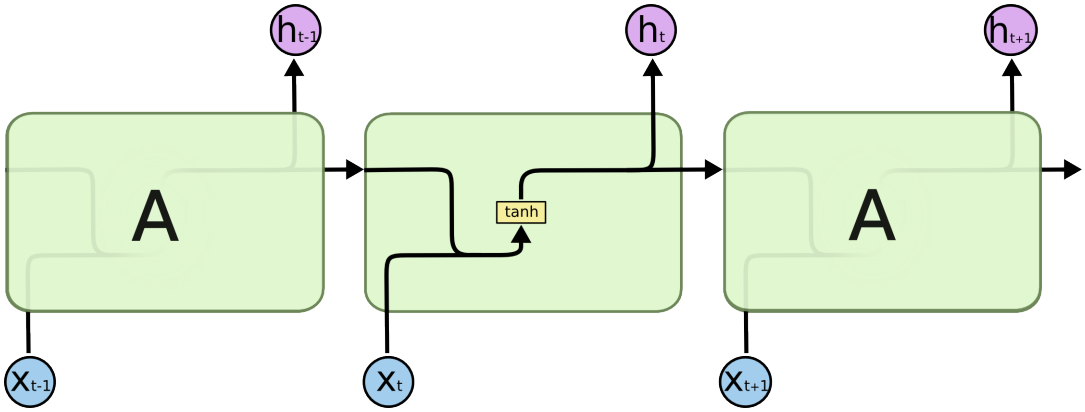

FIGURE 3.6: Repeating module in a standard RNN containing a single layer [Source: Olah (2015)]

The difference of the traditional RNN and the LSTM are the internal operations of the recurrent cell (encircled in Figure 3.5) which are depicted in Figure 3.7.

In a traditional RNN, only one internal state $\boldsymbol{h_t}$ exists which is recomputed in every time step by the following equation:

$$\boldsymbol{h_t} = f(\boldsymbol{W x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b}) \tag{3.5}$$

where $f(.)$ is the activation function (typically the hyperbolic tangent), $\boldsymbol{W}$ and $\boldsymbol{U}$ are the adjustable weight matrices of the hidden state $\boldsymbol{h}$ and the input $\boldsymbol{x}$, and $\boldsymbol{b}$ is an adjustable bias vector.

Differing from this, the LSTM has an additional cell memory $\boldsymbol{c_t}$ in which information is stored, as well as three gates that control the information flow within the LSTM cell (encircled 'x' in Figure 3.7). Introduced by Gers, Schmidhuber, and Cummins (2000), the first gate is the forget gate. This forget gate controls which elements

of the cell state vector $\boldsymbol{c_{t-1}}$ will be forgotten. This is computed by the following equation:

$$\boldsymbol{f_t} = \sigma(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b_f}) \tag{3.6}$$

where $\boldsymbol{f_t}$ is a vector with values in the range $(0, 1)$, $\sigma(.)$ represents the logistic sigmoid function and $\boldsymbol{W_f}$, $\boldsymbol{U_f}$ and $\boldsymbol{b_f}$ define the set of learnable parameters for the forget gate, which are two adjustable weight matrices and a bias vector. For the traditional RNN, the hidden state $\boldsymbol{h}$ is initialized in the first time step by a vector of zeros with a user-defined length.

In the step which follows, an update vector for the cell state is computed from the current input ($\boldsymbol{x_t}$) and the last hidden state $\boldsymbol{h_{t-1}}$ given by the following equation:

$$\tilde{\boldsymbol{c}} = tanh(\boldsymbol{W_{\tilde{c}} x_t} + \boldsymbol{U_{\tilde{c}} h_{t-1}} + \boldsymbol{b_{\tilde{c}}}) \tag{3.7}$$

where $\tilde{\boldsymbol{c}}$ is a vector with values in the range (-1, 1), $tanh(.)$ is the hyperbolic tangent and $\boldsymbol{W_{\tilde{c}}}$, $\boldsymbol{U_{\tilde{c}}}$ and $\boldsymbol{b_{\tilde{c}}}$ are another set of learnable parameters. The input gate is then computed, defining which and how much information of $\tilde{\boldsymbol{c}}$ is used to update the cell state in the current time step:

$$\boldsymbol{i_t} = \sigma(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_i}) \tag{3.8}$$

where $\boldsymbol{i_t}$ is a vector with values in the range (0, 1), and $\boldsymbol{W_i}$, $\boldsymbol{U_i}$ and $\boldsymbol{b_i}$ are a set of learnable parameters, defined for the input gate.

From the results of equations 3.6, 3.7 and 3.8, the cell state, $\boldsymbol{c_t}$ can be updated using the following equation:

$$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \boldsymbol{c} \tag{3.9}$$

where $\odot$ is the Hadamard product (element-wise multiplication). $\boldsymbol{f_t}$ and $\boldsymbol{i_t}$ both have values in the range (0,1) and thus equation 3.9 may be interpreted in such a way that it defines which information is stored in $\boldsymbol{c_{t-1}}$ will be kept (values of $\boldsymbol{f_t}$ which are close to 1) and which will be forgotten ($\boldsymbol{f_t}$ which are close to 0). This equation also decides which new information will be added to thee cell state and which will be ignored. Similar to the hidden state vector, the cell state is initialised by a vector of zeros in the first time step. The final gate is the output gate, which

controls the information of the cell state that flows into the new hidden state. This gate is calculated using the following equation:

$$\boldsymbol{o_t} = \sigma(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{b_o}) \tag{3.10}$$

where $\boldsymbol{o_t}$ is a vector with values in the range (0,1), and $\boldsymbol{W_o}$, $\boldsymbol{U_o}$ and $\boldsymbol{b_o}$ are a set of learnable parameters which are defined for the output gate. From $\boldsymbol{o_t}$, the new hidden state $\boldsymbol{h_t}$ is calculated by combining equations 3.9 and 3.10 to give:

$$\boldsymbol{h_t} = tanh(\boldsymbol{c_t}) \odot \boldsymbol{o_t} \tag{3.11}$$

It is the cell state that allows an effective learning of long-term dependencies. Due to the simple linear interactions with the remaining LSTM cell, it can store unchanged information over a long period of time steps. During training, this characteristic helps to prevent the problem of the exploding or vanishing gradients in the back propagation step (Hochreiter and Schmidhuber, 1997).

As with other continuous hydrological models, the LSTM processes the input data at each time step. At every time step, the input data (rainfall) is used to update several values in the LSTM internal cell states. These cell states can be interpreted as storages which are often used for groundwater storage and soil water content, etc. Updating the states of these internal cells is regulated through several gates which regulate depletion, increase, and outflow of storages. The parameters of each of these gates are adapted and adjusted during a calibration period. In contrast to conventional hydrological models, the LSTM does not "know" the principle of water conservation and the governing processes and is optimized to predict the runoff as good as possible, learning these physical principals during the calibration phase, purely from the data.

3.5 Network Training

In order to train neural networks, some variant of stochastic gradient decent is used. Regardless of the particular algorithm used, a key component in the updating equation will involve evaluating the gradient of the cost function with respect to the parameters of the model. It is possible to derive general expressions for the required

gradients in terms of the model elements. This is what is widely known as back-propagation.

3.5.1 Back-propagation

Formally derived by Bryson and Ho (1969) and implemented to run on computers much as it is today by Linnainmaa (1976), back-propagation is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network.

The goal of the back-propagation training algorithm is to modify the weights of a neural network in order to minimize the error of the network outputs compared to some expected output in response to corresponding inputs. It is a supervised learning algorithm that allows the network to be corrected with regard to the specific errors made. The algorithm is as follows:

1. Present a training input pattern and propagate it through the network to get an output.
2. Compare the predicted outputs to the expected outputs and calculate the error.
3. Calculate the derivatives of the error with respect to the network weights.
4. Adjust the weights to minimize the error.
5. Repeat.

Regression tasks, in the context of supervised learning, usually refer to modeling continuous responses. That is $\boldsymbol{Y_i}$ may take on any value in a continuum of possible outcomes (Pienaar, 2018). A natural measure for the quality of a prediction is the distance between the prediction and the observed response for a particular training example. Assuming N independent examples, we make use of the mean-squared-error (MSE) objective:

$$C_{MSE} = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{3.12}$$

where $\hat{y}_i$ denotes the prediction. C_{MSE} here represents some distance measure to be minimised between the observed responses y_i and the predictions from our model. We then wish to evaluate the gradient of the cost function with respect to

each of the weights and biases, working backwards from the cost function through the layers of the neural network. We define the linear component:

$$z_j^l = \sum_{k=1}^{d_l - 1} a_k^{l-1} w_{kj}^l + b_j^l, \tag{3.13}$$

with working gradient as follows:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}. \tag{3.14}$$

We can now derive the general expressions for the required gradients, starting with the terminal condition and propagating the errors backwards:

$$\begin{aligned} \delta_j^L &= \sum_{k=1}^{d_L = q} \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \sigma\prime(z_j^L). \end{aligned} \tag{3.15}$$

Iterating backwards from the terminal condition:

$$\begin{aligned} \delta_j^{l-1} &= \frac{\partial C}{\partial z_j^{l-1}} \\ &= \sum_{k=1}^{d_l} \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{z_j^{l-1}} \\ &= \sum_{k=1}^{d_l} \frac{\partial z_k^l}{z_j^{l-1}} \delta_k^l \\ &= \sum_{k=1}^{d_l} (w_{jk}^l \sigma\prime(z_j^{l-1})) \delta_k^l \\ &= \sum_{k=1}^{d_l} w_{jk}^l \delta_k^l \sigma\prime(z_j^{l-1}). \end{aligned} \tag{3.16}$$

We can then evaluate the gradient of the cost function with respect to the model parameters. For the biases of the model, we have:

$$\begin{aligned}\frac{\partial C}{\partial b_j^l} &= \frac{\partial C}{\partial z_k^l}\frac{\partial z_k^l}{\partial b_j^l} \\ &= \delta_j^l.\end{aligned} \tag{3.17}$$

Next, for the weights:

$$\begin{aligned}\frac{\partial C}{\partial w_{kj}^l} &= \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{kj}^l} \\ &= a_k^{l-1}\delta_j^l.\end{aligned} \tag{3.18}$$

The updating equations for the back-propagation procedure are defined by Equation 3.17 and 3.18 as well as the terminal condition in Equation 3.15.

3.5.2 Back propagation through time

Training RNNs is very similar to training FFNNs. In fact, there is a variant of the back-propagation algorithm for FFNNs that works for RNNs, called back-propagation through time (BPTT). As the name suggests, this is simply the back-propagation algorithm applied to the RNN backwards through time. BBTT works by applying the back-propagation algorithm to the unrolled RNN. An RNN is unrolled by expanding its computation graph over time, effectively "removing" the cyclic connections. This is done by capturing the state of the entire RNN (called a slice) at each time instant and treating it similar to how layers are treated in FFNNs. Since the unrolled RNN is similar to a FFNN with all elements from the input sequence in the input layer, the entire input sequence and output sequence are needed at the time of training.

BPTT begins similarly to back-propagation, calculating the forward phase first to determine the values of $\boldsymbol{o_t}$ (as defined in equation 3.10) and then back-propagating from $\boldsymbol{o_t}$ to $\boldsymbol{o_1}$ to determine the gradients of some error function with respect to the parameters. The parameters are replicated across slices in the unrolling and therefore gradients are calculated for each parameter at each time slice. The final gradients output by BPTT are calculated by taking the average of the individual, slice-dependent gradients. This ensures that the effects of the gradient update on the outputs for each time slice are roughly balanced.

3.6 Metrics

Although looking at the applicability of ANNs on a hydrological response was a main focus, comparing the different models used was also important. Conventional evaluation metrics used to compare the different models included the root mean squared error (RMSE), the coefficient of determination (R^2), and the Nash–Sutcliffe coefficient of Efficiency (ENS) (Nash and Sutcliffe, 1970). RMSE represents the sample standard deviation of the differences between predicted values and observed values (called residuals). Values range from 0 (perfect fit) to ∞ (no fit). R^2, known as the square of the sample correlation coefficient, ranges from 0 to 1 and describes the amount of observed variance explained by the model. ENS measures the model's ability to predict variables different from the mean and gives the proportion of the initial variance accounted for by the model. The metrics are summarised in Table 3.1, where y_i is actual stream flow, $\hat{y}_i$ is predicted stream flow, $\bar{y}_i$ is the mean of actual stream flow, $\bar{\hat{y}}_i$ is the mean of predicted stream flow, and n is the number of data points

Validation	Expression	Range
RMSE	$\sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$	$0 \leq RMSE \leq \infty$ Perfect:0
R	$\frac{\sum -i = 1^n (y_i - \bar{y}_i)(\hat{y}_i - \bar{\hat{y}}_i)}{\sqrt{\sum -i = 1^n (y_i - \bar{y}_i)^2 \sum_{i=1}^{n} (\hat{y}_i - \bar{\hat{y}}_i)^2}}$	$0 \leq R \leq 1$ Perfect:1
ENS	$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$	$-\infty < ENS \leq 1$ Perfect:1

TABLE 3.1: Showing different metrics used

Chapter 4

About The Data

It is important to give the meteorological aspects of the study area as to gain more understanding of the results found and for the bridging of different communities viz. statistical and hydrological. Before exploring the data, some background on the study area is provided. This chapter is broadly divided into three subsections viz. study area, data organisation, and model formulation.

4.1 Study Area

The study area forms part of the Langrivier catchment area in the Jonkershoek Valley on the boundary of the catchments of the Eerste, Berg and Sonderend rivers[1].

4.1.1 Background

The farm Jonkershoek was named after Jan Andriessen to whom it was granted by Governor Simon van der Stel in 1662. Andriessen had been a midshipman (jonkheer) in the service of the Dutch East India Company and was known as Jan de Jonkheer (Jonker). Jonkershoek and a number of adjoining properties constitute the Jonkershoek State Forest. A small section along the Eerste River is held on a 99-year lease, signed in 1933, from the Municipality of Stellenbosch. A network of rain-gauges was installed, soil and vegetation surveys were completed and work commenced on the building of stream-gauging weirs.

4.1.2 Location

The Jonkershoek valley in the Western Cape Province is a long narrow valley which is situated between the Stellenbosch Mountain (SW) and Jonkershoek Mountains

[1]This section is largely taken from Scott et al. (2000)

(NE) and is enclosed by the Dwarsberg in the south-east (S33°57′ ;E18°55′). All the streams in the valley form the tributaries of the Eerste River that flows through the town of Stellenbosch. The catchment used in this study lies on the south west facing slope of the Jonkershoek mountain with an altitude of 833*m* above sea level, shown in Figure 4.1.

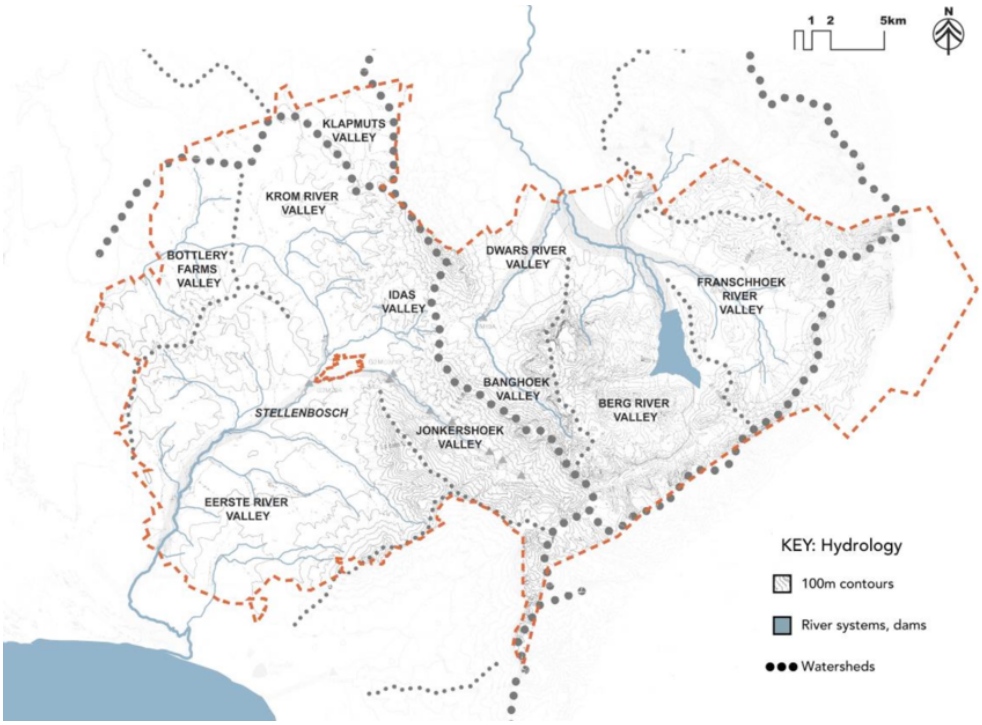

FIGURE 4.1: Topographical map of the Jonkershoek catchment area [Source:Todeschini and Jansen (2016)]

4.1.3 Climatic Characteristics

The climate is of the humid Mediterranean type with warm, dry summers (with prevailing south easterly winds) and cool wet winters with frequent cyclonic rains. The mountainous topography has a significant effect on the rainfall, which is of the highest in South Africa. Snow is not unusual on the higher peaks during the winter months. The average daily temperature for range from 15.6°C in July to 26.3°C in February. The region is coldest during July when the temperature drops to 6.6°C on average during the night (*Stellenbosch climate* 2017).

4.1.4 Geological and Soil Characteristics

The geology comprises of sandstone and quartzite with intermittent thin shale bands of Table Mountain Group (Lower Paleozoic Cape Supergroup)- mostly in the upper

slopes and cliffs of the scarp. These are underlain by Cambrian Cape Granite, which is found mostly on the lower slopes and valley floor. There are several talus screes of granite boulders plus sandstone debris. The Jonkershoek valley is closed at the SE-end by the transverse Dwarsberg block fault.

The soils are fairly complex and of mixed origin, derived primarily of mixed colluvial material from the above geological formations. These forms are acid sandy loams (pH 4.1 to 5.7), low in organic matter and in phosphorous. The soils have a low bulk density, high infiltration capacity and are well-drained. Soil depths range from roughly 1m to 2m, but are underlain by unconsolidated or decomposed material that allows free drainage of water as well as exploration by tree roots.

4.2 Data Organisation

All data was provided by the South African Environmental Observation Network (SAEON).

4.2.1 Collection and Instrumentation

Gauging of the Langrivier weir begun in September 1940 using the Kent[2] water level recorder. The Kent method was then replace by the Belfort[3] water level recorder in March 1961. The South African Environmental Observation Network (SAEON) begun the automated water level monitoring in January 2011, using the Orpheus Mini[4] water level recorder. The stream flow data consist of hourly readings of the water volume flowing over a v-notch weir. Weirs were currently calibrated monthly. Calibrations involved checking the sensor reading against a manual reading of water level in the weir notch. Deviations (sensor drift or disturbance) within $\pm$3mm are tolerated, otherwise the sensor is reset to the correct value. Since SAEON automated water level monitoring in 2011, they have recorded calibration readings and made them available with the raw data so users can see when the sensors have been reset. All stream flow data is recorded in cubic meters per hour.

The rainfall data, measured in millimeters, for two rain gauges in the Langrivier catchment was used. These are the only two stations in Langrivier that have a

[2]More information can be found at https://www.waterworksmuseum.org.uk
[3]More information can be found at http://www.belfort-inst.com/Model_5-FW-1.htm
[4]More information can be found at https://www.ott.com/

historical record predating SAEON and are still being monitored. The rainfall data gauging begun in January 1944 using the Davis[5] tipping bucket rain gauge.

4.2.2 Pre-Analyses

In analysing the data, it is evident that the Langrivier is classified as a perennial river having continuous flow all year round. It was found that the Langrivier revealed a maximum value of 58863.6 m^3/h. This value may be somewhat misleading as this seams to be a cap, which was observed several times. It may, therefore, be more beneficial to look at the summary statistics of daily data instead. Missing stream flow data exists from October 2008 to August 2011. Rainfall data is missing from February 2008 to February 2009. The temporal resolution of records changed over the lifespan of the gauging stations and was different for each station, as is shown in Table 4.1. The rainfall data from July 2011 until February 2018 was provided in a format which was difficult to interpret. The raw data was provided which gave event based rainfall measurements of $0.2mm$ every time the Davis bucket tipped (refer to Appendix A). For the purpose of this project, it was decided that this data would not be used in further evaluation as results and analyses can be compiled based on the previous years.

	Dates	**Resolution**
Rainfall	01/1944 - 02/2008 02/2008 - 02/2009 02/2009 - 09/2012 07/2011 - 02/2018	Daily(8B) &Weekly(14B) Weekly(8B & 14B) Event-based: inappropriate format
Stream Flow	09/1940 - 10/2008 10/2008 - 08/2011 08/2011 - 03/2018	Hourly Missing Data Half-Hourly

TABLE 4.1: Details of data

It was seen that stream flow was much lower in the summer, with the lowest observed cumulative yearly stream flow at $2062013m^2$, during 2003. During this year, cumulative rainfall was among the lowest 10% across the data set. When plotting the mean monthly stream flow (Figure 4.2) for the period of the data set the flow shows to peak in the winter months, falling close to zero in the summer. The black error bars on this plot show the 95% confidence intervals for each month.

[5]See Appendix A

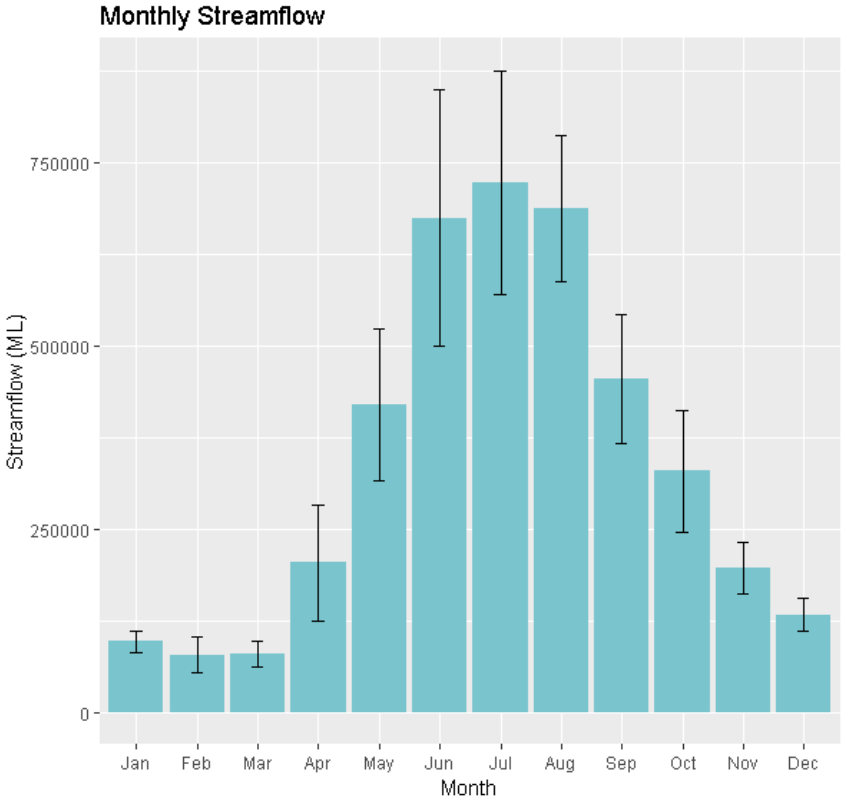

FIGURE 4.2: Bar plot of mean monthly stream flow from 1940 to 2008

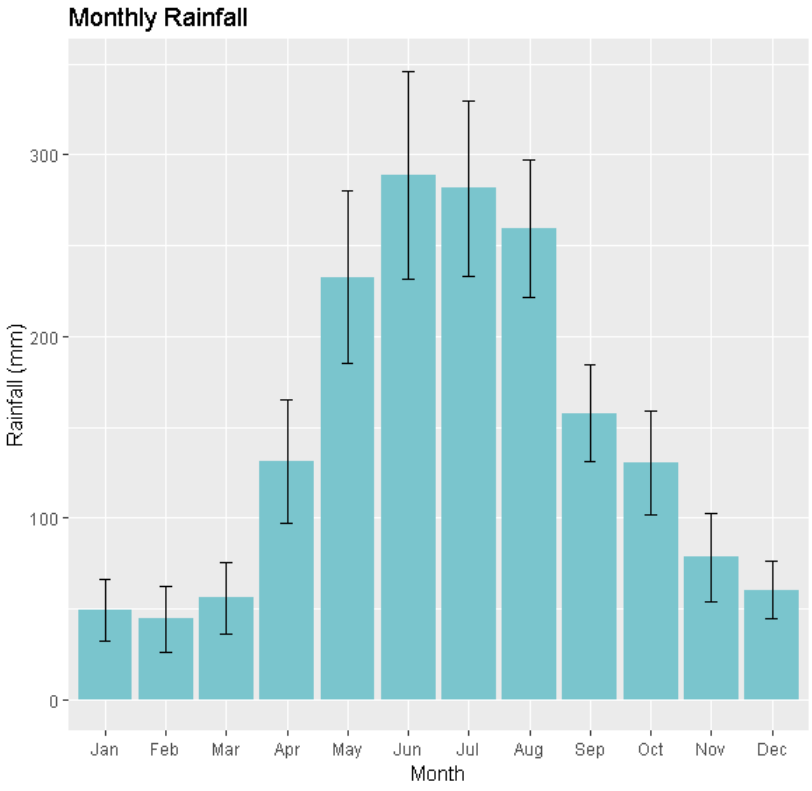

FIGURE 4.3: Bar plot of mean monthly rainfall from 1940 to 2008

Rainfall and stream flow are shown to move together over the period of a year. A full time series of both stream flow and rainfall show this over the whole period. For the purpose of this project, a particular focus will be placed on exploring the seasonality. A first step to exploring this, would be through visualisation techniques showing the cycle of the time series (Figure 4.5) as well as the autocorrelation function plot (Figure 4.6). The cyclic activity of stream flow is easily seen in Figure 4.5 with a seasonal cycle spanning of a single year. Time series of daily, weekly, and monthly discharges belong to the characteristic pattern of time series in which the process level is assumed to vary cyclically over time. For this reason we have decided to use harmonic regression or regression with trigonometric (Fourier) terms to model the seasonal variations in rainfall and runoff separately. Here, a brief theoretical description of the model used will be presented.

Given a time series of n observations, the Fourier representation is the set of q orthogonal trigonometric functions shown below:

$$y_t = \sum_{i=1}^{q} (a_i cos2\pi f_i t + b_i sin2\pi f_i t) + e_t, \tag{4.1}$$

estimated by:

$$\hat{y_t} = \sum_{i=1}^{q} (a_i cost2\pi f_i t + b_i sin2\pi f_i t), \tag{4.2}$$

where $q = n/2$, $a_i = \frac{2}{n} \sum_{t=1}^{n} y_t cos2\pi f_i t$, $b_i = \frac{2}{n} \sum_{t=1}^{n} y_t sin2\pi f_i t$

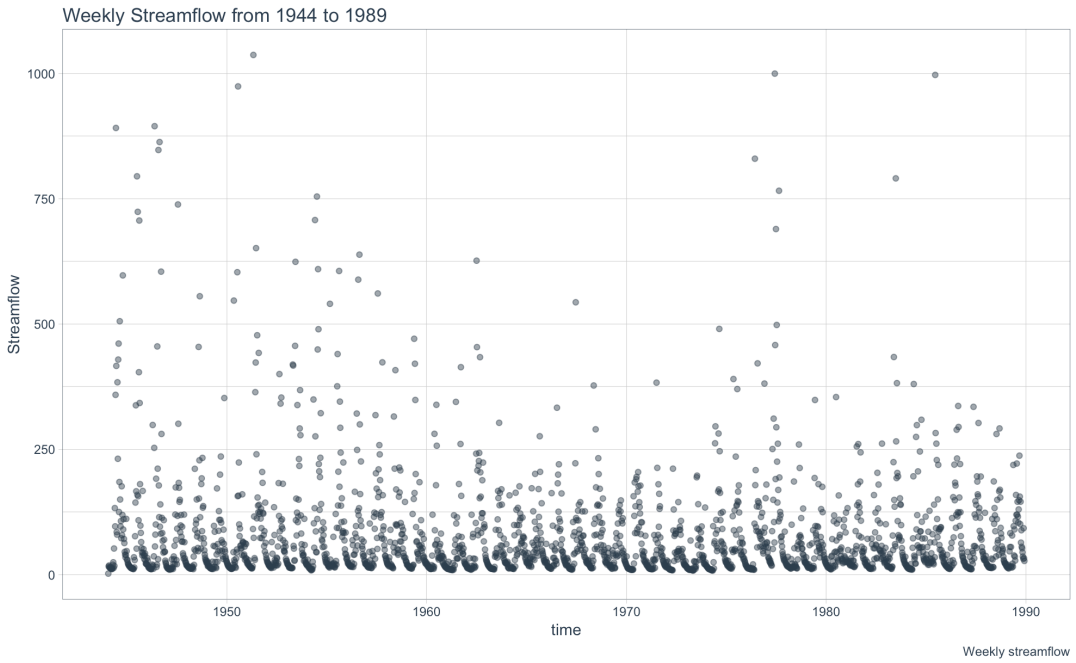

FIGURE 4.4: Showing the full time series of the stream flow data

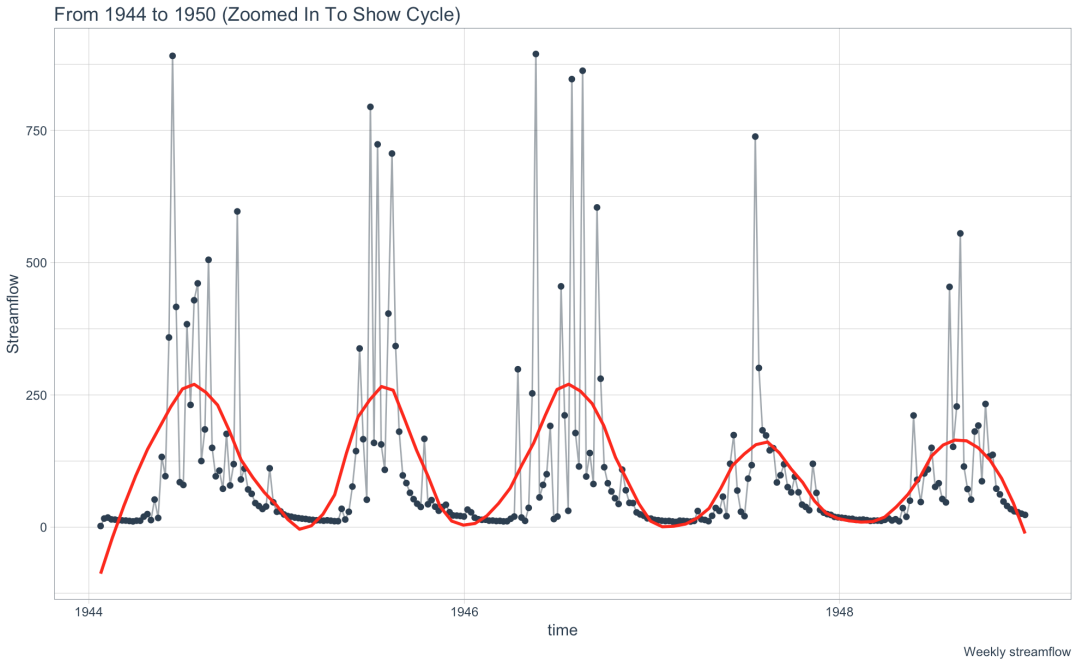

FIGURE 4.5: Showing the full time series of the stream flow data

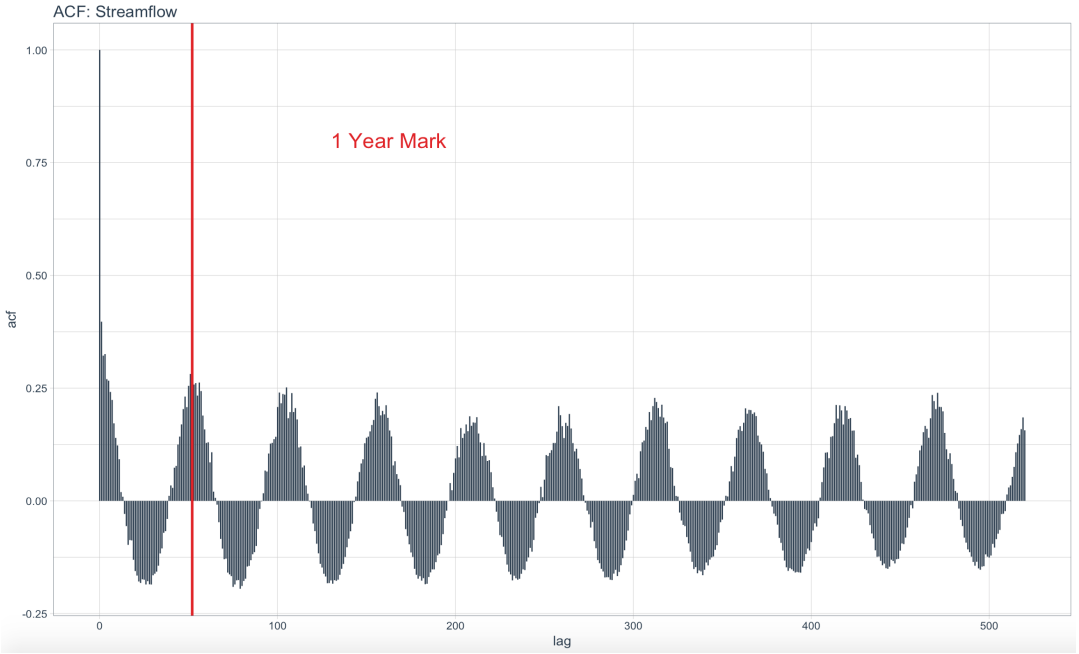

FIGURE 4.6: Autocorrelation function plot of stream flow

4.2.3 Pre-Processing and Normalisation

As with any real world data, some sort of cleaning needed to be done. A series of preprocessing took place to get the data into useful format. Problems included uneven time intervals for weekly rainfall data, as well as missing values spread throughout. Some dates were repeated and others were as extreme as two years apart. A first step to the solution was to find subsets of the data where only minor problems of few missing dates existed. It was also necessary that these subsets had both stream flow and rainfall corresponding data available. The first subset was taken from 1944 to 1989 with the second from 1991 to 2008. Data values were then assigned to equally spaced time intervals, which resulted in a loss of a few observations. The number of lost observations was negligent relative to the size of the dataset. Both sets of rainfall and stream flow data consisted of missing values, thus imputation techniques which deal with data exhibiting strong seasonality were used. The `na.seasplit` function from the `R` package `imputeTS` (Moritz and Bartz-Beielstein, 2017) was used. The imputation algorithm splits the data sets into seasonal slices and then perform interpolation on each of these slices. The algorithm then put each of the imputed slices back together maintaining the sequence of observations to produce the complete dataset. In order to build the model on weekly data, the stream flow had to be aggregated from hourly time steps. The weekly

stream flow data was taken as a sum of all hourly data for that week. Event based rainfall was also provided for a different, yet close by gauge. This event based rainfall consisted of the days rainfall occurred and the total rainfall for that day as the data value. The missing dates in this dataset simply meant that no rainfall occurred on those days.

Normalization or the standardisation of both input and output data is an important procedure in ANN modeling due to the influence it has on the convergence of the system. The activation or transfer function squashes the outputs from the network to be constrained in the range 0 to 1. For proper convergence, the scaling of the data should match the range of the activation function. The observed data, being outside this bounded range has to therefore be normalised or rescaled such that it falls within this bounded range. This will avoid the saturation effect caused by the activation function during the analysis Devia, Pa, and Sa (2015). Theoretically, is isn't strictly necessary to normalise the data beforehand. The reason is that any rescaling of an input vector can be effectively undone by changing the corresponding weights and biases, leaving the exact same outputs as before. However, there are a variety of practical reasons why standardising the inputs can make training faster and reduce the chances of getting stuck in local optima. Also, weight decay and Bayesian estimation can be done more conveniently with standardized inputs.

Our data has, therefore, been normalised before the training begun, then denormalising at the output nodes. This was done using the following:

$$X_{t_{NORM}} = \frac{X_t - \bar{X}_t}{S}, \tag{4.3}$$

where $\bar{X}_t$ is the sample mean and S is the sample standard deviation.

Chapter 5

Results and Discussion

This chapter deals with the application and performance evaluation of the different artificial neural network models, developed for simulation of stream flow for the Jonkershoek catchment in the Western Cape. The different architectures were compared on both weekly time steps (1941-1991) as well as daily time steps (1941-2008), due to reasons described in Chapter 4 above.

5.1 On The Weekly Data

5.1.1 Performance of Feed-Forward neural networks(FFNN)

Data was split in the ratio 70:15:15 corresponding to a training set, validation set and testing set respectively. Since our data set consisted of 2394 observations we decided that 70% of the data would be reserved for the purposes of training the model. We found that this proportion allowed the model to capture the patterns in the data whilst leaving enough data to allow for reliable testing results.

The simplest FFNN model was first fitted, using current rainfall as the only input with current stream flow as the output. In order to efficiently search the different hyperparamters of the models, the `h2o.grid` function from the `h2o` (LeDell et al., 2018) package was used. This function searches through several user defined hyperparameters of the deep learning model, reporting different performance metrics of each combination of hyperparameters. The number of hidden layers was restricted to be at most two, as it is reported that ANN with single hidden layer having sufficient number of neurons can approximate any complex non-linear function to an acceptable degree. Upon analyses of the grid search, a communality in the best models with respect to the activation functions, was noticed. These were seen to be the hyperbolic tangent functions (Section 3.3.2), the ReLU (Section 3.3.3), and

the maxout functions, all with drop out. We therefore decided to perform a further grid search, only varying the activation functions between the three aforementioned functions whilst introducing new values for the other parameters. This was in an effort to decrease computational time. The results of the final grid search, summarized in Table 5.1, indicate a network architecture with a *Rectfier With Dropout* activation function and a single hidden layer containing 3 nodes.

	Activation Function	**No. Hidden nodes**	**l1**	**RMSE**
1	Rectifier With Dropout	[3]	10^{-7}	0.9621951
2	Tanh With Dropout	[3]	0.001	0.9629804
3	Max With Dropout	[2,3]	0.001	0.9634146
4	Max With Dropout	[3]	0.001	0.9635529
5	Tanh With Dropout	[3,3]	0.001	0.9635612

TABLE 5.1: Best five feed forward neural network models based on validation RMSE, with current rainfall as the only input

The RMSE was observed to be marginally different between the best five models, with a range of 0.002. Furthermore, it was seen that the simpler models performed better. We proceeded using the best model to predict stream flow using the testing data. Running the testing dataset through our selected model we obtain a RMSE of 0.862. This value is lower (better) than that of the validation data and the training data. A possible explanation for this is that the testing data did not have to learn to patterns that the training data had to optimize. A plot of the predicted stream flow (red) against the actual stream flow (black) is presented in Figure 5.1. In analysing the plot, it is clear that the model does not perform well in predicting the stream flow as the strong seasonality is not accounted for by the predicted values. It is worth stating that the RMSE has little meaning as of yet as we do not have anything to compare it to. This RMSE will thus be used as a base, above which is considered a bad fit.

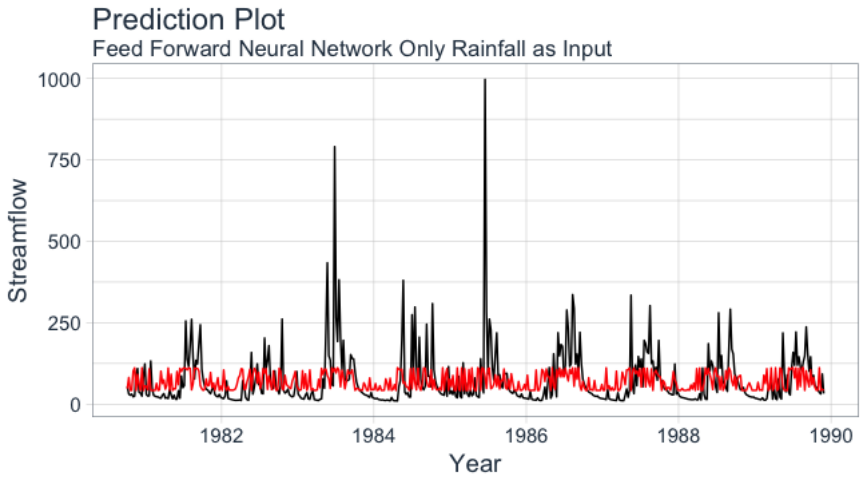

FIGURE 5.1: Prediction plot of the FFNN using current rainfall as the only input

As stream flow is a function of current rainfall, historical rainfall and historical stream flow, we first decided to add lagged rainfall variables to the model to see the impact of historical rainfall on stream flow prediction. The same procedure was followed as in the case of the first model above, except for the selection of the number of lagged rainfall variables to include in the model. By trail and error we found that including five rainfall lags (previous five weeks rainfall data) as well as current rainfall as inputs to the model resulted in a significant decrease in RMSE. A greater number of lagged rainfall variables resulted in marginal decreases in RMSE, carrying the cost of time to create them. The sophisticated grid search yielded the best model using a ReLU activation function with three nodes in both the first and second hidden layers. This model produced RMSE values of 0.587 for both the validation and testing dataset. An improvement of 0.375 on the testing dataset may indicate that the lagged rainfall is of great importance in predicting current stream flow. This was explored through means of variable importance using the Gedeon method (Gedeon, 1997), which considers the weights connecting the input features to the first two hidden layers. The prediction plot of the new model, shown in Figure 5.2, displays the improvement on predicting the stream flow. Now, the model is able to effectively pick up the seasonal patterns of the data. The variable importance plot shows that the lagged variables (coded as r2 is rainfall two weeks previous) are of relatively more importance than current rainfall. This observation is in line with our expectations on the storage of the catchment area.

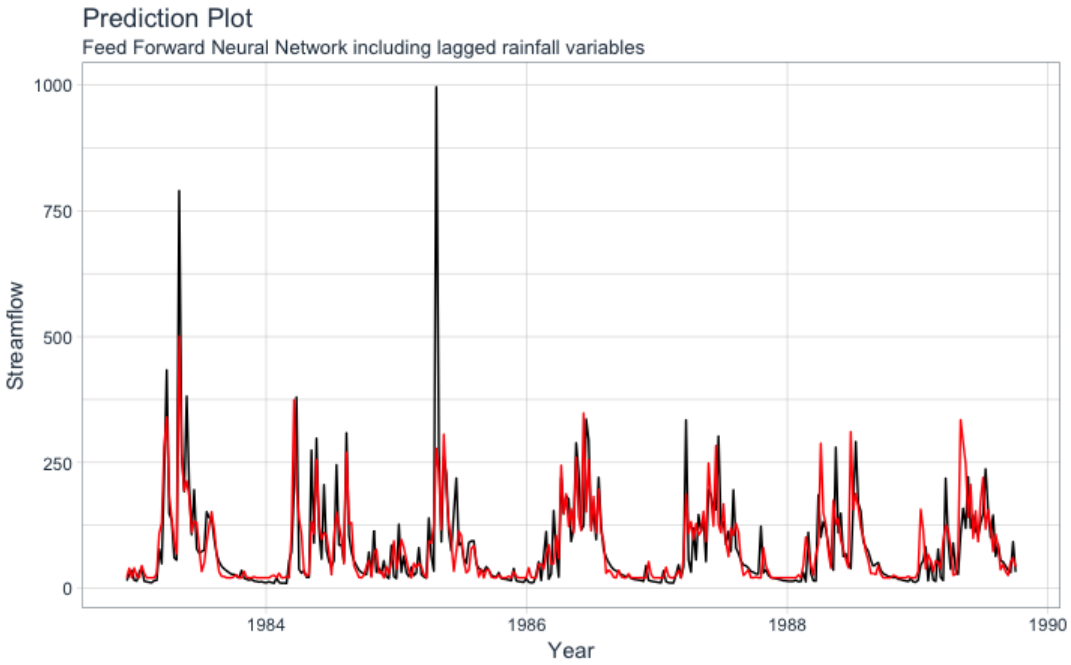

FIGURE 5.2: Prediction plot of the feed forward neural network including five lagged rainfall variables

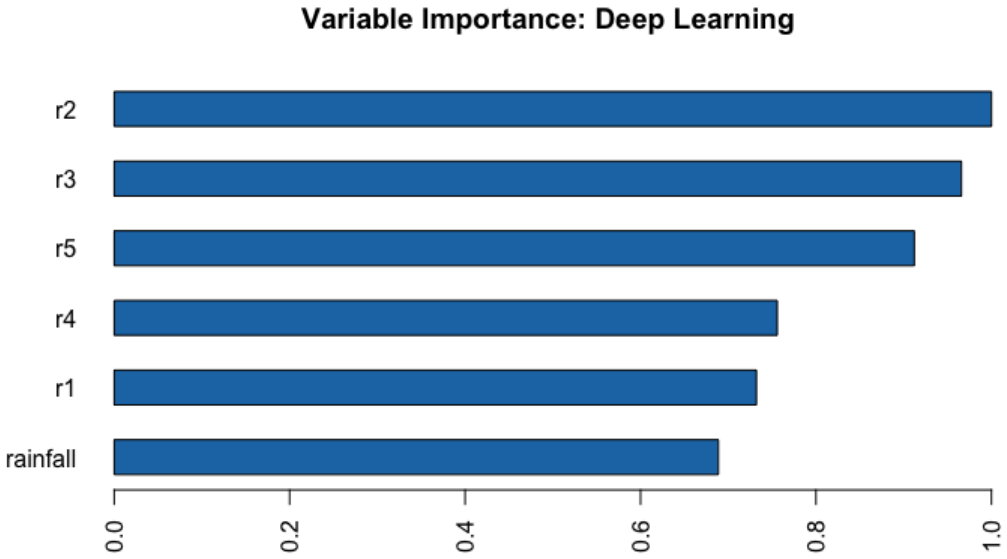

FIGURE 5.3: Variable importance plot of FFNN with current and lagged rainfall as inputs

To see if we could further improve this models performance, we assessed the addition of lagged stream flow variables as input. When assessing the autocorrelation function of the time series data of stream flow, it was seen that the lagged stream flows are correlated with the current stream flow, as was expected. This model will

build on the model consisting of five lagged rainfall variables. We used the same procedure to determine the number of stream flow lags to include. A better model was produced with the addition of eight lagged stream flow variables. Performing a grid search on this configuration yielded an optimized model using a hyperbolic tangent activation function with two nodes in the first hidden layer and three nodes in the second. This model produced an RMSE value of 0.577 for both the validation and testing dataset. This is a further improvement of 0.1 from the previous model, which is an indication that current stream flow is not only dependent on historical rainfall but also historical stream flow. Again, this was assessed using measures of variable importance described above. The prediction plot looks very similar to the one above, however we know that the predictive accuracy is slightly better from the lower RMSE. The variable importance plot in Figure 5.5, shows the high relative importance of lagged stream flow at lag two (two weeks previous), thus justifying the inclusion of lagged stream flow in the model. An interesting result is seen when stream flow eight weeks ago is of higher importance than current as well the the first two lagged rainfall variables.

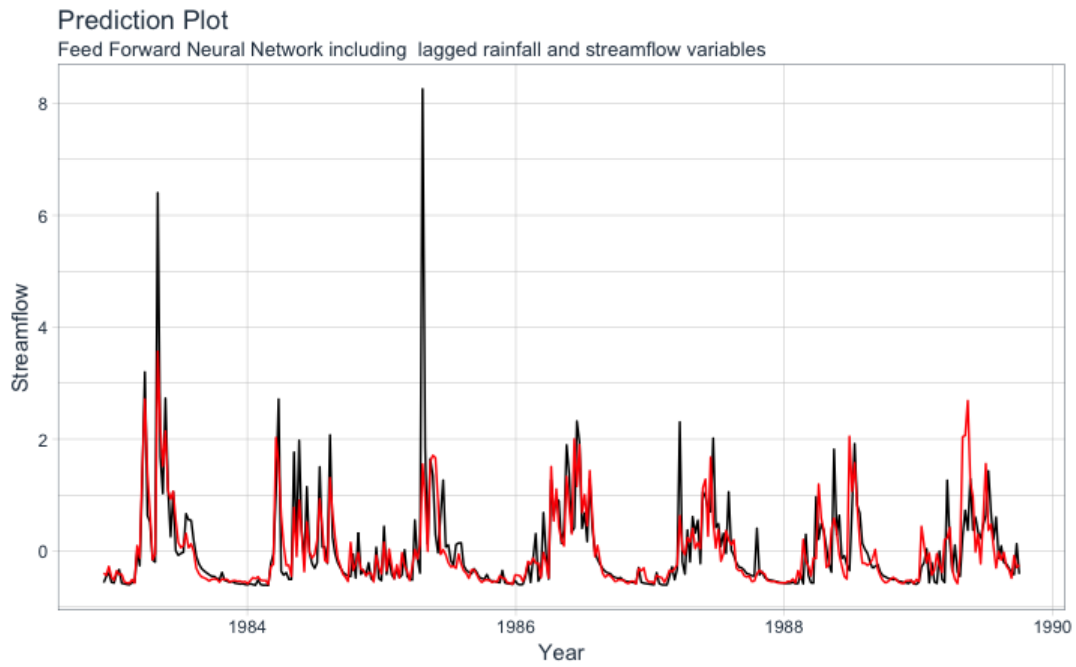

FIGURE 5.4: Prediction plot of the FFNN on weekly data with both lagged rainfall and lagged stream flow as inputs

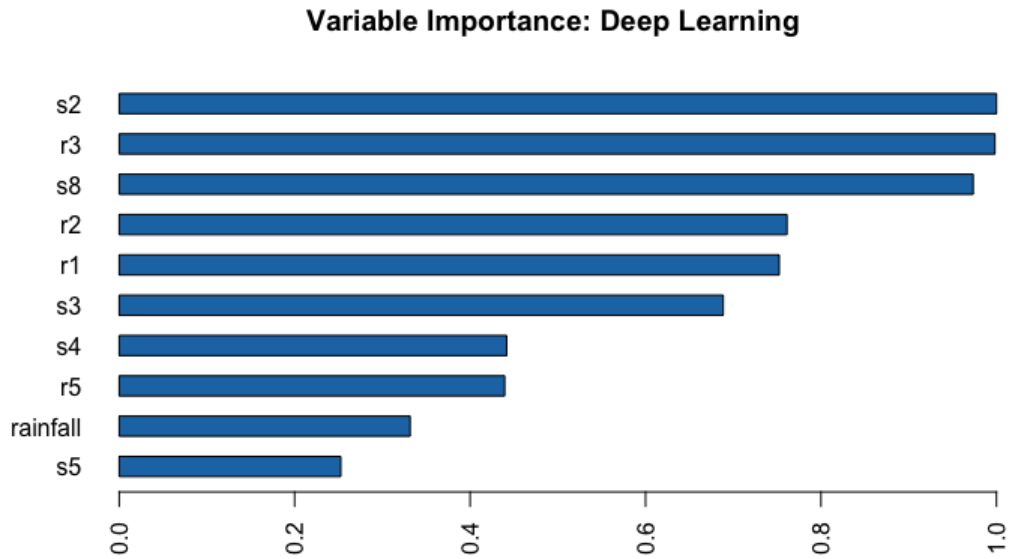

FIGURE 5.5: Variable importance plot of the updated

It is clear that the lagged variables play an important role in the prediction of stream flow, with these variables being relatively more important than the rainfall at the time of the observed stream flow, according to Gedeons measure. This, therefore, sets up the notion for the use of RNNs on the proposed problem.

5.1.2 Performance of simple Recurrent neural network(RNN)

The RNN and LSTM models were developed using the `keras` package (Allaire and Chollet, 2018), which connects to the R `TensorFlow` backend. Through trial and error, a total of 30 epochs showed to give best results without over-fitting the data.

The validation method used to implement our RNNs was different to that of the FFNN. The implementation of validation methods required development of a back testing strategy which involves splitting the data into slices of uninterrupted overlapping shifted windows that can be tested for strategies on both current and past observations. Time series, as preserved in RNNs, is a bit different than non-sequential data when it comes to cross validation. Specifically, the time dependency on previous time samples must be preserved when developing a sampling plan. The strategy is comparable to that which is implemented in financial regression problems. A recent development is the `rsample` package (Kuhn and Wickham, 2017),

which makes cross validation sampling plans very easy to implement. Furthermore, the `rsample` package has backtesting covered.

Each sliced was split into training and testing datasets in the ratio of approximately 80:20. The size of each slice is governed by the size of the training and testing datasets as well as the skip between windows (the number of time steps which the window is shifted by). Given these governing variables the we found, through experimentation, the appropriate slice size consists of 38 years of training data and 7.5 years of testing data, with a skip between windows of 1 month, ultimately dividing the dataset into 6 equal slices. We found that when the training and testing datasets were too small, the model was not able to fully pick up the patterns in the data. We first used a back testing strategy with a nine smaller slices of 12 years each with a skip span of four years. The division of the total data set into slices is shown in Figure 5.6, and each slice is clearly presented in Figure 5.7 The predictions using a different back testing strategy with a smaller testing data size and a skip step of four years produced an average RMSE of 0.771 as shown in Figure 5.8, with some slices performing significantly better than others. The prediction plot for the middle slice (slice 5) is shown in Figure 5.9. This is a higher RMSE than the best model produced in the FFNN, however it can be directly attributed to the size of each slice. To be able to compare the model effectively and fairly to the FFNN, a larger training data set was used. A clear visualisation of the newer (larger sized slices) back testing strategy is presented in Figure 5.10. For the RNN, we will select and visualise the split of the middle slice (Figure 5.11).

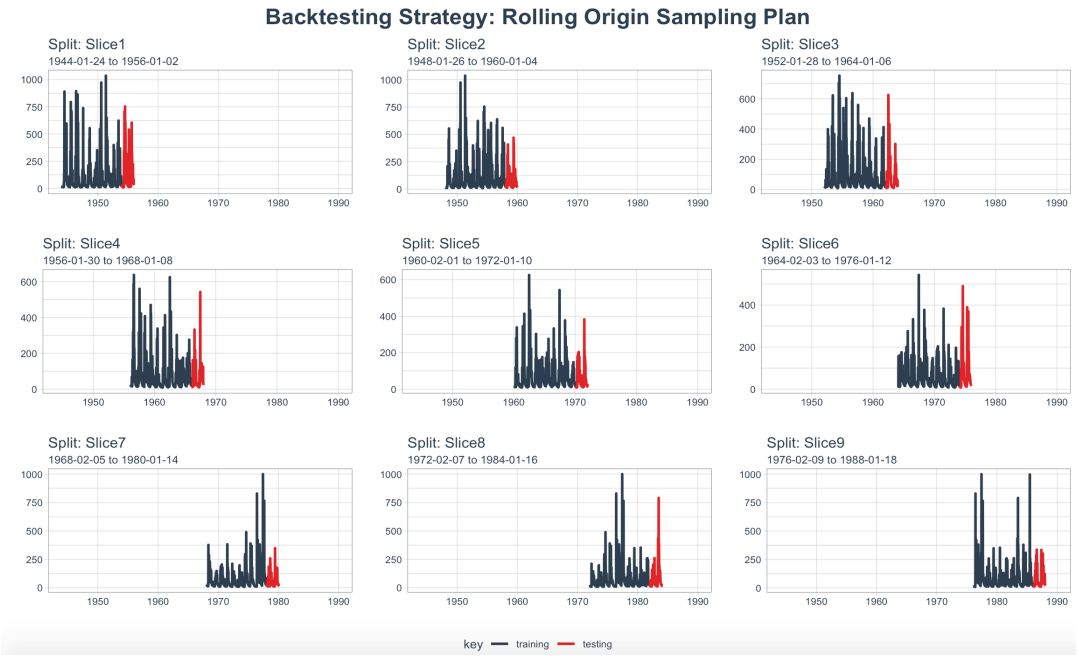

FIGURE 5.6: Back testing strategy for the RNN for slices of 12 years each

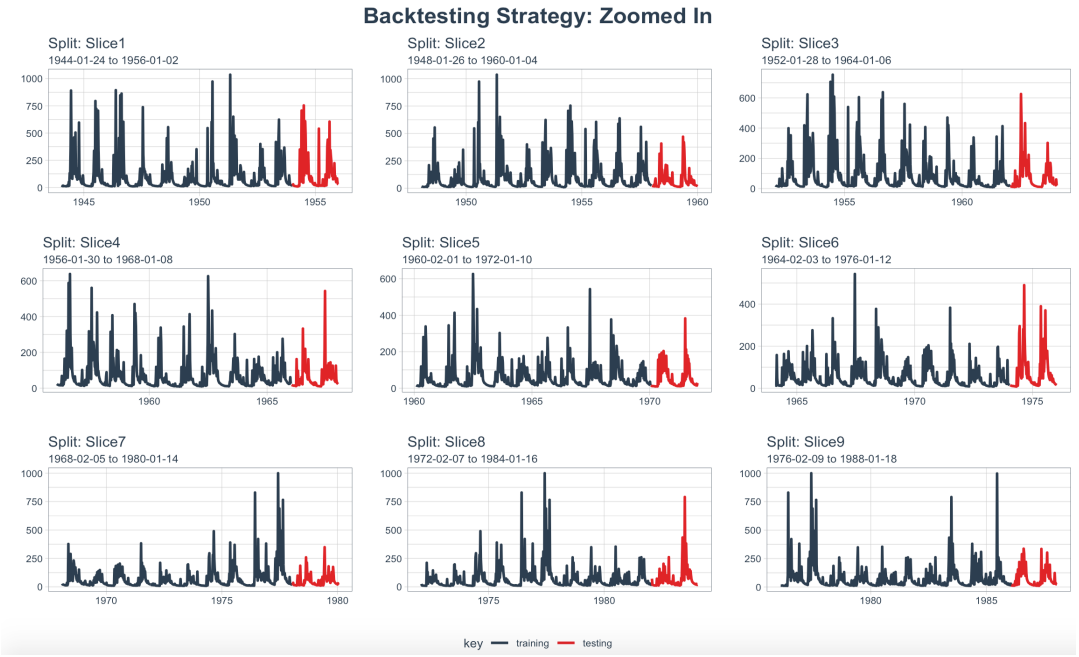

FIGURE 5.7: Back testing strategy for the RNN for slices of 12 years each zoomed in for clarity

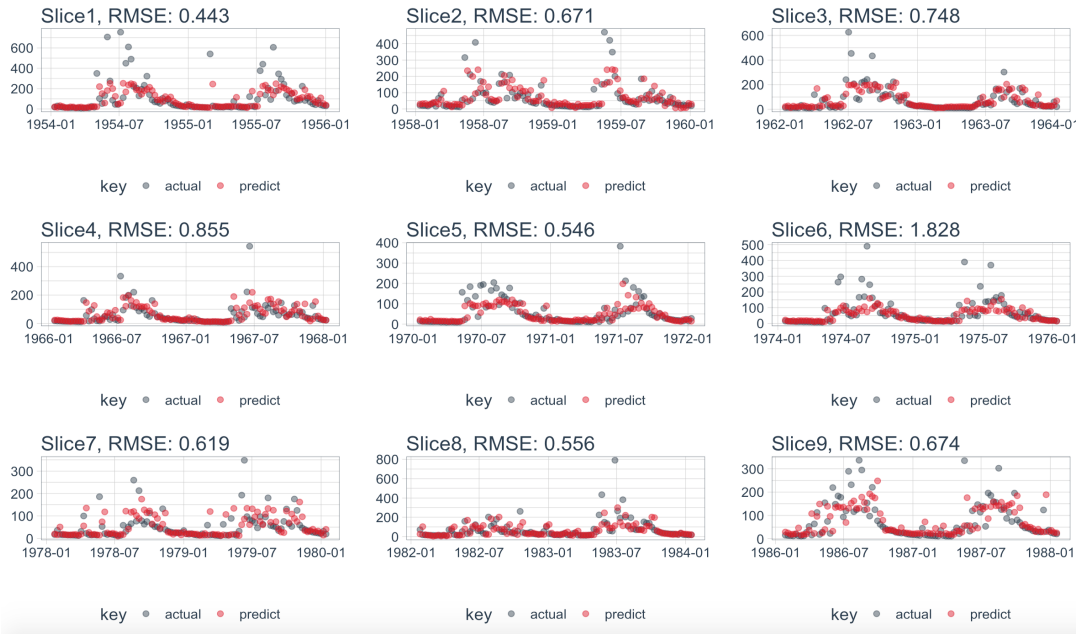

FIGURE 5.8: Prediction plots of each slice of RNN model

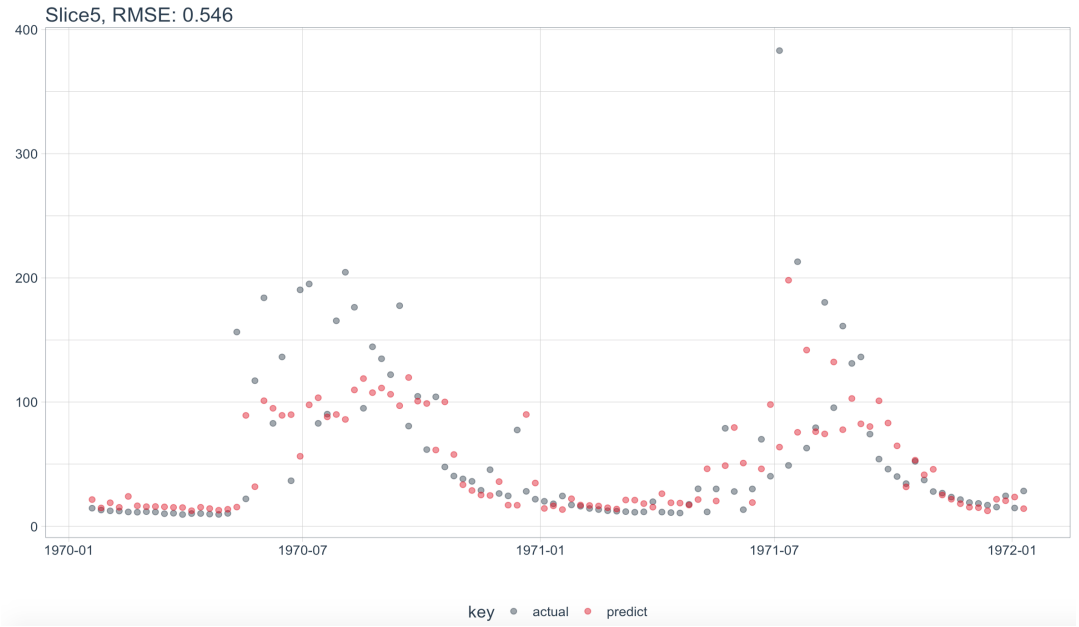

FIGURE 5.9: A better look at the middle slice prediction plot

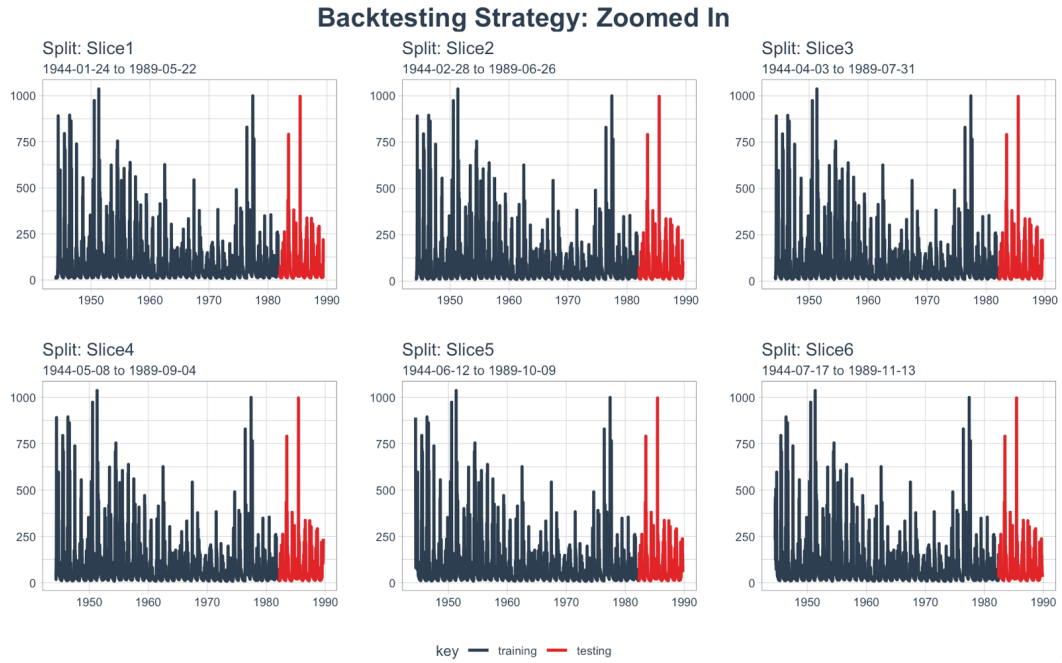

FIGURE 5.10: Showing the training and testing splits of the middle slice of the larger-sized slice back testing strategy

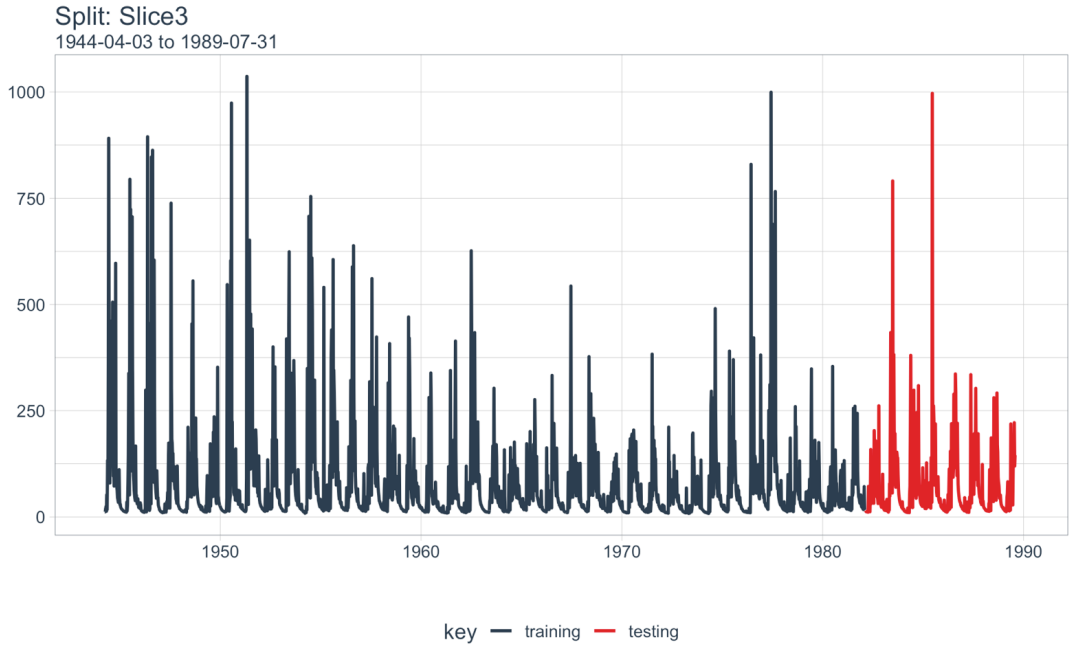

FIGURE 5.11: Showing the training and testing splits of the middle slice

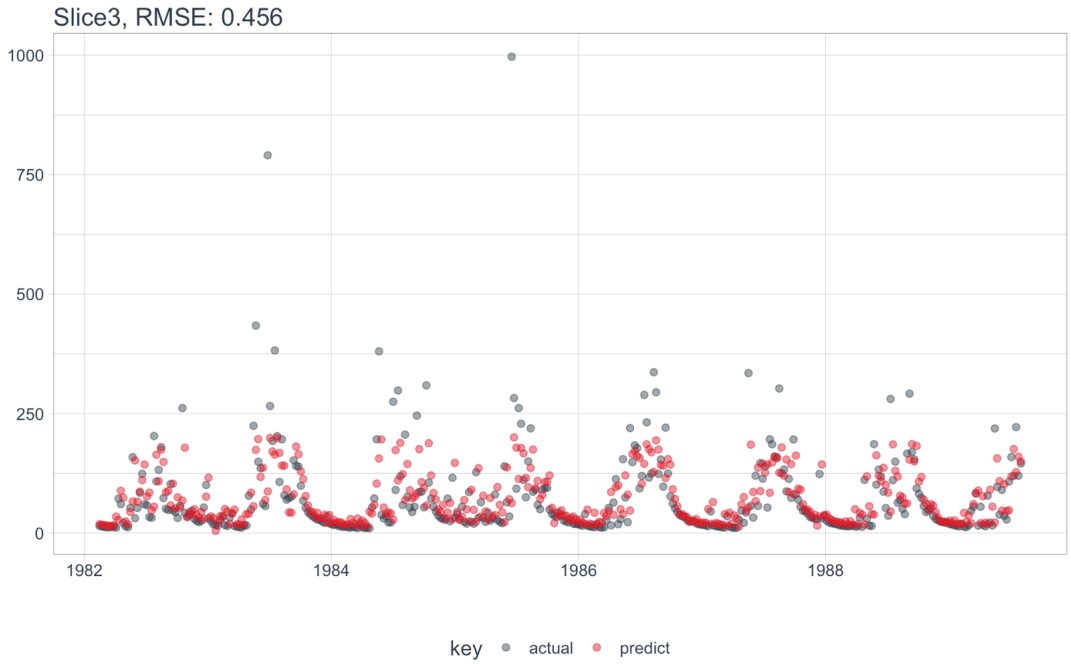

FIGURE 5.12: A clear view of the middle slice prediction plot using the larger-sized slice back testing strategy for the RNN

The RNN model was developed using the `keras` package (Allaire and Chollet, 2018), which connects to the R `TensorFlow` backend. Through trial and error, a total of 30 epochs showed to give best results without over-fitting the data. In assessing the prediction of all slices of larger sizes, the mean RMSE for all the slices was 0.435. This is a slight improvement over the best model of the FFNN architecture. This is a great improvement over using smaller sized slices in the back testing strategy. It is thus clear that the size of the training data plays a significant role in model performance. Figure 5.12 shows the predictions against the observed values of stream flow for the RNN, proving to sufficiently predict points very close to their observed value. The effective seasonality was picked up by this model. The major advantage seen over the FFNN was that the need to create and determine the number of lagged variable became obsolete as this process was directly done in the model architecture of the RNN. We now want to assess whether this model can be further improved by means of the LSTM architecture.

5.1.3 LSTM

The same back testing strategies were used to assess the performance of the LSTM model. On the first (smaller sized slices) back testing strategy, the model was seen as improvement over the RNN with the same back testing strategy. Again, we show

all the slices (Figure 5.13), zooming in on the fifth slice for ease of viewing (Figure 5.14).

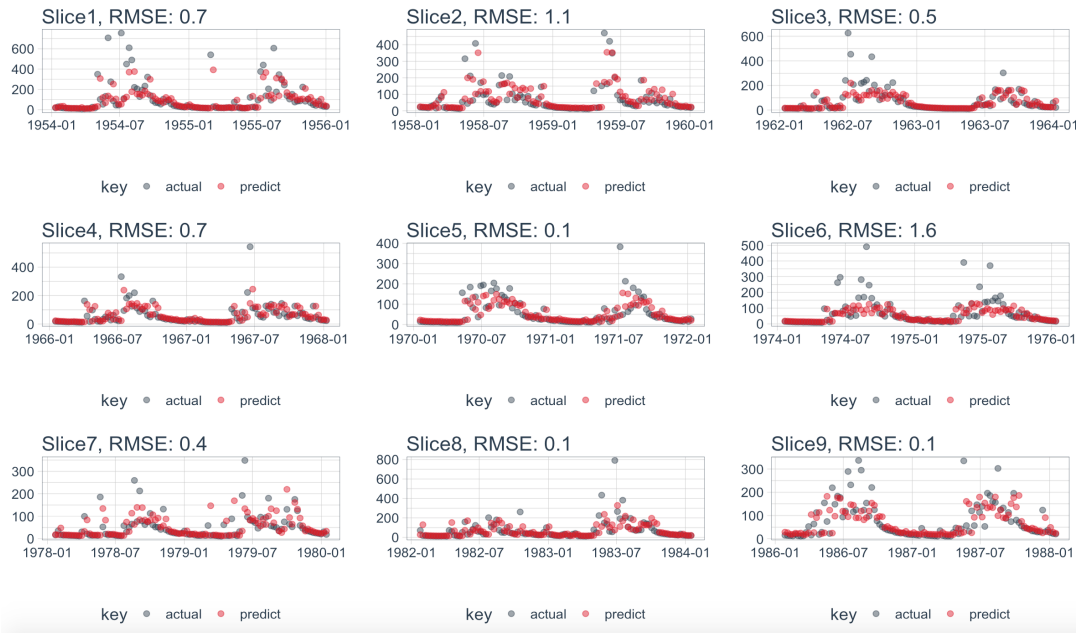

FIGURE 5.13: Showing the predictions of the LSTM of each slice using the smaller-sized slice back testing strategy

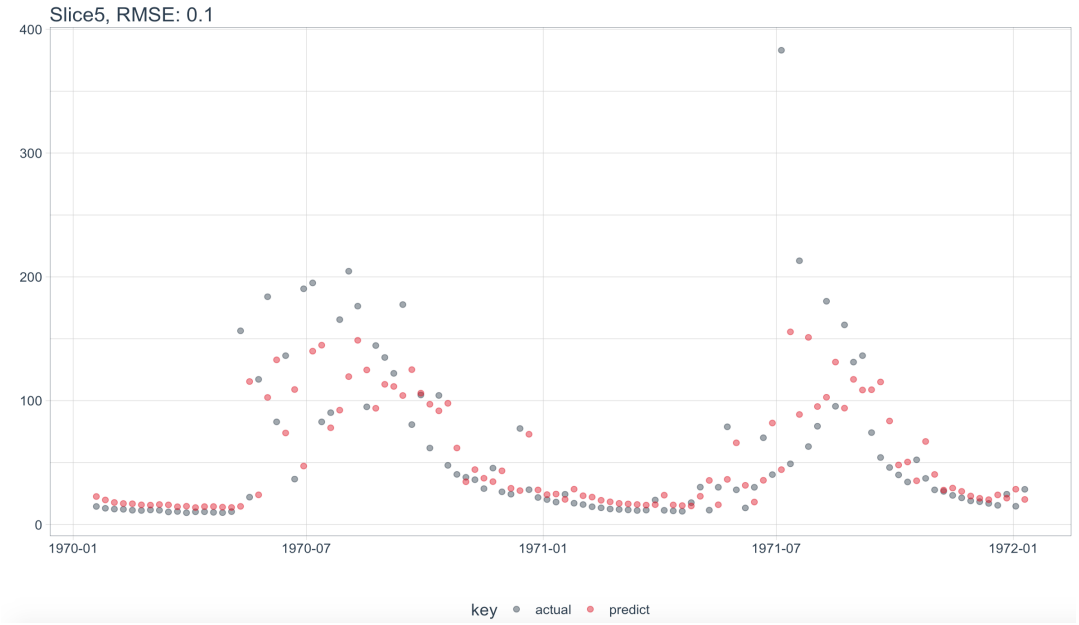

FIGURE 5.14: A clear view of the middle slice prediction plot using the smaller-sized slice back testing strategy for the LSTM

An average RMSE across all the slices was calculated to equal 0.589. This is an improvement of 0.182 from the traditional RNN when using the same back testing strategy. Furthermore, when assessing the fifth slice, the RMSE of this slice for the

LSTM was much lower, at a value of 0.1 as opposed to 0.546. There were, however, some slices that were greater than the RNN. This RMSE for the LSTM is still, however, higher than that found in the best model of the FFNN architecture. As explained above, this could be directly attributed to the size differences in the training data sets. We, therefore, used the second back testing strategy described above which splits the data set into larger slices skipping only one month between each slice. Figure 5.15 shows the third or middle slice prediction plot of this larger sized slice back testing strategy. Predicted points are seen to closely follow those observed, with some exactly the same. When training on a larger sized data set for each slice, the LSTM was seen to perform very well with an average RMSE across all slices of 0.266. The predictions on the first slice produced an RMSE of 0.023. Between the different back testing strategies used, a decrease of 0.323 was seen when training on a larger data set. Again, the size of the training data and the back testing windows made a significant difference in the performance of the model. An important observation is made when assessing the sixth slice of the smaller sized slice back testing strategies. For both the RNN and the LSTM, the models performed relatively poorly on this slice.

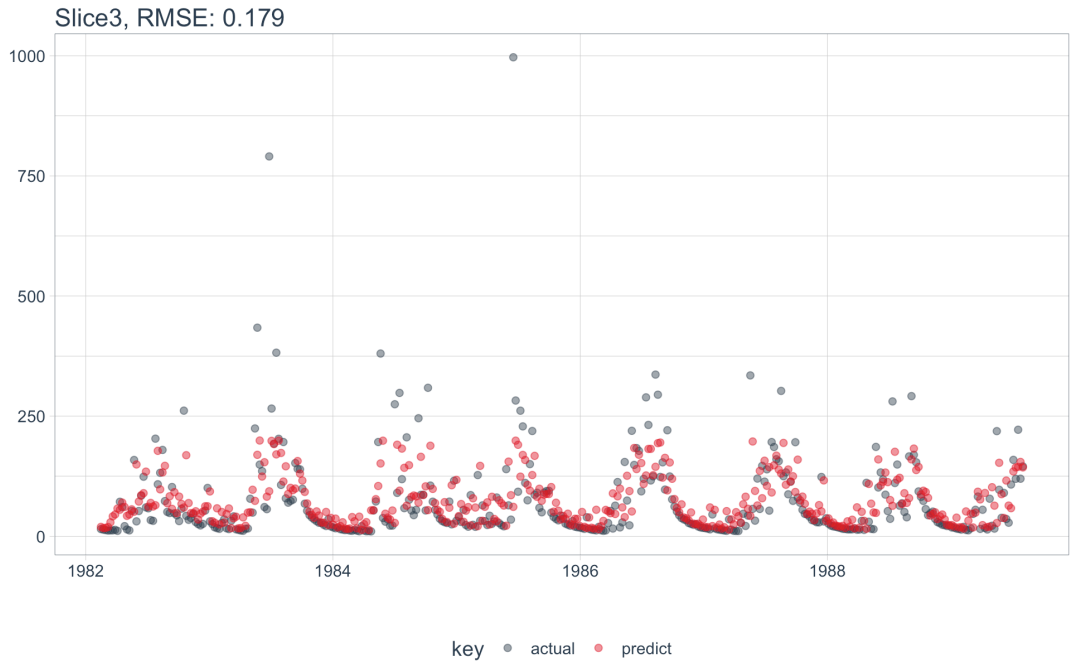

FIGURE 5.15: A clear view of the middle slice prediction plot using the larger-sized slice back testing strategy for the LSTM

After assessing the data in this slice carefully (somewhat visible if Figure 5.10), it was noticed that the testing data set contained outliers which we do not expect

the model to pick up. The problem of the size of the training sets was thought to be improved by adding more data point through using daily data. We do, however, hypothesis that the RMSE could be slightly higher for daily data points as there is more room for error.

5.2 On The Daily Data

5.2.1 FFNN

5.2.2 RNN

When working with the data provided in daily time intervals, similar back testing strategies were developed. We first adopted one which split the data set into nine equal slices consisting of 12 years each with a window shifting four years each slice. The data was then split into six equal slices with a one month shift in the window. The major difference between the two strategies being the size of the data sets in each slice. It is not necessary to show the back testing strategies again as they are the same as those used on the weekly data, however now using the daily format. First, assessing the traditional RNN model using the back testing strategy with smaller slices, an average RMSE of 0.88 was recorded across the different slices. For visualisation purposes, we show the middle slice prediction plot in Figure 5.16. The increased number of points is clearly seen, and the prediction plot shows that the RNN model performs well on the data set, picking up the seasonality very well. When changing the back testing strategy to allow more data for the model to train on in each slice, the average RMSE amoung the slices decreased to 0.667. The middle slice is shown for this in Figure 5.17. This shows how the model predicts the seasonality in the stream flow very well almost to the exact time. The problems are only seen in predicting the outliers, as the model is not entirely accurate in this.

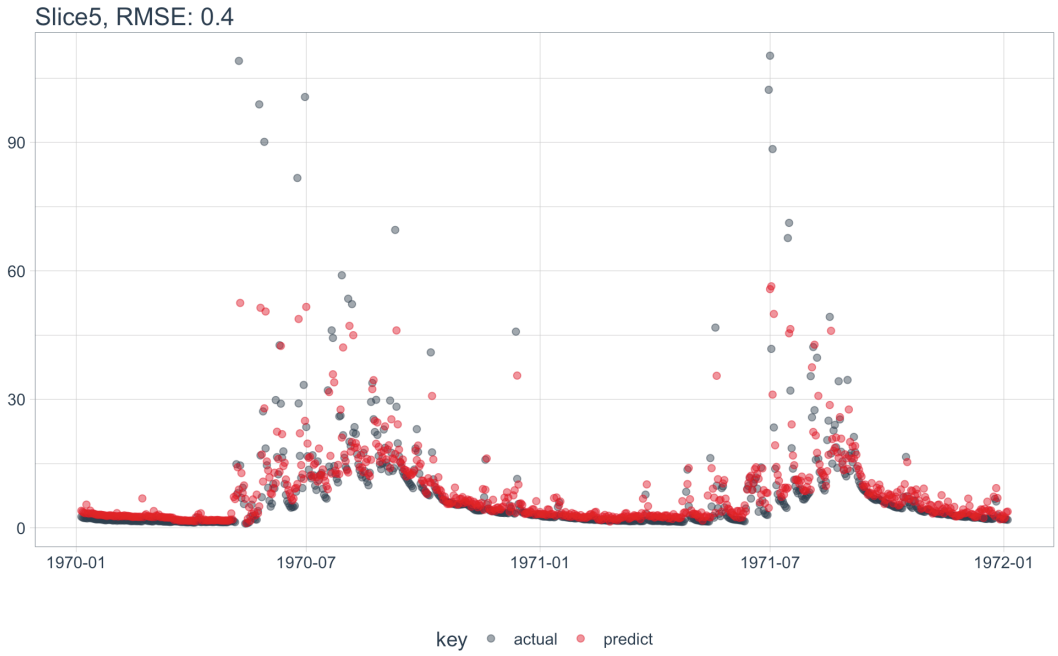

FIGURE 5.16: A clear view of the middle slice prediction plot using the smaller-sized slice back testing strategy for the RNN on daily data

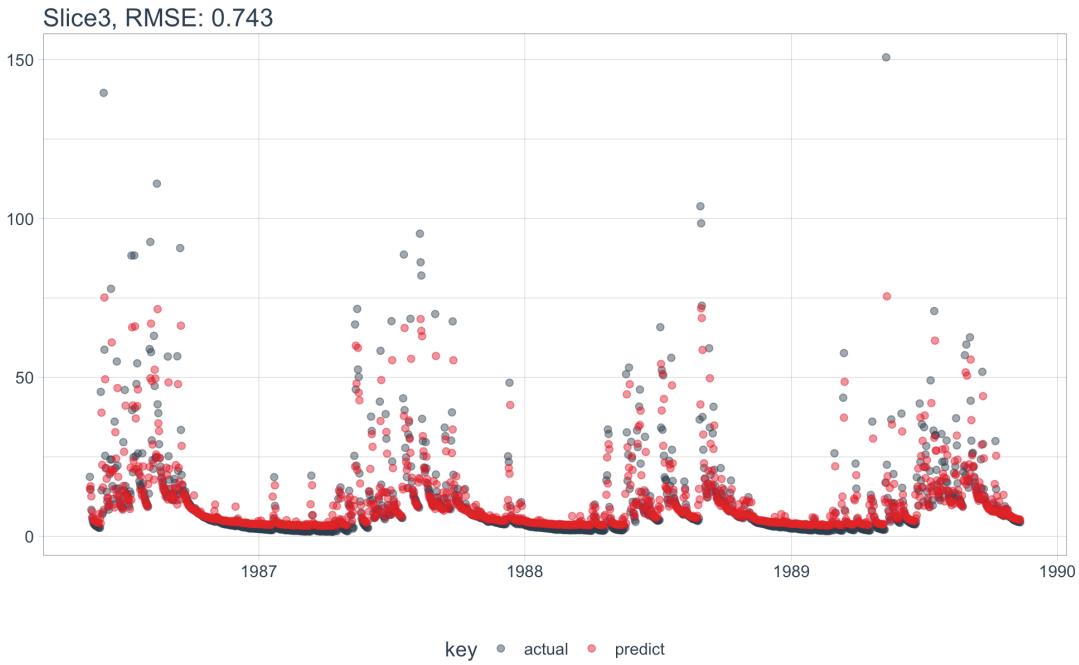

FIGURE 5.17: A clear view of the middle slice prediction plot using the larger-sized slice back testing strategy for the RNN on daily data

5.2.3 LSTM

As with the weekly data, the same back testing strategies used on the RNN were used for the LSTM for the daily data. On the first back testing strategy, the LSTM

model outperformed the traditional RNN. A single slice of the predictions are shown in Figure 5.18. The average RMSE over the entire testing period was 0.47. This slice in particular had a higher RMSE than the same slice of the RNN model. Some slices had RMSE's as low as 0.14 and others as high as 0.7.

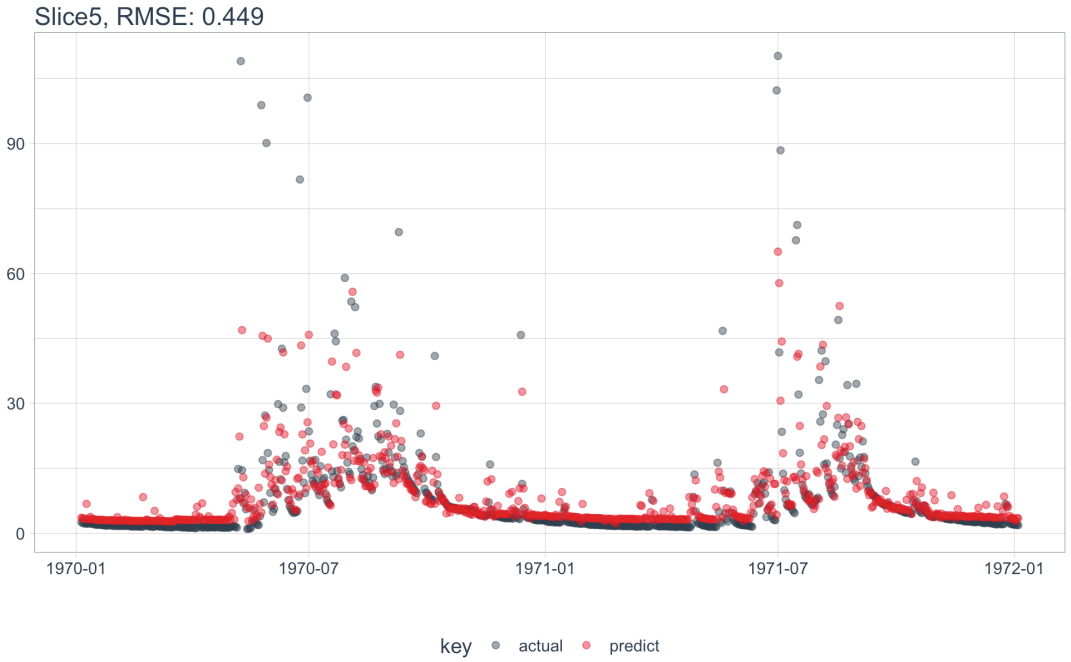

FIGURE 5.18: A clear view of the middle slice prediction plot using the larger-sized slice back testing strategy for the LSTM on daily data

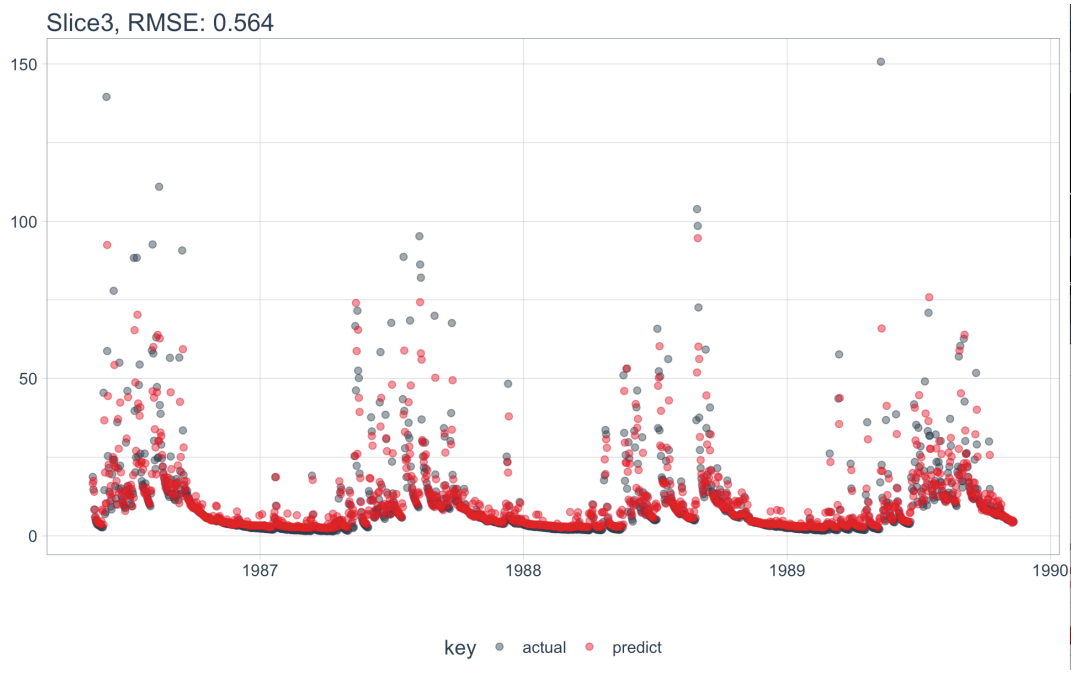

FIGURE 5.19: A clear view of the middle slice prediction plot using the larger-sized slice back testing strategy for the LSTM on daily data

5.3 Validation of Models

5.4 Evaluation of Relative Performance

Chapter 6

Summary and Conclusion

Appendix A

Davis Tipping Bucket

Technical report on calibration of Davis Tipping bucket rain gauges in Jonkershoek – November 2013 By Abri de Buys

A.1 Background:

SAEON Fynbos Node has operated a set of ten Davis Instruments tipping bucket rain gauges since 20 June 2011 in Jonkershoek. Davis Instruments' Quality Assurance Statement claims that their rain gauges are accurate to within $\pm 4\%$ of the actual rainfall amount. We initially installed these gauges straight out of the box because we were satisfied with this level of accuracy. It is routine for instruments to be calibrated from time to time to ensure their long term accuracy. To this end we have been in touch with Davis Instruments to source information on proper calibration methods recommended by them. It should be noted that whenever the issue of calibration was raised with Davis Instruments Tech Support, their replies have been that a) the instruments are factory calibrated and therefore do not need re-calibration or b) that one could set up a paired rain gauge experiment over the course of several rain storms and compare the Davis gauges to that of a standard manual gauge. However, the Davis rain gauges do come with adjustable set screws that can be used to change the volume of water required to tip the gauge. Our rain gauges were due for a calibration test, having been in operation for just over two years. Since other users of the same type of rain gauge (Ferozah Morris and Cobus Pretorius of UKZN) recently raised concerns over the accuracy of some Davis rain gauges, we decided to investigate the accuracy of our Jonkershoek rain gauges in November 2013 and to calibrate them if necessary. This report includes details about the instrumentation, calibration method, calibration results and a calibration protocol. Instrumentation

and calibration method: Tipping bucket mechanism and adjustments The Davis Instruments tipping bucket rain gauge (Part # 7852) has a funnel diameter of 16.5cm giving it a catchment area of 213.82 cm2. The tipping bucket mechanism itself consists of two compartments (Figure) that operate a switch when they tip under the weight of water entering through the funnel. The data logger software then translates this "tip" into a data record of 0.2mm of rainfall. The tipping compartment mechanism rests on set screws (Figure) that can be adjusted to increase or decrease the weight (and thus volume) of water required to initiate a tip. In order to increase the tip volume, the set screws are turned downwards into the gauge platform/base, dropping the resting place of the tipping mechanism lower. In order to decrease the tip volume, the set screws are turned out of the gauge platform, lifting the resting place of the tipping mechanism upwards. It is immediately obvious that to ensure the 0.2mm recorded by the data logger actually represents 0.2mm of rainfall on a $213.82cm^2$ area we need to ensure the mechanism tips at the correct fill volume. In order to do this, the set screws must be in the correct position.

A.2 Calibration Options:

We sourced calibration methods from other manufacturers of tipping bucket rain gauges. A laboratory based method to calibrate rain gauges involves dripping a known volume of water (KVW), say 500ml, through the gauge, counting the number of tips and then calculating the average volume of water per tip. The formula used for this is:

$$\text{Volume per tip} = \frac{\frac{KVW}{\text{Area of Gauge}}}{\text{number of tips}} \times 10 \tag{A.1}$$

An alternative, field based method for calibration is to slowly pour a known volume of water into the gauge using a burette and recording the volume required to make the compartment tip. After several samples the average tip volume is calculated and any necessary adjustments made before the process is repeated to check if the desired result was achieved. We used the formula for the first method to calculate the volume of water that represents 0.2mm (one tip) on an area of 213.82 $cm2$. In other words we solved the equation for KVW:

$$0.2mm = \frac{\frac{KVW}{213.82}}{1} \times 10 \tag{A.2}$$

Therefore: $KVW = 4.276ml$ We rounded this value up to 4.28 ml and attempted to set our tipping compartments to tip as close to this volume as possible.

A.3 Method:

We decided to use the field based method using a burette because we wanted to avoid removing the rain gauges from the field and creating gaps in our data. We also did not have the equipment required to slowly and precisely drip a large volume of water through the gauge in the field or lab. We used a 10 ml burette with 0.05 ml graduations. When first testing the technique it was found that the Davis tipping bucket compartments can sometimes tip at a much lower volume than 4.28 ml, even after adjusting the set screw as low as it can go (increasing the tip volume to the maximum). It was thus impossible to do the calibration by setting both compartments to tip at 4.28 ml as would have been ideal. Instead we aimed to get an average tip of 4.28 ml across both compartments, so that the rain gauge on average records 0.2 mm per tip even though one tip may actually represent slightly higher and one slightly lower amounts of rain. This often meant adjusting one compartment to tip at a volume larger than 4.28 ml if its partner could not be adjusted to tip at a large enough volume. Despite this inconsistency across tipping compartments, we still aimed to get each compartment as close to 4.28 ml as possible and only adjusted compartments away from 4.28 ml if the partner compartment necessitated compensation. Typically the first 4-5 samples per compartment would be a clear indication that the compartment needed adjustment. We would adjust the worst compartment first (average furthest from 4.28 ml after 4-5 samples) and then continue sampling to measure if the adjustment had the desired effect. To quantify the effect of the calibration exercise, a "Factory accuracy" was calculated for each gauge from the overall average tipping volume prior to adjustment. This value reflects the actual amount of rain represented by the average tip of a rain gauge prior to calibration. Note that for each tip, an amount of 0.2mm is recorded in the software. In other words, a rain gauge that tipped at 0.167mm but recorded 0.2mm would have over counted because to register 0.2mm in the data record it actually required less water than that which would represent 0.2mm on a surface area of 213.82 cm2 (i.e. 4.28ml).

References

A. H. Halff, H. M. Halff and M. Azmoodeh (1993). "Predicting Runoff from Rainfall Using Neural Network". In: *Proceeding of Engineering Hydrology, American Society of Civil Engineers*, pp. 760–765.

Agarwal, A. (2002). "Artificial neural networks and their application for simulation and prediction of runoff and sediment yield". PhD thesis. Govind Ballabh Pant University of Agriculture and Technology, Pantnagar.

Agatonovic-Kustrin, S. and R. Beresford (1999). "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research". In: *Journal of Pharmaceutical and Biomedical Analysis*.

Allaire, JJ and Francois Chollet (2018). *keras: R Interface to 'Keras'*. R package version 2.2.0. URL: https://CRAN.R-project.org/package=keras.

Asati, S.R. and S.S. Rathore (2012). "Comparative study of stream flow prediction models". In: 1, pp. 139–151.

ASCE (2000). "Artificial Neural Networks in Hydrology. I: Preliminary Concepts". In: *Journal of Hydrologic Engineering* 5.2, pp. 115–123. DOI: 10.1061/(ASCE)1084-0699(2000)5:2(115). eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%291084-0699%282000%295%3A2%28115%29. URL: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%291084-0699%282000%295%3A2%28115%29.

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181.

Beven, Keith (2012). *Rainfall-Runoff Modelling*. Ed. by Wiley-Blackwell. Second. John Wiley and Sons.

Blume, Theresa, Erwin Zehe, and Axel Bronstert (2010). "Rainfall—runoff response, event-based runoff coefficients and hydrograph separation". In: *Hydrological Sciences Journal*.

Bonne, Jochanan (1971). "Stochastic simulation of monthly streamflow by a multiple regression model utilizing precipitation data". In: *Journal of Hydrology* 12.4, pp. 285 –310.

Boughton, W. C. (1968). "A MATHEMATICAL CATCHMENT MODEL FOR ESTIMATING RUN-OFF". In: *Journal of Hydrology (New Zealand)* 7.2, pp. 75–100. ISSN: 00221708, 24633933. URL: http://www.jstor.org/stable/43944150.

Bryson, Arthur Earl and Yu-Chi Ho (1969). *Optimization, estimation, and control*.

Carcano, Elena C. et al. (2005). "Recurrent Neural Networks in Rainfall – Runoff modeling at daily scale". In:

Carriere, P., S. Mohaghegh, and R. Gaskar (1996). "Performance of a Virtual Runoff Hydrographic System". In: *Water Resources Planning and Management* 122, pp. 120–125.

Cassim, Zaheer (2018). https://www.usatoday.com.

Chandwani et al. (2015). "Soft computing approach for rainfall-runoff modelling: A review". In: *Aquatic Procedia* 4, pp. 1054–1061.

Chen, S.M, Y.M. Wang, and I. Tsou (2013). "Using artificial neural network approach for modelling rainfall-runoff due to typhoon". In: *Journal of Earth System Science* 122.2, pp. 399–405. URL: https://doi.org/10.1007/s12040-013-0289-8.

Daniell, T. M. (1991). "Neural networks. Applications in hydrology and water resources engineering". In: *Proceedings of the International Hydrology and Water Resource Symposium* 3, pp. 797–802. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-0026308790{&}partnerID=40{&}md5=1038e27d8c8640fe3596a93c83dbbd24.

Dawson, C.W. and R.L. Wilby (2001). "Hydrological modelling using artificial neural networks". In: *Progress in Physical Geography*.

De Vos, N.J. and T.H.M. Rientjes (2005). "Constraints of artificial neural networks for rainfall-runoff modelling:Trade-offs in hydrological state representation and model evaluation". In: *Hydrology and Earth System Science* 9, 111–126.

Devia, Gayathri K, Ganasri B Pa, and Dwarakish G Sa (2015). "A Review on Hydrological Models". In: *ScienceDirect*.

Donger, Niklas (2018). *Recurrent Neural Networks and LSTM*. https://towardsdatascience.com.

Ellacott, Steve and Bose D (1996). "Neural Networks: Deterministic Methods of Anayl-sis". In: *London: International Thomson Computer Press*.

Gedeon, T.D. (1997). "Data mining of inputs: Analysing magnitude and functional measures". In: *International Journal of Neural Systems* 8.2, pp. 209–218.

Gers, Felix A., JÃŒrgen Schmidhuber, and Fred Cummins (2000). "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12.10, pp. 2451–2471. DOI: 10.1162/089976600300015015. eprint: https://doi.org/10.1162/089976600300015015. URL: https://doi.org/10.1162/089976600300015015.

Gorgens, A.H.M. (1983). *Intercomparison of conceptual rainfall-runoff models using data from semi-arid research catchments*. Tech. rep. Rhodes University.

Govindaraju, Rao S (2000). "Artificial Neural Networks in Hydrology I: Preliminary concepts". In: *Journal of Hydrologic Engineering Vol 5*.

Gowda and Mayya (2014). "Comparison of Back Propagation Neural Network and Genetic Algorithm Neural Network for Stream Flow Prediction". In: *Journal of Computational Environmental Sciences* 2014. URL: https://doi.org/10.1155/2014/290127.

Grist, Jeremy P. and Sharon E. Nicholson (2001). "A Study of the Dynamic Factors Influencing the Rainfall Variability in the West African Sahel". In: *Journal of Climate* 14.7, pp. 1337–1359. DOI: 10.1175/1520-0442(2001)014<1337:ASOTDF>2.0.CO;2. eprint: https://doi.org/10.1175/1520-0442(2001)014<1337:ASOTDF>2.0.CO;2. URL: https://doi.org/10.1175/1520-0442(2001)014<1337:ASOTDF>2.0.CO;2.

Haan, C. T. (1972). "A water yield model for small watersheds". In: *Water Resources Research* 8.1, pp. 58–69.

Han, H. and P. Felkar (1997). "Estimation of daily soil water evaporation using an artificial neural network". In: *Journal of Arid Environment* 37, pp. 251–260.

Harun, S., A. Nor, and M. Kassim (2002). "Artificial Neural Network Model for Rainfall-Runoff Relationship". In: *Journal of Technology* 37, pp. 1–12.

Hinton, G. et al. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597.

Hochreiter, Sepp and JÃŒrgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

Hornik, K., M. Stinchcombe, and H. White (1999). "Multilayer feed forward networks are universal approximators". In: *Neural Networks* 2, 359–366.

Hsu, Gupta, and Sorooshian (1995). "Artificial neural network modelling of the rainfall-runoff process". In: *Water Resources Research*, 2517–2530.

Hsu, Kuo lin, Hoshin V. Gupta, and Soroosh Sorooshian (1997). "Application of a recurrent neural network to rainfall-runoff modeling". English (US). In: *Proceedings of the Annual Water Resources Planning and Management Conference*. Ed. by D.H. Merritt. ASCE, pp. 68–73.

Hughes, D.A. (2004). "Three decades of hydrological modelling research in South Africa". In: *South African Journal of Science* 100.11-12.

Hunter, D. et al. (2012). "Selection of proper Neural Network sizes and architectures:A comparative study". In: *IEEE Transaction on Industrial Informatics* 8.2, 228–240.

Imrie, C.E., S. Durucan, and A Korre (2000). "River flow predication using artificial neural networks: generalization beyond the calibration range". In: *Journal of Hydrology* 233, pp. 138–153.

Islam, A. et al. (2007). "Designing ANN using sensitivity amp; hypothesis correlation testing". In: *2007 10th international conference on computer and information technology*, pp. 1–6. DOI: 10.1109/ICCITECHN.2007.4579422.

Jakeman, A. J., I. G. Littlewood, and P. G. Whitehead (1993). "An assessment of a dynamic response characteristics of stream flow in catchment". In: *Journal of Hydrology* 195, pp. 337–355.

Jarboe, J. and C. T. Haan (1974). "Calibrating a water yield model for small ungaged watersheds". In: *Water Resources Research* 10.2, pp. 256–262.

Johnson, Mark S et al. (2003). "Application of two hydrologic models with different runoff mechanisms to a hillslope dominated watershed in the northeastern US: a comparison of HSPF and SMR". In: *Journal of Hydrology* 284.1, pp. 57 –76. ISSN: 0022-1694. DOI: https://doi.org/10.1016/j.jhydrol.2003.07.005. URL: http://www.sciencedirect.com/science/article/pii/S0022169403002695.

Jones, J. R. (1976). "Physical Data for Catchment Models". In: *Hydrology Research* 7.4, p. 245.

Kapangaziwiri, E. (2010). "Regional Application of The Pitman Monthly Rainfall-Runoff Model In Southern Africa Incorporating Uncertainty". PhD thesis.

Karmakar, S., M.K. Kowar, and P. Guhathakurta (2009). "Spatial interpolation of rainfall variables using artificial neural network". In: *Proceedings of the International Conference on Advances in Computing, Communication and Control*. Mumbai, India, pp. 547–552.

Karunanithi et al. (1994). "Neural networks for river flow prediction". In: *Journal of Computing in Civil Engineering, ASCE* 8, 201–220.

Kothyari, Umesh C. (1995). "Estimation of monthly runoff from small catchments in India". In: *Hydrological Sciences Journal* 40.4, pp. 533–542. DOI: 10.1080/02626669509491437. eprint: https://doi.org/10.1080/02626669509491437. URL: https://doi.org/10.1080/02626669509491437.

Kratzert, F. et al. (2018). "Rainfall-Runoff modelling using Long-Short-Term-Memory (LSTM) networks". In: *Hydrology and Earth System Sciences Discussions* 2018, pp. 1–26. DOI: 10.5194/hess-2018-247. URL: https://www.hydrol-earth-syst-sci-discuss.net/hess-2018-247/.

Krstanovic, P. F. and V. P. Singh (1991). "A univariate model for long-term streamflow forecasting". In: *Stochastic Hydrology and Hydraulics*.

Kuhn, Max and Hadley Wickham (2017). *rsample: General Resampling Infrastructure*. R package version 0.0.2. URL: https://CRAN.R-project.org/package=rsample.

Kumar, M. et al. (2002). "Estimating evapotranspiration using artificial neural network". In: *Journal of Irrigation and Drainage Engineering, ASCE*, 128, pp. 224–233.

Lane, L.J., M. H. Diskin, and K.G. Renard (1971). "Input-output relationships for an ephemeral stream channel system". In: *Journal of Hydrology* 13, pp. 22–40.

LeDell, Erin et al. (2018). *h2o: R Interface for 'H2O'*. R package version 3.20.0.8. URL: https://CRAN.R-project.org/package=h2o.

Lefkovitch, L.P. *The Study of Population Growth in Organisms Grouped by Stages*.

Lewis, N.D. (2017). *Neural networks for time series forecasting with R. Intuitive Step by Step Blueprint for Beginners*.

Linnainmaa, Seppo (1976). *Taylor expansion of the accumulated rounding error*. DOI: 10.1007/bf01931367.

Linsley, R.K. and N.H Crawford (1960). "Computation of synthetic storm flow record on a digital computer". In: *Int. Associate Science of Hydrology* 51, pp. 526–538.

Lorrai and Sechi (1995). "Neural nets for modelling rainfall-runoff transformations." In: *Water Resources Management* 9, pp. 299–313.

Machado, Fernando et al. (2011). "Monthly rainfall-runoff modelling using artificial neural networks". In: *Hydrological Sciences Journal* 56.3, pp. 349–361. DOI: 10.1080/02626667.2011.559949. eprint: https://doi.org/10.1080/02626667.2011.559949. URL: https://doi.org/10.1080/02626667.2011.559949.

Magette, W. L., V. O. Shanholtz, and J. C. Carr (1976). "Estimating selected parameters for the Kentucky Watershed Model from watershed characteristics". In: *Water Resources Research* 12.3, pp. 472–476.

Mahe, Gil et al. (2008). "Comparing available rainfall gridded datasets for West Africa and the impact on rainfall-runoff modelling results, the case of Burkina-Faso". en. In: *Water SA* 34, pp. 529 –536. ISSN: 1816-7950. URL: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S1816-79502008000500002&nrm=iso.

Marshall, S.J. (2013). "Hydrology". In: *Reference Module in Earth Systems and Environmental Sciences*. Elsevier. ISBN: 978-0-12-409548-9. DOI: https://doi.org/10.1016/B978-0-12-409548-9.05356-2. URL: http://www.sciencedirect.com/science/article/pii/B9780124095489053562.

Martin, Carlos and Lucas Schuermann (2016). *Accessible Machine Learning: Neural Networks*. https://bigtheta.io/2016/02/24/intro-to-neural-networks.html.

Masselota, Pierre et al. (2016). "Streamflow forecasting using functional regression". In: *Journal of Hydrology*.

McCulloch, Warren S. and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

McGonagle, John (2018). *Feed Forward Neural Networks*. https://Brilliant.org.

Molga, E.J. (2003). "Neural network approach to support modeling of chemical reactors: problems, resolutions, criteria of application". In: *Chemical Engineering and Processing: Process Intensification* 42.8–9, 675–695.

Moritz, Steffen and Thomas Bartz-Beielstein (2017). "imputeTS: Time Series Missing Value Imputation in R". In: *The R Journal* 9.1, pp. 207–218. URL: https://journal.r-project.org/archive/2017/RJ-2017-009/index.html.

Mulvaney, T. J. (1850). *On the use of self-registering rain and flood gauges in making observations of the relations of rainfall and of flood discharges in a given catchment*.

Nagesh Kumar, D., K. Srinivasa Raju, and T Sathish (2004). "River Flow Forecasting using Recurrent Neural Networks". In: *Water Resources Management* 18, p. 143. URL: https://doi.org/10.1023/B:WARM.0000024727.94701.12.

Nash, J.E. and J.V. Sutcliffe (1970). "River flow forecasting through conceptual models part I — A discussion of principles". In: *Journal of Hydrology* 10.3, pp. 282 –290. ISSN: 0022-1694. DOI: https://doi.org/10.1016/0022-1694(70)90255-6. URL: http://www.sciencedirect.com/science/article/pii/0022169470902556.

Olah, C. (2015). *Understanding LSTM Networks*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Pan, Tsung yi et al. (2008). *Application of Recurrent Neural Networks to Rainfall-runoff Processes*. Tech. rep. National Taiwan University.

Pendharkar, P.C. and J.A. Rodger (2003). "Technical efficiency based selection of learning cases to improve the forecasting efficiency of neural networks under monotonicity assumption". In: *Decision Support Systems* 36.1, 117–136.

Pienaar, Etienne A.D. (2018). *Introduction to Neural Networks*.

Qu, Yizhong (2004). "An Integrated Hydrologic Model for Multi-Process Simulation Using Semi-Discrete Finit Volume Approach". PhD thesis. Pennsylvania State University.

Rajurkar, M.P., U.C. Kothyari, and U.C. Chaube (2002). "Artificial neural networks for daily rainfall-runoff modelling". In: *Hydrological Sciences Journal* 47(6), pp. 865–877.

Riad, S. et al. (2004). "Rainfall Runoff Model using an Artificial Neural Network Approach". In: *Mathematical and Computer Modelling* 40, pp. 839–846.

Rockwood, D (1958). "Columbia basin stream flow routing by computer". In: *Journal of Waterways and Harbour Division* 84, pp. 1874–1881.

Sawunyama, T and DA Hughes (2010). "Using satellite-based rainfall data to support the implementation of environmental water requirements in South Africa". en. In: *Water SA* 36, pp. 0 –0. ISSN: 1816-7950. URL: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S1816-79502010000400001&nrm=iso.

Saxena, Shubh (2017). *Artificial Neuron Networks(Basics) | Introduction to Neural Networks*.

Schulze, RE and A Pike (2004). *Development and evaluation of an installed hydrological modelling system*. Tech. rep. School of Bioresources Engineering and Environmental Hydrology University of Kwa-Zulu-Natal.

Scott, D.F. et al. (2000). *A re-analysis of the South African catchment afforestation experimental data*. Tech. rep. CSIR Division of Yater Environment and Forestry Technology Stellenbosch.

Sedki, A., D. Ouazar, and E. El Mazoudi (2009). "Evolving neural network using real coded genetic algorithm for daily rainfall-runoff forecasting". In: *Expert Systems with Applications* 36.3, Part 1, pp. 4523 –4527. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2008.05.024. URL: http://www.sciencedirect.com/science/article/pii/S095741740800225X.

Shamseldin (1997). "Application of a neural network technique to rainfallrunoff modeling". In: 199, pp. 272–294.

Sinha, Jitendra (2011). "Artificial Neural Network Approach to Rainfall-Runoff Modelling for Upper Kharun Catchment in Chhattisgarh". PhD thesis. Indira Gandhi Krishi Vishwavidyalaya Raipur.

Sitterson, Jan et al. (2017). *An Overview of Rainfall-Runoff Model Types*. Tech. rep. U.S. Environmental Protection Agency.

Smith, Chris (2016). *iOS 10: Siri now works in third-party apps, comes with extra AI features*.

Somvanshi, V.K. et al. (2006). *Modelling and prediction of rainfall using artificial neural network and ARIMA techniques*.

Stansbury, Dustin (2018). *The Clever Machine*. urlhttps://theclevermachine.wordpress.com.

Stellenbosch climate (2017). urlhttps://http://www.saexplorer.co.za.

Sutskever, I., O. Vinyals, and Q. V. Le (2014). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, 3104–3112.

Suwajanakorn, Supasorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman (2017). "Synthesizing Obama: Learning Lip Sync from Audio". In: *ACN Trans. Graph.* DOI: http://dx.doi.org/10.1145/3072959.3073640.

Tamura, S. and M. Tateishi (1997). "Capabilities of a four-layered feedforward neural network: four layers versus three". In: *IEEE Transactions on Neural Networks* 8.2, 251–255.

Tchircoff, Andrew (2017). *The mostly complete chart of Neural Networks, explained*. https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464. Accessed: 2018-09-01.

Todeschini, Fabio and Liana Jansen (2016). *Heritage Inventory of, and Management Plan for, the Tangible Resources in the Stellenbosch Municipality: Phase 1 Report - Approach,*

Concepts, Methods and Preliminary Findings. Tech. rep. Cape Winelands Professional Practices in Association.

Tokar and Johnson (1999). "Rainfall-runoff modelling using artificial neural networks". In: 4, pp. 232–239.

Tokar and Markus (2000). "Precipitation-Runoff Modeling Using Artificial Neural Networks and Conceptual Models". In: *Journal of Hydrologic Engineering* 5.2, pp. 156–161.

Torfs, P. and P. Warmerdam (2001). "Application of Parzen densities to probabilistic rainfall-runoff modeling". In: *ERB and Northern European Friend Project 5 Conference*.

Valdes, Juan B, Guillermo J Vicens, and Ignacio Rodriguez-Iturbe (1979). "Choosing among alternative hydrologic regression models". In: *Water resources research* 15.2, pp. 347,358. ISSN: 0043-1397.

Vaze, J. et al. (2011). *Guidelines for rainfall-runoff modelling: towards best practice model application*. eWater CRC.

Vogels, Werner (2016). *Bringing the Magic of Amazon AI and Alexa to Apps on AWS. - All Things Distributed*. https://www.allthingsdistributed.com.

Walia, Anish Singh (2017). https://towardsdatascience.com.

Wang, G. T. and Y. Yung (1986). "Estimation of parameters for the discrete linear input-output model". In: *Journal of Hydrology* 85, pp. 15–30.

Wilby, R.L., R.J. Abrahart, and C.W. Dawson (2003). "Detection of conceptual rainfall-runoff processes inside an artificial neural network". In: *Journal of Hydrological Science* 48, pp. 163–181.

Xu, C., J. Seibert, and S Halldin (1996). "Regional water balance modelling in the NOPEX area test on parameter estimation using catchment characteristics". In: *Journal of Hydrology* 13, pp. 353–368.

Yang, C.C., S.O. Prasher, and R Lacroix (1996). "Applications of artificial neural network to land drainage engineering". In: *Transactions of ASAE* 39, pp. 525–533.

Zazo, Ruben et al. (2016). "Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks". In: *PLOS ONE* 11.1, pp. 1–17. DOI: 10.1371/journal.pone.0146917. URL: https://doi.org/10.1371/journal.pone.0146917.

Zbigniew, W. and Jaroslaw J. Napoirkowski (2009). "Nonlinear models of dynamic hydrology". In: *Hydrological Sciences Journal*.

Zealand, Burn, and Simonovic (1999). "Short term stream flow forecasting using artificial neural networks". In: *Journal of Hydrology*, pp. 32–48.

Zhang, Zhenghao, Qiang Zhang, and Vijay P. Singh (2018). "Univariate streamflow forecasting using commonly used data-driven models: literature review and case study". In: *Hydrological Sciences Journal* 63.7, pp. 1091–1111.