Deep Learning for Flood Extent Prediction

Leo Muckley

Master of Science
Data Science
School of Informatics
University of Edinburgh
2020

Abstract

Flooding is among the most destructive natural disasters in the world. As a result of this, there is a great need to develop accurate flood prediction models to prevent flooding ahead of time. However, due to the many factors that are related to predicting flood extent, floods are highly complex to model. Machine learning techniques have shown promise in this space but the current state-of-the-art techniques fail to generalise to other flood events. This study shows that Long Short-Term Memory (LSTM) networks are effective at predicting flood extent by surpassing the current state-of-the-art and generalising to other types of flood events.

Acknowledgements

I would like to thank my supervisor, James Garforth, for his helpful discussions and his vital feedback. I am most grateful for his commitment to his supervision and guidance through unprecedented times.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Leo Muckley*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Floods are among Earth's most common and most destructive natural disasters. According to the Organisation for Economic Cooperation and Development, floods cause more than $40 billion in damage worldwide annually [37]. One of the most impacted areas in the world is the eastern countries of Africa where mass migration due to the impact of flooding is common. As a result of this, there is a great need to develop accurate flood prediction models to prevent flooding ahead of time.

There has been a lot of effort put into detecting floods ahead of time using machine learning techniques with varying success [35]. One problem with flood prediction is the inability of pre-existing models to generalise well to other flood events. This can be attributed to the various types of flooding that exist and the various factors which determine the location and extent of these events. For example, countries can be prone to fluvial flooding (i.e. river flooding), pluvial flooding (i.e. flash floods) and coastal flooding due to storm surges. As a consequence of this, developing a general model that can incorporate all the underpinning features of every variant of flooding to accurately detect a specific flood event ahead of time a is difficult.

One approach to predicting flood extent and location is to use deep learning techniques [30]. A recent Zindi competition [5] challenged competitors to develop machine learning models to predict flood extent and location in Malawi. The winning solution for this competition would be considered the current state-of-the-art in flood extent prediction. However, deep learning techniques have not been investigated for this purpose. The focus of this study will be to develop a novel deep learning solution to the flood extent prediction problem, using the winning solution to the Zindi competition

as a benchmark.

1.2 Problem Statement

The prediction of flood extent and location is a task of trying to predict the level of inundation y, where $0 \leq y \leq 1$, at time t based on M features for the previous k points in time. In this problem, the level of inundation is the fraction (i.e. value in of a 1km sq polygon that is covered in flood water at time t and each feature m, is a sequence of k timesteps (e.g. daily total precipitation). It is these M sequences that are to be utilised to predict flood extent. This study will focus on investigating various deep learning techniques to determine the best method of combining these sequences.

Figure 1.1 demonstrates the modelling process for this problem, where $k-1$ timesteps are considered for each sequence of features. For example, one measurement could be the total precipitation level of a location at timestep $t-k$ with the final timestep at $t-1$. This example would then predict the flood extent for that datapoint (location) at time t.

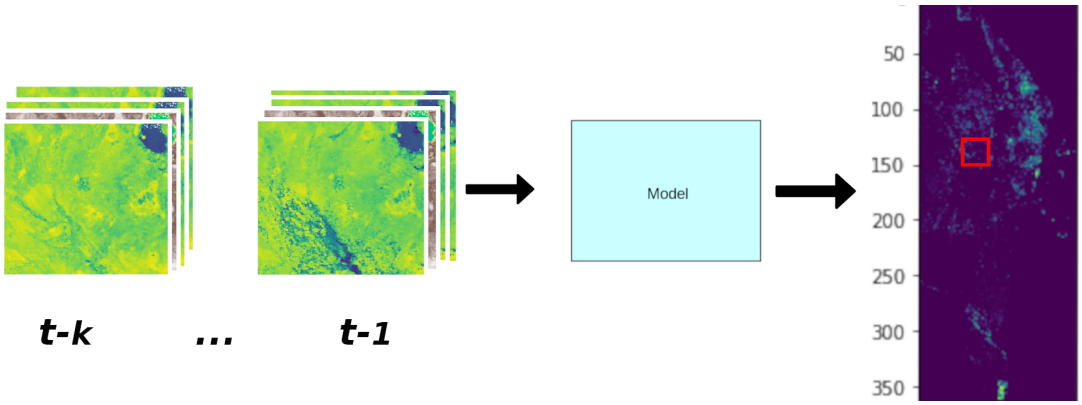


Figure 1.1: The modelling process for the problem of flood extent prediction.

1.3 Objective

Predicting the level of flood extent is a complex task. The difficulty in this task can mostly be attributed to the following reasons: (1) the sparsity of training data as flooding is an extreme rare event; (2) the data that is available for training is generally very noisy due to satellite measurements being approximations and the measurements being taken at varying times leading to fuzzy timesteps for time-series modelling; (3) the detection of spatial outliers can be problematic as each flood can have unique set of

features which determine flooding for that specific area (e.g. land cover could have a dynamic range of values which is positively correlated with flood extent but no correlation exists in a different geo-location).

Deep learning techniques have shown great success across various domains. By utilising techniques in computer vision and image processing, such as Convolutional Neural Networks (CNN), spatial dependencies can effectively be mapped in satellite imagery [42]; Long Short-Term Memory (LSTM) networks are type of neural network that has proven success in modelling sequential unstructured data in the areas of speech recognition [19], natural language processing [45] and in structured time-series applications [22]. Furthermore, one particular deep learning architecture that incorporates both the CNN and LSTM is the ConvLSTM, which was originally developed for precipitation forecasting in a sequence of satellite imagery [46]. The ConvLSTM is particularly useful for spatio-temporal modelling when adequate training data is available so that the spatial dependencies in the data can be modelled over time. However, for the spatio-temporal problem of flood extent prediction, a limited quantity of training data is available.

Therefore the aim of this dissertation will be to investigate the feasibility of utilising deep learning techniques for the spatio-temporal modelling of flood extent with sparse data. The success of these models will be determined by their ability to surpass the current state-of-the-art model in flood extent prediction. To achieve this, three research questions have been created:

- **Can a process be established where only a limited quantity of training data is needed to model the spatial and temporal elements of flood extent prediction?**

  The data augmentation techniques developed in this study show that through spatial grid staggering, spatial autocorrelation in the dataset is maintained. Furthermore, this data augmentation technique achieves the intended outcome of producing larger training sets with a more balanced target distribution. Consequently, this process shows that it is possible to create bigger datasets in the absence of sufficient training data while maintaining the inherent attributes of the original dataset.

- **Can the optimal deep learning architecture be determined for surpassing the current state-of-the-art in flood extent prediction?** The deep learning experiments in this study show that each variant of the LSTM network out perform

the current state-of-the-art in flood extent prediction. The results are greatly in favour of the LSTM model due to their specific architecture being able to effectively utilise the temporal information needed to predict flood extent. More specifically, the Multi-Input LSTM architecture proves to be the optimal deep learning architecture for modelling flood extent.

- **Can deep learning techniques be developed for the purpose of flood extent prediction to be able to generalise to varying types of flood events?** The results obtained from the deep learning techniques show the ability of these techniques to generalise to other flood events under certain conditions. It was observed that when the underlying features that determine one flood event (e.g. distance to water) vary greatly the LSTM networks fail to return accurate predictions. However, when the range of values for the underlying features have similar distributions, the LSTM networks return accurate predictions and greatly outperforming the current state-of-the-art model when generalising to another flood event.

1.4 Structure

The rest of this document is structured as the following:

- Chapter 2 - **Background**: the first section introduces deep learning along with the major architectures, followed by machine learning applications in flood prediction; the second section begins by providing details of a related machine learning competition hosted by [5]. Subsequently, some preliminary work for the competition will be briefly outlined. Finally, the winning solution for the competition will be showcased.

- Chapter 3 - **Data Preprocessing and Analysis**: this chapter starts with providing details of the data collection procedure, followed by an analysis of the datasets. The end of the chapter will then discuss the data augmentation technique that is employed in this study.

- Chapter 4 - **Methodology**: the details of the main methods used in this study are discussed in this chapter. Each model developed for this study will be presented in detail including motivation for using that specific model.

- Chapter 5 - **Evaluation and Results**: the first section provides details on the evaluation process, the characteristics of the training and test sets and the validation strategy; the second section provides a background on the experiments conducted with a brief overview of the hyperparameter configuration for the problem and for the models, which includes the winning solution for the Zindi competition; section 3 presents the results of the experiments conducted for homogeneous data and heterogeneous data; section 4 then provides a deep analysis of the results presented in section 3.

- Chapter 6 - **Discussion**: the final chapter will discuss the overall results of the study before suggesting some future research directions based on the findings. The dissertation will then conclude with some final remarks.

Chapter 2

Background

Flood forecasting is an area of research that has been extensively studied. However, it is only in recent times that machine learning has been applied to this domain [35]. Moreover, there is even less research on deep learning techniques for the prediction of flood extent and location. Nevertheless, this section will introduce the existing research in this space.

This chapter will first give a background on deep learning before discussing applications in flood prediction. Following this, the task of flood extent prediction will be discussed with details of a related an online machine learning competition. This will include details of some preliminary work conducted for the competition and the winning solution for the competition introduced.

2.1 Related Literature

2.1.1 Deep Learning: Feed-forward Networks

Deep learning is a branch of machine learning where the algorithms are inspired by how neurons in the brain interact. These algorithms use, what is called, an Artificial Neural Network (ANN) to model non-linear relationships between an input and an output. This sub-category of machine learning has proven success in modelling structured and un-structured data in areas such as speech [19], computer vision [42], natural language processing [45] and time-series [22] applications.

Deep learning models are composed of multiple layers including the input layer, some intermediate hidden layers and an output layer. The model maps input to output by propagating through the layers where each subsequent layer is a weighted sum of

the previous layer, with the result of this being passed through an activation function. Figure 2.1 outlines this operation for one hidden unit or neuron with corresponding weights. This operation is analogous to a non-linear transformation. It is this non-linear transformation that helps the model to learn different representations of the data with multiple levels of abstraction [30].

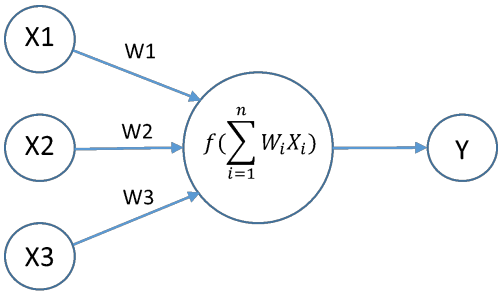


Figure 2.1: Neuron.

When artificial neural networks are trained, an algorithm called backpropagation is applied which updates the weights of the model. This algorithm uses the chain-rule to compute the partial derivatives of the weights with respect to a loss function. This process is optimised using gradient descent and due to the loss function being non-convex, only a local minimum can be reached. Nevertheless, numerical methods have shown to be sufficient in training artificial neural networks to learn complex mappings between input and output [30].

In general, traditional prediction tasks would require domain knowledge to effectively identify the most important features for a given task. Domain experts may need to reduce the complexity of the data to make the signal visible to the learning algorithm before training. Conversely, deep learning is effective in the absence of domain knowledge as it is able to learn high-level features. Therefore, the this ability to learn in the absence of domain knowledge would make this technique a viable option for predicting flood extent for the purpose of this study.

2.1.2 Deep Learning: Recurrent Networks

In the previous section, feed-forward networks were discussed outlining their ability to learn complex mappings through non-linear transformations as the data is propagated through the hidden layers of the network. In tasks that deal with sequential data, additional measures need to be provided to effectively learn from this type of data. Recur-

rent neural networks are an extension of the feed-forward neural network architecture and have been specifically developed for the purpose of learning from sequential data.

In sequential data modelling, to effectively predict at timestep t, it is necessary to have enough information at timestep $t-1$ to make that prediction. Thus, to have the adequate information to make that accurate prediction, information from the previous timesteps (i.e. $t-2, t-3$, etc.) need also to be retained. This information can be retained in a single recurrent layer where the recurrent neuron contains the state of the previous time step, and combines with the state at time t. For longer sequences of data, the recurrent layer can combine all the previous timesteps. However, since the back-propagation algorithm computes partial derivatives, the recurrent neural network can be prone to the vanishing gradient problem which leads to all the derivatives tending to zero. Subsequently, the weights of the model do not to update and no learning can take place. Long Short Term Memory (LSTM) networks were developed to tackle this problem [21].

LSTM networks have proven success in a variety of sequential and temporal modelling problems such speech recognition [19], natural language processing [45] and time-series forecasting [22]. As previously stated, the success of LSTM networks can mostly be attributed to their ability to tackle the vanishing gradient problem and can be therefore applied to practical problems. As a result of this, LSTM networks are often a good model candidate for problems that contain inherent patterns within sequential data.

2.1.3 Deep Learning: Autoencoder

In data science, if a learning task has labelled data available for training, we would associate that task with *supervised learning*. If the data has no labels, we employ *unsupervised learning* techniques. One deep learning technique for unsupervised learning is called an autoencoder [9].

An autoencoder is a type of neural network algorithm but the architecture of this model differs slightly. In the case of an autoencoder, a bottleneck layer is introduced into one of the intermediate layers of the network to compress the input data. This operation forces the network to learn a compressed representation of the input where only the most important features of the data is kept. In the case of unsupervised learning, it can be utilised for feature selection [20]. Although autoencoders were originally devised for unsupervised learning tasks, autoencoders have also been applied to su-

pervised learning tasks, with one example being the LSTM Autoencoder [33]. This variation of the unsupervised learning task is utilised for sequential learning tasks.

One successful application of LSTM autoencoders is in anomaly detection where its has been applied to fall detection [36], video surveillance [32] and computer network intrusion [34]. In these cases, the LSTM autoencoder attempts to recreate the input fed into the network. Then any element of the data that the network fails to recreate would be labelled as an anomaly. As the LSTM autoencoder is successful in applications where a disproportionate amount of the data is negatively labelled (e.g. anomaly detection), one potential application of LSTM autoencoders would be in extreme rare event problems such as flooding.

2.1.4 Deep Learning: Convolutional Neural Networks

When data contains spatial dependencies in the data such as images, Convolutional Neural Networks (CNN) are generally used [28, 15, 26]. These types of networks apply filters to the data to extract the most pertinent or meaningful features which represent the image. The process begins by first sliding the filter over the image while computing a calculation called a convolution, resulting in what are called convolved features. Following this, a pooling layer is applied to the convolved features, resulting in the reduction of the dimensions of these features and retaining only the high-level features of the image.

There are a wide variety of CNN based architectures for dealing with different computer vision and image processing tasks [42]. Moreover, additional architectures have been developed for spatio-temporal modelling of sequences of images, namely, the Convolutional LSTM (ConvLSTM) [46]. This architecture adopts operations from both the CNN and LSTM networks to effectively model both spatial and temporal dependencies in the data. In the ConvLSTM, original LSTM internal multiplications are replaced with convolution actions, resulting in the input maintaining the original dimension of the image.

The benefits of implementing a ConvLSTM is to effectively model both spatial and temporal domains. However, for the model to learn it would need sufficient amount of image data to train on. In the context of flood extent prediction, an inadequate amount of satellite imagery is available for training a *full* ConvLSTM due to the extreme rarity of flooding events. Nonetheless, a variation of the ConvLSTM model may still be able to extract local spatial dependencies by applying 1-Dimensional filter to sequential

features, such as the total precipitation of a location over time.

2.1.5 Flood Conditioning Factors

Closely related to the modelling of flood extent prediction is the modelling of flood susceptibility. These are closely related as factors that determine flood susceptibility are likely to have predictive power in modelling flood extent. A GIS-based spatial prediction approach which modelled flood prone areas in the Xing Guo region of China was proposed using an ensemble of techniques [43]. The goal was to accurately classify the flood prone areas using a test set of ground truth levels of flood susceptibility. The ensemble of Logistics Regression and Weight of Evidence models achieved the highest performance of accuracy 90.36% when modelled on the following conditioning factors: (1) altitude, (2) slope, (3) aspect, (4) geology, (5) distance from river, (6) distance from road, (7) distance from fault, (8) soil type, (9) LULC, (10) rainfall, (11) Normalized Difference Vegetation Index (NDVI), (12) Stream Power Index (SPI), (13) Topographic Wetness Index (TWI), (14) Sediment Transport Index (STI) and (15) curvature. The result of this report supports the prior expectation that less vegetated areas at low altitude with high levels of rainfall are more susceptible to flooding.

Flood susceptibility is not equivalent to flood extent, however, for an area with a high susceptibility of flooding it would be expected that there would be a positive correlation between the two values. In general, it would be plausible to say that a high level of flood susceptibility would equate to a high level of flood extent for a given flood event. As a result of this, the features used to model susceptibility can be expected to be useful for modelling flood extent.

2.1.6 Flood Forecasting

Flood forecasting is a widely studied area and various type of statistical and machine learning models have been developed [35]. Generally these predictive models are specifically tuned for one area or for a specific type of flood (i.e.pluvial, fluvial flooding etc.) resulting in the inability of the model to generalise to other flood events. Nevertheless, machine learning techniques have shown some success in flood forecasting.

Early applications of feed-forward neural networks have been applied to modelling river water level to effectively forecast riverine flooding [13] in Tagliamento, Italy. In this scenario, the neural network used information from various rain gauges along the river as the features for the model. The results of this model were quite accurate with

a mean square error less than 4% when used within a 1-hour time window. However, when the time window is increased, the prediction accuracy decreases as the neural network is note able to model the time lag between the water level and the rainfall, which is an important aspect of flood forecasting.

More recent developments in flood forecasting utilise LSTM networks utilising discharge and rainfall information in the Da River basin in Vietnam [29]. The LSTM network modelled one-day, two-day and three-day flow rate forecasting with Nash–Sutcliffe efficiency (NSE) reaching 99%, 95%, and 87% respectively. These results demonstrate how well-suited LSTM networks are for flood forecasting due to their ability to effectively deal with time lags and exploit the temporal dependencies in the data. However, the LSTM network does not take into account the spatial relationships between the different hydrological stations. The outcome of this shortcoming is that it would be unlikely that the model would generalise well to other river basins.

2.1.7 Flood Extent Prediction

The difficulty of modelling flood extent is partly due to the sparsity of training data but also due to pre-existing models not being able to generalise well to other flood events at different geo-locations. This is the case as flooding in different areas can be attributed to different factors. For example, training on fluvial flooding (i.e. river flooding) and pluvial flooding (i.e. flash floods) data when trying to predict coastal flooding would likely be ineffective. As a result of this, many studies generally focus on one geographic area to find those unique local factors that determine flood extent.

For the problem of predicting flood extent and location, exploiting spatial correlations in the data is necessary. One solution would be to use a convolutional LSTM network which would generally take 2D data as input. To effectively implement this approach, it would be necessary to use a large sequence of images to model both spatial and temporal patterns in the data. However, due to the fact that the floods are an extreme rare event, labelled data can be sparse. For example, training set may consist of only a single image. Therefore, using a 2D convolutional LSTM to model flood extent based on the available satellite imagery would be infeasible. Nevertheless, this study will investigate the feasibility of modelling with sparse data.

In the next section, an online machine learning competition will be introduced where the goal is to solve the problem of predicting flood extent and location in Malawi.

2.2 Zindi Competition

Zindi is an online platform that hosts machine learning competitions [5]. The platform recently hosted a competition titled *UNICEF Arm 2030 Vision: Flood Prediction in Malawi* for which this research problem is based. In this competition, the goal was to predict the extent and location of flooding in the Lower Shire valley in Malawi. The map of Malawi was divided into 1km sq rectangles where each rectangle represents a datapoint. Each datapoint then has certain features attached (e.g. elevation; precipitation etc.) with the target value y being the fraction of the rectangle that was flooded, where $0 \leq y \leq 1$.

2.2.1 Preliminary Work

For the competition, some initial modelling was conducted to gain better insight into how the different models perform. A summary of the models investigated are as follows:

- **Hurdle Regression** This is a type of hurdle model [41], which aims to tackle zero-inflated datasets, originally devised for count data. The motivation to use this model was to account for the disproportionate quantity of non-flooded data. However, this method was not effective at predicting flood extent mostly due to the slightly bimodal nature of the target distribution (i.e. the first mode equal to 0 and the second mode equal to 1) at the upper and lower bounds, at 0 and 1 respectively.

- **Mixture Density Network** This model is a type of neural network which can provide a probability distribution over predictions. This technique was investigated as it incorporates mixture of Gaussians [10] which is better able to model multimodal target distributions, similar to the target distribution in the flood extent prediction problem. However, this model did not perform well as it could not deal with the multiple features need for modelling suitably well.

- **Ensemble Methods** These methods ultimately combine several weak learners or base models to produce one strong learner or meta-learner [48]. For the task of flood extent prediction, an ensemble model called Light Gradient Boosting Model (LGBM) [25]. However, this technique was not able to model the temporal features that were provided in the data.

2.2.2 Winning Solution

The winning solution for the competition was composed of a the gradient boosting framework of the LGBM with model averaging [6]. The true success of this approach was due to the feature engineering applied to the weekly mean precipitation values. The precipitation values were considered in various contexts and time windows (e.g. max-precipitation over a two-week period; the rate of change in mean precipitation between weeks etc). Despite these features being temporal by nature, these were modelled as constant features and achieved RMSE of 0.0734 on the Zindi test set.

Considering that flood extent prediction is a task that contains many conditioning features, which are both temporal and constant by nature, an intuitive approach to this task would be to develop a model that can exploit both the temporal and constant attributes of these features. Therefore, an approach that captures such behaviour would be a good model candidate for this problem.

The type of deep learning architecture chosen for this study is the LSTM network. The reason for utilising an LSTM is based on the winning solution. Since the main success of this method was due to the effective engineering of the precipitation features (such as the determination of the most effective precipitation window size), an approach that contained this inherent learning process seemed appropriate. As a result, utilising the LSTM architecture for learning the high level features in sequential data, such as precipitation, is a good approach for this study.

Chapter 3

Data Preprocessing and Analysis

This chapter presents the datasets utilised for the training and testing of the flood extent prediction models. Details of the data collection and feature extraction process will be discussed. This will include the process of approximating and segmenting flooding from satellite images, and the corresponding quantification of flood extent for a given area. Furthermore, data analysis will be conducted on the given datasets, followed by a description of data augmentation process.

3.1 Data Collection

3.1.1 Datasets

In order to effectively train a neural network, a sufficient quantity of data is needed. As a result of this, additional data collection is often required. Each dataset was downloaded from Google Earth Engine (see Appendix A.1). The datasets utilised are the following (where N represents the number of datapoints/grid squares containing the various features):

- Mozambique Floods 2019: Coastal Flooding + Riverine Flooding (N = 32,039)
- Malawi Floods 2019: Riverine Flooding (N = 51,546)
- Kenya 2020: Flash Flooding (N = 26,271)

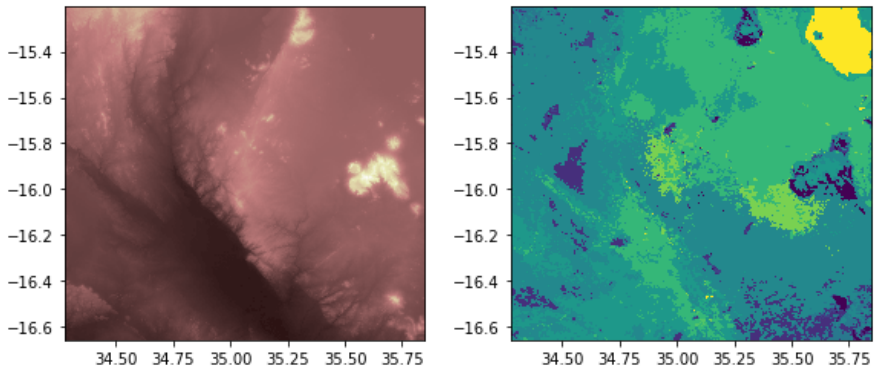
3.1.2 Method for extracting features

Considering that each flood event can have features unique to that specific flood which determine the extent of flooding, it is important to attain the most pertinent features in flood extent prediction. This is particularly important as a single flood event may have multiple sources of flooding for one particular area. Previous research outlines a list of flood condition factors that vital for modelling flood susceptibility [43]. Therefore these features will be used for modelling in this study.

In order to attain these features, Google Earth Engine is used to download the raster images where it is then possible to extract the relevant features. Figure 3.1 presents some examples of raster images downloaded from Google Earth Engine for a specific area. Once these images are downloaded, statistics (e.g. mean, mode, standard deviation) can be applied to extract the different features or information to be used in the modelling. The following is a list of features used in this study for the modelling of flood extent (where r is the repetition rate of the satellite i.e. how long it takes to capture a full image):

- **Elevation** - constant: Scalar-valued feature that models elevation at a resolution of 1 arc-second (approx. 30m) [16].
- **Slope** - constant: Scalar-valued feature derived from the Elevation raster image using the slope calculation.
- **Soil Moisture** - repetition rate, r=3: Scalar-valued feature that represents the surface soil moisture at a resolution of 0.25 arc degrees [12].
- **Soil Organic Carbon** - constant: Scalar-valued feature that represents the amount of soil organic matter [38].
- **Clay Content** - constant: Scalar-valued feature that represents the quantity of clay content (g/kg) in the soil [38].
- **Total / Max / Standard Deviation Precipitation** - repetition rate, r=0.25: Scalar-valued feature with different statistics applied to high resolution data at 0.05 arc degrees [18].
- **Distance to River** - constant: Scalar-valued feature derived from calculating the distance from the centroid of a grid to the closest river in a mask image, at a resolution of 0.01 arc degrees [4].

- **Land Cover Type** - constant: Categorical data with a range from 1-17 representing the dominant land type at a resolution of 500m [17].
- **Normalised Difference Vegetation Index (NDVI)** - repetition rate, r=1: Scalar-valued feature which represents live green vegetation at a resolution of 1km [14].



(a) Elevation. (b) Land Cover.

Figure 3.1: Example of raster images based on the EPSG:4326 coordinate system.

3.1.3 Extracting flood fraction

One challenge with using machine learning techniques for flood prediction is the lack of accurate data for training purposes. However, due to satellite technology it is possible to approximate flooding which can be used for this purpose. To acquire flood data, a change point detection algorithm is employed [23]. This approach applies a calculation on the before-flood satellite image and the after-flood satellite image which results in a raster layer showing the degree of change per pixel. Then a predefined threshold function is applied where values above a certain threshold will be flagged as flooding. To reduce false positives within the flood extent layer, additional datasets are utilised to mask out all areas which are covered by water for more than 10 months of the year.

The next step is to calculate the fraction of flooding for each square in the gridded map, where the flood mask can be used in parallel with a grid of coordinates to extract the flood fraction for each square. This is achieved by calculating the fraction of overlap between the flood polygon (i.e.flood) and the grid polygon (i.e. each square). The resulting flood fraction can be then assigned to that specific square, which in the context of this study is equivalent to one datapoint.

3.2 Data Analysis

To better understand the data, some analysis on the datasets used in this study was conducted. Table 3.1 presents the summary statistics for the Mozambique dataset, Malawi dataset and the Kenya dataset. The results show that all datasets have near identical measures of centrality for flood extent with all values close to 0. Additionally, for each dataset the target distribution is highly positively skewed (i.e. long tail to the right of the distribution) and high-level of kurtosis (i.e. the sharpness of the peak in the distribution). As the target feature is a fraction (i.e. range 0-1) and therefore bounded, we can see that the distribution tends to the lower bound of 0.

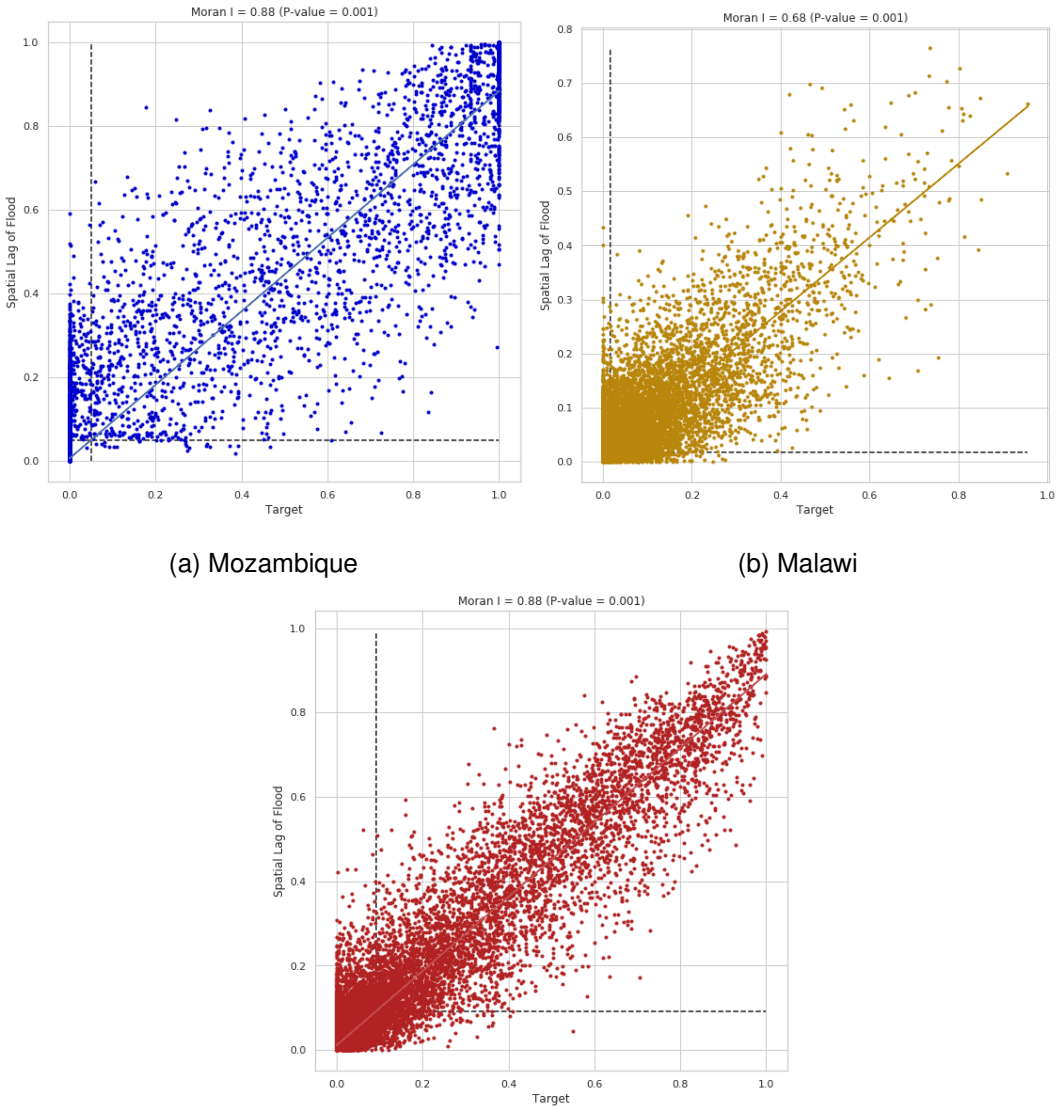
Dataset	**Mozambique**	**Malawi**	**Kenya**
Count	32200	51546	26271
Mean	0.0505	0.0177	0.0915
Median	0.0	0.0	0.0
Mode	0.0	0.0	0.0
Std	0.1935	0.0654	0.2020
Kurtosis	14.74	39.22	5.76
Skewness	3.97	5.63	2.55

Table 3.1: Summary statistics for flood extent in the datasets used for this study.

The Malawi dataset has the highest level of skewness (5.63) and kurtosis (39.22). Then Mozambique has the second highest level of skewness (3.97) and kurtosis (14.74) with Kenya having the least amount of skew (2.55) and kurtosis (5.76). Distributions with high levels of skew can be problematic in learning tasks. The reason for this is due to the model tending to over predict values towards the measure of centrality and failing to predict values in the long tail of the distribution. One solution to this problem is to use data augmentation techniques such as oversampling the areas of the distribution that have less occurrence in the data (e.g. flood values = 1) or by other techniques that can maintain the original attributes of the dataset (see section 3.3).

To assess the spatial dependencies in the various datasets, we look at the level of spatial autocorrelation present in the data. Spatial autocorrelation looks at the level of similarity in values between observations, related by specific locations of those values. The spatial autocorrelation metric of Moran's I is similar to correlation: positive spatial autocorrelation means that values of similar value are generally grouped together; neg-

ative spatial autocorrelation means that values of similar values tend to be scattered; zero spatial autocorrelation means that there is no spatial dependencies in the data and therefore *spatial randomness* exists. Consequently, the greater the Moran's I statistic, the greater the amount of local spatial dependencies exist in the data.



(a) Mozambique

(b) Malawi

(c) Kenya

Figure 3.2: Spatial Autocorrelation in Training and Test Sets.

Figure 3.2 presents the spatial autocorrelation that exists in the three datasets used in this study. The Mozambique and Kenya datasets both report a spatial autocorrelation of 0.88 while the Malawi dataset reports a lower spatial autocorrelation value of 0.68. Each statistic reports a low p-value (0.001) which indicates that all the tests for spatial autocorrelation are statistically significant. As a result, this shows that each dataset has

relatively high levels of spatial autocorrelation which reinforces the idea of developing a solution that can effectively exploit local spatial dependencies that exist in the data.

3.3 Data Augmentation

Deep learning techniques are only effective when a sufficiently large dataset is available for model training [8]. Therefore, data augmentation is often needed to supplement the existing training datasets. Data augmentation is particularly important in the case of modelling flood events as positively labelled data is uncommon due to the irregularity of flood events. In addition, due to the repetition rate of satellites, missing data is quite common. Nevertheless, to solve the problem of inadequate data for training, a spatial grid staggering data augmentation technique is used.

Spatial grid staggering utilises the existing grid to augment the existing labelled dataset. This approach uses different staggering along the axes to extract additional flood data. For instance, one approach would be to stagger the x-axis by 0.05 degrees, with the y-axis values remaining constant. For the task of flood prediction, the gridded map consists of 1km sq coordinate areas. So in this case, staggering the grid by 0.05 would overlap the original gridded squares (approximately by half). Then the opposite could be applied where the y-axis is staggered and the x-axis remains constant. Furthermore, another approach would be to use a combination of both.

We see that after applying data augmentation to the Mozambique dataset, the number of transactions increases to 86,581 datapoints. Upon data augmentation, we see that the Moran's I statistics remains high with a value of 0.87. The invariance of this statistic in the context of data augmentation is important so that the original attributes of the data are maintained. Data augmentation techniques can only be effective if the inherent spatial features in the data can be maintained.

In addition, we see that the intended outcome of creating a more balanced dataset is achieved This is the case as we see the mean target value in the augmented dataset increase to 0.075 with the corresponding skewness being reduced. Although the change in mean and skewness are small, this observation shows that through simple data augmentation techniques, such as spatial grid staggering, we can create a larger, more balanced dataset while maintaining the original attributes of the dataset.

Chapter 4

Methodology

4.1 Modelling

This chapter provides details of the deep learning models used in this study. The Long Short-Term Memory (LSTM) network is the main architecture applied to all models. Therefore this will be described in detail. Following this, each extension of the LSTM will be introduced while outlining the difference and providing motivation for using the specific model.

4.1.1 Long Short-Term Memory Networks

Long short-term memory (LSTM) networks are a type of recurrent neural network (RNN) used in the field of deep learning. The RNN architecture is different then feed-forward neural networks, as they contain a feedback loop. These type of networks are generally used to model sequential data such as speech [19].

In temporal problems, the LSTM architecture is designed to make certain information from the past redundant by *forgetting* certain information through a sigmoid function (also known as a forget gate). This follows from (4.1). Equation (4.2) represents the input gate where the sigmoid function processes the hidden state h_{t-1} and the state x_t and outputs a number i between 0 and 1. Moreover, a value of $i = 0$ equates to *forget* all this information and $i = 1$ equates to *remember* all this information. This process has shown to be effective at modelling long-term sequential dependencies in data with a temporal nature while mitigating the vanishing gradient problem [21].

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{4.1}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{4.2}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{4.3}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4.4}$$

To decide exactly what information is to be stored in the cell state, a tanh layer is introduced which outputs a vector of values $\tilde{C}_t$ as shown in (4.3). The input gate is then combined with this to create an update to the state. The update consists of multiplying the old state C_{t-1} with the forget layer f_t and the input layer i_t is multiplied with the vector $\tilde{C}_t$ and their addition results in the the cell state C_t in (4.4). It is this process that implements the *forgetting* of previous redundant information.

In the context of flood extent prediction, the temporal nature can be modelled through this process. For example, the cell state could include information about the level of precipitation at time $t-k$, then a new precipitation value at $t-k+1$ is introduced to the model. At this point the model may want to encode some of this information regarding the precipitation levels at timestep $t-k$ into the cell state at timestep $t-k+1$. This new information could be more recent levels of heavy rainfall, that make the earlier rainfall less important. At this point, the updating of the cell state would learn how much information should be kept.

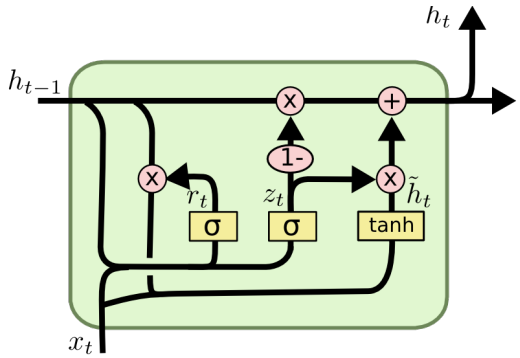


Figure 4.1: LSTM cell. Image taken from [3]

The remaining step is to then decide the output based on the cell state. This follows (4.5) where the output gate o_t utilises a sigmoid function applied to the weighted sum of the input, hidden state and the cell state. Furthermore, the hidden state h_t is found by first squeezing the output of the cell state C_t with a tanh function and then multiplying with the output gate o_t.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{4.5}$$

$$h_t = o_t * \tanh(C_t) \tag{4.6}$$

The rest of this section outlines the models used in this study. Each model consists of the LSTM architecture, described previously, to model flood extent. These variants of the LSTM network will be discussed, including motivation for extending the original architecture.

4.1.2 LSTM Autoencoder

In general, an autoencoder is an unsupervised learning task (i.e. where no labelled data is present) which utilises an artificial neural network [33]. In unsupervised learning, the architecture of the artificial neural network is designed in such way that the information is forced through a bottleneck which ultimately compresses the representation of the original input. If a relationship exists between features, this compressed representation can be learned. Additionally, this architecture can leveraged in supervised tasks where the result of this process would be a feature vector used for further modelling.

One variation of the LSTM is the LSTM-Autoencoder. This implementation leverages the autoencoder to learn a feature vector for sequential data using an Encoder-Decoder architecture [33]. In this scenario, the datasets containing sequential data is configured to read in a sequence, encode it, decode it and try to re-create the input sequence. If the model is able to recreate the sequence, the model has learned the compressed representation which is sufficient for modelling. Once this compressed representation of the data has been learned, the decoder element is not needed and the remaining encoder can be used for a supervised learning task such as an LSTM network.

The use of the LSTM-Autoencoder in predicting flood extent is motivated due to successful applications in anomaly detection [33] and rare-event classification [44]. The idea is that an autoencoder is applied to data to effectively learn the *normal* labels (i.e. non-anomalous data) to learn a compressed representation of the data. If the error in the re-creation of the data is high, then the label would be classified as anomalous. In the context of flood prediction, the autoencoder would learn the compressed representation of the sequential data extracted from the satellite imagery to effectively label given coordinates as a flood or not and output the labels for the future timesteps. However, in this study the focus is not on outputting a sequence of classifications for future timesteps such as ($t, t+1, t+2$, etc.). Instead, this study focuses on the flood extent where we want to predict how much of a given square is flooded at timestep t only. So

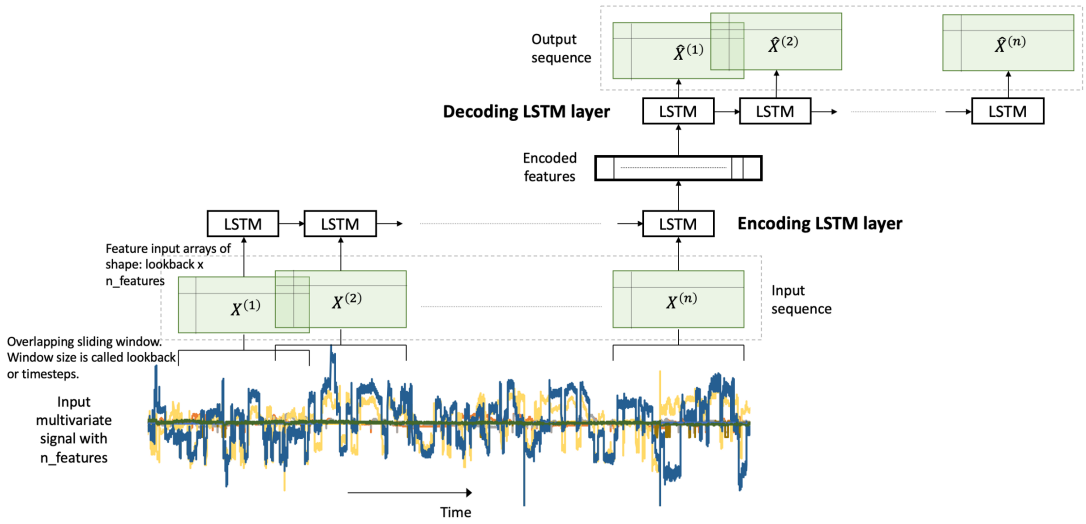


Figure 4.2: Multivariate LSTM-Autoencoder. Image taken from [2]

only a linear activation in the output layer is needed for the regression task.

4.1.3 Multi-Input LSTM

When utilising satellite imagery for machine learning tasks, it is often the case that the features extracted have a mix of data types. In addition, these images are generally not available for every time step due to satellite repetition rates. To summarise, an effective models needs to be effective at dealing with the following two cases: (1) the data is temporal by nature but it is not available for every timestep (e.g. soil moisture); (2) the data has a very long repetition rate (e.g. 365 days) and can be thus considered a constant feature (e.g. elevation). So a model that is robust to to different data types is desirable.

For LSTM networks, the input layer consists of a number features to be modelled for each timestep. In this scenario, the LSTM is expecting that each feature to be modelled to be temporal. A naive approach would be to consider any constant features as temporal features that do not vary with time. However, adding constant features to a LSTM may add redundant information through the layers and at the cost of greater model performance, which would therefore be counter-productive. Another approach would be to consider temporal features and constant features as two separate inputs to the network. In cases where the input into a neural network has different data types, it is commonly referred to as multi-input [39].

Figure 4.3 outlines the overall interactions between the different types of layers in the multi-input LSTM. This architecture is divided into two components: (1) for mod-

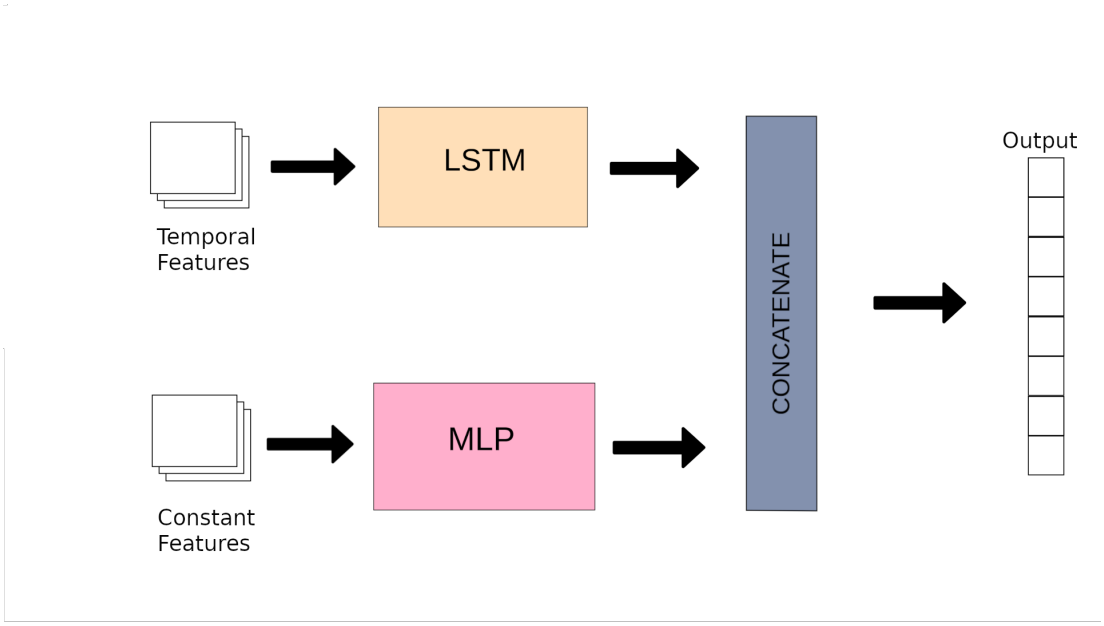


Figure 4.3: Multi-Input LSTM flow of operations.

elling the temporal features, the LSTM model will be utilised; (2) for modelling the constant features a feed-forward network (MLP) will be used. These two components will be combined at a subsequent layer before being propagated to the output layer.

4.1.4 Convolutional LSTM Network

When modelling spatial data such as images, Convolutional Neural Networks (CNN) are generally used. The reason is that the CNN architecture is able to leverage the fact that nearby pixels are more strongly correlated than distant pixels and thereby exploiting the local spatial dependencies [42]. The spatial dependencies in the data are analysed through filters. This filter moves across the image with a predefined size (e.g. 3x3 pixels) and a stride length (similar to a step size) which determine the amount of overlap when moving through the while image. For each filter, a calculation is made called a convolution. This step reduces the amount of weights to learn in the model while focusing on the most interesting areas in space. Once the convolution step is executed a feature map is created for each filter. Following this normal deep learning protocol is applied.

To effectively model spatio-temporal relationships ConvLSTM was developed [46]. This architecture was devised to be used for spatio-temporal sequence forecasting of rainfall intensity over a short period of time. This method was shown to surpass state-of-the-art predictions by utilising satellite imagery. However, in order to achieve this

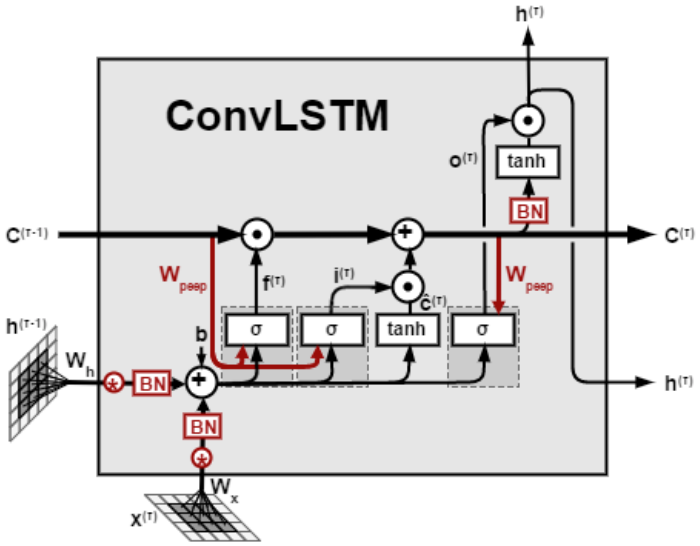


Figure 4.4: ConvLSTM Cell. Image taken from [1]

lots of data was available as rainfall is not an extreme rare event. In the case of flood extent prediction, a sufficient amount of data is not available to train a ConvLSTM on full image data. This is due to the fact that it uses a 2D filter to extract features from the images.

Figure 4.4 shows the operation of the ConvLSTM cell for 2D data, such as an image. Considering this study does not have the adequate amount of satellite imagery to be used as training data, an alternative approach will be used. A ConvLSTM with 1D filter will be used to exploit local spatial dependencies that exist within various features (e.g. elevation). By utilising this technique, local spatial dependencies can be learned without the need for additional training data, as sequential data is sufficient for this purpose.

4.1.5 Multi-Input ConvLSTM

To effectively model the temporal and spatial dependencies in the data and removing any redundant constant features from the temporal modelling component, a novel multi-input ConvLSTM was developed. The Multi-Input ConvLSTM adopts a similar architecture to the Multi-Input LSTM, except now replacing the LSTM operation with the ConvLSTM component.

Figure 4.5 outlines the flow of operations for the Multi-Input ConvLSTM. This architecture is also divided into two components: (1) for modelling the temporal features, the ConvLSTM model will be utilised; (2) for modelling the constant features a

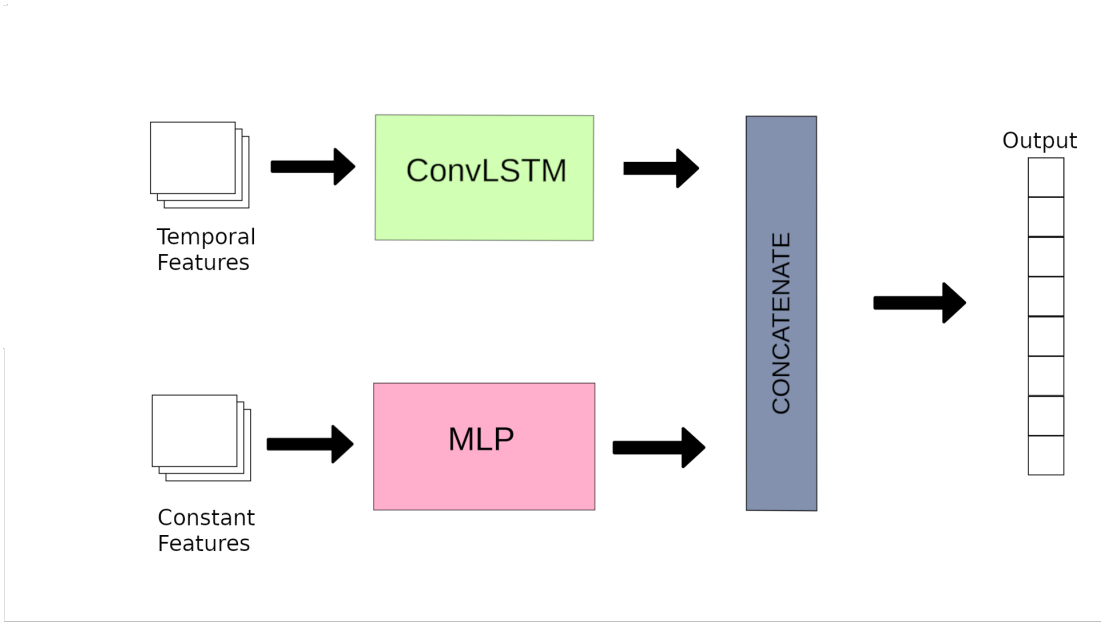


Figure 4.5: Multi-Input ConvLSTM flow of operations.

feed-forward network (MLP) will be used. These two components will be combined at a subsequent layer for before being propagated to the output layer.

In the next chapter, the results of applying these models to various flood events will be discussed. For each model, the hyperparameters used in the experiments will also be outlined.

Chapter 5

Evaluation and Results

This chapter outlines the evaluation process, hyperparameter configuration and the results of the deep learning experiments conducted in this study. Furthermore, analysis of the results will be provided.

The first section will outline the data used for the evaluation along with the validation strategy used. The second section will present the models used in the experiments including justifications for using certain hyperparameters. The chapter will conclude with a critical analysis of the results.

5.1 Evaluation

To determine the accuracy of the models, the Root Mean Squared Error (RMSE) is applied:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

where n is the number of datapoints, $\hat{y}_i$ is the predicted value and y_i is the target value that we are trying to predict. This metric quantifies the prediction error (difference between the predicted values and target values), averaged over the number of samples. In view of the fact that the errors are squared before they are averaged, this assigns a greater weight to large errors. Then the squared root is taken to revert back to the original scale of values.

The motivation for using this metric is due to the fact that it penalises large errors. In the problem of predicting flood extent, it is desirable to penalise a prediction more

if there is a greater discrepancy between the predicted value and the ground truth. For example, if a flooded square has ground-truth value of 0.5, it makes sense to penalise a prediction of 0.10 more than a prediction of 0.49. So this metric should be used where precise predictions are needed and when it is expected that the errors will vary in magnitude. Additionally, this was the metric chosen in the Zindi competition and would therefore allow for appropriate comparisons with the winning solution.

5.1.1 Data

The data used for training and testing were carefully selected. The reason for careful selection was to test if the models could generalise to various flood events. To test this the following data sets were used:

- 1. Malawi used for training and testing
- 2. Mozambique used for training and Malawi used for testing
- 3. Mozambique used for training and Kenya used for testing

First the models were trained and tested on homogeneous data to better understand the ability of the models to perform on data of a similar nature. This is a similar to the setup for the Zindi competition, as both the training and test set were extracted from identical coordinates in Malawi. Thus, the majority of the underlying features in both training and test sets being similar (e.g. elevation does not vary over time).

To test on heterogeneous data, other datasets were selected. These datasets were selected due to their different geographic features (e.g. land cover type; elevation etc.) but also due to the varying types of flood events. For instance, if a model was trained using pluvial flood data and the target flood event was due to fluvial flooding, this would make the prediction problem a much more difficult task. Therefore, this would test the ability of the models to generalise well to other flood events.

5.1.2 Validation

To validate the model as it was being trained, the training data was further split into a training set and a validation set. This consisted of 80% for training the model and 20% for validation. The purpose of this split is to test the performance of a model in parallel allowing for a predefined stopping condition for the training process to be set.

For each model that was trained, a specific validation strategy was used. Each model was trained using a max epoch value of 1000 and early-stopping applied. The early-stopping strategy evaluates the model on the validation set as it is training. Once the validation error stops decreasing beyond a pre-defined error threshold for a number of epochs, the training then terminates and the weights with the lowest RMSE on the validation set are kept.

In the experiments, each model was cross-validated (K-Fold=3) with the reported result representing the mean over the three prediction results. In addition, the cross-validations enables the experiments to report the standard deviation for each model and therefore allowing a measure of confidence for each model to be reported.

5.2 Experiments

5.2.1 Hyperparameter Configuration

The choice of loss function used for training the models was the Huber loss. The reason for choosing the Huber loss instead of using the RMSE was because the Huber loss is robust to outliers. As seen in section 5.1, the RMSE is shown to penalise predictions that have a greater error magnitude which is desirable in testing scenarios, as it is desirable to have an accurate prediction for the flood extent. However, when it comes to optimising the loss function, it is important to be robust to outliers. To summarise, the Huber loss was chosen in the context of flood extent prediction as it would better handle the infrequent cases of high-valued flood extent.

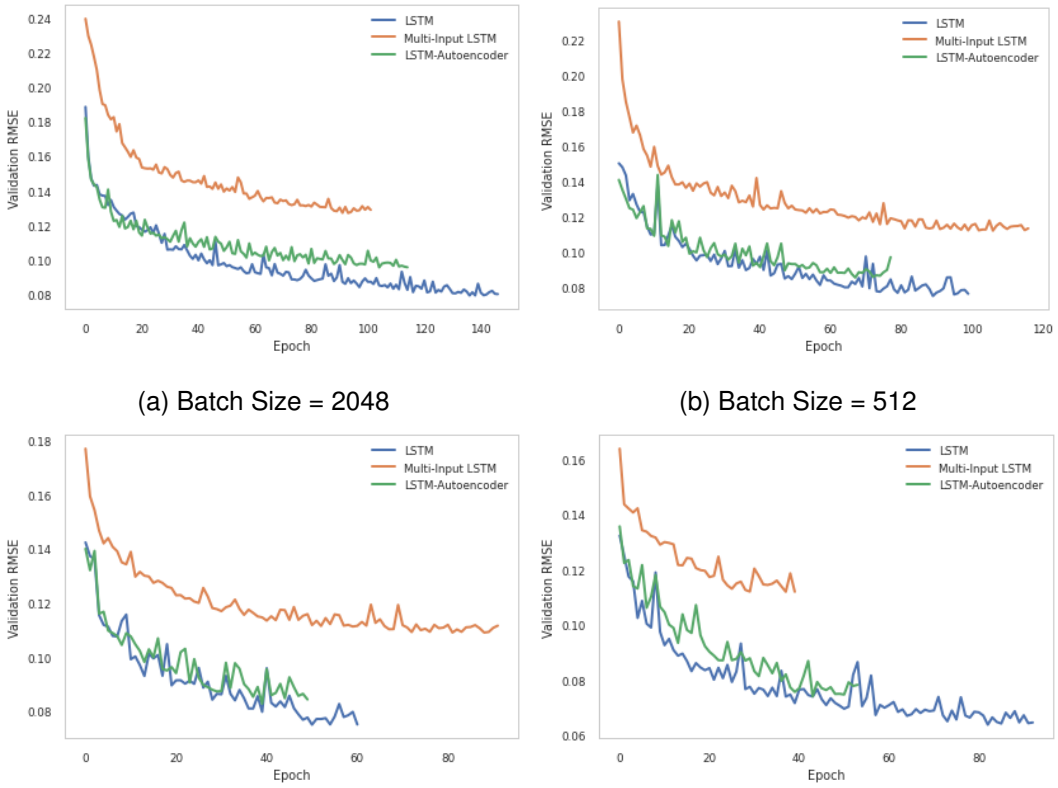
$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{5.1}$$

Equation (5.1) shows the Huber loss where it is quadratic for errors of small magnitude and linear in errors of greater magnitude. Due to the nature of flooding data, outliers are common. As a result of this, it is necessary to have a loss function robust to outliers.

The Adam optimiser [24] was chosen for the learning process as it has an adaptive learning rate which reduces the need to optimise the learning rate hyperparameter. As it uses an adaptive learning rate, the optimiser achieves greater training speed and increases the rate of convergence. The optimiser achieves this by utilising an exponential moving average of the gradient and the parameter. In this study, the default values that

are used in TensorFlow [7] (i.e beta1 and beta2) were used, which control the decay rates of these averages.

The models in this study all used mini-batching which updates the error gradient after seeing a only a subset of the data. This is in contrast to full-batching where the error gradient is computed after one full epoch of the data. In general, the batch size of a neural network controls the dynamics of training. When choosing an optimal batch size, it is common to choose a batch size based on the lowest validation error. However, lower validation errors attributed to lower batch sizes are less trustworthy, as they have not seen as much of the data, and may have a poor generalisation error.



(a) Batch Size = 2048

(b) Batch Size = 512

(c) Batch Size = 128

(d) Batch Size = 32

Figure 5.1: Validation RMSE

In order to decide the optimal batch size, several models were compared. Figure 5.1 outlines the dynamics of the validation RMSE for several models. As you can see, the LSTM and LSTM-Autoencoder have the lowest RMSE with a batch-size=32, while the Multi-Input LSTM achieves a better result with a batch-size=128. However, considering the data is highly positively skewed, lower batch sizes are less reliable. For instance, a lower batch size of 32 would likely be zero-inflated (i.e. a batch containing all zero target values). In the validation RMSE, this would show that the model is mak-

ing accurate predictions which would be misleading. When looking at a batch size, a stable validation error is desirable. This can then be attributed to the greater likelihood of a batch with more positive values. Therefore, despite the lower validation error in smaller batch sizes, a batch-size of 2048 was chosen based on the stable validation error and the greater likelihood of generalising better to the test set.

5.2.2 LSTM

For the LSTM model the most important hyperparameters to choose are the number of hidden layers and the number of units in each layer. For this task the number of layers were chosen from between 2-4 layers and the hidden units between (32, 128). As the difference in performance between 64-128 units and 2-3 layers was negligible, the smaller network was chosen. Therefore, an architecture with 2 hidden layers and (64, 32) hidden units was chosen, respectively. As the model was not sensitive to changes in these hyperparameters, those specific hyperparameter values were used because the smaller the network, the less amount of parameters are needed to be learned, resulting in faster convergence. To reduce chances of overfitting, the second layer applied dropout. The purpose of the dropout is to randomly remove a certain percentage of hidden units to help tackle overfitting and improve generalisation. For this model, a dropout rate of 0.10 was used.

5.2.3 Multi-Input LSTM

Considering the Multi-Input is a variant of the LSTM, this model adopted many of the hyperparameters from the previous model to allow adequate comparison between models. Therefore the temporal element of this model uses the same architecture and hyperparameters as the LSTM.

For the Multi-Input LSTM the changes in hyperparameters are implemented on feed-forward layers that model the constant features. In conjunction with the temporal element, the best performing model consisted of a feed-forward network with one hidden layer, containing 16 hidden units with a ReLU activation function.

5.2.4 ConvLSTM

As the ConvLSTM model introduces convolutions, additional hyperparameters needed to be considered. For the ConvLSTM layer, the number of filters and the size of the

kernel are to be considered. For the filter size, values of 32 and 64 were chosen with the 64 size having a lower validation error. Considering the ConvLSTM layer will be used for sequential data the kernel size must be one-dimensional. Nevertheless, a kernel size of (1,3) and (1,5) were compared. The kernel of size (1,5) had a lower validation error due to ability of the model to capture a greater amount of the spatial dependencies in the data. Therefore, this kernel was chosen for the experiments.

5.2.5 Multi-Input ConvLSTM

The Multi-Input ConvLSTM is a combination of both the multi-input LSTM and the ConvLSTM. Therefore it adopts many of the same hyperparameters as before. As this has the most weights to be learned a limited hyperparameter search was done. So for the purpose of this study, this model inherited the hyperparameters with the lowest validation error from the Multi-Input LSTM and the ConvLSTM models.

5.2.6 Benchmark

The benchmark to be utilised in the experiments for this study, is the winning solution for the Zindi competition (see section 2.2.2). The solution of a LGBM with model averaging and is considered current state-of-the-art in flood extent prediction. As this model is not a neural network but a variant of gradient boosted tress, different hyperparameters are considered. The hyperparameters chosen for the winning solution were as follows:

- The number of boosting iterations = 221
- The learning rate = 0.07
- The maximum depth for the tree model = 8

5.3 Results

This section presents the results for the experiments of this study. The first part of this section presents the results of the models where the training data and test data are homogeneous (i.e. same flood event). The second part of this section presents the results of experiments for the models where the training and testing data is heterogeneous (i.e. different flood events).

5.3.1 Homogeneous Data

Table 5.1 exhibits the results for the models utilised in this study. These models have been trained and tested on homogeneous data, where the training and testing data are sourced from the same flood event and type. In this case, the models were trained and tested on Malawi flood data. The results a report the RMSE for each model and standard deviation.

Model	**Dataset**
	Malawi
LSTM	0.0525 ± 0.0004
LSTM-Autoencoder	0.0542 ± 0.01
Multi-Input LSTM	0.0532 ± 0.001
ConvLSTM	$\mathbf{0.0519 \pm 0.0002}$
Multi-Input ConvLSTM	0.0572 ± 0.001
LGBM	0.1021 ± 0.0001

Table 5.1: Results of training on Malawi flood data and testing on Malawi flood data using RMSE.

In the scenario of homogeneous data, the ConvLSTM surpasses the other models. It achieves the lowest RMSE of 0.0519 with the second lowest RMSE coming from the LSTM model achieving 0.0525. The LSTM-Autoencoder and Multi-Input LSTM report slightly higher RMSE than these models, with 0.0542 and 0.0532 respectively. The Multi-Input ConvLSTM is the worst performing deep learning technique with an RMSE of 0.0572. However, the deep learning techniques developed for this study all surpass the LGBM model as that model reports an RMSE of 0.1021. From Table 5.1 we see for all the deep learning techniques developed for this study had a RMSE range of between 0.0519 and 0.0572.

5.3.2 Heterogeneous Data

Table 5.2 presents the results for the models developed for testing the ability for the models to generalise to other flood events. These models have been trained and tested on heterogeneous data, where the training and testing data are sourced from the different flood events with different flood types. In this case, the models were trained on Mozambique data and tested on both Malawi and Kenya flood data. The results in the

table report the RMSE for each model with standard deviation.

Model	**Dataset**	
	Malawi	**Kenya**
LSTM	0.1471 ± 0.03	0.2270 ± 0.01
LSTM-Autoencoder	0.1117 ± 0.02	0.2333 ± 0.01
Multi-Input LSTM	$\mathbf{0.0803 \pm 0.01}$	$\mathbf{0.2120 \pm 0.01}$
ConvLSTM	0.1350 ± 0.02	0.2180 ± 0.01
Multi-Input ConvLSTM	0.0890 ± 0.01	0.2169 ± 0.02
LGBM	0.2815 ± 0.002	0.2554 ± 0.01

Table 5.2: Results of training on Mozambique flood data and testing on Malawi and Kenya flood data.

For the Malawi data, the Multi-Input LSTM achieves the lowest RMSE. This model has a RMSE of 0.0803 and the the second lowest RMSE coming from the Multi-Input LSTM model achieving 0.0890. The LSTM-Autoencoder reports a slightly higher RMSE of 0.1117. The LSTM and ConvLSTM report the worst RMSE results with 0.1471 and 0.1350, respectively. However, the deep learning techniques developed for this study all surpass the LGBM model as that model reports an RMSE of 0.2815.

For the Kenya data, the Multi-Input LSTM again achieves the lowest RMSE. This model has a RMSE of 0.2120 and the the second lowest RMSE coming from the Multi-Input ConvLSTM model with a RMSE of 0.2169. The ConvLSTM reports a slightly higher RMSE of 0.2180. The LSTM and LSTM-Autoencoder report the worst RMSE results with 0.2270 and 0.2333, respectively. However, the deep learning techniques developed for this study all surpass the LGBM model as that model reports an RMSE of 0.2554.

5.4 Analysis of Results

In the previous section, the experiments were conducted on two types of data: homogeneous data and heterogeneous data. The purpose of the former experiment was to compare the models developed in this study with the current state-of-the-art model in flood extent prediction on flood events of a similar nature. The latter experiment was conducted to assess the ability of each model to generalise to different type of flood event.

Section 5.3.1 demonstrates the ability of each variant of LSTM network to model homogeneous flood data and to outperform the LGBM model. The evidence suggests that the LSTM network is well-suited to the problem of flood extent prediction due to their ability to incorporate the temporal features instead of simply considering all the features to be constant, similar to case of the LGBM model. However, based on these results, one approach cannot be preferred over another as the difference in the RMSE is not statistically significant.

One reason why the models that utilise convolutions do not perform any better here is due to Malawi having less spatial autocorrelation. In section 3.2, the Moran's I statistic was computed for each dataset used in the study. The results of this these statistical tests showed that Malawi has a lesser degree of spatial autocorrelation (0.68), leading to less spatial dependencies in the data to exploit. If a poses no spatial attributes the convolutions become redundant in the data resulting in no increase in the performance for those particular models.

In section 5.3.2, the ability of each model to generalise to different types of floods is assessed. The general trend here is that both the Multi-Input LSTM and Multi-Input ConvSTM outperform the other models for both the Malawi and Kenya datasets. When comparing the performance across datasets, we see the stronger performance on the Malawi dataset in comparison to the Kenya dataset.

Figure 5.2 presents the distance to water (e.g. river) for the Mozambique, Malawi and Kenya datasets. We see that the Mozambique and Malawi have similar distributions. However, we see that Kenya has a different distribution with a the distribution having a greater distance to water and more variable distances. Considering that the Mozambique dataset was used for training and Malawi dataset used for testing, it is plausible that the similar distributions added to the ability of the models to generalise better. On the contrary, we see that the models do not generalise as well to the Kenya dataset. This can be attributed to the fact that the underlying distribution for the dis-

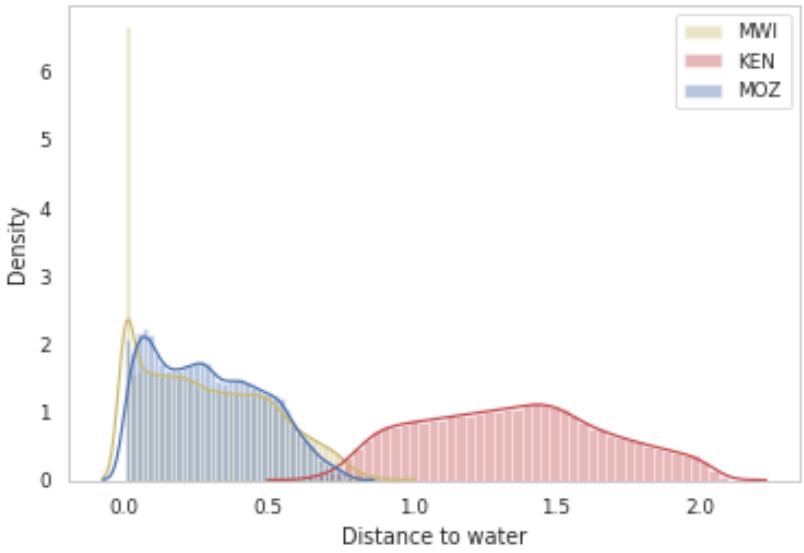


Figure 5.2: Distance to water for each of Mozambique, Malawi and Kenya datasets.

tance values are vastly different.

Furthermore, we see that the LSTM-Autoencoder improves upon the standard LSTM architecture for the Malawi dataset. As discussed earlier, the LSTM-Autoencoder architectures generally perform well for extreme rare event or anomaly detection tasks. In Table 3.1 we seen that Malawi had highest level of skewness in the data (5.63). The LSTM-Autoencoder performs less well on the Kenya dataset where the data has the least amount of skew and therefore, the least amount of outliers. This evidence suggests that that the LSTM-Autoencoder model was able to learn the compressed version of the data and to effectively distinguish flooded squares and non-flooded squares on highly-skewed data.

The best performing models for both test datasets is the Multi-Input LSTM. We seen this model is an extension of the standard LSTM architecture by dealing with temporal features and constant features separately. We also notice that the Multi-input ConvLSTM performs second best, showing that implementing an LSTM layer in conjunction with the multi-input architecture is more effective than implementing the ConvLSTM. Considering the high levels of spatial autocorrelation in the Kenya dataset, it was unexpected to see that the 1-Dimensional filter applied in the ConvLSTM layer is not able exploit any local spatial dependencies in the data.

Chapter 6

Discussion

Flood extent prediction is a complex task, challenged by spatial outliers, the sparsity of training data and the inaccurate measurements of the features used to build predictive flood models. The main objectives of this study were to establish a process in which limited data could be utilised for training spatio-temporal models, to determine the best neural network architecture for the problem and to develop a model that could generalise to other flood types. The models explored were the LSTM, LSTM-Autoencoder, ConvLSTM, Multi-Input LSTM and the Multi-Input ConvLSTM. Each model was chosen for specific reasons but with the overall goal of solving the flood prediction problem.

The results of the study showed that converting the original flood extent prediction problem into a temporal problem outperforms the current state-of-the-art model in generalising to other flood events. The success of this can be attributed to using data augmentation to acquire the sufficient quantity of data necessary to apply LSTM networks to a temporal problem. The best performing model in generalising to other flood events was the Multi-Input LSTM as it could model both constant features and temporal features, without adding redundant information to the temporal problem. The ConvLSTM did outperform the standard LSTM architecture on certain tasks but the Multi-Input ConvLSTM failed to outperform the Multi-Input LSTM. In datasets with a high level of positive skew in the target distribution, the LSTM-Autoencoder performed well. This was attributed to the ability of the model to learn the compressed features of the data and to effectively distinguish the outliers (i.e. high levels of flood) from the highly frequent values (i.e no flood). Considering their ability to model this effectively, a potential extension to the successful Multi-Input LSTM model would be to incorporate an intermediate bottleneck layer to inherit these qualities from the

LSTM-Autoencoder model. Nevertheless, the LSTM-Autoencoder architecture proves to be a good model candidate for flood extent prediction.

This study also pointed out current limitations of flood extent prediction. We showed that the data augmentation process developed was effective for creating bigger datasets while maintaining spatial dependencies. However, despite the high levels of spatial autocorrelation in the datasets, utilising a 1-D filter for the ConvLSTM architecture proved ineffective. This showed that the absence of sufficient satellite imagery data and the absence of a 2D filter to extract spatial features, does prohibit the ability of the ConvLSTM to exploit local spatial dependencies.

6.1 Future Research

The success of the multi-input architecture leads to interesting research directions. The ability for this type of architecture to effectively model problems with both temporal and constant elements make this architecture applicable to other similar problems. For instance GIS problems such as drought prediction [27], the tracking of bird migration [40] and environmental issues [47] all utilise satellite imagery for their respective problems. Therefore, deep learning techniques, such as the Multi-Input LSTM, that can incorporate both temporal and constant features extracted from satellite imagery would be desirable for various problems.

Considering the LSTM-Autoencoder showed ability to effectively model problems with a large skew, one future research direction would be to investigate a larger network for this model. One possible extension would be to investigate the efficacy of incorporating the multi-input architecture and as a result, combine the desirable traits of both models. An effective implementation of this architecture would likely improve on the multi-input LSTM developed in this study.

Recently, ConvLSTM models have shown to be very effective at spatio-temporal problems [46]. However, this study only had sparse training data available for the ConvLSTM which ultimately constrained the ability of the model to effectively exploit local spatial dependencies through a 1D filter. Therefore an interesting approach would be to investigate potential opportunities in spatial specific batch learning [11] which may alleviate this problem. If adequate data is available for training, the ConvLSTM model should be investigated for future flood extent prediction tasks. Alternatively, current research in sparse convolutional neural networks may extend to the ConvLSTM architecture [31].

6.2 Conclusion

This study showed the following: (1) it is possible to develop a data augmentation process that can maintain inherent spatial features to be used for spatio-temporal problems such as flood extent prediction; (2) the overall LSTM deep learning architecture consistently outperformed the state-of-the-art in flood extent prediction, with the best performing model being the multi-input LSTM model; (3) each model was able to generalise to other flood events effectively once certain conditions were met.

The work presented in this study showed the ability of deep learning techniques when applied to flood extent prediction. Not only has this work provided a useful resource to the efforts to provide reliable flood forecasting, but also provides a foundation for future research in the area of deep learning for flood extent prediction.

Appendix A

Data Acquisition

A.1 Introduction to Google Earth Engine

The majority of the data used in this study comes from a cloud-based platform called Google Earth Engine [?]. This platform is a worldview geospatial analysis tool that allows the user of leverage Google's computational resources to help better understand societal issues such as environmental issues and natural disasters. Google Earth Engine utilises large-scale cloud computing resources to collate satellite imagery for the user. This in turn allows the user to download the multi-spectral imagery and allow the necessary features to be extracted for further analysis.

Bibliography

[1] Conv-lstm cell image. https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7. Accessed: 2020-08-16.

[2] lstm-auto. https://towardsdatascience.com/lstm-autoencoder-for-extreme-rare-event-classification-in-keras-ce209a224cfb. Accessed: 2020-08-16.

[3] Lstm cell image. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2020-08-16.

[4] River mask images. http://www.diva-gis.org/. Accessed: 2020-08-16.

[5] Unicef arm 2030 vision: Flood prediction in malawi. https://zindi.africa/competitions/2030-vision-flood-prediction-in-malawi. Accessed: 2020-08-16.

[6] Winning solution: Flood prediction in malawi. https://github.com/belkhir-aziz/Flood-Prediction-in-Malawi-winning-solution-. Accessed: 2020-08-16.

[7] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[8] Ahmad Alwosheel, Sander van Cranenburgh, and Caspar G Chorus. Is your dataset big enough? sample size requirements when using artificial neural networks for discrete choice analysis. *Journal of choice modelling*, 28:167–182, 2018.

[9] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

[10] Christopher M. Bishop. Mixture density networks. Technical report, 1994.

[11] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.

[12] JD Bolten and WT Crow. Improved prediction of quasi-global vegetation conditions using remotely-sensed surface soil moisture. *Geophysical Research Letters*, 39(19), 2012.

[13] Marina Campolo, Paolo Andreussi, and Alfredo Soldati. River flood forecasting with a neural network model. *Water Resources Research*, 35(4):1191–1197, 1999.

[14] K Didan. Mod13a2 modis/terra vegetation indices 16-day l3 global 1km sin grid v006 [data set]. *NASA EOSDIS LP DAAC https://doi.org/10.5067/MODIS/MOD13A2*, 6, 2015.

[15] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016.

[16] Tom G. Farr, Paul A. Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, David Seal, Scott Shaffer, Joanne Shimada, Jeffrey Umland, Marian Werner, Michael Oskin, Douglas Burbank, and Douglas Alsdorf. The shuttle radar topography mission. *Reviews of Geophysics*, 45(2), 2007.

[17] M Friedl and D Sulla-Menashe. Mcd12q1 modis/terra+ aqua land cover type yearly l3 global 500m sin grid v006 [data set]. *NASA EOSDIS Land Processes DAAC*, 10, 2015.

[18] Chris Funk, Pete Peterson, Martin Landsfeld, Diego Pedreros, James Verdin, Shraddhanand Shukla, Gregory Husak, James Rowland, Laura Harrison, Andrew Hoell, et al. The climate hazards infrared precipitation with stations—a new environmental record for monitoring extremes. *Scientific data*, 2(1):1–21, 2015.

[19] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

[20] Kai Han, Yunhe Wang, Chao Zhang, Chao Li, and Chao Xu. Autoencoder inspired unsupervised feature selection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2945. IEEE, 2018.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang. Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, 57(6):114–119, 2019.

[23] Minmin Huang and Shuanggen Jin. Rapid flood mapping and evaluation with a supervised classifier and change detection in shouguang using sentinel-1 sar and sentinel-2 optical data. *Remote Sensing*, 12(13):2073, 2020.

[24] Imran Khan Mohd Jais, Amelia Ritahani Ismail, and Syed Qamrun Nisa. Adam optimization algorithm for wide and deep neural network. *Knowl. Eng. Data Sci.*, 2(1):41–46, 2019.

[25] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154, 2017.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[27] H Kuswanto, IL Yuliatin, and HA Khoiri. Statistical downscaling to predict drought events using high resolution satelite based geopotential data. In *IOP Conference Series: Materials Science and Engineering*, volume 546, page 052040. IOP Publishing, 2019.

[28] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[29] Xuan-Hien Le, Hung Viet Ho, Giha Lee, and Sungho Jung. Application of long short-term memory (lstm) neural network for flood forecasting. *Water*, 11(7):1387, 2019.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[31] Yao Lu, Guangming Lu, Bob Zhang, Yuanrong Xu, and Jinxing Li. Super sparse convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4440–4447, 2019.

[32] Weixin Luo, Wen Liu, and Shenghua Gao. Remembering history with convolutional lstm for anomaly detection. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 439–444. IEEE, 2017.

[33] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[34] Ali H Mirza and Selin Cosan. Computer network intrusion detection using sequential lstm neural networks autoencoders. In *2018 26th signal processing and communications applications conference (SIU)*, pages 1–4. IEEE, 2018.

[35] Amir Mosavi, Pinar Ozturk, and Kwok-wing Chau. Flood prediction using machine learning models: Literature review. *Water*, 10(11):1536, 2018.

[36] Jacob Nogas, Shehroz S Khan, and Alex Mihailidis. Fall detection from thermal camera using convolutional lstm autoencoder. In *Proceedings of the 2nd workshop on Aging, Rehabilitation and Independent Assisted Living, IJCAI Workshop*, 2018.

[37] OECD. *Financial Management of Flood Risk*. 2016.

[38] OECD. *Financial Management of Flood Risk*. 2016.

[39] Ozan Oktay, Wenjia Bai, Matthew Lee, Ricardo Guerrero, Konstantinos Kamnitsas, Jose Caballero, Antonio de Marvao, Stuart Cook, Declan O'Regan, and Daniel Rueckert. Multi-input cardiac image super-resolution using convolutional neural networks. In *International conference on medical image computing and computer-assisted intervention*, pages 246–254. Springer, 2016.

[40] Mattia Pancerasa, Matteo Sangiorgio, Roberto Ambrosini, Nicola Saino, David W Winkler, and Renato Casagrandi. Reconstruction of long-distance bird migration routes using advanced machine learning techniques on geolocator data. *Journal of the Royal Society Interface*, 16(155):20190031, 2019.

[41] KM Sakthivel and CS Rajitha. A comparative study of zero-inflated, hurdle models with artificial neural network in claim count modeling. *International Journal of Statistics and Systems*, 12(2):265–276, 2017.

[42] Chandrama Sarker, Luis Mejias, Frederic Maire, and Alan Woodley. Flood mapping with convolutional neural networks using spatio-contextual pixel information. *Remote Sensing*, 11(19):2331, 2019.

[43] Mahyat Shafapour Tehrany, Farzin Shabani, Mustafa Neamah Jebur, Haoyuan Hong, Wei Chen, and Xiaoshen Xie. Gis-based spatial prediction of flood prone areas using standalone frequency ratio, logistic regression, weight of evidence and their ensemble techniques. *Geomatics, Natural Hazards and Risk*, 8(2):1538–1561, 2017.

[44] Balaram Sharma, Prabhat Pokharel, and Basanta Joshi. User behavior analytics for anomaly detection using lstm autoencoder-insider threat detection. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–9, 2020.

[45] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

[46] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[47] Bijan Yeganeh, Michael G Hewson, Samuel Clifford, Luke D Knibbs, and Lidia Morawska. A satellite-based model for estimating pm2. 5 concentration in a sparsely populated environment using soft computing techniques. *Environmental Modelling & Software*, 88:84–92, 2017.

[48] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman amp; Hall/CRC, 1st edition, 2012.